UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
NOVI SAD

Vladimir Vujović, mr

Modelom upravljani razvoj arhitekture Senzor Web mreža

– doktorska disertacija –

Novi Sad, 2016.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска публикација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Докторска дисертација
Аутор, **АУ**:		Владимир Вујовић
Ментор, **МН**:		Бранко Перишић
Наслов рада, **НР**:		Моделом управљани развој архитектуре Сензор Веб мрежа
Језик публикације, **ЈП**:		Српски
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Србија
Уже географско подручје, **УГП**:		Нови Сад, Војводина
Година, **ГО**:		2016
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Факултет техничких наука, Трг Доситеја Обрадовића 6, Нови Сад
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		6/226/335/18/73/0/1
Научна област, **НО**:		Електротехничко и рачунарско инжењерство
Научна дисциплина, **НД**:		Рачунарство и аутоматика
Предметна одредница/Кључне речи, **ПО**:		Моделом управљан развој софтвера, доменски специфични језици, доменски специфично моделовање, интернет ствари, сензор Web, сензорске мреже
УДК		
Чува се, **ЧУ**:		Библиотека ФТН, Трг Доситеја Обрадовића 6, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Примјена Интернет протокола у уређајима са ограниченим ресурсима, доводи до радикалне промјене Интернета и појаве потпуно новог концепта под називом *Интернет ствари – Internet of Things* (IoT), чији је један од основних градивних елемената *Senzor Web* (SW) чвор. SW чвор представља елементарни "*ресурс*" у SW мрежи која се по својој природи може посматрати као неструктурирана колекција градивних елемената који се могу динамички оркестрирати у виртуелне кластере, односно у архитектуру. Циљ дисертације представља унапређење процеса развоја архитектуре система базираних на SW мрежама уз ослонац на динамичко генерисање сервисног слоја у сврху повећања продуктивности, одрживости и смањења трошкова развоја. Под унапређењем процеса развоја архитектуре сматра се анализа, интеграција и прилагођавање постојећих система и приступа пројектовања архитектуре сензорских мрежа, као и система базираних на IoT концептима. У ту сврху дефинисана је архитектура SW мрежа, креиран доменски специфичан језик, интерактивни графички едитор и алат за аутоматску трансформацију модела у имплементационе класе. У склопу тезе извршена је и експериментална верификација предложеног модела и развојног окружења, чиме је доказана њихова практична примјена.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	Проф. др Горан Стојановић
	Члан:	Проф. др Драган Јанковић
	Члан:	Проф. др Гордана Милосављевић
	Члан:	Доц. др Игор Дејановић
	Члан, ментор:	Проф. др Бранко Перишић

Потпис ментора

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Word printed document	
Contents code, **CC**:		Ph.D. Thesis	
Author, **AU**:		Vladimir Vujović	
Mentor, **MN**:		Branko Perišić	
Title, **TI**:		Model Driven Development of Sensor Web Networks Architecture	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Serbia	
Locality of publication, **LP**:		Novi Sad, Vojvodina	
Publication year, **PY**:		2016	
Publisher, **PB**:		Authors reprint	
Publication place, **PP**:		Faculty of Technical Sciences, Trg Dositeja Obradovica 6, Novi Sad	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		6/226/335/18/73/0/1	
Scientific field, **SF**:		Electrical and Computer Engineering	
Scientific discipline, **SD**:		Computing and Control Engineering	
Subject/Key words, **S/KW**:		Model Driven Development, Domain Specific Language, Domain Specific Modeling, Internet of Things, Sensor Web, sensor networks	
UC			
Holding data, **HD**:		Library of Faculty of Technical Sciences, Trg Dositeja Obradovica 6, Novi Sad	
Note, **N**:			
Abstract, **AB**:		The use of Internet protocols in limited resources devices contributes to radical changes in the Internet and the emergence of an entirely new concept called the *Internet of Things* (IoT), consisted of the *Sensor Web* (SW) nodes as one of the basic building blocks. SW node is the elementary "*resource*" in the SW Network, which by their nature can be seen as an unstructured collection of blocks that can be dynamically orchestrated into the virtual cluster, or in the architecture. The aim of this thesis is to improve the process of developing a system architecture based on SW networks, relying on the dynamic generation of the service layer in order to increase productivity, sustainability and cost of development. The improvement of the architecture development process includes analysis, integration and adaptation of existing systems and sensor network architecture design approaches, as well as systems based on the IoT concepts. For this purpose, the architecture of the SW Network is defined, a domain-specific language has been created as well as interactive graphics editor and a tool for automatic transformation of models into the implementation class. As part of the dissertation, the experimental verification of the proposed model and the development environment were carried out demonstrating their practical application.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Prof. Goran Stojanovic, PhD	
	Member:	Prof. Dragan Jankovic, PhD	
	Member:	Prof. Gordana Milosavljevic, PhD	Menthor's sign
	Member:	Assist. prof. Igor Dejanovic, PhD	
	Member, Mentor:	Prof. Branko Perisic, PhD	

Predgovor

Projektovanje informacionih sistema predstavlja interdisciplinarnu djelatnost čija kompleksnost direktno zavisi od sinergije koju je neophodno ustanoviti između generičkih koncepta, kakvi su informacija i sistem, i postupka inženjerskog projektovanja koji je dominantno tehnološki zavisan. Specifičnosti koje unosi domen problema, odnosno oblast ljudske djelatnosti koja je predmet automatizacije, uslovljavaju potrebu pronalaženja balansa između mogućnosti primjene univerzalnih principa pod ograničenjima koja su specifična za domen. Primjena koncept *inteligentnih informacionih sistemi* (IIS), iako svoje začetke duguje devedesetim godinama prošlog vijeka, i dalje predstavlja značajan naučni i stručni izazov posebno u uslovima globalnog informacionog povezivanja. Oslonjen na integraciju širokog spektra različitih, često nezavisno razvijanih, tehnologija koje su u izolovanim primjenama pokazale značajan potencijal za podizanje efektivnosti i efektivnosti upotrebe informacionih i komunikacionih servisa, IIS podrazumijevaju kontinuirani razvoj i unapređenje efikasnosti alata za njihovo projektovanje. Nivo sofisticiranosti gradivnih elemenata IIS-a dovodi do značajnog usložnjavanja informacione infrastrukture što stvara širok jaz između koncepata domena problema i domena izvršavanja. Poseban izazov predstavlja upotreba senzorskih mreža za izgradnju mjerno-akvizicione infrastrukture, neophodne za praćenje širokog skupa parametara složenih sistema, koja je doživjela snažnu ekspanziju osloncem na bežični prenos podataka između senzorskih uređaja i baznih stanica. Mogućnost izgradnje fleksibilnih topologija i velika pokrivenost prostora, u prvi plan upotrebe senzorske infrastrukture stavlja bežične senzorske mreže.

U posljednjoj dekadi razvoja informacionih tehnologija i Interneta kao dominantan se izdvaja koncept *Internet stvâri – Internet of Things* (IoT). Kao osnovni gradivni blok, on uvodi *Senzor Web – Sensor Web* (SW) i radikalno mijenja odnos između elemenata senzorske mreže i Interneta. Primjena dobro definisanih standarda *Internet Protocol*-a (IP) u uređajima sa ograničenim resursima, kakvi su senzorski elementi, obezbjeđuje

njihovu globalnu dostupnost. Osnovni problem leži u pronalaženju adekvatnog implementacionog rješenja koje obezbjeđuje efikasnu primjenu u uređajima sa limitiranim hardverskim mogućnostima i procesnom moći. Ovakav pristup značajno proširuje koncept IoT i transformiše ga u *Web of Things* (WoT) te otvara potpuno nove perspektive upotrebe globalne mreže. Kao posljedica prethodnog, projektovanje, upotreba i održavanje senzorskih mreža prerasta u interdisciplinarnu oblast koja uključuje sve aspekte SW-a i proširuje skup ekspertskih znanja neophodnih za efikasno i efektivno inženjerstvo u ovoj oblasti.

Dodatni problem predstavlja obezbjeđenje adekvatne podrške klijentima koji su zainteresovani za prikupljanje podataka sa SW mreže bez njene stalne rekonfiguracije. Posljednji zahtjev u prvi plan stavlja podizanje nivoa apstrakcije i razvoj domenski specifičnog rješenja koje skriva unutrašnju složenost topologije SW i obezbjeđuje njenu platformski nezavisnu upotrebu.

Primjenom *Domenski specifičnog modelovanja – Domain-Specific Modeling* (DSM) moguće je značajno podići nivo apstrakcije prilikom razvoja softvera. Modeli na visokom nivou apstrakcije kreirani korištenjem nekog domenskog jezika – *Domain-Specific Languages* (DSL) čija konkretna sintaksa odgovara domenu primjene, omogućavaju lakšu komunikaciju korisnika i projektanata, kao i brži razvoj zahvaljujući mogućnosti potpune ili djelimične transformacije DSL specifikacija u izvršni kôd. Uvođenjem DSM-a, pored dizajna SW, olakšava se i implementacija i razvoj *softverskih sistema – Software Systems* (SS) za podršku rada SW-a.

Enkapsulacijom detalja niskog nivoa i omogućavanje stručnjacima iz domena problema da opišu ponašanje SW mreža uz upotrebu njima bliskih koncepta, fokus se pomjera sa implementacije na dizajn. Dinamičkim generisanjem servisnog sloja SW mreže modelovana arhitektura se integriše u jedinstveni izvršni sistem. Osloncem na prateće alate bazirane na DSM pristupu i podizanjem nivoa apstrakcije, moguće je adekvatnije podržati postupak kreiranja SW mreža. U ovom slučaju, koncepti i sintaksa modela moraju odgovarati kognitivnim oblastima i intuiciji samih eksperata, dok se dizajn u potpunosti prepušta korisnicima.

Cilj disertacije jeste unapređenje procesa razvoja arhitekture sistema baziranih na Senzor Web mrežama uz oslonac na dinamičko generisanje servisnog sloja u svrhu povećanja produktivnosti, održivosti i smanjenja troškova razvoja, gdje se pod unapređenjem procesa razvoja arhitekture

smatra analiza, integracija i prilagođavanje postojećih sistema i pristupa projektovanja arhitekture senzorskih mreža, kao i sistema baziranih na IoT konceptima.

Ostvarivanje osnovnog cilja disertacije je postignuto kroz dekomponovanje istraživanja na skup pojedinačno jednostavnijih zadataka:

a. Detaljnom analizom domena problema je omogućeno prepoznavanje zahtjeva u procesu rješavanja i kreiranja domena rješenja, kao i prepoznavanje potrebnih metodologija i tehnologija pri njegovom rješavanju. Na osnovu postojećih rješenja dobijenih iz pregleda literature, analize potreba i stanja u oblasti domena problema, dat je prijedlog za realizaciju teorijsko-konceptualnog modela arhitekture SW mreže u svrhu uvida u trenutno stanje i nedostatke koji moraju biti otklonjeni.
b. Kreiranjem dobro definisanog pristupa za prilagođavanje postojećih metodologija i tehnika za implementaciju sistema, izvršeno je prepoznavanje ključnih elemenata i metodoloških pristupa te analiza pogodnosti različitih distribuiranih athitektura kao kandidata za integraciju u konačni IS. U kontekstu ostvarenja postavljenih ciljeva, razmotreni su pristupi modelovanja koji definišu sintaksu jezika kojim se opisuju koncepti sistema. Specifikacija apstraktne sintakse izvedena je definisanjem metamodela (dinamičkog servisnog sloja, SW čvorova i REST servisa), relacija koje postoje između definisanih konstrukcija kao i skupa pravila kojima se upravlja interakcija između objekata.
c. Definisanjem mehanizma za preslikavanje i automatsku transformaciju elemenata SW mreže u implementaciju, stvorene su podloge za brz razvoj ove forme IIS-a. Kreiranjem adekvatnih modela specifičnih za domen, koji obuhvataju elemente SW mreže i osiguravaju preslikavanje domena problema na domen rješenja, podržana je transformacija modela u objekte implementacije uz spriječavanje gubitak informacija i osiguranje konzistentnost procesa transformacije. Uključivanjem dodatnih tehnologija i proširenjem postavljenih metamodela, kreirano je arhitekturno rješenja koje predstavlja podlogu za daljnju analizu i implementaciju domenski specifičnog okruženja. U svrhu demonstracije predloženih teorijsko-konceptualnih modela i arhitekture, razvijeno je domenski specifično okruženje za modelovanje arhitekture SW mreže na bazi dinamičkog servisnog sloja – *Sensor Web Services* (SWServ).

d. Adekvatnim izborom slobodno dostupnih alata za modelovanje, postavljeni ciljevi su u potpunosti realizovani. Detaljnom analizom, kao primarno opredjeljenje, zbog svoje jednostavnosti, raširenosti i podrške, izabran je *Eclipse Modeling Framework* (EMF) *ECore* meta-metamodel koji obezbjeđuje adekvatnu podršku za kreiranje apstraktne sintakse. Oslanjajući se na kreirani metamodel, razvijeno je aplikativno okruženje (grafički editor) za kreiranje grafičkih modela, koje uz oslonac na *Eclipse integrisano razvojno okruženje – Integrated Development Environment* (IDE) okruženje i *Sirius* okvir pruža podršku za kreiranje skupa alata za uređivanje modela baziranih na apstraktnoj sintaksi. U posljednjoj fazi izrade, oslanjajući se na *Acceleo* okvir, implementirana je automatska transformacija modela u izvršni kôd čija je ciljana razvojna platforma Java objektno orijentisani jezik, kao i prateće datoteke za specifikaciju SWServ mreže. Kreiranjem aplikativne podrške za definisani proces brzog razvoja arhitektura SW mreža, primjenom konkretnog skupa alata baziranih na modelima i pravilima kroz različite scenarije, verifikovan je predloženi pristup.
e. Verifikacija kreiranog teorijsko-konceptualnog modela i domenski specifičnog okruženja je obavljena nizom eksperimenata sprovedenih u vidu simulacija, kreiranja semantičkih modela i generisanjem kôda. Na osnovu kreiranih semantičkih modela, automatskom transformacijom je generisan niz implementacionih rješenja za ciljanu platformu. Pored simulacija, u završnoj fazi, verifikacija je primjenjena i na prototip namjenski kreirane SW mreže. Simulacijom u realnom okruženju, ispitano je ponašanje generisanog sistema, njegov odziv, promjene, rekonfiguracija (dinamička promjena servisnog sloja i topologije SW elemenata), te dodavanje i uklanjanje senzorskih čvorova. Rezultati analize su predstavljeni u vidu numeričkih i tekstualnih podataka, a njihovom klasifikacijom na osnovu definisanih parametara koji utiču na povećanje produkcije i održavanja SW mreža, ustanovljene su korektivne akcije za optimizaciju sistema. Analizom dobijenih rezultata i korekcijom predloženih modela izvršeno je poboljšanje i unapređenje tezom formulisanog metodološkog pristupa.

Na osnovu model-baziranog kreiranja arhitekture SW mreže i dinamičkog servisnog sloja, te transformacijom u implementacione elemente, osigurana je konzistentnost sistema, kao i redukovanje broja grešaka koje nastaju manuelnom implementacijom. Bazirajući rješenje na dobro poznatim pristupima za razvoj senzorskih mreža, servisnih

arhitektura, kao i standardima i tehnologijama metamodelovanja, otvorena je mogućnost za daljnje unapređenje i razvoj tezom formulisanog metodološkog pristupa. Dobijeni rezultati i konkretni doprinosi nastali kao rezultat rada na ovoj disertaciji mogu se sumirati u sljedećem:

a) Kreiran je dobro definisan proces i jezik za opis arhitekture SW mreža i servisnog sloja. Jezik obezbijeđuje jednostavno kreiranje arhitekture bez suštinskog poznavanja problema arhitekture SW mreže, kao i dinamičkih servisnih elemenata.
b) Kreirana su pravila za transformaciju modela baziranih na jeziku za opis arhitekture SW mreža i servisnog sloja u implementaciju. Mehanizam transformacije modela u implementaciju je bez gubitaka semantike, čime je osigurana konzistentnost samog postupka.
c) Kreirano je radno okruženje i aplikativna podrška za automatizaciju procesa projektovanja arhitekture SW mreža i servisnog sloja na osnovu predloženih modela i pravila transformacije. Integracijom sa postojećim razvojnim okruženjem *Eclipse* IDE, obezbjeđena je podrška za modelom-upravljan razvoj softvera.

Tekst disertacije je organizovan u šest poglavlja, uključujući uvodna razmatranja i zaključak sa diskursom.

Na početku disertacije priložen je popis slika, popis tabela i popis korištenih skraćenica

U prvom poglavlju predstavljen je problemski okvir rada i predmet istraživanja. Uvedeni su ključni koncepti koji su primjenjeni za realizaciju disertacije, kao i kratak pregled publikovanih istraživanja koja se odnose na definisane koncepte.

Drugo poglavlje sadrži detaljan pregled postojećih istraživanja u okviru teze. Izvršena je analiza arhitekture SW mreža i definisana veza između softverskih sistema, komunikacionih tehnologija i senzorskih mreža. Koncepti IoT, WoT i SW su prošireni uključujući detaljan opis implemenatacije i integracije sa informacionim sistemima. Na kraju poglavlja predstavljeni su trenutno dostupni softverski alati za projektovanja senzorskih mreža.

Treće poglavlje sadrži analizu primjene modelom upravljanog razvoja softvera u oblasti SW mreža. Sumiranjem zahtjeva koji su uvedeni u drugom poglavlju, izdvojene su ključne činjenice koje su neophodne za implementaciju MDA rješenja. Dati su odgovori na pitanja: koje dijelove

sistema treba modelovati; koji su to statički a koji dinamički dijelovi sistema; i koja tehnologija predstavlja optimalan izbor za implementaciju predloženog rješenja.

Četvrto poglavlje opisuje dizajn domenski specifičnog jezika i implementaciju pratećih alata za modelovanje i generisanje kôda.

U petom poglavlju je izvršena verifikacija dizajna i implementacije kreiranog domenski specifičnog jezika i pratećih alata za modelovanje i generisanje kôda. Prikazana je implementacija SW čvora i kreiranje modela arhitekture SW mreža za dva odabrana slučaja:

- monitoring parametara životne sredine u zatvorenom prostoru,
- monitoring parametara životne sredine na ograničenom geografskom području.

Provjera postavljenih koncepta je izvršena generisanjem arhitekture SW mreže i servisnog sloja, te njihovim povezivanjem sa softverskim sistemima.

Zaključni dio disertacije u kojem su sumirani rezultati istraživanja, prikazani doprinosi disertacije i dati prijedlozi za daljnja istraživanja i budući rad su prikazani u šestom poglavlju.

Kao prilog, uz spisak korištene literature, na kraju teksta disertacije navedena je i lista objavljenih radova koji su nastali u postupku izrade teze.

Resume

Information Systems Design is a challenging interdisciplinary activity whose complexity is directly dependant on the synergy established between the information and the system, that are more generic concepts, and the process of engineering, that is highly technologically dependent. The specifics introduced by the particular problem domain, namely the area of human activity that is the subject of automation, always demand a fine balance that has to be established between the universal design principles and the specific domain constraints. The application of the concept of *Intelligent Information Systems* (IIS), although introduced in the early nineties of the last century, still represents a significant scientific and technical challenge especially when the global information connectivity is concerned. Relying on the integration of a wide range of different often unrelated technologies that, when independently used, have demonstrated significant potentials for enhancing efficiency and effectiveness of information and communication services delivery, IIS application is impossible without the continued development and improvement of supporting design tools. The level of sophistication of the IIS' building blocks lead to significant complexity of the information infrastructure that creates a wide gap between the concepts of the problem domain and the execution domain. A particular challenge is the use of sensor networks for creating instrumentation and data acquisition infrastructure that is necessary when tracking a huge set of complex systems parameters, especially with the proliferation of the wireless transmission of data between the sensor devices and base stations. The possibility of building flexible topologies as well as the large area coverage raises the importance the wireless sensor networks usage.

In the last decade of the information technology and the Internet development, the concept of the *Internet of Things* (IoT) emerges. This concept introduces *Sensor Web* (SW) as a basic building block and radically changes the relationship between the sensor networks elements and the Internet. The use of well-defined standards of *Internet Protocol* (IP) in sensor elements aids their global reachability. The main problem is

how to gain the efficient implementation solutions with devices that have limited hardware capabilities and the processing power. This approach significantly expands the concept of IoT and transforms it into a *Web of Things* (WoT) and opens up completely new perspectives for the global network utilization. Therefore, the design, use and maintenance of sensor networks becomes an interdisciplinary field that includes all aspects of the SW and expand the collection of expert knowledge necessary for efficient and effective engineering in this field.

An additional problem is the provision of adequate support to customers who are interested in gathering data from the SW network without its constant reconfiguration. The last request consequently raises the abstraction level and inaugurates the development of domain-specific solutions that hide the inner complexity of the SW topology and provide their platform-independent use.

Domain-Specific Modeling (DSM) can significantly raise the level of abstraction when engineering software. High level abstraction models, created by *Domain-Specific Languages* (DSL) with the application domain compliant syntax, enable easier communication between users and designers, as well as the faster development due to the total or partial transformation of DSL specifications into the executable code. The introduction of the DSM, in addition to SW design, facilitates the implementation and development of *Software Systems* (SS) that support the SW operation.

Encapsulation of low level abstraction details accompanied with the experts' ability to describe the behavior of the SW network in the problem domains using the concepts they are familiar with, shifts focus from the implementation to the design phase. Through dynamically generated service layer of SW network, modeled architecture may be integrated into a compact executing system. Relying on supporting tools based on the DSM approach and raising the level of abstraction, it is possible to adequately support the process of creating arbitrary SW network. In this case, the concepts and syntax of the model must match the cognitive areas and experts' intuition, while the design is completely left to the users.

The aim of this thesis is to improve the process of developing a system architecture based on Sensor Web networks based on the dynamic generation of the service layer with regard to the increase of productivity, sustainability and the development costs. The improvement of the architecture development process includes: analysis, integration and

adaptation of existing systems and sensor networks architecture design approaches, as well as systems based on the IoT concepts.

The main objective of the dissertation is achieved through research decomposition to the following, simpler, tasks:

a. A detailed analysis of the problem domain enabled requirements identification and the creation of domain solutions, as well as identification of the methodologies and technologies necessary for the solution finding process. In order to get an insight into the current state and the deficiencies that must be corrected, based on the related work analysis and the domain specific needs and conditions, the proposition of theoretical and conceptual model of SW network architecture implementation has been formulated
b. Based on well-defined adaptation process of the existing system implementation methodologies and techniques, the identification of key elements and the methodological approaches has been performed, as well as the analyzes of various distributed architectures as suitable candidates for integration into the final IS. In the context of defined set of goals, the modeling approaches, that define the language syntax used to describe the concepts of the system, were discussed. Specification of abstract syntax is derived by defining the metamodel (dynamic service layer, SW nodes and REST services), the existing relations between the defined structures and the set of rules that govern the interaction between objects.
c. The foundation for the rapid development of this form of IIS has been created by specifying a mapping mechanism and the automatic transformation of SW network elements into the executable implementation. Creation of appropriate domain specific models, which include elements of the SW Network and provide a problem domain mapping into domain solutions, support the transformation of the model to the implementation objects by preventing the loss of information and ensuring the consistency of the transformation process. The architectural solution, representing the basis for further analysis and implementation of domain specific environment, has been created by incorporating new technologies and defined metamodel extensions. To demonstrate the proposed theoretical and conceptual model and the architecture, domain specific environment for modeling SW network architecture based on dynamic service layer - *Sensor Web Services* (SWServ) has been developed.

d. Defined objectives have been fully implemented through the adequate selection of open source modeling tools. Through the detailed analysis, the *Eclipse Modeling Framework* (EMF) *ECore* meta-metamodel that provides adequate support for the creation of an abstract syntax was elected as primary choice, due to its simplicity and widespread support. Referred to the designed metamodel, an applicative environment (graphical editor) has been developed in order to support the creation of graphical modeling. Based on the abstract syntax, the *Eclipse Integrated Development Environment* (IDE) and *Sirius* framework the set of tools that support models management (editing) have been created. In the final phase of development, relying on *Acceleo* framework, the automatic transformation of models into executable code whose target development platform is Java object-oriented language, as well as accompanying files to specify SWServ network, were implemented. The proposed approach is verified by creating application support for a defined process of rapid SW network architecture development.
e. The verification of created theoretical and conceptual model and domain specific environment has been performed through a series of experiments conducted in the form of simulation, semantic models' creation and code generation. Based on the created semantic models, a series of implementing solutions for the target platform have been generated through the automatic transformation process. In addition, in the final stage, the created prototype of SW Network has been verified. By simulating the real environment, the behavior of generated system is examined as well as its response, changes, reconfiguration (dynamic changes in the service layer and topology of SW elements), and the addition and removal of sensor nodes. The analysis results are presented in the form of numeric and textual data. Their classification, based on defined parameters that influence the production increase and maintenance of the SW Network, have established the corrective actions for system optimization purposes. The improvements and advancements of the methodological approaches, formulated in this thesis, have been performed via results analysis and the set of proposed model refinements.

With model driven SW Network architecture development, based on the automatic generation of dynamic service layer, the reduction of errors that usually occur with manual transformation is achieved, thereby aiding

the overall system consistency. An opportunity for further improvement and development of the thesis formulated methodological approach is based on the well-known solution approaches for the development of sensor networks, service architectures, as well as standards and metamodeling technologies. The results and the specific contributions as an outcome of the work on this thesis can be summarized as follows:

a) The well-defined process and description language for SW network architecture and service layer are established. The created language provides rapid architecture development without substantial knowledge of the SW Network architecturing, as well as accompanied dynamic service elements.
b) The rules for models transformation, based on language for SW network architecture description and dynamic generation of service layer, are defined. The mechanism of the model transformation into the implementation is without semantics losses, thus ensuring the consistency of the overall procedure.
c) The operating environment and application support to automate the process of SW network architecture and service layer design, based on the proposed models and the transformation rules are created. By integrating it with existing *Eclipse* IDE, the support for model-driven software development is provided.

The text of the dissertation is organized into six chapters, including introductory sections and the conclusion with the discourse.

At the beginning of the thesis, the lists of: figures, tables and the abbreviations are presented.

The first chapter discusses problem domain and the research subject. The key concepts that have been applied for the realization of the thesis, as well as a brief overview of published research related to the defined concepts, are described.

The second chapter contains a detailed review of existing published researches related to the thesis. The analysis of SW network architecture is performed and relationship between software systems, communication technologies and sensor networks is defined. The IoT, WoT and SW concepts have been expanded, including a detailed description of their implementation and integration with information systems. In the concluding part of the chapter, the currently available software tools for designing sensor networks are illustrated.

The third chapter provides an analysis of the application of model driven software development in the area of SW networks. By adding together the requirements that were introduced in the second chapter, the key facts that are necessary for the implementation of MDA solutions, are pointed out. The answers to the following questions are provided: which parts of the system have to be modeled; which parts of the system are static and which are dynamic; and which technology is the optimal choice for the implementation of the proposed solutions?

The fourth chapter describes the detailed design of domain specific language and the implementation of supporting tools for modeling and code generation.

In the fifth section the verification of the design and implementation of the created domain-specific language and related tools for modeling and code generation is performed. The implementation of SW node and creation of a SW network architecture model is shown through two selected cases:

- Monitoring of environmental parameters in indoor space,
- Monitoring of environmental parameters in outdoor space.

The verification of the formulated concept is performed by SW network architecture and service layers generation together with their integration with arbitrary software systems.

The final part of the dissertation summarizes the outcomes of research, presents contributions of the dissertation and gives suggestions for further research and future work directions are presented in chapter six.

As an attachment, along with a list of the references used, at the end of the dissertation a list of research related publications, presented on the conferences and published in international Journals, is attached.

Sadržaj

Popis slika

Poglavlje 4

Poglavlje 5

Popis tabela

Popis skraćenica

6LoWPAN	IPv6 over Low-power Wireless Wrea Networks
AP	Access Point
API	Application Programming Interface
ARM	Architectural Reference Model
ARM	Advanced RISC Machine
ATL	Atlas Transformation Language
BMS	Building Management System
BSM	Bežičnih senzorskih mreža
CBIS	Computer-Based Information System
CDMA	Code-Division Multiple Access
CIM	Computation Independent Model
CoAP	Constrained Application Protocol
CoRE	IETF Constrained RESTful Environments
CPU	Central Processing Unit
CRUD	Create-Read-Update-Delete
CSI	Camera Serial Interface
DAQ	Data Aquisition Systems
DIY	Do-It-Yourself
DSI	Display Serial Interface
DSL	Domain-Specific Languages
DSM	Domain-Specific Modeling
DSML	Domain-Specific Modeling Language
DTN	Delay Tolerant Network
EDGE	Enhanced Data rates for GSM Evolution
EEF	Extended Editing Framework
EMF	Eclipse Modeling Framework
EMOF	Essential MOF
EMP	Eclipse Modeling Project
EPC	Electronic Product Code
ETL	Epsilon Transformation Language
ETSI	European Telecommunications Standards Institute
FCC	Federal Communications Commission
FTP	File Transfer Protocol
GEF	Graphical Editing Frameworka
GMF	Graphical Modeling Framework
GPIO	General Purpose Input and Output

GPL	General Purpose Programming Languages
GPML	General Purpose Modelling Language
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HDMI	High Definition Multi-Media Interface
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
I^2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IERC	IoT European Research Cluster
IIS	Intelligent Informtion Systems
IKT	Informaciono Komunikacione Tehnologije
IoC	Internet of Content
IoE	Internet of Events
IoL	Internet of Locations
IoP	Internet of People
IoT	Internet of Things
IP	Internet Protocol
IS	Informtion Systems
ISM	Industrial, Scientific and Medical
ITU	International Telecommunication Union
JET	Java Emitter Templaes
JSON	JavaScript Object Notation
KS	Komunikacioni Sloj
M2M	Model-to-Model
M2T	Model-to-Text
MDA	Model-Driven Architecture
MDE	Model-Driven Engineering
MDSD	Model-Driven Software Development
MIME	Multipurpose Internet Mail Extension
MOF	Meta Object Facility
MVC	Model-View-Controller
O&M	Observation and Measurement
OCL	Object Constraint Language
OGC	Open Geospatial Consortium
OS	Operating System
OSWA	Open Sensor Web Architecture
PIC	Programmable Intelligent Computer
PIM	Platform Independent Model
PSM	Platform Specific Model
QoS	Quality-of-Service
QVT	Query/View/Transformation
RBAC	Role Base Access Control
REnvDB	RESTful Environmental DataBase
REST	Representational State Transfer

RFID	radio frekvencijski identifikatori
ROA	Resource Oriented Architecture
RPi	Raspberry Pi
RSS	Really Simple Syndication
SANET	Wireless Sensor and Actuator Networks
SAS	Sensor Alert Service
SCADA	Supervisory Control and Data Acquisition
SCS	Sensor Collection Service
SensorML	Sensor Modeling Language
SM	Senzorske Mreže
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SoC	System on a Chip
SOS	Sensor Observation Service
SoS	System-of-Systems
SPI	Serial Peripheral Interface Bus
SPS	Sensor Planning Service
SS	Software Systems
SSD	Solid-State Drive
SSL	Secure Sockets Layer
SW	Sensor Web
SWE	Sensor Web Enablement
SWServ	Sensor Web Services
SWT	Standard Widget Toolkit
TDMA	Time-Division Multiple Access
TIA	Telecommunications Industry Association
TransducerML	Transducer Model Language
UDP	User Datagram Protocol
ULSC	Ultra-Large-Scale Systems
UML	Unified Modeling Language
UMS	USB Mass Storage
UMTS	Universal Mobile Telecommunication System
URI	Uniform resource identifier
URI	Uniform Resource Locator
VPN	Virtual Private Network
VSM	Viewpoint Specification Model
VSP	Vewpoint Specification Projects
W3C	World Wide Web Consortium
WAR	Web application ARchive
WBAN	Wireless Body Area Network
WFI	Wait for Interupt
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WNS	Web Notification Service
WoT	Web of Things

WPAN	Wireless Personal Area Network
WRS	Web Register Service
WWAN	Wireless Wide Area Network
WWW	World Wide Web
XHTML	EXtensible HyperText Markup Language
XMI	XML Metadata Interchange
XML	EXtensible Markup Language
XMPP	Extensible Messaging Presence Protocol

Poglavlje 1

Uvodna razmatranja

1.1. Problemski okvir rada

Primjena koncept inteligentnih informacionih sistemi (IIS), iako svoje začetke duguje devedesetim godinama prošlog vijeka [Kerschberg90, Wiederhold92, Seligman93, Potter93, Perisic94, Gomaa96, Kerschberg96, Kerschberg00] i dalje predstavlja značajan naučni i stručni izazov posebno u uslovima globalnog informacionog povezivanja. Oslonjen na integraciju širokog spektra različitih, često nezavisno razvijanih, tehnologija koje su u izolovanim primjenama pokazale značajan potencijal za podizanje efektivnosti i efektivnosti upotrebe informacionih i komunikacionih servisa, IIS podrazumijevaju kontinuirani razvoj i unapređenje efikasnosti alata za njihovo projektovanje. Nivo sofisticiranosti gradivnih elemenata IIS-a dovodi do značajnog usložnjavanja informacione infrastrukture što stvara širok jaz između koncepata domena problema i domena izvršavanja.

Kao posljedica toga, savremeni informacioni sistemi (IS) predstavljaju skup heterogenih, međusobno povezanih elemenata ili komponenti koje prikupljaju (*input*), manipulišu (*process*), skladište i distribuiraju (*output*) podatke i informacije, te osiguravaju korektivne akcije (povratni mehanizam) koje dovode do poboljšanja ili ispunjenja postavljenih ciljeva sistema [Stair12]. Oni integrišu hardver, softver, skladišta (baze) podataka, telekomunikacione jedinice, korisnike i procedure za prikupljanje, manipulisanje, skladištenje i obradu podataka i njihovu konverziju u informacije. Potpuna automatizacija sistema realnog svijeta koja uključuje neposrednu integraciju segmenata koji rade u realnom vremenu, predstavlja izazov na koji je neophodno odgovoriti adekvatnim naučno-stručnim metodama i rješenjima. Normativno regulisanje kompleksnih segmenata ljudske djelatnosti uslovljava intenzivniji oslonac na savremene

informacione i komunikacione tehnologije koje obezbjeđuju podloge za provjeru stepena usaglašenosti normativnog i stvarnog.

Na primjer: zakon o zaštiti životne sredine u nekim državama uključuje i odredbe o praćenju stanja čovjekove okoline, čime se dobijaju detaljne informacije o uticaju budućih i realizovanih projekata, planova i programa na čovjekovu okolinu [Hart06]. Ispunjenje tih odredbi, zahtjeva kreiranje hardversko/softverske infrastrukture za nadgledanje distribuiranih parametara životne sredine koji mogu dovesti do debalansa i/ili narušavanja sigurnosti [Gupta12, Strandell02]. Također, primjena informacionih tehnologija je sve više prisutna u poljoprivredi [Mendez13], medicini [Yang14] i drugim složenim sistemima formiranim od strane čovjeka.

Osnovni gradivni blok sistema za monitoring i prikupljanje parametara iz okruženja, predstavljaju *senzorske mreže* (SM) preko kojih posvećeni segment IS-a prikuplja podatke iz okoline. Priroda i kompleksnost primjene SM u prošlosti indukovali su dvije osnovne grupe problema:

- intenzivan nivo operativne podrške u procesu prikupljanja podataka zbog nemogućnosti korištenja globalne komunikacione infrastrukture, i
- visoki troškovi implementacije SM i rigidnost arhitekture zbog zahtjeva vezanih za robustnost [Sadilek08].

Pojavom Interneta, IS i SM postaju distribuirani i dostupni na globalnom nivou, ispunjavajući glavni uslov - "*bilo kad i bilo gdje*", dok uvođenje *bežičnih senzorskih mreža* (BSM) uspješno rješava probleme visokih troškova i rigidnosti arhitekture SM.

BSM predstavljaju kompoziciju prostorno distribuiranih, autonomnih, senzorskih čvorova opremljenih:

- senzorskom jedinicom (za prikupljanje podataka iz okoline),
- procesorskom jedinicom (za obradu prikupljenih informacija),
- jedinicom za komunikaciju (najčešće, bežični primo-predajnik),
- ugrađenom memorijom (jedinica za smještanje podataka) i
- autonomnim izvorom napajanja (energije).

Uprkos postojanju ograničenja, kao što su: neadekvatan izvor napajanja, ograničena procesorska snaga i operativna memorija, BSM predstavljaju oblast koja se veoma brzo razvija u raznovrsnim poljima istraživanja i komercijalnoj upotrebi, pa se BSM mogu pronaći uveliko u vojnoj primjeni, u sistemima za praćenje stanja životne sredine, inteligentnim poljoprivrednim sistemima, sistemima za nadzor zdravlja,

nacionalnoj sigurnosti, sigurnosti saobraćaja te kontroli i praćenju u industriji i razvoju [Ali10]. Savremene BSM postaju dio globalne mreže - Interneta, zbog čega je potrebno tehnološki riješiti infrastrukturne probleme: sa jedne strane globalizacija i Internet protokoli, a sa druge strane topologija i realizacija senzorskih mreža. U literaturi se sreću dva gruba pristupa rješavanju navedenih problema. Prvi podrazumijeva oslonac na komunikaciju posredstvom izlazne tačke ("*gateway*"-a), koja predstavlja adapter između Internet-a i protokola na kojima su implementirane BSM mreže. Drugi pristup koristi tzv. komunikaciju s *kraja-na-kraj* ("*end-to-end*") zasnovanu na autonomnim uređajima koji posjeduju vlastitu IP adresu ("*IP enabled things*") [Yazar09].

Unapređenje tehnologija BSM, te primjene *Internet protokola* (IP) u uređajima sa ograničenim resursima (kao što su senzorski čvorovi), radikalno mijenjaju Internet, kreirajući potpuno novi koncept pod nazivom *Internet stvari – Internet of Things* (IoT). Jedan od osnovnih i najvažnijih gradivnih elemenata IoT je senzorski čvor, odnosno *Senzor Web – Sensor Web* (SW). SW se može definisati kao Web međusobno povezanih heterogenih senzora koji su: interoperabilni, inteligentni, dinamički, skalabilni i fleksibilni. Drugi pristup posmatra SW kao grupu interoperabilnih web servisa usklađenih sa određenim skupom ponašanja i interfejsa koji odgovaraju senzorskom čvoru [Di07], pri čemu su razlike između običnih senzorskih čvorova i SW-a za krajnjeg korisnika transparentne [Delin05].

Prepoznavajući prednosti koje nudi koncept SW, *Open Geospatial Consortium* (OGC) *Network* zajednica je kreirala standarde pod nazivom *Sensor Web Enablement* (SWE), koji osiguravaju ključne ciljeve SW-a. Standard podrazumijeva da svi senzori: imaju konekciju na Web, prijavljuju svoju poziciju, posjeduju meta-podatke i imaju mogućnost daljinskog očitavanja podataka i upravljanja.

Glavni cilj SWE-a predstavlja detekcija i očitavanje podataka sa bilo kojeg senzora, prema potrebama krajnjih korisnika, nezavisno od toga gdje se oni (senzori ili korisnici) nalaze.

U tom smislu funkcionalnost SW-a mora da uključi i [Martinez11]:

- otkrivanje senzorskih sistema, zapažanja i posmatranja procesa koji zadovoljava oblast primjene ili potrebe korisnika,
- određivanje mogućnosti senzora, te kvaliteta mjerenja njihovih podataka,

- omogućavanje pristupa parametrima senzora koji automatski dozvoljavaju očitavanje mjerenih i geopozicionih podataka,
- rad u realnom vremenu (*real-time*) ili rad u vremenskim intervalima (*time-series*),
- paralelni (multitasking) pristup informacijama,
- mogućnost za alarmiranje od strane senzora ili sistema u zavisnosti od postavljenih uslova.

Na osnovu specifikacija SWE standarda, SW se može posmatrati kao skup Web servisa, za koje je neophodno pronaći adekvatan način implementacije.

Zbog hardverskog ograničenja SW čvorova, izbor adekvatnog stila implementacije interakcije između servisa i korisnika preko Interneta predstavlja glavno polje istraživanja i ključ ekspertnosti dizajnera i programera IS i fokus disertacije kada je u pitanju problemski okvir istraživanja.

Sadržana interdisciplinarnost problemskog okvira istraživanja zahtjeva preciznu formulaciju pojmovne strane domena problema. U sklopu paragrafa 1.2. formulisan je osnovni skup pojmova koji čine polaznu ontologiju disertacije.

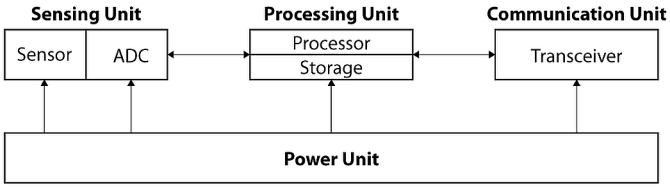
1.2. Pojmovni okvir rada

Da bi se na adekvatan način izvršila analiza i postiglo suštinsko razumijevanje domena problema, neophodno je uvesti precizne definicije osnovnih pojmova korištenih, kako u fazi analize, tako i u fazama dizajna i implementacije. U nastavku poglavlja su date definicije senzorskih mreža i bežičnih senzorskih mreža kao i novih paradigmi *Internet of Things* (IoT), *Web of Things* (WoT) i *Sensor Web* (SW). Servisno orijentisane arhitekture predstavljaju okosnicu za implementaciju pomenutih koncepta, dok metodologija modelom upravljanog razvoja softvera omogućava kreiranje novog pristupa u procesu razvoja SW mreža.

Opredjeljenje za primjenu metodologije modelom upravljanog razvoja softvera i implementaciji grafičkih alata za podršku formulisanju i transformacijama domenski specifičnih modela zahtjeva pojmovno definisanje i ovog segmenta disertacije.

1.2.1. Senzorske mreže i bežične senzorske mreže

Senzorske mreže (SM) predstavljaju skup malih i jeftinih senzora koji prikupljaju i distribuiraju podatke iz realnih sistema, a pomoću kojih je moguće pratiti i upravljati parametrima posmatranog sistema sa udaljenih lokacija [Bharat01, Buratti09]. SM predstavlja uređeni skup senzorskih čvorova, koji su najčešće opremljeni: senzorskom i procesorskom jedinicom, jedinicom za komunikaciju, ugrađenom memorijom i izvorom napajanja, dok su senzorski čvorovi kod *bežičnih senzorskih mreža* (BSM) najčešće opremljeni i sa: bežičnim primo-predajnikom te autonomnim izvorom napajanja. Na senzorski čvor se mogu povezati razni mehanički, termički, biološku, hemijski, optički i magnetni senzori [Yick08] koji obezbjeđuju prikupljanje vrednosti parametara posmatranog sistema. Tipična arhitektura senzorskog čvora prikazana je na slici (Slika 1.2.1.1) [Nack09].



Slika 1.2.1.1. Pojednostavljena arhitektura tipičnog senzorskog čvora [Nack09]

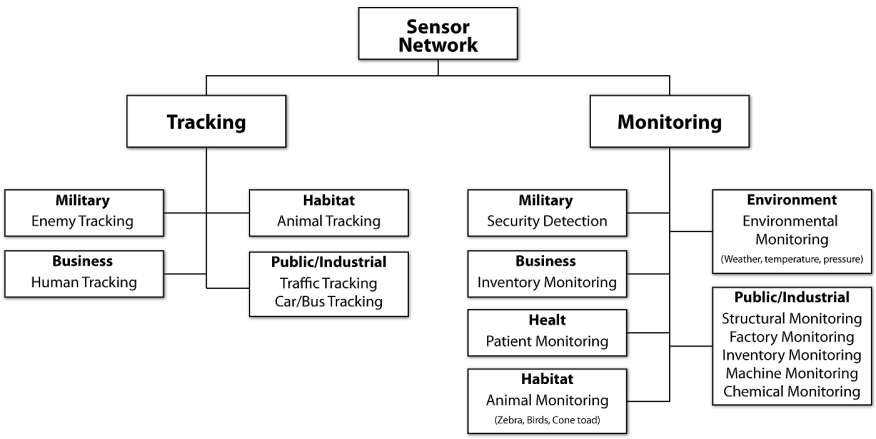
Opcione komponente senzorskog ili bežičnog senzorskog čvora obuhvataju: jedinicu za određivanje lokacije senzora (npr. GPS prijemnik), mobilizator (blok za pokretanje senzorskog čvora, koristi se kada senzor treba da postane mobilan), generator energetskog napajanja (blok koji vrši konverziju energije, npr. solarna baterija).

Zbog relativno niske cijene, malih dimenzija senzorskih čvorova, mogućnosti aplikacije na nepristupačnim terenima i potpuno autonomnog rada, BSM nalazi sve veću upotrebu kako u svakodnevnim primjenama tako i u istraživanjima. Analizirajući više od 50 slučajeva primjene BSM Heart i Martinez-ov [Heart06] iznose tvrdnju da će BSM postati standardan istraživački alat budućnosti koji obuhvata:

- globalne senzorske mreže sa jednom funkcijom,
- lokalizovane višenamjenske senzorkse mreže,
- biosenzorske mreže, i
- heterogene senzorske mreže.

Dosadašnji razvoj tehnologije BSM uglavnom se odvijao kroz specifičnu upotrebu koja je zahtijevala posebne uslove za neometano funkcionisanje senzorskih čvorova, Neki od njih su: visoki pritisci, ekstremne hladnoće i vrućine, velike brzine kao i prisistvo visokog nivoa šumova, namjernih ometanja i računarskih napada [Sharma10]. Analiza izvršena za skup dosadašnjih primjena BSM mreža, pokazala je velike razlike po pitanju zahtjevanih karakteristika što dodatno otežava definisanje tipičnih zahtjeva po pitanju hardverskih rješenja, komunikacionih tehnika i softverske podrške [Romer04]. Zbog multidisciplinarne prirode BSM, navedena situacija unosi dodatne probleme u implementaciju, pa je potrebna bliska saradnja korisnika i različitih stručnjaka iz oblastima primjene, razvoja hardvera, softvera i telekomunikacionih tehnologija. Klasifikacija BSM u zavisnosti od upotrebe, dovodi do uniformih rješenja i smanjenja cijene razvoja hardverskih platformi i softverske podrške.

Analiza literaturnih izvora pokazuje da se BSM najčešće primjenjuju u: automatizaciji životnog prostora (pametna kuća, sistem za upravljanje zgradom), vojsci, industriji, kontroli saobraćaja, građevinarstvu, poljoprivredi, medicini i zdravstvu [Katiyar10, Buratti09, Hart06, Nack09, Yick08, Sharma10]. Yick i dr. u svom radu [Yick08] kategorizuju BSM u dvije osnovne grupe primjena (Slika 1.2.1.2): *praćenje* (*tracking*) i *nadgledanje* (*monitoring*).



Slika 1.2.1.2. Pregled primjene senzorskih mreža [Yick08]

Veliki broj autora [Katiyar10, Hussain09, Doumit02, Akyildiz02, Chinru06, Lorincz04, Wood06, Anliker04, Holger05, Buratti09,

Sharma10, Townsend05, Yick08] daje svoj doprinos poboljšanju primjene BSM u navedenim oblastima.

Primjena BSM baziranih na heterogenim ugrađenim sistemima, senzorima i komunikacionim protokolima zahtijeva ispunjenje sljedećih ciljeva:

- sistem mora biti jednostavan za korištenje. BSM mora biti samostalno konfigurisana kako bi se inicijalna ulaganja i troškovi koji su uključeni u njeno održavanje zadržali na racionalnom nivou,
- sistem mora da ima visok nivo tolerancije na greške. U slučajevima otkaza čvora, BSM mora biti u stanju da pronađe alternativne pravce za isporuku informacija,
- fleksibilna arhitektura. Senzorski čvorovi BSM moraju biti locirani u različitim oblastima ukoliko je to izvodljivo,
- rad BSM mora biti energetski efikasan,
- fleksibilna periodičnost (komunikacija sa centralnim čvorom) mora biti nezavisno podesivi parametar u svakom čvoru i njena modifikacija mora biti moguća u svakom trenutku, čak i kada je čvor u režimu spavanja,
- rad BSM mora da bude pouzdan što podrazumijeva isporuku podataka sa niskim procentom grešaka,
- sistemsko kašnjenje mora biti svedeno na minimum,
- BSM mora imati izraženu sistemsku fleksibilnost,
- sistemska bezbjednost i privatnost su od suštinske važnosti i zato BSM mora da bude otporna na zlonamjerne napade.

Najčešći zahtjevi koji se postavljaju prilikom kreiranja BSM istovremeno predstavljaju i najznačajnije izazove istraživanja u domenu BSM i obuhvataju [Holger05, Townsend05, Yick08, Nack09, Culler04]:

- energetsku efikasnost i vrijeme života mreže,
- pokrivanje i mogućnost povezivanja (konektivnost),
- topologiju mreže,
- samo-organizovanje,
- pokretljivost čvorova,
- prikupljanje podataka,
- sigurnost i problem privatnosti, i
- kvalitet servisa - *Quality-of-Service* (QoS).

U radovima [Marković, Jardosh08, Camilo08, Rajago06, Li09, Wang06, Abidin09, Townsend05, Yick08, Nack09, Culler04] elaborirana su poboljšanja arhitekture i metodologije projektovanja koje je moguće primjeniti na pojedinačne zahtjeve, sa ciljem optimizacije BSM.

BSM predstavljaju izuzetno interesantnu multidisciplinarnu oblast istraživanja, koju odlikuje veliki broj domena i načina primjene zbog male potrošnje, dimenzija i niske cijene, posebno u oblastima gdje je upotreba postojećih sistema za mjerenje ekonomski neopravdana ili logistički neizvodiva. Savremeni trendovi razvoja BSM usmjereni su na:

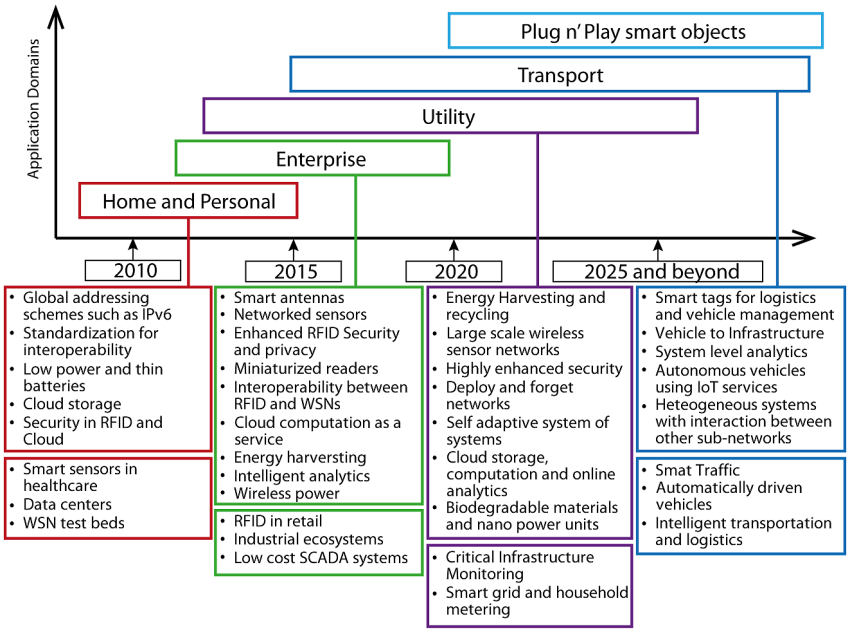
- kognitivno opažanje,
- upravljanje opsegom djelovanja,
- koordinaciju u heterogenim mrežama,
- vremenski kritične primjene,
- eksperimentalne postavke i nove primjene,
- probleme nepokrivenosti,
- probleme vremenske sinhronizacije i lokalizacije,
- nove modele i arhitekture.

1.2.2. Internet of Things

Razvojem Interneta i mrežne infrastrukture globalizacija postaje primarni fenomen koji danas ima dominantnu ulogu u svim elementima ljudskog djelovanja. Preplitanje virtualnog i realnog svijeta postaje ne samo imaginacija i fikcija, već i potreba koja dovodi do razvoja novih koncepta i naprednih tehnoloških pristupa. U eri kada su usluge dostupne putem Interneta sa pametnih telefona, više od dvanaest milijardi senzora je već povezano na: prirodne izvore, vozila, proizvodne linije, električne mreže, urede, i domove, konstantno obezbjeđujući informacije korisnicima. Sve ono što nas okružuje, postaje sastavni dio Interneta i novog koncepta zvanog *Internet stvari – Internet of Things* (IoT). U radu [Vermesan13] IoT je predstavljen kao koncept i paradigma koja razmatra rasprostranjene stvari/objekte koji, bežičnim ili žičanim vezama uz primjenu jedinstvenih adresnih šema, omogućavaju komunikaciju i saradnju u cilju kreiranja novih aplikacija i/ili usluga.

Osnovna ideja IoT jeste da gotovo svaka fizička stvar postane računar koji je povezan na Internet. Pod pojmom "*postane računar*" podrazumjeva se koncept, koji se često referencirane kao "*pametne stvari*" ("*smart things*") [Fleisch10], a predstavlja integraciju malih računara i stvari. Uz oslonac na postojeću infrastruktru i novu verziju Internet protokola – *Internet Protocol* (IP) – IPv6, primjena IoT postaje praktično neograničena. U radovima [Vermesan13, Fantana13, Gubbi13, Fleisch10, Santucci08] analizirane su oblasti primjene IoT kao i ekonomska opravdanost ovog pristupa. U [Gubbi13] predstavljene su četiri kategorije

primjene IoT: lična i kućna, poslovna, pomoćna (utility) i transportna (Slika 1.2.2.1).



Slika 1.2.2.1. Smjernice ključnog tehnološkog razvoja u IoT kontekstu [Gubbi13]

Moguće je uočiti tri komponente IoT koje podržavaju djelotvorno sveprisutno računarstvo [Gubbi13]:

- hardver – sastoji se od senzora, aktuatora i ugrađenih primo-predajnika,
- posrednički sloj (*middleware*) – alat za skladištenje i obradu analitike podataka, i
- prezentacija – alati za vizuelizaciju i interpretaciju kojima se može pristupiti sa različitih platformi i iz konteksta različitih aplikacija.

Neke od osnovnih tehnologija koje navedene komponente podržavaju su [Gubbi13, Pande14, Haller10]: radio frekvencijski identifikatori (RFID), bežične senzorske mreže, adresne šeme, čuvanje i analiza podataka i vizuelizacija.

1.2.3. Web of Things

IoT se ne nalazi na istom nivou kao Internet. One samo predstavljaju njegovu primjenu i u tom smislu su mnogo bliže Web servisima. Na osnovu toga koncept IoT je transformisan u koncept *Web stvari – Web of Things* (WoT) [Fleisch10]. Pojam WoT se prvi put pojavljuje 2011. godine u doktorskim disertacijama Trifa i Guinard-a [Trifa11, Guinard11] kao arhitekturni stil koji omogućava povezivanje relnih objekata (stvari) sa *World Wide Web*-om (WWW) [Guinard10, Guinard09, Guinard11a].

U [Zeng11] WoT se posmatra kao nova vizija IoT u kojoj su svakodnevni predmeti i uređaji/objekti, koji imaju ugrađene Web servise, povezani i u potpunosti integrisani sa Web-om. Za razliku od mnogih sistema koji se primjenjuju pri implementaciji IoT, WoT koriste dobro poznate i opšte prihvaćene Web standarde i protokole kao što su: *Uniform resource identifier* (URI), *Hyper Text Transfer Protocol* (HTTP), *Representational State Transfer* (REST), *Really Simple Syndication* (RSS), itd.

U radovima [Guinard09, Guinard10, Pérez14] autori za implementaciju WoT predlažu primjenu standardnih Web tehnologija, odnosno RESTful Web servisa i HTTP protokola. Na ovaj način realni objekti se predstavljaju kao RESTful resursi kojima se može direktno pristupiti putem Web-a. WoT koncept podrazumijeva da: svaka sadržana informacija posjeduje jedinstveni URI, kodira se standardnim *Multipurpose Internet Mail Extension* (MIME) tipovima i prenosi uz oslonac na HTTP protokol [Tridium09].

1.2.4. Senzor Web

Senzor Web (SW) koncept se prvi put pojavljuje 1999. godine u radu Delina i dr. [Delin99], nastalog u okviru *NASA Sensor Web Applied Research Planning Group*, gdje je predstavljen kao autonomno organizovana bežična senzorska mreža koja se može koristiti za nadzor i praćenje parametara okruženja. U [Delin02, Delin05] ilustrovano je konstantno unapređenje SW koncepta i njegovih primjena.

Glavne funkcionalnosti SW-a se mogu predstaviti kao [Botts06, Botts07]:

- pronalaženje senzora i senzorskih podataka,
- opis senzora i mjerenja (npr. pouzdanost i tačnost),

- pristup senzorskim parametrima,
- pristup mjerenim podacima (podaci u realnom vremenu i periodični podaci) na osnovu standardizovanih formata podataka,
- dodjeljivanje zadataka senzoru,
- mehanizmi upozoravanje na osnovu senzorskih mjerenja i kriterijuma za upozoravanje.

Zbog velikog broja različitih proizvođača senzora koji kreiraju vlastite protokole, spajanje senzorskih elemenata u monitoring sistem i razvoj jedinstvenog aplikativnog rješenja postaje kompleksan zadatak [Funk11]. Potencijalno rješenje ovog problema predstavlja integracija TCP/IP pristupa senzorima preko mrežnog ili bežičnog interfejsa. Pojavom *Sensor Web Enablement* (SWE) okvira, formulisanog od strane OGC zajednice, došlo je do znatnijeg pomaka i usvajanja potpuno novog značenja koncepta SW-a [Botts06, Botts07, Funk11].

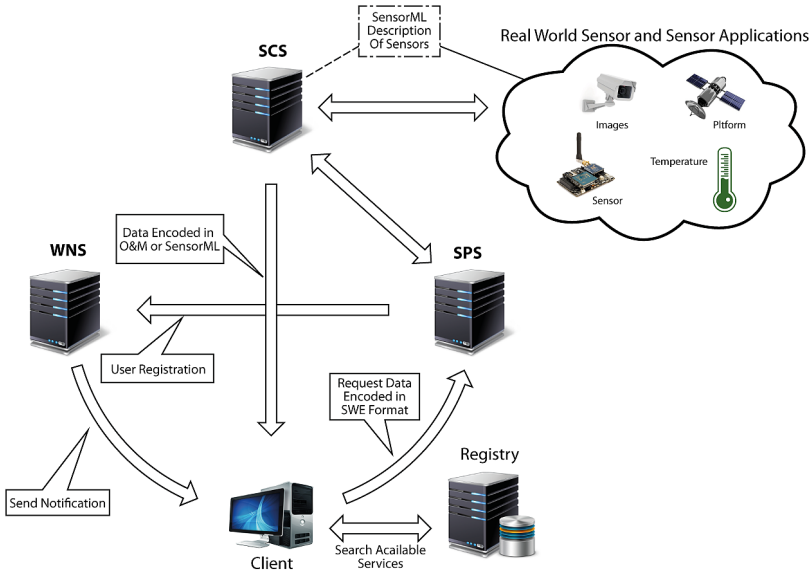
Osnovna vizija SWE je pojednostavljenje veza i saradnje između različitih senzorskih mreža uz oslonac na WWW infrastrukturu, kao i definisanje standarda za globalnu upotrebu senzorskih mreža. SWE predstavlja *de facto* standard za razvoj SW-a [Chu07] pri čemu ne nudi konkretno implementaciono rješenje, već definiše skup protokola i Web servisa koji će biti korišteni [Funk11].

SWE definiše sljedeći skup osnovnih specifikacija [Chu07, Randrian12] koje se koriste za kreiranje Web baziranih senzorskih mreža na osnovu SWE okvira (Slika 1.2.4.1):

- Jezik za modelovanje senzora – *Sensor Modeling Language* (SensorML).
- Praćenje i zapažanje – *Observation and Measurement* (O&M).
- Servis za prikupljanje senzora – *Sensor Collection Service* (SCS).
- Servis za planiranje senzora – *Sensor Planning Service* (SPS).
- Servis za obavještavanje putem Web-a – *Web Notification Service* (WNS).
- Jezik za modelovanje formata podataka – *Transducer Model Language* (TransducerML).
- Servis za opažanje senzora – *Sensor Observation Service* (SOS).
- Servis za objavu podataka senzora – *Sensor Alert Service* (SAS).

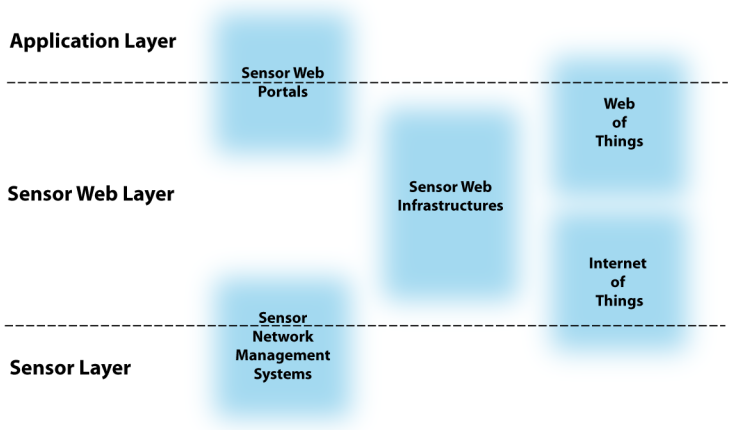
U sklopu [Di07] Di uvodi novu paradigmu u kojoj SW posmatra u odnosu na SOA i Web servise i definiše SW kao *grupu interoperabilnih Web serivsa u kojoj je specificiran određeni skup interfejsa i ponašanja*. Vodeći se novim konceptima, on tvrdi da sve odlike Web servisa mogu biti primjenjene i na SW elemente. Oslanjajući se na predloženi pristup,

Chu i Buyya [Chu07] definišu *Open Sensor Web Architecture* (OSWA) kao proširenje SWE standarda, dodajući servisno orjentisan pristup senzorskim elementima.



Slika 1.2.4.1. Prikaz tipične saradnje u SWE okviru [Chu07]

U radu [Bröring11] autori SW razmatraju kao posrednik (*middleware*) između fizičkih senzora i aplikacija, pa na osnovu toga definišu tri glavna arhitekturna sloja sa kojim predstavljaju odnose između konceptualno sličnih tehnologija (Slika 1.2.4.2):



Slika 1.2.4.2. Senzor Web sloj i locirane klase posredničkog sloja [Bröring11]

- *Senzorski sloj* – predstavlja sloj u kojem se nalaze fizički senzori koji podržavaju različite protokole za komunikaciju.
- *Senzor Web* sloj – predstavlja sloj koji povezuje fizičke senzore i aplikativni sloj.
- *Aplikativni sloj* – predstavlja sloj koji obezbjeđuje interfejse i servise za direktnu komunikaciju sa klijentima (korisnicima ili drugim računarskim sistemima).

1.2.5. Distribuirane servisne arhitekture

Distribuirane servisne arhitekture uveliko odgovaraju paradigmi *Servis Orijentisanih Arhitektura – Service Oriented Architecture* (SOA), mada je tačnu definiciju koja ih opisuje veoma teško pronaći jer postoji veliki broj različitih definicija što ukazuje na ontološki problem. U sklopu [Nicolai07] utvrđeno je da je svim definicijama zajedničko to da SOA predstavlja *paradigmu* koja poboljšava fleksibilnost. SOA se ne može posmatrati kao konkretna arhitektura već kao skup pravila koja "vode" ka konkretnoj arhitekturi. Ona predstavlja stil, paradigmu, koncepte, perspektive, filozofiju ili reprezentaciju na osnovu koje je moguće realizovati konkretnu arhitekturu.

Osnovna uloga SOA je da osigura povezivanje davaoca (servera ili servisa) i korisnika servisa. Da bi navedeni cilj bio postignut, SOA treba da riješi ključni problem komunikacije između heterogenih elemenata složenih sistema [Nicolai07]. Implementacija SOA arhitekture je obično zasnovana na Web servisima gdje se distribuisani sistem implementira kao skup javnih procedura, postavljenih na udaljene servere, koje je, najčešće putem HTTP protokola, moguće koristiti.

Kao alternativa SOA arhitekturi, uvodeći "*resurs*" kao osnovnu jedinicu komunikacije između servisa, pojavljuje se *Resurs Orijentisana Arhitektura – Resource Oriented Architecture* (ROA). ROA implementira Web servise kao skup resursa (entiteta) kojima se može pristupati, za razliku od SOA koja ih tretira kao procedure.

Osnovni principi ROA arhitekture su:

- svaka informacija predstavlja resurs,
- informacije nisu samo statičke stranice, nego i datoteke koji sadrže informacije, tabele u bazi podataka ili bilo koji drugi podaci kojima treba pristupiti,

- resursi predstavljaju stvarne informacije kojima se može pristupiti u sistemu – Svaki resurs ima jedinstvani *Uniform Resource Locator* (URL) (ili URI) koji ga identifikuje i pomoću kog mu se može pristupiti. Način pristupa resursima se definiše pomoću HTTP protokola,
- servisi nemaju stanje – svaki poziv servisa predstavlja jednu zaokruženu akciju koja ili čita ili mijenja podatke.

Postoji više različitih koncepta Web servisa koji se razlikuju po načinu realizacije i zahtjevima prema obradi podataka, od čega zavisi izbor koji je optimalan za razmatrani domen problema [Kalin09]. Tradicionalna poslovna rješenja koriste *Simple Object Access Protocol-u* (SOAP) koji se za predstavljanje podataka oslanja na *EXtensible Markup Language* (XML). U novije vrijeme akcenat se prebacuje na HTTP bazirane usluge, kao što su REST i Web servisi (*RESful Web servisi*)

1.2.5.1. Simple Object Access Protocol (SOAP)

SOAP se pojavljuje 1998. godine, ali postaje aktuelan tek nakon preuzimanja od strane *World Wide Web Consortium*-a (W3C) 1999. godine. Oslanjajući se na XML, SOAP definiše pravila za strukturiranje podataka koji trebaju biti razmjenjeni između učesnika [Tidwell01]. Zbog svoje velike rasprostranjenosti često se poistovjećuje sa SOA, tako da se danas SOAP može smatrati i *Service Oriented Application Protocol*-om. Do ekspanzije SOAP-a dovode njegove sljedeće karakteristike [Xmlpsd]:

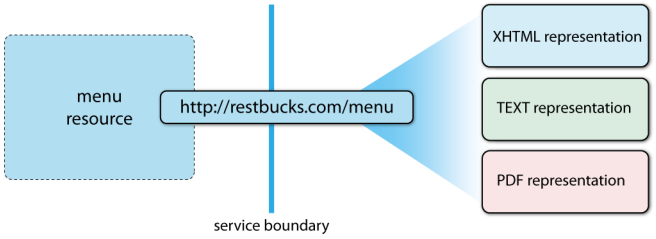
- ***jednostavnost*** – fokus se stavlja na pozivanje jednostavnih procedura između klijenta i servera. Današnje platforme i programski jezici imaju ugrađenu podršku za implementaciju SOAP-a.
- ***sveprisutnost*** (raširenost) – oslanja se na HTTP koji *de facto* predstavlja univerzalni protokol za komunikaciju putem Interneta.
- ***pogodnosti za zaštitne barijere (firewall)*** – SOAP koristi port 80 za komunikaciju i time obezbjeđuje globalnu mogućnost pristupa.
- ***sigurnost*** – kao osnovno sredstvo komunikacije SOAP koristi XML koji je sastavljen od tekstualnih podataka. Svi postojeći mehanizmi zaštite Web-a mogu biti primjenjeni i na SOAP (*Hypertext Transfer Protocol Secure* (HTTPS) ili *Secure Sockets Layer* (SSL)).
- ***proširivost*** – proširenjem XML-a moguće je obuhvatiti sve potrebe korisnika.

Glavni nedostatak SOAP-a jeste upravo ono što ga čini jakim – XML pri razmjeni podataka između korisnika. XML struktura i podaci koji se

razmjenjuju mogu biti veoma kompleksni i robusni, čime se otežava sâmo parsiranje i prikupljanje informacija iz XML-a. Ovo je pogotovo izraženo kod sistema sa ograničenim resursima kod kojih primarni cilj predstavlja brza obrada podataka i ispunjavanje postavljenih zahtjeva. Upravo iz tog razloga, pojavljuju se nove tehnologije koje pružaju značajan doprinos u ovoj oblasti.

1.2.5.2. Representational State Transfer (REST)

Pojam REST servisa se prvi put pojavljuje 2000. godine u doktorskoj disertaciji Roy Fieldinga [Fielding00]. Oslanjajući se na HTTP protokol i resurs kao osnovni gradivni blok Web servisa, Fielding generalizuje principe Web arhitekture i prikazuje ih kao skup ograničenja za kreiranje Web servisa, odnosno kao arhitektonski stil. Primjena ovakvog pristupa zahtjeva jedinstvenu identifikaciju resursa na mreži i standardizaciju načina rukovanja resursima (kreiranje, brisanje, mijenjanje stanja i sl.). Za ovu namjenu Web se proširuje sa uniformnim identifikatorom resursa *Uniform Resource Identifier* (URI) koji, uz oslonac na HTTP protokol, omogućava jedinstveno identifikovanje resursa [Webber10]. Vodeći se ovim principom, da bi bio adresibilan na mreži, svaki resur mora imati najmanje jedan identifikator, a svaki od njih je povezan sa jednim ili više načina prikaza - reprezentacijom (različiti formati kao što su: *EXtensible HyperText Markup Language* (XHTML), XML, *JavaScript Object Notation* (JSON), tekst ili slike). Za razliku od početnih konvencija REST-a, koje su eksplicitno zahtijevale različite URI-je za svaki različit prikaz resursa, danas se koristi sofisticiraniji pristup, odnosno kontekstni protokol HTTP zaglavlja koji obezbjeđuje informacije o očekivanom tipu resursa (Slika 1.2.5.2.1), dok se za sam pristup resursima koriste HTTP metode: GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE, CONNECT i PATCH [Richardson10].



Slika 1.2.5.2.1. Višestruki pikaz resursa sa jednim URI identifikatorom [Richardson10]

Osnovne operacije za rukovanje resursima obuhvataju: kreiranje, čitanje, promjenu i brisanje – *Create-Read-Update-Delete* (CRUD) i direktno se preslikavaju na HTTP metode (Tabela 1.2.5.2.1.).

Tabela 1.2.5.2.1. Preslikavanja između CRUD operacija i HTTP metoda

CRUD Operacija	HTTP metoda	URI	Opis
`Create`	`POST`	`/narudzba`	Kreiranje nove narudžbe
`Read`	`GET`	`/narudzba/{Id}`	Zahtjev o podacima o narudžbi sa određenim ID brojem
`Update`	`PUT`	`/narudzba/{Id}`	Promjena podataka narudžbe sa određenim ID brojem
`Delete`	`DELETE`	`/narudzba/{Id}`	Brisanje narudžbe pod određenim ID brojem

1.2.6. Model-Driven Software Development (MDSD)

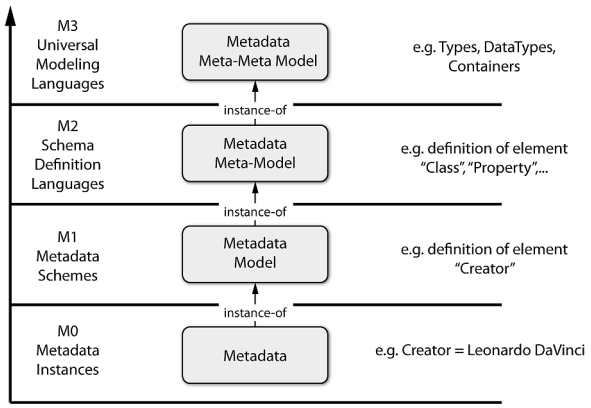
U opštem slučaju model se može definisati kao: apstrakcija (pojednostavljen prikaz) stvarnog svijeta, formalan opis stvarnog svijeta ili pogled na svijet sa nekog gledišta (aspekta). Modeli predstavljaju sredstvo za borbu sa kompleksnošću kroz fokusiranje na esencijalne karakteristike modelovanog sistema realnog svijeta uz maksimalno očuvanje njegovih funkcija, strukture i ponašanja. Odgovori koje nudi model moraju biti isti kao oni koje nudi realan sistem pod uslovom da se pitanja nalaze u obuhvatu modela.

Oslanjajući se na modele koji opisuju dijelove funkcionalnosti, strukture i/ili ponašanje sistema, *Object Management Gropu* (OMG) 2003. godine uvodi novu konkretizaciju *Model Driven Engineering* (MDE) paradigme u razvoj softvera, odnosno *Arhitekturu Upravljanu Modelima – Model Driven Architecture* (MDA). Za razliku od tradicionalnog razvoja softvera, koji kroz svoj životni ciklus susreće probleme kao što su: ponovno korištenje dijelova kôda, prenosivost, interoperabilnost, održavanje i dokumentovanje, MDA definiše okvire i principe za razvoj softvera na visokom nivou apstrakcije korištenjem formalnih metoda pri čemu rješava sve navedene probleme tradicionalnog pristupa [Kleppe03, Gianni15].

Prema pomenutoj paradigmi, softverski sistemi inicijalno se određuju pomoću modela na visokom nivou apstrakcije (metamodela), koji se koriste za generisanje drugih modela na nižem nivou apstrakcije, sve dok se ne dođe do profinjenog modela koji može biti izvršan. MDA pristup razvoja softvera je baziran upravo na ovom principu, odnosno na principu izgradnje softvera određivanjem skupa transformacija modela sa višeg na

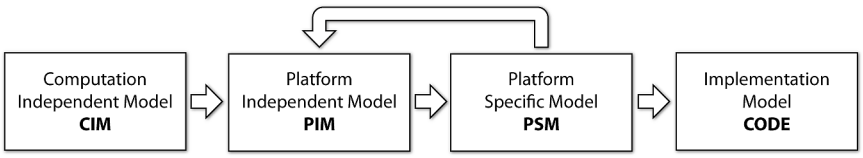
niži nivo apstrakcije. Za postizanje ovih ciljeva, MDA se oslanja na standarde [Gianni15]:

- *Meta Object Facility* (MOF) (Slika 1.2.6.1): za određivanje tehnološki neutralnih metamodela (modeli koji opisuju druge modele),
- *XML Metadata Interchange* (XMI): za serijalizaciju MOF metamodela/modela u XML bazirane šeme/dokumente,
- *Query/View/Transformation* (QVT): za određivanje transformacija modela.



Slika 1.2.6.1. MOF arhitektura [Gianni15]

MDA pristup se oslanja na tri stanovišta (*viewpoint*) sistema, odnosno na: stanovište nezavisno od načina obrade (*computation independent viewpoint*), stanovište nezavisno od implementacione platforme (*platform independent viewpoint*) i stanovište zavisno od implementacione platforme (*platform specific viewpoint*); koji obezbjeđuju fokus na specifične dijelove problema u zavisnosti od nivoa apstrakcije. Navedena stanovišta su najčešće predstavljena modelima koji sačinjavaju osnovne nivoe apstrakcije u MDA (Slika 1.2.6.2) [Dejanović11, Miller03, Kardoš10]:



Slika 1.2.6.2. Osnovni nivo apstrakcije u MDA

Računski nezavisan model (Computation Independent Model (CIM)) – predstavlja model najvišeg nivoa apstrakcije koji definiše procese nezavisno od njihove krajnje implementacije, a ponekad se naziva i domenskim modelom jer se za njegovu specifikaciju koriste opisi i riječnici sa kojima su upoznati domenski eksperti. Polazi se od pretpostavke da korisnici CIM modela, odnosno domenski ekspert, ne posjeduje znanje o modelima i artefaktima koji su korišteni za implementaciju funkcionalnosti predstavljene CIM modelom, pa tako CIM igra važnu ulogu u komunikaciji između domenskih eksperata koji definišu zahtjeve sistema i onih koji ih dizajniraju i konstruišu.

Platformski nezavisan model (Platform Independent Model (PIM)) – predstavlja model sistema koji je nezavisan od implementacione platforme, pa je kao takav pogodan za upotrebu na različitim platformama sličnog tipa. PIM opisuje domenski problem, skrivajući detalje primjene konkretne tehnologije, pa je kao takav pogodan za kreiranje specifikacije sistema bez tehničkih detalja zavisnih o platformi. Često se postavlja pitanje da li je posmatrani model platformski nezavisan ili ne, pa je prilikom definisanja nezavisnosti potrebno uvesti i definiciju konteksta u kojem taj iskaz vrijedi. Uobičajena tehnika za postizanje platformske nezavisnosti jeste primjena tehnološki neutralnih virtuelnih mašina koje obezbjeđuju koncepte i servise potrebne za modelovanje PIM-a. Virtuelna mašina jeste platforma pa je model zavisan od te platforme ali je model nezavisan u kontekstu različitih platformi na kojima je vitruelna mašina implementirana.

Platformski zavisan model (Platform Specific Model (PSM)) – predstavlja model koji koristi koncepte konkretne implementacione platforme. Kombinacijom specifikacija dobijenih iz PIM-a i detaljima koji određuju kako se oni implementiraju na određenoj vrsti platforme, kreira se PSM model. PSM model se u većini slučajeva ne kreira manuelno već se dobija automatskom ili poluautomatskom transformacijom PIM modela i najčešće predstavlja implementacioni model – kôd koji definiše sve potrebne informacije za kreiranje sistema. Nakon transformacije, PSM može postati PIM za narednu transformaciju u PSM koji će se naknadno transformisati u kôd.

Pod modelovanjem se podrazumijeva postupak izrade modela [Gianni15]. Neke od ključnih apstrakcija, na koje se oslanja proces modelovanja, mogu se svrstati u jednu od predloženih kategorija [Blackburn08]:

- struktura – sistemi, podsistemi, komponente, moduli, klase i interfejsi (ulazni i izlazni),
- ponašanje (funkcionalnost),
- vrijeme (konkurencija, interakcija),
- resursi (sredina),
- metamodeli (modeli modela).

U zavisnosti od cilja modelovanja, modeli se mogu podijeliti u dvije osnovne skupine [Brambilla12]:

- deskriptivne – ukoliko modeluju realne sisteme, ili
- preskriptivne – ukoliko predstavljaju "plan" sistema koji treba da se izgradi.

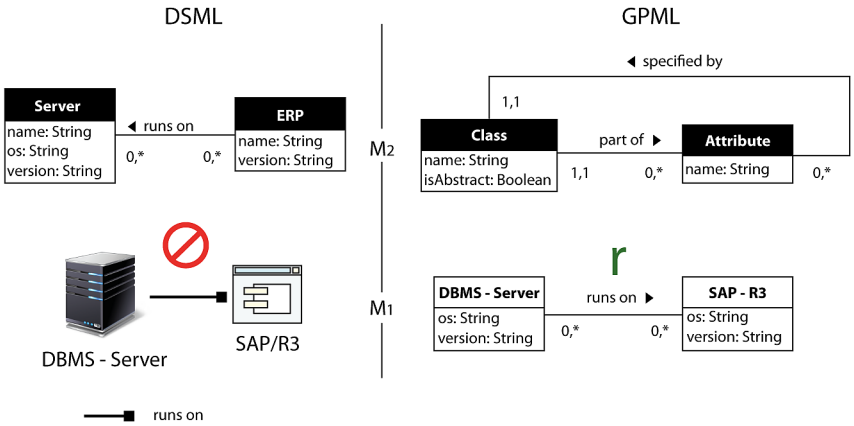
Modelom upravljan razvoj softvera koristi model kao primarni artefakat procesa razvoja softvera, pa umjesto izvornog kôda nastaju izvorni modeli koji podižu nivo apstrakcije i sakrivaju unutrašnju kompleksnost modelovanog sistema [Kelly08]. Projektanti generalno prave suštinsku razliku između modelovanja i kodiranja, pa tako modele najčešće koriste za dizajniranje sistema, njihovo bolje razumijevanje, kreiranje funkcionalne specifikacije i dokumentacije, dok kodiranje služi za implementaciju predloženog dizajna, kao i za pronalaženje grešaka, testiranje te održavanje softvera. Iako ovakva primjena modela, najčešće, ne prati životni ciklus razvoja softvera, radi velike cijene održavanja i sinhronizacije modela i kôda, veliki broj projektanata ipak kreira modele zbog prednosti koje pruža podizanja nivoa apstrakcije [Kelly08].

1.2.6.1. Domenski specifično modelovanje (DSM)

Pored *jezika za modelovanje opšte namjene – General Purpose Modelling Language* (GPML), koji su već duže vrijeme prisutni u metodološkim okvirima za razvoj softvera, u posljednjoj deceniji sve veći značaj dobijaju *namjenski jezici* za modelovanje ili *domenski specifični jezici za modelovanje – Domain-Specific Modeling Language* (DSML) (Slika 1.2.6.1.1) [Ulrich10].

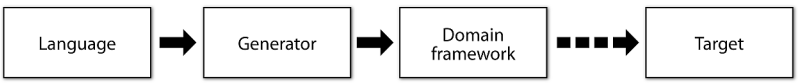
Domenski specifično modelovanje – Domain-Specific Modeling (DSM) se pojavljuje kao rješenje problema sinhronizacije između koncepata domena problema i domena rješenja, pri čemu podiže nivo apstrakcije iznad postojećih programskih jezika specificirajući rješenje upotrebom koncepata domena problema, dok se finalni proizvod dobija generisanjem iz kreirane specifikacije. DSM metodologija definiše dva osnovna cilja:

- podizanje nivoa apstrakcije iznad implementacije, kreirajnjem specifikacije rješenja na jeziku koji direktno koristi koncepte i pravila domena problema, i
- generisanje rješenja na nekom od programskih jezika ili drugoj formi, na osnovu apstraktne specifikacije rješenja.



Slika 1.2.6.1.1. Ilustracija poređenja DSML-a i GPML-a [Ulrich10]

DSM omogućava projektantima da se fokusiraju na rješenje problema umjesto na njegovu tehničku implementaciju, koja se najčešće dobija direktnom transformacijom iz modela, osiguravajući tako povećanje produktivnosti (rast produktivnosti od 3 do 10 puta), smanjenje troškova testiranja, razvoja, obuke i održavanja, kao i značajnije neposredno uključivanje krajnjih korisnika u proces razvoja softvera [Hulshout07, Kelly00, Wegeler13]. Da bi se sve pomenute koristi DSM pristupa mogle ostvariti, potrebno je automatizovati transformaciju modela u izvršni kôd (kompajler modela), za čije ispunjenje je predložena tronivovska arhitektura [Kelly08] (Slika 1.2.6.1.2).



Slika 1.2.6.1.2. Osnovna arhitektura DSM-a [Kelly08]

Pojedinačni nivoi arhitetkure obuhvataju [Kelly08]:

- *Domenski specifičan jezik* – pruža mehanizam apstrakcije za rješavanje kompleksnosti u kontekstu domena. Apstrakcija je ostvarena definišući koncepte i pravila unutar jezika koji

predstavljaju stvari u kontekstnom domenu, umjesto koncepta u određenom programskom jeziku.

- *Generator* – određuje kako se informacija ekstrahuje iz modela i transformiše u kôd. U najjednostavnijem slučaju, svaki modelovani simbol proizvodi određeni fiksni kôd, uključujući i vrijednosti koje su unesene u simbol kao argumenti. Generator može generisati različit kôd ovisno o vrijednostima dodjeljenih simbolu, odnosima simbola sa drugim simbolima ili drugim informacijama prisutnim u modelu. Ovaj kôd će biti povezan s okvirom i kompajliran u finalnu izvršnu verziju. Prilikom kreiranja DSM rješenja cilj je da se eliminiše dodatni manuelni rad na izmjenama ili proširenjima generisanog kôda nakon generisanja.
- *Domenski okvir* – obezbjeđuje interfejs između generisanog kôda i platforme. Drugim riječima, domenski okvir je često kreiran kako bi generisanje kôda bilo lakše jer obezbjeđuje dodatni sloj između generisanog kôda i postojećeg kôda platforme.

U radovima [Kelly08 Hulshout07, Ulrich10, Luoma04] predstavljeni su principi i metode ali i suštinski problemi na koje se nailazi prilikom primjene DSM metodologije. Kelly i Pohjonen [Kelly09] posebno ističu najčešće greške koje mogu nastati prilikom kreiranja novog DSM rješenja.

Primjena DSM metodologije za razvoj softvera, iako ne predstavlja jednostavan zadatak, nalazi široku primjenu u razvoju: interaktivnih sistema, sistema za rad u realnom vremenu, Web 2.0 dizajna, poslovnih arhitektura, modernih računarskih igara, industrijske automatizacije, hardvera i softvera, mehatronike, poslovnih procesa, dizajna procesa, itd., i predstavlja jedan od ključnih pravaca u MDA pristupu.

1.2.6.2. Domenski specifični jezici za modelovanje

Jedan od osnovnih razloga propadanja softverskih projekata jeste nedostatak komunikacije između poslovnih korisnika (koji poznaju domen problema) i projektanata softvera (koji poznaju domen rješenja). Nedostatak komunikacije je najčešće prouzrokovan suštinskim nerazumijevanjem terminologije domena problema koja je projektantima softvera najčešće strana i/ili nerazumljiva. Sa druge strane, implementacija softverskog rješenja se najčešće obavlja korištenjem nekog od programskih jezika visokog nivoa, koja, u određenim slučajevima može biti veoma robusna, kompleksna i u većini slučajeva nepoznata za naručioca. Kompleksnost najčešće predstavlja posljedicu razlika koje postoje između domena problema (koji opisuje procese, entitete i

ograničenja a dijelom su poslovne logike posmatranog sistema) i domena rješenja (koji predstavlja alate, tehničke detalje i implementaciju) [Ghosh11, Ghosh11a].

Za implementaciju rješenja, uglavnom se koriste dvije vrste jezika [Oliveira09]:

- *jezici opšte namjene – General-purpose Programming Languages* (GPL) – koji su pogodni za rješavanje velikog broja problema, bez obzira kojem domenu pripadaju, i
- *domenski specifični jezici – Domain-Specific Languages* (DSL) – koji su pogodni za rješavanje problema usko vezanih uz jedan domen.

Analizom prednosti i mana [Oliveira09, Kosar10], navedenih jezika, pri rješavanju pomenutog jaza između domena problema i domena rješenja, može se doći do zaključka da DSL premoščuje semantičke razlike između poslovnih korisnika i projektanata, podstičući bolju saradnju kroz zajednički riječnik. Kako DSL definiše odgovarajuću sintaksu i semantiku domena problema, pruža mogućnost korisnicima da postanu dio razvojnog tima i verifikuju poslovna (domenska) pravila tokom čitavog životnog ciklusa projekta [Ghosh11, Ghosh11a, Fowler10].

Fowler definiše DSL kao [Fowler10]: *programski jezik ograničene ekspresivnosti koji je fokusiran na određeni domen*.

Postoje četiri ključna elementa u Fowler-ovoj definiciji [Fowler10]:

- *Programski jezik* – DSL se koristi od strane ljudi za definisanje računarskih instrukcija. Kao takav, DSL mora biti poput modernih programskih jezika – jednostavan za razumijevanje ali izvršiv na računaru.
- *Priroda jezika* – DSL je programski jezik i kao takav mora imati osjećaj lakoće gdje ekspresivnost dolazi ne samo iz pojedinih izraza, već i iz načina na koji izrazi mogu biti kombinovani.
- *Ograničena ekspresivnost* – GPL jezici nude puno mogućnosti: podržavaju različite podatke, kontrole i apstraktne strukture, ali su samim tim i znatno teži za učenje. Za razliku od GPL jezika, DSL podržava minimalan broj funkcija potreban za opis domena problema.
- *Fokusiranje na domen* – ograničeni jezik je koristan samo ako ima jasan fokus na uzak domen i to je upravo ono što ga čini vrijednim.

Primjenom DSL-a se ostvaruju značajne prednosti, pa je pitanje koje se nameće: zašto uvijek ne primjenjivati DSL-ove? Sam razvoj DSL-a je

skup i može znatno povećati troškove razvoja softvera. Po Fowleru [Fowler10] jedini razlog zbog kojeg ne bi trebalo koristiti DSL jeste ako ne postoje nikakve koristi nakon primjene DSL-a ili ako su troškovi razvoja DSL-a znatno veći od troškova razvoja finalnog sistema. U radovima [Fowler10, Oliveira09, Ghosh11a, Mernik05] su navedene analize, prednosti i mane DSL pristupa, odnosno razlozi za korištenje DSL-a u procesu razvoja softvera.

1.2.7. Okviri za kreiranje grafičkih alata

Koristeći grafičke modele za predstavljanje međusobno zavisnih elemenata i alate za njihovu vizuelizaciju, olakšava se razumijevanje domena problema [Deshpande09] ali i sam proces modelovanja. Oslanjajući se na dvije ključne činjenice da [Kelly04]:

- ljudski mozak mnogo brže pamti i čita grafičke dijagrame nego tekst, hijerarhijski ili tabelarni prikaz,
- dijagrami imaju mogućnost prikaza višestrukih relacija između objekata za razliku od teksta ili tabela,

korištenje grafičke reprezentacije domenskih objekata predstavlja prirodno rješenje. Primjena grafičkih alata danas se može pronaći u velikom broju oblasti, pa tako i u procesu modelom upravljanog razvoja softvera. U MDSD metodologiji, *Unified Modeling Language* (UML) predstavlja najčešće korišten vizuelni jezik za specificiranje, konstrukciju i dokumentovanje artefakata sistema [Larman04]. Upravo zbog toga se na tržištu može pronaći širok spektar alata koji osiguravaju razvoj grafičkih modela baziranih na UML notaciji, a neki od njih su Power Designer (*Sybase*), Enterprise Architect (*Sparx Systems*), Rational Rose (*IBM*) i UML Designer (*Eclipse*). Za razliku od UML alata, koji najčešće ne dozvoljavaju prilagođenje jezika za modelovanje, u slučaju DSL-a, potrebno je pristupiti tehnologijama baziranim na metamodelima koje uklanjaju ovo ograničenje i omogućavaju fleksibilnost domenskih jezika za modelovanje [Kelly08].

U slučaju primjene EMF-a za kreiranje DSM alata, osigurava se samo djelimično rješenje koje podržava: smještanje podataka, prikaz svojstava objekata, hijerarhijski ili tabelarni prikaz objekata i modela, te okvir za generisanje Java kôda. Primjenom *Graphical Editing Frameworka* (GEF) [GEF] dobija se neophodno proširenje za izgradnju grafičkih editora baziranih na EMF oviru [Kelly04]. GEF *Eclipse* projekat predstavlja okvir za kreiranje kompleksnih grafičkih editora baziranih na modelima,

pružajući strukturu za smještanje, izgled i prikaz (renderovanje) grafičkih objekata [Buchmann07, Modica09, Bender14]. Sama struktura GEF-a je bazirana na *Model-View-Controller* (MVC) arhitektonskom šablonu, dok model koji predstavlja može biti bilo koji, između ostalog i EMF model [Kelly04, Moore04, Buchmann07, Modica09, Amyot06].

GEF *Eclipse* projekat se sastoji od tri gradivna elementa [Rubel12, Gronback09]:

- *Draw2D* [Draw2D] – predstavlja jednostavan okvir za iscrtavanje i prikaz informacija na *Standard Widget Toolkit* (SWT) kontejneru bez podrške za interakciju sa korisnikom,
- *Zest* [Zest] – zasnovan na *Draw2D* okviru, osigurava interfejse za povezivanje Java modela sa *Draw2D* dijagramima,
- *GEF* – zasnovan na *Draw2D* okviru, pruža veoma bogat API za kreiranje interaktivnih dijagrama naprednih mogućnosti, uključujući palete alatki, povuci-i-ispusti funkciju, poništavanje i ponovno izvršavanje akcija, štampanje i dr.

Koristeći GEF bazirane editore koji posjeduju veliki broj funkcionalnosti, moguće je vršiti jednostavne modifikacije modela, kao što je mijenjanje svojstava elemenata, ali i kompleksne operacije kao što su višestruke strukturalne promjene modela [Moore04].

U [Amyot06] je izvršena uporedna analiza komercijalnih alata i EMF+GEF okvira za kreiranje DSM rješenja i na osnovu postavljenih kriterija za evaluaciju, EMF+GEF je ocjenjen kao najbolji izbor. Međutim, iako se izabrana kombinacija pokazala kao najbolja, zbog svoje kompleksnosti i potrebe za značajnim manuelnim prilagođenjem i implementacijom, ona ne predstavlja optimalo rješenje. U [Modica09] je predstavljeno poboljšanje GEF okvira koje enkapsulira složenost EMF-a i GEF-a, ali tek sa pojavljivanjem novih okvira kao što su *Graphical Modeling Framework* (GMF) [GMP, GMF, Amyot06], *Graphiti* [Graphiti] i *Sirius* [Sirius], učinjen je ozbiljan pomak u pojednostavljenju razvoja grafičkih editora za DSM. Detalji vezani za pojedinačne okvire opisani su u sklopu Poglavlja 2.

Poglavlje 2

Pregled postojećih istraživanja

S obzirom na izuzetno veliki broj izvora i interdisciplinarnu prirodu oblasti kojoj pripada teza, izolovanje relevantnog skupa istraživanja predstavlja poseban izazov. U prethodnom poglavlju, prilikom elaboriranja pojmovnog okvira rada, dat je prikaz dijela referenci koje su značajne sa aspekta ontologije domena problema. Kako je osnovni cilj disertacije usmjeren na aspekte kreiranja alata za podršku razvoja IIS (u uslovima globalnog povezivanja sofisticiranih uređaja), u sklopu ovog poglavlja dat je prikaz najznačajnijih elemenata koji su direktno uticali na disertacijom predloženi metodološki pristup i iz njega izvedena konkretna rješenja.

Oslonac na dobro poznate principe savremenih informacionih i komunikacionih tehnologija dovodi do pojave potpuno novih metoda i tehnika projektovanja IIS čiju osnovnu infrastrukturu čine SW mreže.

U procesu projektovanja i izgradje kooperativnih sistema zasnovanih na definisanom konceptu, u literaturi i konkretnim implementacijama, elaborirani su brojni problemi i ograničenja. Projektovanje arhitekture SW mreža, koje se dijelom ili u potpunosti oslanjaju na dinamičke servise, predstavljaja kompleksan problem koji uključuje visok stepen interoperabilnosti heterogenih servisa različitog stepena granulacije. Njegovo rješavanje jako zavisi, kako od tehnologije, tako i od načina na koji se implementira SW infrasrtuktura.

Zbog toga, ključni dio analize je usmjeren na reference i publikovana istraživanja koja se bave različitim aspektima projektovanja arhitekture SW mreža. Poseban akcenat je stavljen na nove koncepte, kao što su IoT, WoT, kao i različite načine implementacije SW mreža.

U implementacionom smislu, osnovne pretpostavke na kojima se bazira disertacija podrazumjevaju da se, oslanjanjem na jednostavniju tehnologiju (RESTful servise) kao "kičmu" sistema i posmatrajnjem

elemenata senzorske mreže (senzorske čvorove) kao resursa, omogućava kreiranje održivih sistema koji zadovoljavaju većinu zahtjeva koji se postavljaju pred arhitekturu SW mreža.

Razvoj namjenskog jezika za modelovanje arhitekture senzorskih mreža i pratećeg alata za modelovanje i generisanje kôda predstavlja originalan istraživački doprinos disertacije. Zbog toga je uporedna analiza uglavnom usmjerena na aspekte koji su uticali na izbor tehnologija pogodnih za implementaciju razvojnog okruženja za projektovanje arhitekture SW mreža (Poglavlja 3., 4. i 5.). Analiza je obuhvatila najčešće korištene, slobodno dostupne alate i okvire, koji pružaju podršku za DSM pristup razvoju softvera.

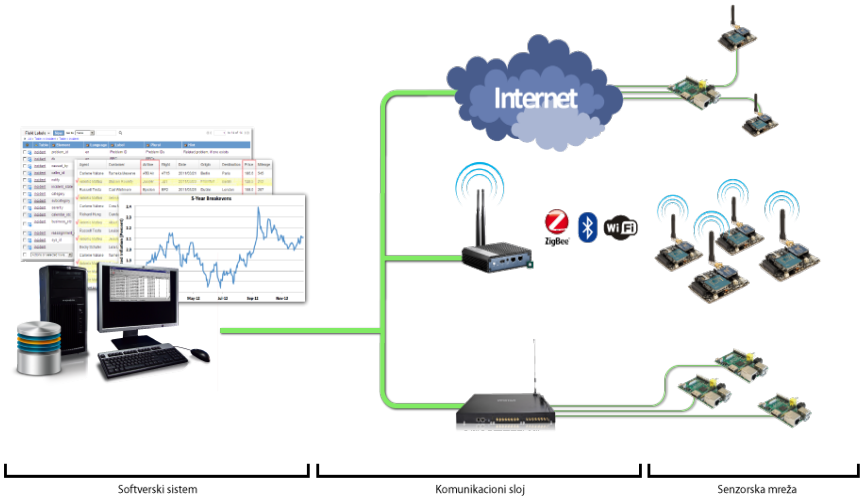
U sklopu Poglavlja 2 izvršeno je grupisanje postojećih istraživanja u paragrafe koji se odnose na:

- elemente arhitekture SW mreža ,
- servisno orijentisane senzorske mreže, i
- alate za projektovanje senzorskih mreža.

2.1. Elementi arhitekture SW mreža

Polazni skup analiziranih rezultata istraživanja, obuhvata pojam SW mreže, odnosno razvoj koncepta koji utiču na projektovanje njene arhitekture. Sa stanovišta arhitekture SW mreža moguće je izdvojiti tri osnovne cjeline koje predstavljaju gradivne elemente savremenih sistema koji sadrže softver (Slika 2.1):

- ***softverski sistem (SS).*** Najčešće predstavlja element koji je izložen korisnicima i koji nudi interfejse za pristup informacijama, skladištenje podataka, administraciju prava pristupa i dr. SS obezbjeđuje podršku semantičkim aspektima senzorske mreže u skladu sa profilom neposrednih korisnika.
- ***komunikacioni sloj (KS).*** Ima ulogu posrednika između SS i SW mreže. Finalna rješenja mogu znatno varirati u zavisnosti od korištene tehnologije i infrastrukture, posmatrano sa aspekta implementacije i kompleksnosti projektovanja.
- ***senzorska mreža*** (***SM***). Mrežna arhitektura koju čine senzorski, bežični senzorski ili SW čvorovi i mrežni posrednički uređaji (npr. izlazne tačkre – *gateway*).



Slika 2.1. Osnovna arhitektura SW mreža

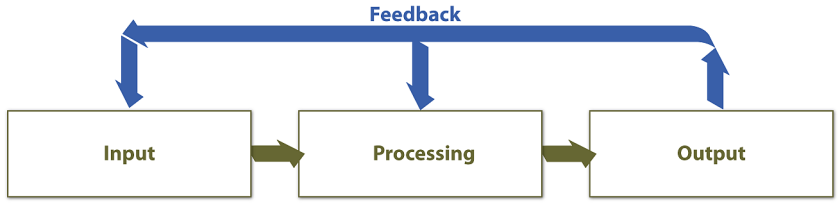
Projektovanje sistema baziranih na SW mrežama zahtijeva jednako tretiranje sva tri gradivna elementa, pri čemu, svaki od njih predstavlja vrlo kompleksan domen istraživanja za čije rješavanje je potreban studiozan pristup. Zbog toga je obezbjeđivanje adekvatnog ekspertskog znanja prilikom projektovanja i implementacije arhitekture SW mreže ogroman izazov, kako sa aspekta postojanja integrisanog ekspertskog znanja, tako i sa aspekta saradnje pojedinačnih domenskih eksperata. Navedene činjenice imaju znatan uticaj na ukupne troškove razvoja sistema baziranih na SW mrežama.

2.1.1. Softverski sistem (SS)

Osnovna uloga SS u sklopu arhitekture SW mreže predstavlja enkapsulaciju implementacionih detalja, kao i podršku semantici domena problema. Jedan od bitnih aspekata istraživanja u ovoj oblasti tiče se odnosa podataka/informacija i sadržanog znanja. Kod savremenih SS, primarni cilj više ne predstavlja prikupljanje podataka, niti sami podaci koji predstavljaju neobrađene činjenice, već način na koji oni smanjuju stepen nepoznavanja (entropije) posmatranog sistema i postaju informacije. Informacija predstavlja skup činjenica koje su organizovane i obrađene na smislen način, tako da u konkretnoj organizaciji mogu dobiti i prošireno značenje u odnosu na značenje pojedinačnih činjenica. Zbog toga što se informacije najčešće koriste za donošenje odluka na svim

nivoima odlučivanja, potrebno je obezbijediti mogućnost redefinisanja veza/odnosa između činjenica, pri čemu se kreiraju novi skupovi informacija. Za ovaj proces je neophodno znanje, odnosno svijest i razumijevanje niza informacija i način na koji one mogu biti korisne u određenim zadacima ili donošenju odluka. Dalje transformacije informacija u znanje predstavljaju izazov koji obezbjeđuje neposrednu podršku implementaciji sistema za podršku odlučivanju koji su zasnovani na predstavljanju i interpretaciji znanja.

U posljednjih nekoliko godina, evolucijom informaciono komunikacionih tehnologija (IKT), dolazi do prelaska iz predominantno "analognih" u "digitalna" okruženja, pri čemu nastaju značajne promjene u načinu komunikacije i poslovanja. Od savremenih SS se očekuje da obezbjede podršku za: analizu, obradu i odlučivanje. Oni su najčešće proaktivni i predstavljaju skup heterogenih, međusobno povezanih elemenata ili komponenti, koje koriste korektivne akcije za predviđanje budućih koraka u svrhu: izbjegavanja problema, uvođenja poboljšanja ili ispunjenja postavljenih ciljeva [Stair12] (Slika 2.1.1.1.):



Slika 2.1.1.1. Komponente informacionog sistema [Stair12]

SS danas gotovo bez izuzetka predstavljaju *računarski bazirane informacione sisteme – Computer-based information system* (CBIS) koji predstavljaju kolekciju: hardvera, softvera, baza podataka, telekomunikacione infrastrukture (mreža i Internet), korisnika i procedura za prikupljanje, manipulisanje, skladištenje i transformaciju podataka u informacije. Neizbježna heterogenost podataka, proizašla iz primjene računarskih sistema u procesu prikupljanja i obrade podataka, u procesu projektovanja otvara potrebu razmatranja sljedećeg skupa činjenica CBIS [Purvis99]:

- zahvaljujući automatizmu, prikupljena količina podataka je enormna,
- podaci se na mreži nalaze na različitim platformama, uključujući različite hardverske platforme i operativne sisteme,
- podaci se skladište na različite vrste medija i u različitim formatima,

- opisi podataka su organizovani prema različitim standardima i mogu imati različite interpretacije za isti skup podataka,
- podacima i informacijama je neophodno obezbijediti pristup sa različitih platformi, uključujući i Web pretraživače.

1998 godine, Mark Maier u svom radu [Maier98] uvodi osnovne smjernice koje razdvajaju velike monolitne sisteme od "*sistema-sistema" – System-of-Systems* (SoS):

- operativna nezavisnost elemenata: ako se SoS rastavi na sastavne komponente, iste moraju biti u stanju da funkcionišu nezavisno; SoS se sastoji od sistema koji su nezavisni i korisni sami po sebi,
- upravljačka nezavisnost elemenata: komponente sistema ne samo da rade nezavisno, već moraju da rade samostalno; Komponente sistema su odvojeno kreirane i integrisane, a njihovo održavanje je nazavisno od SoS,
- evolutivni razvoj: SoS se ne pojavljuje u potpunosti formiran. Njegov razvoj nastaje evolucijski: dodavanjem, uklanjanjem i modifikovanjem funkcionalnosti,
- primarno ponašanje: SoS ima jedinstvenu svrhu i obavlja funkcije koje ne postoje na komponentama sistema. Ovo ponašanje predstavlja okosnicu čitavog SoS i ne može biti lokalizovano na komponentama sistema.
- geografska distribucija: geografska rasprostranjenost komponenti sistema ne smije rezultirati ograničenjima u pogledu razmjene informacija.

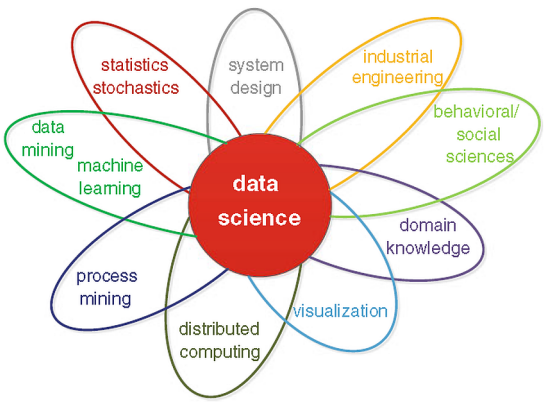
U skladu sa definisanim činjenicama i Mark Maier-ovim smjernicama, lako se može doći do zaključka da se CBIS sistemi, koji se baziraju na prikupljanju informacija sa senzorskih mreža, mogu posmatrati kao SoS [Chua11], ali ujedno i kao *sistemi izrazito velikih razmjera – Ultra-Large-Scale Systems* (ULSC) [Feiler06]. Najčešće se oslanjaju na mreže za praćenje pojava na određenom geografskom prostoru (geosenzorske mreže), čija veličina varira od okruženja jedne prostorije do cijelih regiona. Čvorovi mreže mogu biti statični ili mobilni, vezani za mobilne objekte ili čovjeka [Nittel08].

Geosenzorske mreže nalaze primjenu u različitim oblastima, kao što su monitoring životne sredine (posmatranje staništa, očuvanje okeana i obala), preciznoj poljoprivredi i ribarstvu, transportu, nadzoru i vojnoj upotrebi. U literaturi [Nittel08, Jolma15, Mukhop13] su navedene neke od primjena senzorskih mreža i CBIS sistema koji pružaju softversku podršku u analizi i donošenju odluka na osnovu prikupljenih podataka.

U radu [Aalst14] autori koriste pojam "*Podaci događaja*" – *Event data* i iznose tvrdnju da su "događaji", koji predstavljaju sastavni dio svih procesa i sistema koji nas okružuju, najvažniji izvori podataka. Oslanjajući se na definisani pojam, autori uvode i pojam "*Interneta događaja*" – *Internet of Events* (IoE), koji obuhvata sve poznate izvore događaja prisutne danas na Internetu:

- *Internet sadržaja – Internet of Content* (IoC) – predstavlja sve informacije koje je stvorio čovjek u svrhu povećanja znanja iz određenih oblasti. IoC uključuje tradicionalne Web stranice, enciklopedije, video zapise, elektronske knjige i dr.
- *Internet osoba/ljudi – Internet of People* (IoP) – predstavlja podatke koji se odnose na društvenu interakciju, a uključuje elektronsku poštu, društvene mreže, forume, itd.
- *Internet stvari – Internet of Things* (IoT) – predstavlja sve fizičke predmete/uređaje koji su spojeni na mrežu i posjeduju jedinstveni ID.
- *Internet lokacija – Internet of Locations* (IoL) – predstavlja sve geoprostorne podatke (geopozicione lokacije).

Zbog sve veće potrebe za analizom prikupljenih podataka, putem SM i/ili IoE, dolazi do pojave novih naučnih disciplina koje podatke koriste kao osnovnu podlogu za donošenje strateških odluka. "*Nauka o podacima" – Data Science* (Slika 2.1.1.2.) predstavlja novu disciplinu koja izlazi van okvira jednostavne analitike/statistike i uključuje razne aspekte bihevioralnih/društvenih i inženjerskih nauka.



Slika 2.1.1.2. Nauka o podacima [Aalst14]

Izazovi vezani za efikasno funkcionisanje CBIS u realnom vremenu i manipulacija sa velikom količinom podataka prikupljenih sa SM, koji su

potencijalno neprecizni i/ili nestrukturirani, otvara četiri osnovna pitanja koja je neophodno analizirati i koja ujedno predstavljaju moguće pravce daljnjeg istraživanja [Aggarwal13]:

- *prikupljanje i* "*pročišćavanje*" *podataka* – brojni problemi nastaju prilikom prikupljanja podataka sa senzora, koji su sami po sebi najčešće neizvjesni, sa šumovima, a ponekad i nekompletni ili redudantni. Da bi podaci bili spremni za korištenje, neophodno je izvršiti njihovo prilagođavanje i "*čišćenje*",
- *upravljanje podacima* – velike količine prikupljenih podataka predstavljaju značajan izazov za njihovo skladištenje. Ponekad je obim prikupljenih podataka toliko veliki, da je za njihovo skladištenje potrebno izvršiti redukciju ili kompresiju, što otvara novo pitanje: koji podaci će biti redukovani ili kompresovani?
- *dubinska analiza podataka (Data Mining) i obrada* – velike količine senzorskih podataka zahtijevaju primjenu efikasnih jedno-prolaznih (*one-pass*) algoritama za njihovu obradu. Ponekad se može primjeniti i obrada podataka na nivou mreže čime se smanjuje slanje podataka koji su irelevantni za naredne faze obrade,
- *specifične primjene* – podaci sa senzora se mogu pojaviti u različitim oblastima primjene: trgovini, vojnoj primjeni, astronomiji, zaštiti okoliša, itd., što može dovesti do različitih zahtjeva prema njihovom skladištenju i obradi.

2.1.2. Komunikacioni sloj (KS)

Komunikacija između CBIS i BSM (pojedinih senzorskih ili SW čvorova), predstavlja jedan od ključnih elemenata cjelokupnog sistema. Razvojem IKT-a nastaje znatan broj tehnologija i tehnika za bežični prenos podataka, što otvara pitanje: *koja od trenutno dostupnih tehnologija predstavlja optimalan izbor za implementaciju BSM i njeno povezivanje sa CBIS-om?* Kako danas postoji više različitih bežičnih tehnologija koje su u javnoj upotrebi, 1985. godine, federalna komisija za komunikacije – *Federal Communications Commission* (FCC) izdvaja poseban segment u radio spektru za potrebe ne-licencnog (*license-free*) prenosa. Ovi segmenti su danas najčešće korišteni spektri. Poznati su kao industrijski, naučni i medicinski – *Industrial, Scientific and Medical* (ISM) i nalaze se u tri opsega: 900MHz, 2.4GHz i 5.8GHz. Koji od ovih opsega će biti korišten zavisi od tehnologije kao i od namjene BSM.

U zavisnosti od udaljenosti i brzine pristupa, postojeće tehnologije se mogu svrstati u pet kategorija [Grzechca13, IoTIEC14, Tanenbaum11]:

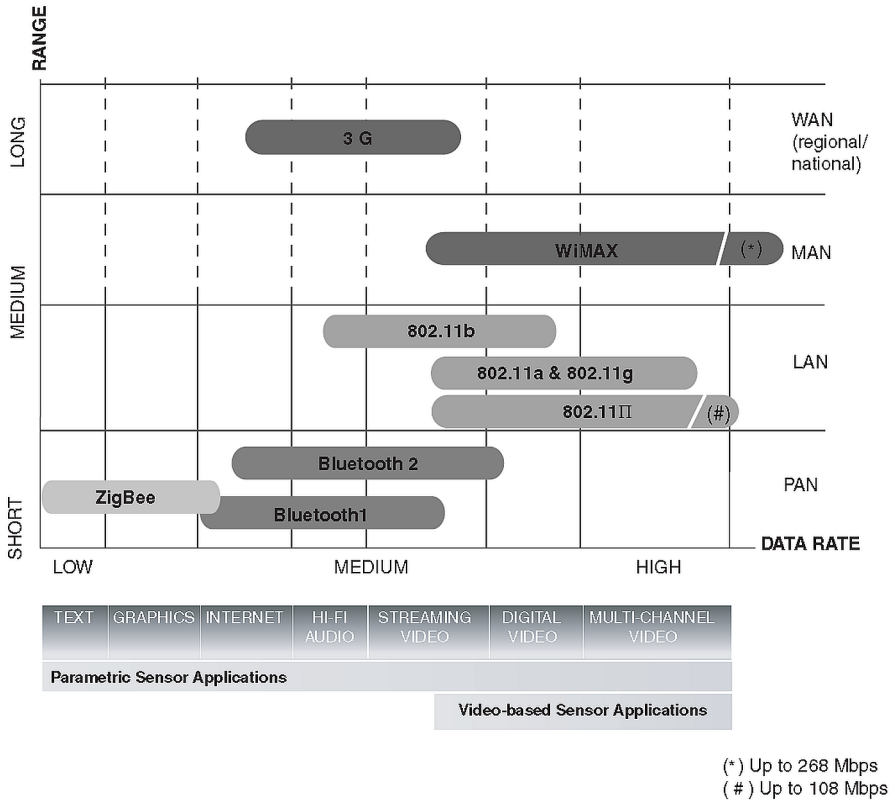
- bežične personalne mreže (mreže na nivou osobe) – *Wireless Body Area Network* (WBAN) – služe za monitoring i prikupljanje vitalnih parametara osobe,
- bežične mreže kratkog opsega – *Wireless Personal Area Network* (WPAN) – obezbjeđuju komunikaciju uređaja na kratkom opsegu. Obično služe za bežično povezivanje hardverskih komponenti, a najpoznatiji protokoli PAN-a su Bluetooth i RFID (IEEE 802.15.1),
- bežične lokalne mreže – *Wireless Local Area Network* (WLAN) – obezbjeđuju komunikaciju na ograničenom prostoru (zgrada, kuća, kancelarija, preduzeće, fabrika). Najčešće se koristi za povezivanje računara u jedinstvenu mrežu za razmjenu informacija i dijeljenje resursa. Također poznata i kao Wi-Fi mreža. (IEEE 802.11a/b/g/n).
- bežične gradske mreže – *Wireless Metropolitan Area Network* (WMAN) – mreže koje obezbjeđuju komunikaciju na nivou grada, npr. WiMAX za brzu bežičnu Internet konekciju (IEEE 802.16)
- bežične širokopojasne mreže – *Wireless Wide Area Network* (WWAN) – povezuju velike geografske oblasti (države ili kontinente). Najčešće su realizovane optičkim, mobilnim ili satelitskim prenosom podataka, a karakteriše ih velika brzina prenosa.

Na tržištu danas ne postoji jedna univerzalna bežična tehnologija koja zadovoljava sve potrebe industrije. U literaturi su prepoznata tri ključna faktora koji otežavaju njeno formulisanje [IoTIEC14]:

- *pouzdanost* – BSM obično rade u uskim opsezima frekventnog spektra u kojem nastaju šumovi i interferencija signala, što komunikaciju čini nepouzdanom,
- *sposobnost prenosa u realnom vremenu* – aplikacije koje zahtijevaju rad u realnom vremenu imaju strožije zahtjeve u pogledu prenosa podataka preko BSM. Malo kašnjenje u prenosu može dovesti do velikih (fatalnih) otkaza sistema.
- *energetska efikasnost* – niska potrošnja predstavlja ključ za nezavisne uređaje, pogotovo ako su oni teško dostupni. Na ovaj način se znatno smanjuju troškovi održavanja senzorske mreže.

Na osovu definisanih faktora, kreiran je širok spektar protokola koji se razlikuju po opsegu djelovanja, brzini prenosa podataka, latenciji signala, pouzdanosti i sigurnosti, a najčešće korišteni u oblasti industrije, nauke i medicine su [Grzechca13, Christin10, IoTIEC14, Sohraby07, Dickinson14,

Tanenbaum11]: IEEE 802.11a/b/g/n (Wi-Fi), IEEE 802.15.1 (Bluetooth), IEEE 802.15.4 (*Wireless Interface for Sensor and Actuators* - WISA), IEEE 802.16 (WiMax), Cellular (GPRS). Na slici (Slika 2.1.2.1.) je prikazan uporedni pregled najčešće korištenih protokola u zavisnosti od brzine prenosa podataka i opsega djelovanja.



Slika 2.1.2.1. Uporedni pregled najčešće korišćenih protokola [Sohraby07]

Bitno je pomenuti da pojavom novih koncepta kao što su IoT, WoT, SW, postoji tendencija za integracijom IP protokola u standardima i tehnologijama za komunikaciju u cilju obezbjeđenja globalne komunikacije zasnovane na dobro definisanim Web tehnologijama (SOA, SOAP, REST, HTTP, HTTPS, *Constrained Application Protocol* (CoAP) i sl.).

2.1.2.1. IEEE 802.11a/b/g/n

WLAN (Wi-Fi), koji je baziran na IEEE 802.11 standardu, danas predstavlja najrasprostranjeniju tehnologiju za bežičnu komunikaciju.

Zahvaljujući svojim karakteristikama, Wi-Fi ima široku primjenu u kreiranju bežičnih mreža unutar kuća, poslovnih objekata i kancelarija, definisanju javnih pristupnih tačaka (*Public Access Points*), sigurnosnim i kontrolnim sistemima, te kućnoj tehnici [Grzechca13, IoTIEC14]. Wi-Fi se može pronaći i u industrijskim primjenama (za povezivanje računara, programabilnih logičkih kontrolera, senzora, te drugih uređaja koji se koriste u sistemima kontrole) [IoTIEC14, Dickinson14].

Wi-Fi mreže se najčešće koriste na dva načina [Tanenbaum11], kao:

- pristupna tačka – *Access Point* (AP) – pruža mogućnost povezivanja više klijenata, kao što su računari i pametni telefoni, sa drugom mrežom (Intranetom ili Internetom). Ovo predstavlja dominantan način korištenja IEEE 802.11 tehnologije, te
- ad-hoc mreža – kreiraju se kolekcije vezanih uređaja, pa se razmjena podataka može vršiti direktno.

IEEE 802.11 standard definiše pod-standarde koji se razlikuju po tipovima modulacije i brzinama prenosa podataka, a prikazani su u tabeli (Tabela 2.1.2.1.1).

Tabela 2.1.2.1.1 Poređenje IEEE 802.11 standarda [Grzechca13, Dickinson14]

Type	Frequency [GHz]	Modulation type	Data transfer [Mbit/s]	Range (indoor/outdoor) [m]
802.11	2.4	IR/FHSS/DSSS	2	20/100
802.11a	5	OFDM	54	35/120
802.11b	2.4	DSSS	11	28/140
802.11g	2.4	OFDM, DSSS	54	38/140
802.11n	2.4 or 5	OFDN	600	70/250
802.11y	3.7	OFDN	54	-/5000

Glavni nedostatak Wi-Fi tehnologije jeste velika potrošnja energije, što značajno degradira performanse i vrijeme rada sistema koji se napajaju iz autonomnih izvora (baterija, solarni paneli i sl.) [Grzechca13].

2.1.2.2. IEEE 802.15.1

Bluetooth je baziran na IEEE 802.15.1 standardu, a predstavlja tehnologiju bežičnog prenosa male snage i kratkog dometa. Najčešće se koristi za povezivanje prenosnih i računarskih perifernih uređaja kao što su [Sohraby07, Dickinson14, Tanenbaum11]: mobilni telefoni, satovi, konzole za igru, tastatura, miš, printeri, uređaji za navigaciju, i dr. Međutim, Bluetooth tehnologija je našla primjenu i u industriji (bliska

komunikacija između uređaja, kao i zamjena za kablove na pokretnim dijelovima uređaja) [Dickinson14] te medicini [Grzechca13]. IEEE 802.15.1 standard definiše tri klase uređaja na osnovu snage prenosa i opsega djelovanja [Grzechca13, Dickinson14]:

- klasa 1 – 100mW, opseg do 100m,
- klasa 2 – 2,5mW, opseg do 10m,
- klasa 3 – 1mW, opseg do 1m,

međutim, u zavisnosti od postojanja optičke vidljivosti, moguće je ostvariti i domete veće od 100m.

Bluetooth tehnologija od svog nastanka pa do danas je implementirana kroz više verzija koje se razlikuju po brzini prenosa i potrošnji energije. Tako je Bluetooth verzija 1.2 dozvoljavala prenos podataka maksimalnom brzinom od 1Mb/s, dok je verzija 2 mogla doseći i do 3Mb/s [Sohraby07]. 2012 godine, Bluetooth dobija verziju 4 koja donosi znatna poboljšanja u pogledu potrošnje energije čime osigurava trajanje baterije i po nekoliko godina [IoTIEC14].

2.1.2.3. IEEE 802.15.4

IEEE 802.15.4 standard definiše ključne elemente za bežične poligonalne (*mesh)* mreže male snage, te kao takav predstavlja osnovu za druge standarde kao što su ZigBee, WirelessHart, WIA-PA i ISA.100.11a [Christin10, IoTIEC14, Dickinson14]. Karakteristika IEEE 802.15.4 mreža su veoma slične WPAN mrežama malih brzina, a dodatno ih odlikuje i mala potrošnja energije (višemjesečno ili višegodišnje trajanje baterija), mala brzina prenosa, rad na ne-licencnom spektru frekvencija (2.4GHz) i veoma mala kompleksnost [IoTIEC14, Sohraby07, Dickinson14]. IEEE 802.15.4 bazirane bežične mreže, mogu biti organizovane kao zvijezde, stabla ili *mesh* topologije [Christin10]. Ovakva organizacija pruža znatne prednosti pri smanjenju potrošnje (komunikacija se vrši samo između susjednih čvorova) i povećanju pouzdanosti i opsega mreže (postojanje višestrukih puteva kojima podaci mogu biti dostavljeni centralnom čvoru), ali i mana zbog povećane latencije (veliki broj čvorova dovodi do velikog broja "skokova") [Dickinson14]. 2004 godine, ZigBee Alliance predstavljaju prvu ZigBee specifikaciju koja je bazirana na IEEE 802.15.4 standardu, a koja danas predstavlja *de facto* vodeći standard za izgradnju energetski efikasnih i jeftinih bežičnih senzorskih mreža male brzine prenosa [Sohraby07]. ZigBee danas ima široku primjenu u daljinskom

nadzoru, kontroli stambenih prostora, automatizaciji zgrada, industrijskoj primjeni i dr.

2.1.2.4. IEEE 802.16

WiMAX (*Worldwide Interoperability for Microwave Access*) tehnologija predstavlja implementaciju standarda IEEE 802.16 [Tanenbaum11] koja proširuje bežični pristup WLAN (IEEE 802.11) mreža na opseg WMAN i WWAN [Bhandare08]. Tipična WiMAX mreža se sastoji od bazne stanice koja ima repetitor (antenski toranj ili antene koje su montirane na zgradama) i direktne veze sa zemaljskom mrežom (Internet, LAN, itd.) [Sohraby07]. IEEE 802.16 obezbjeđuje visoku mobilnost širokopojasnih usluga čak i pri brzinama većim od 120km/h, a u zavisnosti od konfiguracije antene i načina modulacije, brzina prenosa podataka se kreće do 75 Mb/s. Najčešće primjene WiMAX mreža su [Sohraby07]: *point-to-point* komunikacija između stanica, *point-to-multipoint* komunikacija između bazne stanice i klijenata, Wi-Fi (802.11) pristupne tačke, širokopojasni Internet pristup za korisnike, te povezivanje korisnika na udaljenim lokacijama.

2.1.2.5. Celularne mreže

Celularne mreže (poznate i kao Mobilne mreže) koriste se za prenos glasa i podataka na velikim geografskim prostorima čime omogućavaju globalno povezivanje [Dickinson14, Tanenbaum11]. Celularne mreže se sastoje od više primopredajnika koji se nazivaju "bazne stanice", a osiguravaju komunikaciju unutar određene oblasti u zavisnosti od frekventnog opsega (1800MHz pokriva površinu od 614km^2 sa radijusom ćelije od 14km, 950MHz pokriva površinu od 2200km^2 sa radijusom ćelije od 27km) [Grzechca13].

Dva osnovna standarda koji se primjenjuju u celularnim mrežama su: *Time-Division Multiple Access/Code-Division Multiple Access* (TDMA/ CDMA) razvijan od strane *Telecommunications Industry Association* (TIA) u sjevernoj Americi i *Global System for Mobile Communications* (GSM) koji je razvijan od strane *European Telecommunications Standards Institute* (ETSI) iz Evrope [Sohraby07]. Usljed razvoja tehnologija i povećanjem mobilnosti korisnika, ovi standardi prerastaju u *General Packet Radio Service* (GPRS), *Enhanced Data rates for GSM Evolution* poznat i kao *Enhanced GPRS* (EDGE), te *Universal Mobile Telecommunication System* (UMTS) [Grzechca13, Tanenbaum11] koji se

razlikuju po svojoj primjeni, brzini prenosa i rasprostranjenosti. UMTS, poznat još i kao 3G, danas predstavlja najpopularniji standard za mobilne mreže pošto se odlikuje velikom brzinom prenosa podataka (govornih i video poziva, te prenos podataka). Celularne mreže (2G, 2.5G, 3G, 4G) su našle primjenu u industriji za daljinski nadzor i upravljanje sistemima, prikupljanje i *Machine-to-Machine* komunikaciji, kao i u sistemima za supervizorsku kontrolu i prikupljanje podataka – *Supervisory Control and Data Acquisition* (SCADA) sistemima, koji često zahtjevaju komunikaciju sa geografski rasprostranjenim elementima [Dickinson14].

Osnovne prednosti celularnih mreža su: veoma široka oblast djelovanja i oslanjanje na IP protokol (što omogućava jednostavno povezivanje sa postojećim mrežama), dok se kao mane navode velika potrošnja energije i ograničena brzina prenosa [Grzechca13]. Važno je napomenuti da celularne mreže ne predstavljaju besplatan servis i da cijena često zavisi od količine prenesenih podataka [Grzechca13, Dickinson14], kao i da gubitak komunikacije može dovesti do trajnog gubitka informacija [Dickinson14].

2.1.3. Senzorske mreže (SM)

Nedavnim napretkom u oblasti mikro-elektro-mehaničke tehnologije, bežičnih komunikacija i digitalne elektronike, omogućen je razvoj jeftinih, multifunkcionalnih senzorskih čvorova male potrošnje i dimenzija, koji su sposobni za komunikaciju na malim udaljenostima [Akyildiz02a, Yick08, Gűrgen10, Othmana12]. Senzorski čvorovi su po svojoj arhitekturi slični računarima, pa tako imaju procesorsku jedinicu ograničene snage, memoriju, senzore, komunikacionu jedinicu i autonomni izvor napajanja (baterije). Opremljeni sa senzorskim elementima, senzorski čvorovi mogu da opažaju, mjere i prikupljaju podatke iz okruženja i po potrebi ih proslijede krajnjim korisnicima. Ponekad, umjesto slanja neobrađenih podataka, senzorski čvorovi koriste mogućnost obrade, pa se korisniku šalju djelimično obrađene informacije.

Senzorske mreže predstavljaju distribuirane mreže senzorskih čvorova, koji su gusto raspoređeni unutar ili blizu fenomena koji se posmatra, a prate okruženje ili sistem mjerenjem skupa fizičkih parametara od interesa. Pozicija senzorskih čvorova obično nije unaprijed određena, čime se pruža mogućnost nasumičnog raspoređivanja na nepristupačnim terenima ili područjima pogođenim katastrofama, ali i povećava kompleksnost njihove izrade [Akyildiz02a].

U zavisnosti od potreba sistema, senzori se klasifikuju na osnovu karakteristika kao što su: veličina, potrošnja, način rada, tipovi senzora i dr. Uključujući taksonomiju koja klasifikuje senzore na osnovu veličine, dobija se podjela na velike senzore (radari, sonari), mikrosenzore (male senzore), nanosenzore, senzore za čitanje oznaka, te druge senzore, dok se na osnovu potrošnje energije senzori dijele na pasivne i aktivne [Sohraby07]:

- *pasivni senzori* – predstavljaju uređaje male potrošnje koji mogu biti u obliku jednog elementa (temperatura, vlažnost, pritisak, i dr.) ili kao skup elemenata (optički i biohemijski),
- *aktivni senzori* – su uređaji velike potrošnje (radari i sonari).

Tipični parametri koji se mjere sa senzorima, mogu se podijeliti u dvije skupine [Sohraby07, Akyildiz02a]:

- *mjerenje fizičkih parametara* – temperatura (termistori), vlažnost, atmosferski pritisak, magla, prašina, zvuk, ubrzanje, brzina, pritisak, kretanje, svjetlost, UV zračenje, radijacija, radio i mikrotalasi, lokacija,
- *mjerenje hemijskih i bioloških parametara* – mjerenje prisustva ili koncentracije određenih supstanci u analiziranom uzorku (oblasti).

Za razliku od senzora, senzorski čvorovi se uglavnom razlikuju po tome u koji tip senzorskih mreža se ugrađuju, pa tako mogu biti namjenjeni za [Stojčev05]:

- *proaktivne mreže*: čvorovi u mreži periodično prelaze u aktivno stanje, izvrše mjerenje okruženja/pojave i proslijede informaciju korisnicima, te
- *reaktivne mreže*: čvorovi su sve vrijeme aktivni i trenutno reaguju na nagle promjene u mreži. Ovi tipovi čvorova su pogodni za sisteme koji rade u realnom vremenu,

dok same senzorske mreže mogu biti orijentisane na [Stojčev05]:

- *podatke* (*data-centric*) – upiti se izvršavaju nad regionima koje čine topološki uređene grupe senzora,
- *adrese* (*address-centric*) – upit se izvršava na tačno određenom čvoru sa definisanom adresom.

U radovima [Othmana12, Gűrgen10, Sohraby07, Akyildiz02a] su navedene primjene senzorskih mreža u različitim oblastima ljudskog djelovanja. Kako svaka oblast ima karakteristične elemente i zahtjeve, dolazi do razvoja velikog broja specifičnih hardverskih i softverskih platformi, što rezultira velikim brojem heterogenih rješenja. Sveprisutnost

BSM tako nameće brojne izazove i prepreke koje moraju biti detaljno razmotrene i prevaziđene u procesu implementacije. U sklopu [Sohraby07, Akyildiz02a] obrađene su:

- *ograničena funkcionalnost* – redukovana procesna moć senzorskih čvorova umanjuje mogućnost primjene kompleksnih algoritama za procesiranje prikupljenih podataka,
- *veličina senzorskih čvorova* – u zavisnosti od polja primjene, potrebna je minijaturizacija senzorskih čvorova, što može predstavljati problem u procesu implementacije,
- *efikasno napajanje* – limitirani izvor napajanja zahtijeva posebne hardverske i softverske elemente koji bi optimizovali potrošnju,
- *cijena* – tehnologija izrade i implementacije senzorskih čvorova može znatno povećati cijenu čitavog sistema,
- *radna okolina čvorova* – toksična ili korozivna okruženja znatno umanjuju vijek trajanja senzorskih čvorova,
- *komunikacija i prenos podataka* – potrebni adekvatni algoritmi za razmjenu i prenos podataka koji značajo umanjuju i optimizuju potrošnju energije,
- *kompleksnost topologije mreže* – senzorski čvorovi moraju imati mogućnost prilagođavanja različitim topologijama,
- *distribuisanje čvorova* – u zavisnosti od primjene, način postavljanja senzorskih čvorova može znatno da se razlikuje (deterministički ili nasumično)
- *primjena standarda ili pojedinačnih rješenja* – primjena standarda znatno povećava integraciju sa drugim rješenjima, ali često i troškove u pogledu implementacije istih,
- *skalabilnost* – proširljivost senzorske mreže sa dodatnim senzorskim čvorovima ili senzorskim jedinicama.

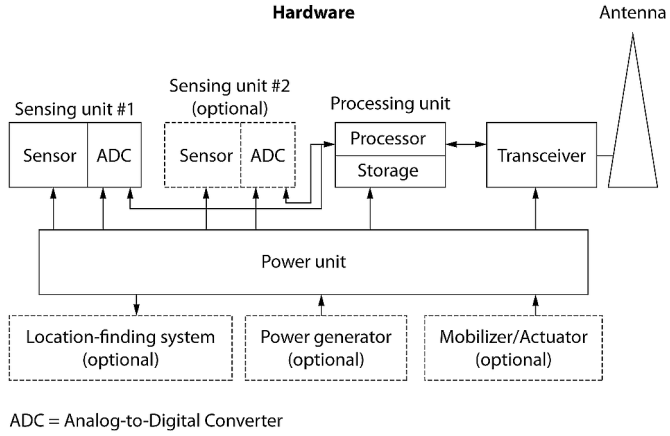
Dodatna kompleksnost proizilazi iz činjenice da se BSM po svojim karakteristikama znatno razlikuju od LAN mreža u pogledu [Yick08, Akyildiz02a]:

- broja elemenata u mreži koji može biti i za nekoliko redova veličine veći od onih u LAN mrežama,
- gustine rasporeda elemenata,
- čestih otkaza elemenata,
- česte promjene topologije mreže,
- načina komunikacije između elemenata,
- ograničenja napajanja, obrade i skladištenja,
- jedinstvenog globalnog identifikatora za pristup (zbog velikog broja senzora)

Iako se senzorski čvorovi, u implementacionom domenu, znatno razlikuju u zavisnosti od primjene, dva osnovna elementa čine svaki od njih, a to su *hardver* i *softver*. Projektovanje ovih elemenata mora biti oslonjeno na dobro definisane elemente računarske arhitekture.

2.1.3.1. Hardver

U radovima [Hart06, Sohraby07] identifikovano je preko 200 primjena senzorskih mreža, od kojih mnogi koriste specifične tehnologije za implementaciju senzorskih čvorova. Zajednička karakteristika većine pristupa je postojanje četiri osnovna hardverska podsistema koji čine tipičan senzorski čvor [Sohraby07, Townsend05, Nack09] (Slika 2.1.3.1.1):



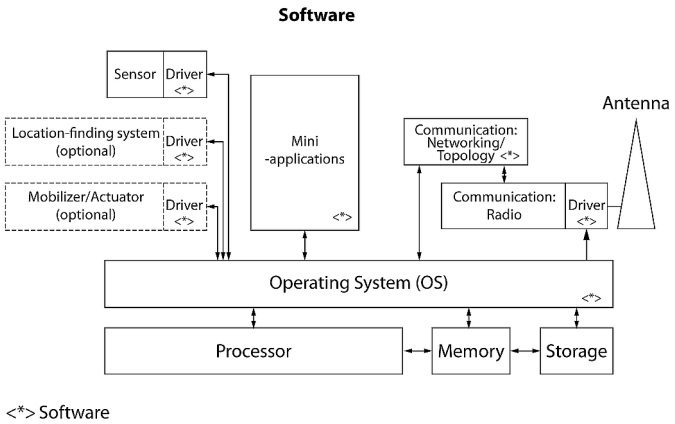
Slika 2.1.3.1.1. Arhitektura hardverskog sistema senzorskog čvora [Sohraby07, Nack09]

1. *Napajanje* – odgovarajuća energetska infrastruktura koja podržava rad od nekoliko sati do nekoliko mjeseci ili godina (u zavisnosti od primjene). Najčešće se implementira autonomnim izvorom napajanja – baterijom.
2. *Jedinica za obradu i skladištenje podataka* – koriste se za trenutnu obradu i manipulisanje podataka, privremeno smještanje, enkripciju, ispravljanje grešaka, digitalnu modulaciju i digitalni prenos. Zahtjevi za obradom su različiti i najčešće se mogu izvršiti na hardveru u rasponu od 8-bitnih mikrokontrolera do 64-bitnih mikroprocesora, dok je skladištenje tipično u rasponu od 0.01 do 100 gigabajta (GB).

3. *Senzorske jedinice sa analogno-digitalnim konvertorom* (ADC) – predstavljaju posrednika između okoline (parametara koji se posmatraju) i BSM. Osnovne senzorske jedinice mogu biti mjerači brzine (akcelorometri), senzori vlage, svjetlosti, temperature, pritiska i dr. ADC transformiše analogne vrijednosti koje su očitane na senzoru u digitalne signale koji mogu biti obrađeni od strane procesora.
4. *Primopredajnik i antena* – BSM moraju imati mogućnost komunikacije u *mesh*, *point-to-point* ili *point-to-multipoint* sistemima.

2.1.3.2. Softver

Ograničenja koja unose hardverske platforme senzorskih čvorova zahtjevaju oslonac na specijalizovane operativne sisteme (OS) i programske jezike koji olakšavaju razvoj aplikacija za senzorske čvorove. Pristupom hardverskoj arhitekturi (procesoru, memoriji, senzorima i dr.) uz oslonac na OS i aplikativni softver, omogućeno je "fino" upravljanje pojedinim komponentama i optimizacija potrošnje energije. Softverska arhitektura senzorskog čvora (Slika 2.1.3.2.1) obično posjeduju pet osnovnih podsistema [Sohraby07, Nack09] koji su orkestrirani od strane OS:



Slika 2.1.3.2.1. Arhitektura softverskog sistema senzorskog čvora [Sohraby07, Nack09]

1. *Operativni sistem* – *Operating system* (OS) – poznat i kao *middleware*, predstavlja mikrokôd koji se koristi od strane softverskih modula sa ciljem pružanja podrške različitim funkcionalnostima. OS također enkapsulira funkcionalnosti mikroprocesora na niskom nivou (nivo mašinskog kôda) čime

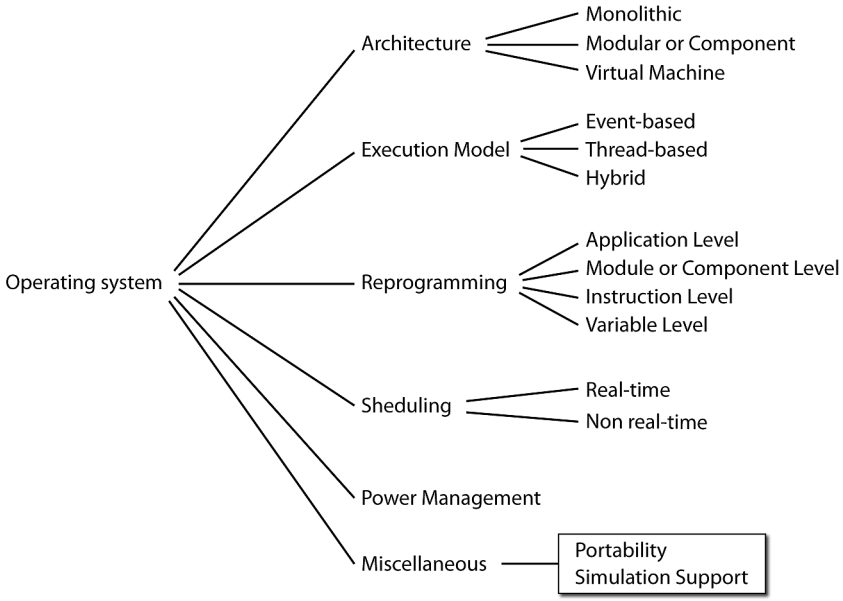
pojednostavljuje njegovo rukovanje. Poželjno je da OS bude slobodno dostupan i namjenski kreiran za rad sa senzorskim čvorovima čime se obezbjeđuje: optimalno korišćenje ahritekture, smanjuje veličina programskog kôda i skraćenje vremena potrebnog za implementaciju rješenja. TinyOS predstavlja najčešće korišten OS za ove namjene.

2. *Drajveri senzora* – predstavljaju softverske module koji upravljaju osnovnim funkcijama senzorskih jedinica. Senzori mogu biti modularnog/*plug-in* tipa, pa je potrebno podesiti odgovarajuću konfiguraciju senzorskog čvora prije očitavanja podataka sa senzora. Drajveri enkapsuliraju funkcionalnosti senzora na niskom nivou pružajući korisnicima interfejs za njegovo rukovanje.
3. *Komunikacioni podsistem* – modul koji upravlja komunikacionim funkcijama uključujući i rutiranje, baferovanje i prosljeđivanje paketa, održavanje topologije, kontrole pristupa i šifrovanje.
4. *Komunikacioni drajveri* – softverski modul koji upravlja: prenosom podataka, usklađivanjem takta i sinhronizaciju, kodiranjem signala, oporavkom i brojanjem bitova, nivo signala i modulacijama.
5. *Aplikacije za obradu podataka* (*mini applications*) – aplikacije za numeričku obradu, skladištenje i manipulisanje podacima ili bilo koje druge aplikacije koje su podržane na nivou senzorskih čvorova.

Operativni sistem (*middleware*) predstavlja najvažniji element sistema, jer obezbjeđuje interfejse za programiranje – *Application Programming Interface* (API), pristup hardveru, te organizuje sve ostale module sistema. Definisani API igra ključnu ulogu pri razdvajanju funkcionalnosti na niskom nivou od one na aplikativnom, najčešće obuhvatajući [Mallika07]: Mrežni API, API za čitanje podataka sa senzora, API za manipulaciju sa memorijom, API za rad sa napajanjem i API za upravljanje zadacima. Rad savremenih OS za BSM zasnovan je na različitim implementacionim modelima koji obuhvataju: agenate (*agent-based*), upite (*query-based*) i događaje (*event-based*). Zbog posebne karakteristike i prirode BSM u prvi plan se stavljaju na događajima zasnovani (*event-based)* modeli kao najprikladnija paradigma za razvoj aplikativnih rješenja [Taleghan07, Ammari14].

Ključni faktori koji moraju biti podržani u sklopu OS namjenjenih za podršku BSM-a [Taleghan07, Farooq11, Mallika07, Hill05, Fröhlich08, Ammari14] obuhvataju: fleksibilnu arhitekturu, model izvršavanja, komunikaciju, upravljanje resursima (memorijom, energijom, skladištem

podataka), upravljanje izvršnim organima, reprogramiranje i rad u realnom vremenu. U radu [Taleghan07] je predložen okvir za klasifikaciju OS u skladu sa sljedećim karakteristikama: arhitektura, model izvršenja, reprogramiranje, raspoređivanje poslova i upravljanje potrošnjom energije (Slika 2.1.3.2.2.) i drugo (miscellaneous).



Slika 2.1.3.2.2. Okvir za klasifikaciju OS u BSM [Taleghan07]

Na osnovu predloženih klasifikacija, u radovima [Taleghan07, Cecílio14, Mallika07, Farooq11, Hill05, Fröhlich08, Novkov07, Albert14, Ammari14] data je detaljna analiza postojećih OS za BSM iz koje slijedi da TinyOS predstavlja *de facto* standard u okviru OS za BSM. On podržava događajima upravljan komponent-baziran razvoj softvera [Taleghan07, Cecílio14, Hill05, Albert14] i dekompoziciju programa na funkcionalne komponente povezane interfejsima. Glavna prednost ovakvog pristupa predstavlja mogućnost ponovnog korištenja razvijenih komponenti [Cecílio14]. Kao namjenski operativni sistem, TinyOS reaguje na hardverske događaje, rukuje njima i kroz koncept zadataka (*tasks*), koji su ekvivalentni funkcijama u drugim programskim okruženjima, vrši njihovu obradu i izvršavanje. Kompletan TinyOS sistem, njegove biblioteke i aplikacije su implementirane na namjenskom programskom jeziku nesC, koji je po sintaksi sličan programskom jeziku C [Novkov07]. Kao glavne nedostatke obrađeni izvori navode: odsustvo podrške za

dinamičko alociranja memorije i izvršavanje programa isključivo unutar jednog procesa.

2.2. Servis orijentisane senzorske mreže

Razvojem IKT industrije kreiraju se senzorski čvorovi manjih dimenzija sa većom snagom obrade informacija koji uz primjenu IP protokola grade ključne elemente za implementaciju novih vizija senzorskih mreža (IoT, WoT, SW). Pored primjene IEEE 802.11 protokola uz oslonac na IPv4 i IPv6, *Internet Engineering Task Force* (IETF) je razvio standard za primjenu IPv6 protokola u 6LoWPAN mrežama, odnosno u IEEE 802.15.4 baziranim mrežama. Iako pomenute tehnologije predstavljaju spregu između BSM i Interneta, mogućnost povezivanja na mrežnom nivou ne predstavlja dovoljan uslov za potpuno pružanje servisa na globalnom nivou. Da bi se ispunili svi uslovi, rješenje je potrebno potražiti u standardnim Web tehnologijama koje omogućavaju komunikaciju između udaljenih računara korištenjem IP protokola i SOA arhitekture.

Implementacija SOA arhitekture se može realizovati korištenjem Web servisa koji su podjeljeni u dvije implementacione grupe: SOAP i REST. Potti i dr. [Potti12] u svom radu vrše komparativnu analizu ova dva pristupa uz elaboraciju prednosti i mana njihove upotrebe. Prednosti RESTa u odnosu na SOAP su u jednostavnost. REST Web servisi su jednostavniji od SOAP servisa jer ne koriste složeno formatiranje parametara i rezultata u SOAP-XML formatu. Ovo omogućava da jednostavni klijenti, kao što su JavaScript Ajax pozivi, direktno pozivaju REST servise i prihvataju rezultate. Podaci koji se vraćaju klijentima mogu biti u proizvoljnim formatima, pa tako različiti klijenti mogu da zatraže podatke u formatu koji im najviše odgovara, što je jednostavnije od SOAP formata koji se, iako je standardizovan, mora parsirati.

Zbog toga što senzori posjeduju ograničenja u pogledu snage obrade, energije i komunikacije, propusnog opsega, neophodan je oslonac na "laganije" (*light-weight*) mehanizme. U radu [Rouached12] je predložen RESTful pristup baziran na SWE servisima za interakciju sa BSM. U ovom slučaju svaki senzorski čvor se posmatra kao RESTful resurs kojem se može pristupiti preko SWE usluge koristeći za razmjenu podataka JSON format kao alternativu XML-u. Drugi pristupi se uglavnom zasnivaju na direktnoj implementaciji REST API-ja na svakom

senzorskom čvoru u cilju obezbjeđenja komunikacije između senzora, automatskog otkrivanja senzorskog čvora kao i efikasne komunikacije sa Web korisnicima.

Radovi [Muracevic10] i [Yazar09] prikazuju RESTful Web servis sa IP-orijentisanim BSM. Za obezbjeđivanje neometane komunikacije između čvorova preko Web servisa i olakšavanje direktne integracije u moderne IT sistema u sklopu ovih radova usvojen je sistem "*IP orijentisane senzorske mreže*". Implementacija RESTful API-ja na senzorskim čvorovima u cilju obezbjeđenja pristupa senzorima i aktuatorima preko Web-a prikazana je u [Adama]. Kako se senzorski čvor može povezati sa različitim senzorima i aktuatorima, u prisustvu velikog broja senzorskih čvorova, ručna konfiguracija centralnog sistema nadzora postaje neizvodljiva. Uz oslonac na RESTful API-je uvodi se *plug-and-play* pristup koji omogućava automatsko otkrivanje senzorskih čvorova u bežičnoj mreži, ali i funkcionalnost koju oni pružaju. Za povezivanje različitih senzora i aktuatorskih čvorova u monitoring sistem, u [Schor09] je elaboriran *plug-and-play* pristup zasnovan na primjeni RESTful API-ja. Ovaj API je direktno realizovan na Web uređajima (bez upotrebe *gateway*) u cilju automatskog otkrivanja senzorskih čvorova u bežičnoj mreži. Obezbjeđenje konzistentnost informacija na serveru zahtjeva da senzorski čvorovi periodično šalju statusne poruke. Kao alternativu ovom pristupu, Schor i dr. predlažu pribavljanje stanja senzora na zahtjev od strane servera. Ovaj pristup se smatra efikasnijim zbog toga što se stanja, i senzora i aktuatora, mogu ekstrahovati i modifikovati uz oslonac na isti interfejs. Uprkos navedenim pogodnostima, potrebno je definisati dodatne zahtjeve, kao što su: definisanje i ažuriranje posrednika, pronalaženje pravih usluga i rješavanje neusaglašenosti podataka.

Veliki broj istraživača je zainteresovan za pružanje efikasnog i rigoroznog načina preuzimanja, organizovanja i čuvanja heterogenih podataka/metapodataka sa senzora. Za povezivanje različitih vrsta BSM, ekstrakciju senzorskih podataka i njihovo predstavljanje na Internetu u [Stefani10] autori predlažu novu bazu podataka pod nazivom *RESTful Environmental DataBase* (REnvDB). Ova baza obezbjeđuje zajednički RESTful interfejs koji omogućava pristup podacima pomoću HTTP operativnih poziva u URL-u i prevođenje razmjenjenih poruka preko odgovarajućeg *gateway*-a na strani BSM. Zbog svoje jednostavne konekcije zasnovane na RESTful interfejsu ovaj pristup može da smanji kompleksnost komunikacije. Vremenske uštede koje se ostvare na strani baze podataka gube se zbog korištenja *gateway*-a za prosljeđivanje senzorskih podataka u REnvDB. Prevazilaženje ovog nedostatka uz

oslonac na standardizaciju poruka elaborirano je u [Rouached10] i [Mohamed11].

Analizirajući primjenu i koncepte REST arhitekturnog modela Rouached i dr. [Rouached12] potvrđuju potrebne i dovoljne uslove za primjenu u BSM, odnosno za kreiranje SW. Kroz niz radova [Ludovici10, Potti12, Yazar09] autori verifikuju korištenje REST servisa u implementaciji SW-a ističući prednosti navedenog pristupa. Korištenjem REST arhitekturnog stila za SWE efikasno se poboljšava arhitektura, prevazilazi velika potrošnja energije i ograničenja vezana za resurse. Detaljnija elaboracija novih koncepta i primjena servis orijentisanih senzorskih mreža prikazana je u paragrafima 2.2.1., 2.2.2. i 2.2.3.

2.2.1. Internet of Things

Pojam "*Internet of Things*" prvi put je predložen od strane suosnivača *Auto-ID* centra na *Massachusetts Institute of Technology – MIT*, Kevin Ashton-a 1999. godine i izvorno je bio povezan sa RFID i *Electronic Product Code* (EPC) tehnologijama [Mukhop14, Bassi13, Yang14]. Danas se pojam IoT veže za novu paradigmu koja predstavlja korak naprijed u Internet revoluciji i najčešće se koristi da opiše sadržane (embedded) uređaje ("*pametne objekte*" ili "*stvari*") koji su opremljeni senzorima i aktuatorima a imaju mogućnost povezivanja sa Internetom. Oslonac na Internet, kao komunikacionu infrastrukturu, znatno povećava pouzdanost, održivost i efikasnost u pristupu informacijama, ali i upravljivost samih "*pametnih objekata*"/"*stvari*" [Mukhop14, Delicato13].

International Telecommunication Union (ITU) definiše IoT kao "*globalnu infrastrukturu za informatičku zajednicu, koja omogućava napredne usluge povezivanjem (fizičkih i virtuelnih) stvari na osnovu postojeće i nove interoperabilne informacione i komunikacione tehnologije*" [Wortmann15], dok *IoT European Research Cluster* (IERC) opisuje IoT kao "*globalnu mrežu međusobno povezanih pametnih objekata/stvari koji pružaju mogućnost međusobne komunikacije i komunikacije sa okolinom, razmjenjujući podatke koji se prikupljaju iz okruženja, dok pokretanje procesa na reakcije uzrokovane stanjem okruženja mogu biti realizovane sa ili bez direktne veze sa čovjekom*" [Vermesan14] (Slika 2.2.1.1.).

Obje definicije sadrže pojam "*pametni objekat*" odnosno "*stvar*", a u radu [Miorandi12] se posmatraju kao entiteti koji:

Slika 2.2.1.1. Internet of Things vizija [Vujović15d]

- imaju fizičku reprezentaciju i posjeduju niz povezanih karakteristika (veličina, oblik, itd.),
- imaju redukovani skup komunikacionih funkcija, kao što su mogućnost detekcije, te prihvatanje/odgovoranje na poruke,
- posjeduju jedinstveni identifikator,
- su povezani sa najmanje jednim imenom i adresom. Ime, po pravilu, predstavlja opis objekta koji može biti pročitan od strane čovjeka, dok adresa predstavlja niz znakova koji se najčešće koristi za komunikaciju,
- posjeduju mogućnost obrade. Osobina obrade varira od sposobnosti očitavanja parametara, do kompleksne obrade informacija i upravljanja mrežnim zadacima,
- mogu imati sredstvo za očitavanje fizičkih pojava (temepratura, svjetlo, nivo elektromagnetnog zračenja, i dr.) ili mogućnost pokretanja akcija koje imaju uticaj na fizičke objekte (pokretanje pogona).

Međutim, i pored ovako detaljne definicije, sa razvojem IoT-a potrebno je proširiti i sam koncept "*pametnog objekta*"/"*stvari*" kako bi bio obuhvaćen bilo koji realni ili virtuelni "*učesnik*", koji predstavlja realne objekte, ljude, virtuelne podatke ili inteligentne softverske agente [Yang14].

U [Miorandi12] su prepoznate tri osnovne karakteristike sistema baziranih na IoT:

- *sve komunicira*: pametne stvari imaju mogućnost bežične komunikacije između sebe i između međusobno povezanih objekata unutar ad-hoc mreže,
- *sve se identifikuje*: pametne stvari se identifikuju preko digitalnog imena,

- *sve interreaguje*: pametne stvari mogu interreagovati sa lokalnim okruženjem kroz očitavanja i aktiviranja postojećih mogućnosti.

Kako je predstavljeno u [Taleghan07], za realizaciju IoT zahtjeva potrebna je dramatična promjena sistema, arhitekture i komunikacije koja mora biti fleksibilna, prilagodljiva, sigurna, ali ne i nametljiva. Za ispunjenje postavljenih zadataka, nameću se naučni i tehnički izazovi koji zahtijevaju različite kompetencije na [Mukhop14, Yang14, Delicato13]:

- *tehnološkom nivou* – izazovi vezani za integraciju pametnih "*network enabled*" objekata sa velikim ograničenjima u pogledu energije i okruženja u kojem će biti postavljeni,
- *nivou komunikacije i povezivanja* – izazovi vezani za pružanje sigurnih usluga u velikim, dinamičkim i fleksibilnim mrežama,
- *nivou inteligencije* – izazovi vezani za spajanje podataka i detekcije usluga, za podatke prikupljene od strane pojedinih "*network enabled*" objekata kao što su RFID i bežični senzori na zahtjev distribuiranih korisnika.

Primjena koncepta IoT moguća je u različitim područjima [Wortmann15, Mukhop14, Song14, Miorandi12, Vermesan14, Vermesan15]: napredne elektroenergetske mreže, sigurnost, praćenje saobraćaja, farmacija (produkcija, logistika i maloprodaja lijekova) i zdravstvo, praćenje porijekla i kvaliteta hrane kao i praćenje mnogih drugih različitih procesa, automatizacije istih, dostave proizvoda i dr.

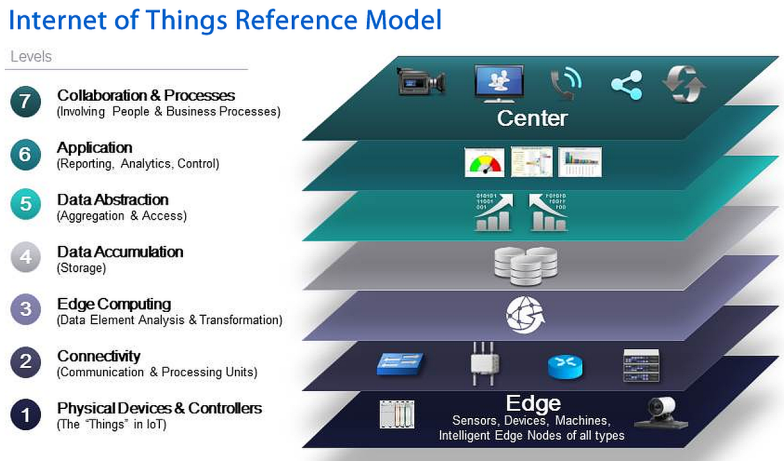
Analizom upotrebe IoT-a mogu se identifikovati ključne karakteristike sistema koje IoT treba podržavati [Miorandi12]:

- *heterogenost povezanih uređaja* – veliki broj uređaja sa različitim karakteristikama i mogućnostima,
- *skalabilnost* – zbog velikog broja uređaja na globalnoj infrastrukturi, problem skalabilnosti se dešava na više različitih nivoa: imenovanje i adresiranje, komunikacija, servisi, obrada informacija i dr.,
- *razmjenu podataka putem bežičnih tehnologija* – veliki broj uređaja može znatno da smanji spektar za komunikaciju,
- *energetsku optimizaciju* – optimizacija potrošne energije (često i na račun performansi sistema),
- *lociranje i mogućnost praćenja* – veoma bitno za aplikacije u logistici i praćenju životnog ciklusa proizvoda,
- *mogućnost samo-organizovanja* – potrebno da se ljudska intervencija smanji na najmanji mogući nivo,

- *interoperabilnost i menadžment podataka* – potrebno usvojiti standardizovane formate, modele i semantičke opise podataka koristeći dobro definisane jezike i formate,
- *sigurnost i zaštita privatnosti* – sigurnost se treba razmatrati na sistemskom nivou i biti sastavni dio dizajna arhitekture sistema.

Zbog velike heterogenosti arhitekture IoT rješenja, pojavila se potreba za jednim, uniformnim rješenjem, koje bi zadovoljilo sve zahtjeve korisnika. Oslanjajući se na dobro poznate slojevite arhitekture računarskih mreža, kao što su ISO OSI model i Internet TCP/IP model, u radu [Dai12] je predložena arhitektura IoT na osnovu: ITU Y.2002 modela za arhitekturu sveprisutnih mreža, arhitekture RFID baziranih EPC mreža i standardne arhitekture BSM.

U periodu od 2010. do 2013. godine, finansiran od strane Evropske unije, pokrenut je projekat pod nazivom *IoT-A17* koji je za cilj imao razvoj arhitektonskog referentnog modela - *Architectural Reference Model* (ARM) IoT-a [Bassi13, Vermesan14, Vermesan15] (Slika 2.2.1.2.).



Slika 2.2.1.2. Arhitektonski referencijalni model IoT-a [Vermesan14]

Osnovna ideja je da IoT ARM pruži zajedničku strukturu i smjernice za razvoj, korištenje i analizu IoT sistema, uz obezbjeđenje kompatibilnosti sa postojećim sistemima, ali i usvajanje konkretnih rješenja datih u raznim aspektima njihove primjene.

Povezujući "*pametne objekte*"/"*stvari*" u BSM, a onda tako kreirane BSM sa Internetom, formira se IoT informaciona infrastruktura. Za

povezivanja BSM i Interneta postoji mnogo načina koji su u zavisnosti od njihove komunikacione arhitekture kategorisani u tri grupe [Yang14]:

1. *Proksi interfejs* – komunikacija između TCP/IP korisnika i senzorskih čvorova se ne obavlja direktno, već preko *proksi* računara po slobodno izabranom komunikacionom protokolu. *Proksi* proaktivno prikuplja podatke iz BSM i skladišti ih u svoju bazu, nakon čega do željenih podataka korisnici dolaze postavljajući SQL upite ili putem Web interfejsa. Mana ovakvog pristupa jeste što sva komunikacija sa BSM prestaje u slučaju otkaza *proksi* računara, kao i činjenica da njegova implementacija zavisi od konkretnog zadatka ili određenog skupa protokola (za svaku aplikaciju potrebna je nova implementacija *proksi* servisa).
2. "*gateway*" – predstavlja jedan od osnovnih uređaja koji obezbjeđuje vezu između TCP/IP i BSM mreža. *Gateway* obavlja nekoliko zadataka kao što je konverzija protokola, prenos poruka, i dr., a kao rezultat toga senzorski čvorovi i Internet klijenti mogu direktno razmjenjivati informacije. Rješenja koja koriste *gateway* pristup, mogu biti grupisana u dvije kategorije: aplikacioni *gateway* (predstavlja jednostavan *gateway* pristup koji preslikava adrese BSM u IP adrese. Preslikavanje se obavlja na aplikativnom sloju.) i mreže tolerantne na odgodu – *Delay Tolerant Network* (DTN) (DTN predstavlja rješenje slično prethodnom pristupu, s tim da se u obje mreže, BSM i TCP/IP, dodaje "*bundle*" sloj koji skladišti i prosljeđuje komunikacione pakete.).
3. *TCP/IP enkapsulacija* – predstavlja implementaciju TCP/IP protokola na mikroračunarskim sistemima ograničenih resursa. Kod ovakvog pristupa, susreću se brojni problemi, npr. kako senzorskom čvoru dodijeliti IP adresu, te kako efikasno koristiti rutiranje na osnovu adresa i podataka, u skladu sa mrežnim saobraćajem. IPv6 preko bežičnih mreža male potrošnje - *IPv6 over Low-power Wireless Wrea Networks* (6LoWPAN), nastao od strane IETF-a, predstavlja tipičnu implementaciju TCP/IP enkapsulacije. 6LoWPAN definiše način prenosa IPv6 paketa preko IEEE 802.15.4 MAC sloja, pa tako Internet korisnici mogu direktno pristupiti pojedinačnim senzorima pomoću njihove IPv6 adrese.

U radovima [daCosta13, Puliafito10, Alphand10, Eisenhauer10, Giordano14] su prikazani različiti pristupi za realizaciju IoT, pa se tako u [Giordano14] predlaže korištenje orkestratora koji vrši sinhronizaciju senzorskih čvorova, a ujedno omogućava pristup pojedinačnim senzorima

ili obrađenim podacima. U [Alphand10] predložen je novi pristup integracije mreže bežičnih senzora i aktuatora – *Wireless Sensor and Actuator Networks* (SANET) koji je zasnovan na novom principu komunikacione infrastrukture *diseminacija podataka ili sadržaja* po modelu *objave i pretplate* ("*publish/subscribe*"). U radovima [Puliafito10, Eisenhauer10] autori se oslanjaju na SOA arhitekturu za kreiranje *middleware* sloja za povezivanje BSM i aplikativnog sloja, pri čemu korisnik može vidjeti samo interfejs koji mu pruža servis. Francis de Costa u knjizi [daCosta13] iznosi niz argumenata kojima se protivi korištenju standardnih tehnologija BSM za implementaciju IoT-a zbog velikog prenosa podataka i kompleksnosti zaglavlja paketa. Francis predlaže uvođenje novih protokola "*Chirp*" koji bi znatno redukovali ove probleme.

U procesu analize IoT tehnologija, veoma je bitno sagledati ekonomske i sigurnosne aspekte sistema. U proizvodnim industrijama, Internet se uglavnom koristi za pojednostavljenje procesa što znatno smanjuje troškove uz povećanje kvaliteta i raznovrsnosti ponude [Fleisch14].

Sigurnosni aspekti predstavljaju jedan od najznačajnijih elemenata IoT-a i zbog svoje prirode moraju biti razmatrani na više različitih nivoa. Identifikovane su tri ključne komponente sigurnosti IoT-a [Miorandi12]:

- *tajnost podataka* – predstavlja osnovno pitanje IoT-a koje garantuje da samo ovlašteni subjekti mogu pristupati i mijenjati podatke,
- *privatnost* – definiše pravila pod kojima se može pristupiti podacima pojedinačnih (individualnih) korisnika,
- *povjerenje* – se odnosi na proces razmjene certifikata koji omogućavaju da servisi ili resursi jedne strane izvrše provjeru validnosti usluga ili resursa druge strane.

U skladu sa predstavljenim komponentama, u radovima [Han12, Zhang10] su razmatrane primjene modifikovanih kontrola pristupa zasnovanih na ulogama – *Role Base Access Control* (RBAC), dok je u [Ukil11] analizirana primjena raznih tehnika za sigurnost embedded sistema. Autori radova [Hasić14, Weber10] su akcenat stavili na zaštitu IoT-a kroz pravne aspekte propisanih zakonskih rješenja (legislativa) na državnom i globalnom nivou.

2.2.2. Web of Things

Analizirajući IoT, njegovu implementaciju i integraciju sa Internetom, nameće se potreba usvajanja otvorenih standarda komunikacionih protokola zasnovanih na interoperabilnim embedded IP i Web tehnologijama, koje znatno smanjuju kompleksnost IoT arhitekture i dugoročno predstavljaju optimalno rješenje [Colitti14]. Integracijom IoT-a sa Web-om, znatno se olakšava izgradnja "*pametnih objekata*"/"*stvari*" uz pomoć Web tehnologija (npr. HTML, JavaScript, Ajax, PHP, Ruby) koje mogu biti korištene uz dobro poznate Web mehanizme pregledanja, pretraživanja, povezivanja i dr. [Guinard11a]. Danas, najčešće korišten pristup integracije, predstavljaju Web servisi koji korisnicima pružaju platformsku nezavisnost, višekratno korištenje i drastično pojednostavljenje procesa razvoja aplikacija [Colitti14]. Ovakav pristup, predstavlja osnovu u kojoj "*pametni objekti*"/"*stvari*" mogu međusobno komunicirati istim "jezikom" kao i ostali resursi Interneta, pružajući tako jednostavan način integracije bilo kojih fizičkih uređaja putem Web-a [Guinard09]. U poglavlju 1.2.5 su prikazani osnovni principi SOA arhitekture koja za primjenu u IOT-u mora da pruži podršku na globalnom nivou za više aplikacija, izvora podataka i korisnika [Yang14]. Međutim, priroda IoT-a u kojoj "*resurs*" predstavlja osnovni gradivni blok, nameće korištenje ROA arhitekture koja je najčešće implementirana REST tehnologijom [Vujović15c]. Ključnu asptrakciju RESTful Web servisa predstavljaju resursi, gdje pod pojmom resursa može biti bilo što dovoljno važno da bi se predstavilo kao "*pametni objekat*"/"*stvar*" (npr. dokumenti, podaci iz baze, rezultat izvršavanja nekog programa, fizički objekti ili apstraktni koncepti). U poglavlju 1.2.5.2. i radovima [Vujović15c, Delicato13, Mathew13] su navedene uopštene karakteristike RESTful servisa, ali za njihovo bolje razumijevanje potrebno je dodati pet osnovnih principa na kojima se temelje [Delicato13, Guinard10, Mathew13]:

- *identifikacija resursa na globalnom nivou* – identifikacija resursa pomoću URI-ja koji obezbjeđuje globalnu adresu za resurse i servise. Resurs može biti server, dokument, Web stranica, video zapis ili fizički uređaj,
- *jedinstveni interfejs za pristupanje resursima* – definiše pristup resursima preko jedinstvenog interfejsa koji ima dobro definisanu semantiku, npr. HTTP protokol,
- *samo-opisne poruke* – predstavlja format poruka za razmjenu između klijenta i servera. Resurs može imati nekoliko različitih formata kao što su HTML, XML, JSON, i dr.,

- *promjena stanja aplikacije putem hiperlinkova* – je osnovno načelo REST-a, i predstavlja kontrolu Web aplikacija putem hiperlinkova koji ukazuju na Web resurse. Hiperlinkovi za pristup resursima koriste globalnu identifikaciju putem URI-ja,
- *interakcija bez čuvanja stanja* – server ne čuva sesije pa je neophodno da svaki zahtjev od klijenata sadrži sve informacije koje je potrebno obraditi, uključujući zaglavlja, URL, parametre upita i dr.

Primjenjujući standardnu Web tehnologiju ili RESTful Web servise i HTTP protokol, objekti su predstavljeni kao RESTful resursi kojima se može direktno pristupiti putem Web-a. REST implementacija je veoma "lagana", pa su danas sve češći slučajevi da HTTP klijenti i serveri mogu biti ugrađeni direktno u male "*IP enabled*" platforme, pružajući na taj način odlično poklapanje REST i IoT arhitekture [Vujović15c, Delicato13]. Inspirisana ovim konceptima, nastaje nova paradigma za razvoj aplikacija pod nazivom Web stvari – *Web of Things* (WoT) [Delicato13, Guinard11a, Tridium09, Delgado13]. WoT paradigma je zasnovana na korištenju protokola i standarda koji su široko prihvaćeni i već u upotrebi, kao npr. HTTP i URI za razmjenu informacija i inteoperabilnost uređaja, te XML i JSON za reprezentaciju istih [Pérez14]. Ovi standardi, u kombinaciji sa drugim elementima, kao što su REST i ROA, omogućavaju da se senzorske jedinice BSM tretiraju kao bilo koji drugi resurs na Web-u [Delicato13, Pérez14]. Generalno govoreći, WoT pokušava da premosti jaz između sadržanih (embedded) uređaja i Web-a, u svrhu integrisanja različitih i izolovanih IoT mreža sa korisničkim okruženjima i drugim mrežama putem Web tehnologija [Pérez14]. IoT tako svakodnevnim uređajima omogućava IP adresu i pristup Internetu, dok WoT pomjera ovu granicu integrišući uređaje u "tkivo" samog Web-a [Tridium09]. Da bi se realizovala WoT paradigma, potrebno je obezbjediti da "*pametni objekti*"/"*stvari*" budu adresabilni, da se mogu pretraživati, kontrolisati i da im se može pristupati preko Web-a. Iako danas Web serveri postaju masovno ugrađeni u hardverske uređaje, teško je pretpostaviti da će svi ovako kreirani "*pametni objekti*"/"*stvari*" imati implementirane RESTful interfejse. Upravo zbog navedenog, u praksi se susreću dva moguća rješenja [Delicato13, Pérez14, Guinard11a, Guinard09, Delgado13]: Web povezivanje direktno na "*pametne objekte*"/"*stvari*" ili indirektno povezivanje putem *proksi*-ja. U prvom pristupu, ugrađeni Web serveri su integrisani direktno u "*pametne objekte*"/"*stvari*" omogućavajući da svojstva ovih uređaja budu dostupni na Web-u kao RESTful resursi. Međutim, kad god "*pametni*

objekat"/"*stvar*" nema dovoljno hardverskih resursa da pokrene ugrađeni server, potreban je drugačiji pristup za integraciju WoT-a baziran na primjeni WoT *Gateway*-a ili *Smart Gateway* uređaja koji rutiraju saobraćaj između Interneta i BSM.

Iako se HTTP protokol značajno koristi pri implementaciji IoT i WoT paradigmi, zbog svoje kompleksnosti (prvenstveno zbog TCP protokola koji ima velika zagavlja) ovakav pristup često nije moguće primjeniti u uređajima sa ograničenim hardverskim mogućnostima. Pored toga, prilikom primjene principa Web servisa na uređaje ograničenih hardverskih mogućnosti, potrebno je u obzir uzeti i druge faktore kao što su: životni vijek baterije i kašnjenje u procesu prenosa podataka. IoT aplikacije su najčešće kratkog životnog vijeka i izvršavaju se na baterijski napajanim uređajima koji su aktivni samo kada postoji potreba za razmjenom podataka. Ovakve aplikacije zahtjevaju *multicast* (*jeda-na-više*) i asinhronu komunikaciju u odnosu na *unicast* (*jeda-na-jedan*) i sinhroni pristup koje koriste standardne Web aplikacije [Colitti14]. Ove karakteristike IoT-a ukazuju na to da HTTP protokol ne predstavlja optimalan izbor za implementaciju WoT-a, te da je potreban novi, REST baziran pristup. Vođeni ovim problemom, *IETF Constrained RESTful environments* (CoRE) radna grupa definiše novi REST baziran protokol pod nazivom *Constrained Application Protocol* (CoAP) koji je definisan karakteristikama [Mathew13]:

- dizajn baziran na RESTful principima koji minimizira složenost mapiranja sa HTTP protokolom,
- zaglavlja male veličine i jednostavna za parsiranje,
- kontekstno zavisna i URI podrška,
- pronalaženje resursa koje nude CoAP servisi,
- prijava za resurse (pretplata) i njihova obavještenja,
- mehanizam keširanja podataka.

CoAP predstavlja optimizovani Web transfer protokol za uređaje sa ograničenim hardverskim mogućnostima, baziran na *User Datagram Protocol*-u (UDP) [Colitti14, Delgado13]. Pored toga, dodatno su modifikovane postojeće i kreirane nove funkcionalnosti kako bi protokol bio odgovarajući za IoT.

Poboljšanje implementacije WoT pristupa je razmatrano u više radova. Tako u [Chen14] autori predlažu proširenje HTTP standarda sa dodatnim metodama BRANCH i COMBINE za podršku grupnoj komunikaciji, dok je u radu [Nie12] integracija BSM sa WoT urađena putem *Extensible Messaging Presence Protocol* (XMPP) protokola.

Kao i kod IoT-a, sigurnosni aspekti predstavljaju jedan od ključnih elemenata, s tim što WoT nasljeđuje sve sigurnosne elemente SOA arhitekture. U radu [Barka15] su razmatrane sigurnosne prijetnje i izazovi koje se postavljaju pred WoT i predloženo rješenje bazirano na RBAC.

2.2.3. Senzor Web Enablement standardi

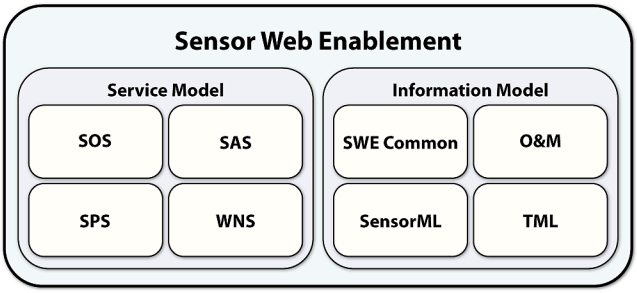
Da bi se IoT paradigma mogla primjeniti na senzorske mreže, OGC je definisao standarde pod nazivom *Sensor Web Enablement* (SWE), koji osiguravaju ključne ciljeve primjene senzorskih čvorova kao IoT elemenata pod nazivom *Senzor Web* (SW). Kroz standarde [Reed13, Cox10, Botts14, Randrian12] OGC definiše servise, interfejse, formate razmjene podataka, ali i najbolju praksu kao i jezik za opis SW elemenata i njihovu primjenu u okruženju. SWE specifikacija omogućava prikupljanje opisa, otkrivanje, pristupanje i postavljanje zadataka senzorskim čvorovima putem Interneta. Podjeljena je u dvije kategorije [Randrian12, Botts08]:

1. opis senzora i definicije podataka,
   - Jezik za modelovanje senzora – *Sensor Modeling Language* (SensorML) – daje opisni model senzora i XML šemu za opisivanje modela senzora koji mogu biti fizički ili apstraktni. SensorML šema se može koristiti za opis metapodataka senzora kao što su karakteristike, lokacija i svojstva senzora.
   - Jezik za modelovanje formata podataka – *Transducer Model Language* (TransducerML) – obezbjeđuje konceptualni pristup i XML šemu za podršku razmjene podataka prema/od senzorskom/g sistema.
   - Praćenje i opažanje – *Observation and Measurement* (O&M) – obezbjeđuje konceptualni model i XML šemu za kodiranje zapažanja i mjerenja senzora.
2. servise
   - Servis za opažanje senzora – *Sensor Observation Service* (SOS) – obezbjeđuje standardan Web servis za pristupanje, filtriranje i registrovanje ili dodavanje senzora ili posmatrača.
   - Servis za objavu podataka senzora – *Sensor Alert Service* (SAS) – pruža standardan Web servis za objavu podataka koristeći "*publish and subscribe*" metodu.
   - Servis za obavještavanje putem Web-a – *Web Notification Service* (WNS). – pruža standardan Web servis za asinhrono

dostavljanje poruka ili upozorenja sa bilo kojeg drugog Web servisa.

- Servis za planiranje senzora – *Sensor Planning Service* (SPS) – pruža standardan Web servis za postavljanje zadataka ili korisničkih zahtjeva.

Predloženi SWE okvir se sastoji od niza standarda koji definišu formate za senzorske podatke, metapodatke i interfejse Web servisa za obezbjeđenje funkcionalnosti senzora. SWE okvir se može podijeliti u dva dijela: *model servisa* i *model informacije* [Rouached12] (Slika 2.2.3.1.).



Slika 2.2.3.1. Sensor Web Enablement okvir [Rouached12]

SWE model servisa obuhvata standarde koji preciziraju interfejse za različite senzorske Web servise. Četiri usluge interfejsa su definisane u prvoj verziji SWE: SOS, SAS, SPS i WNS. *Protocol Transducing Service Specifications* prve verzije SWE prvenstveno je bio koncentrisan na definiciju XML šeme koja odražava servisnu funkcionalnost. U novoj generaciji SWE, prvi konceptualni model servisa je definisan (korištenjem UML notacije), prije nego što je specificirana XML implementacija modela. Ovo olakšava usvajanje drugih oblika implementacije konceptualnog modela, kao što je JSON implementacija.

SWE model informacije obuhvata skup standarda koji definišu modele podataka prvenstveno za kodiranje senzorskih opažanja, kao i senzorskih metapodataka. Za ovu svrhu, prva verzija SWE je sadržila tri specifikacije: O&M, SensorML i TransducerML. Pošto se TransducerML vrlo rijetko koristi u praksi, danas više ne predstavlja sastavni dio SWE okvira.

Primarni izazov u izgradnji SW jeste kako automatski pristupiti i integrisati različite vrste prostorno vremenskih podataka dobijenih od heterogenih senzorskih uređaja ili generisanih korištenjem simulacionih modela.

Sekundarni izazov predstavlja činjenica da većina aplikacija još uvijek integriše resurse senzora kroz pogodne mehanizme umjesto da se grade na dobro definisanom i ustanovljenom integracijskom nivou. S druge strane, senzorske mreže su trenutno razvijene od strane različitih zajednica proizvodžača senzora i korisnika, pri čemu se svaka zajednica oslanjanja na sopstvene sisteme, semantiku metapodataka, format podataka i softver.

Integrisanje različitih senzora IS nije jednostavno jer nekompatabilni servisi i usluge mogu da izazovu nemogućnost kooperativnog rada različitih senzorskih čvorova, čak i u okviru iste BSM.

U radu [Ali10] autori razmatraju zahtjeve za *middleware* koji omogućavaju da bilo koja aplikacija neprimjetno integriše/koristi podatke sa različitih senzora koji pripadaju istim ili različitim mrežama. Rad se zasniva na SWE standardima i formuliše sljedeće zahtjeve:

1. Informacije o senzorima i njihovi podaci trebaju biti semantički obilježeni koristeći jednostavnu dobro definisanu sintaksu. Ovi komentari treba da pruže informacije o tipu posmatranja, jedinici mjerenja, tačnosti i fizičkoj lokaciji senzora itd. SensorML obezbjeđuje ovu funkciju, dok Q&M može da se koristi za opisivanje jedinica mjerenja i tipa posmatranja.
2. Senzorske mreže zainteresovane za dijeljenje svojih podataka sa drugim aplikacijama trebaju biti u stanju da registruju svoje senzore u dobro poznate repozitorije. S druge strane, aplikacije zainteresovane za korištenje senzorskih podataka trebalo bi da imaju mogućnost lokalizovanja senzora od interesa unutar repozitorijuma, kao i mogućnost primanja podataka dobijenih od strane senzora, na sopstveni zahtjev. Ove osobine mogu biti sprovedene u serveru koristeći standarde opisane u SOS.
3. Aplikacije mogu biti zainteresovane za primanje obavještenja (upozorenja) iz senzorskih mreža. Obavještenje se može definisati kao poslata/primljena poruka ili kao rezultat posmatranja koje zadovoljava određeni kriterijum (senzorsko očitanje vrijednosti iznad ili ispod praga). Kao što je rečeno u 2, ove aplikacije bi trebale imati mogućnost da pronađu senzor od interesa u repozitorijumu servera, sposobnog za slanje obavještenja, a zatim se "pretplatiti" na primanje obavještenja od senzora. SAS obezbjeđuje standarde za ove operacije, dok SES predstavlja unaprijeđenje SOS i može da se koristiti za postavljanje filtera i obavještenja o isporuci koristeći javni pretplatnički model.

4. SAS zahtijeva komunikacioni kanal između senzora i korisničke aplikacije. Upozorenja/obavještenja iz senzora i znanja iz korisničkih aplikacija će se razmjenjivati kroz ovaj kanal. Postoji nekoliko za njegovu izgradnju, pa su autori diskutovali o ulozi XMPP-a za postavljanje višekorisničkog sistema za razmjenu poruka između senzora i korisničke aplikacije. Također, može da se koristi *e-mail* ili RSS *feeds* kao znak isporuke.
5. Kako senzor kontinuirano generiše podatke, serveri SOS i SAS sistema primaju podatke u kontinuiranim tokovima. ESP filtrira tokove podataka prema potrebama raznih aplikacija.

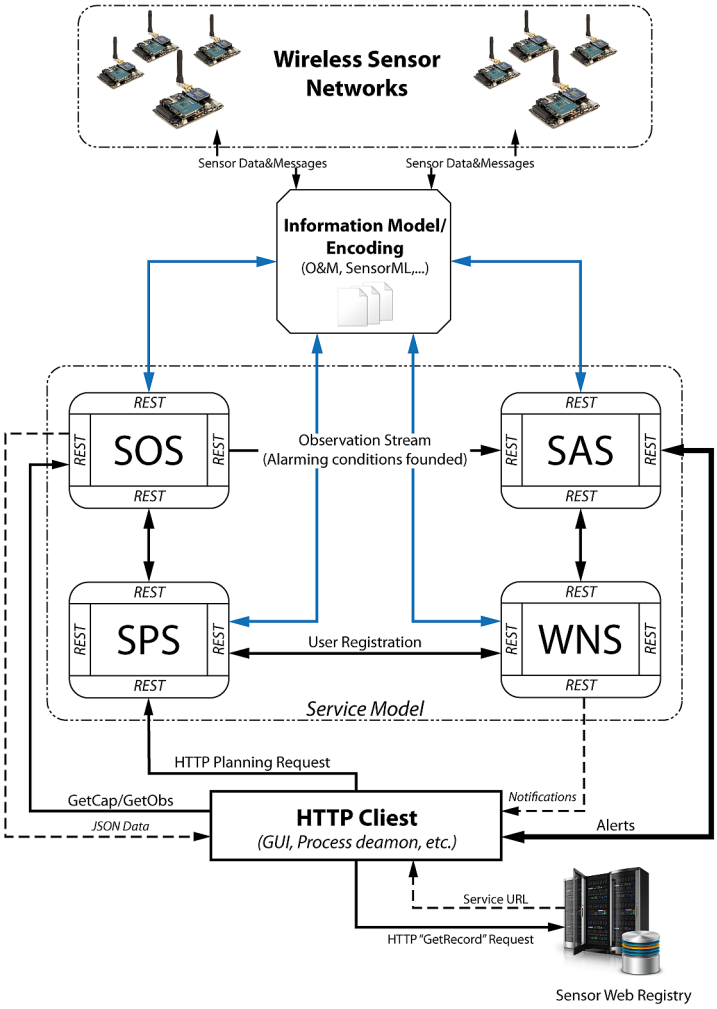
Na osnovu SWE standarda i donesenih zaključaka nakon analize problema, potrebno je na adekvatan način implementirati SM i servise koji odgovaraju datom standardu.

2.2.3.1. Implementacija Senzor Web-a

Koncept SWE zahtjeva globalnu interoperabilnost, pa je potrebno razmotriti adekvatne tehnologije i protokole za njegovu implementaciju. Autori u radu [Ali10] predlažu korištenje *posrednika* koji će djelovati kao virtuelna veza između korisničke aplikacije i prikupljenih podataka, sakrivajući sve detalje koji su uključeni u otkrivanje, filtriranje i isporuku podataka. Međutim, ovakav pristup ne osigurava rješenje problema na globalnom nivou. Zbog toga je potrebno implementaciju SWE okvira potražiti u dobro definisanim tehnologijama koje su *de facto* osnova komunikacije na Internetu. Danas postoje tri vodeća standarda koja definišu okruženje WWW i omogućavaju interoperabilnost između različitih sistema i aplikacija: HTTP, HTML i XML uz oslonac na pomenute tehnologije i SOA arhitekturu (SOAP i REST) moguće je znatno ubrzati razvoj sistema koji povezuju SM i Internet.

U radovima [Fairgrieve09, Priyantha08, Botts08] za implementaciju SWE okvira, odnosno servisa koje pruža SWE, korišten je SOAP/WSDL implementacija SOA arhitekture, dok je za razmjenu podataka između korisnika i sistema korišten XML. Iako predloženo rješenje uvodi znatne koristi, ono zbog svoje "težine" (*heavy-weight*) ne predstavlja optimalnu implementaciju. Da bi se omogućila neophodna operativnost između heterogenih čvorova i pristupačnih usluga, a pored toga postigla i maksimalna optimizacija, autori u [Rouached12] predlažu adaptaciju SWE okvira i njegovo prilagođenje RESTful arhitekturi kroz dvije

karakteristike servisa (Slika 2.2.3.1.1.): model interfejsa i model informacija (kodiranje podataka).

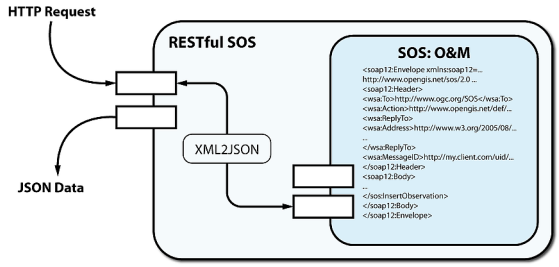


Slika 2.2.3.1.1. RESTful arhitektura za SWE [Rouached12]

Autori su tako razvili REST interfejse za svaku SWE uslugu (SOS, SAS, SPS i WNS), koji enkapsuliraju postojeće implementacije serverskih aplikacija i rade kao proksi za pristup podacima (Slika 2.2.3.1.2.).

U sklopu predloženog rješenja (Slika 2.2.3.1.1.) svaki HTTP klijent može predstavljati servis ili aplikaciju koja zahtijeva senzorske podatke/metapodatke ili događaj. Zahtjev se prenosi preko okvira

određenom senzorskom čvoru, koji nakon toga izdaje odgovor kroz različite OGC servise. Svaki senzorski čvor ima jedinstven ID preko kojeg se pridružuje mreži, šalje i prima podatke pomoću SWE standarda (SOS, SPS, SRS i WNS) i REST operacija (Get, Post, Put, Delete, itd.).



Slika 2.2.3.1.2. RESTful SOS implementacija [Rouached12]

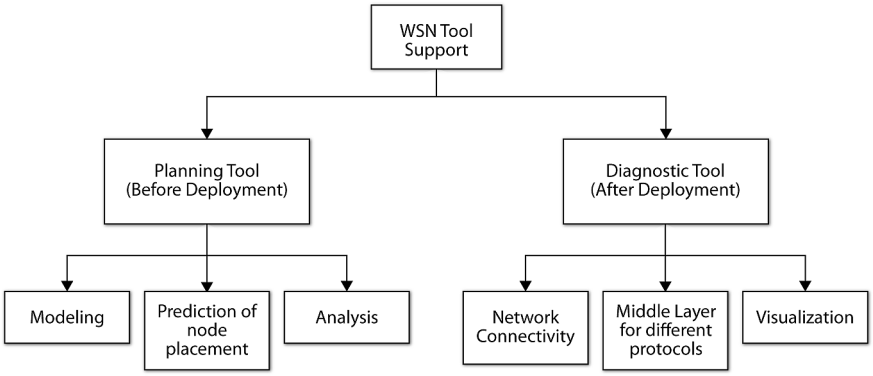
Web Register Service (WRS) je posrednik za povezivanje klijenta i servisa i opslužuje *GetRecords* zahtjev, te vraća dokument u JSON formatu koji sadrži krajnje tačke URL svih postojećih servisa koji zadovoljavaju postavljeni upit. U tipičnom scenariju, poslije dobijanja URL-a servisa, klijent šalje zahtjev *GetFeasbility* ka SPS-u sa željenim parametrima, nakon čega, ako je zadatak izvodljiv, prosljeđuje se GET/POST zahtjev prema željenom čvoru. SOS počinje prikupljanje zapažanja i skladištenje podataka i nakon što je zadatak završen, SPS obavještava/upozorava korisnika da su traženi podaci (događaj) dostupni (izvršen) i spremni za korištenje.

Predloženo implementaciono rješenje SWE okvira (RESTful arhitektura i JSON format podataka) i činjenica da se IoT danas može poistovjetiti sa ROA arhitekturom, predstavlja znatan napredak u povezivanju SM sa Internetom, pri čemu je znatno smanjeno vrijeme komunikacije, količina prenesenih podataka, a povećana efikasnost cjelokupnog sistema. U radu [Randrian12] ovaj pristup je iskorišten u svrhu kreiranja sistema za praćenje podzemnih voda.

2.3. Alati za projektovanje senzorskih mreža

Primjena BSM najčešće zahtjeva veliki broj senzora, što znatno povečava kompleksnost projektovanja i implementacije mreža i CBIS koji ih opslužuju. Modelovanjem parametara BSM (ponašanje, topologija, rutiranje, i dr.) prije implementacije i raspoređivanja senzorskih čvorova, je od ključnog značaja za efikasan razvoj i smanjenje troškova

projektovanja. Pored parametara BSM-a, također je potrebno poznavati i realno okruženje u kojem će biti raspoređeni čvorovi mreže (vazduh, tečnost, fizičke strukture) kako bi se dobila detaljna slika (percepcija) o broju čvorova, njihovoj fizičkoj lokaciji, ponašanju i povezanosti [Jurdak04, Ray09]. U sam proces modelovanja, danas su uključeni softverski alati, koji pružaju podršku u skoro svim aspektima projektovanja BSM, a dijele se na alate za [Ray09] (Slika 2.3.1.):

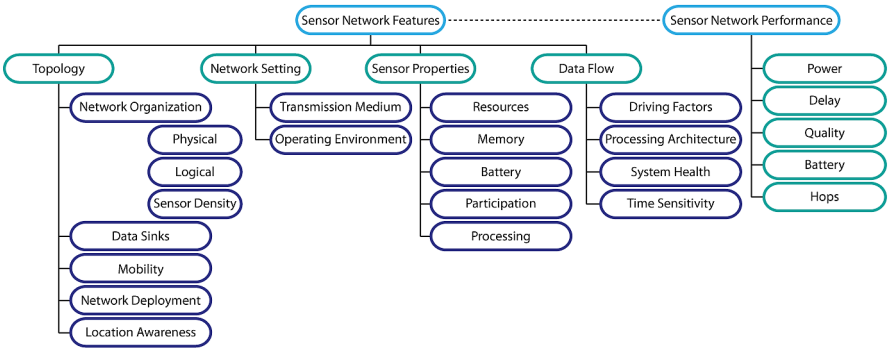


Slika 2.3.1. Taksonomija alata ta podršku projektovanja i analize BSM [Ray09]

- *Planiranje* – planiranje smanjuje vrijeme i troškove dizajna BSM prije njihovog raspoređivanja, a obuhvata funkcije:
  1. *modelovanja* – predstavlja najvažniji element u procesu planiranja, a opisuje senzorske čvorove (hardverske i softverske karakteristike), okruženje u kojem se senzorski čvorovi nalaze, tip senzorske mreže i dr.,
  2. *predviđanja rasporeda čvorova* – proces planiranja koji predviđa optimalan broj čvorova BSM i poziciju na kojima će čvorovi biti postavljeni u okruženju, i
  3. *analize kreiranog plana* – analiza ponašanja ukupne mreže i provjera da li je kreirani plan optimalan (raspored, potrošnja, povezanost, i dr.).
- *Dijagnosticiranje* – se koristi za praćenje stanja BSM nakon raspoređivanja senzorskih čvorova i puštanje mreže u eksploataciju, čime se osiguravaju korektivni faktori mreže, a obuhvata funkcije:
  1. *mrežne povezanosti* – analizira povezanost senzorskih čvorova sa drugim čvorovima, *gateway* uređajima te IS,

2. *adaptera za različite protokole* – transformaciju podataka u određene forme, kako bi bili korišteni od strane različitih protokola,
3. *vizuelizaciju* – grafički prikaz topologije mreže i prikupljenih rezultata u svrhu detaljne analize.

U radu [Jurdak04] je predložen okvir za modelovanje SM baziran na osnovu opštih karakteristika koje su identifikovane analizom postojećih SM. Definišući ontologiju, okvir tako klasifikuje SM prema željenim karakteristikama, pri čemu pruža skup ciljeva koji moraju biti ispunjeni da bi se dobile optimalne performanse (Slika 2.3.2.).



Slika 2.3.2. Predložena ontologija SM [Jurdak04]

Predložena klasifikacija obuhvata samo određeni skup elemenata koji imaju znatan uticaj na rad, razvoj i projektovanje SM i u zavisnosti od potrebe može biti dodatno proširena.

Autori u radovima [He12, Guinard09a, Wehrle10] predlažu modelovanje niza parametara BSM, pa tako za modelovanje topologije BSM u [He12] su definisane dvije osnovne komponente koje moraju biti razmatrane prije dizajna ili optimizovanja mreže. Prva predstavlja specifikaciju okruženja u kojem će se mreža nalaziti, dok druga opisuje ulogu senzorske mreže u okruženju. U radu [Guinard09a] su razmatrani principi i alati za modelovanje i simulaciju rasporeda čvorova, dok su u [Wehrle10] izvršene analize modelovanja WWW saobračaja (HTTP i *File Transfer Protocol* (FTP)) kao i Internet mrežne topologije. Alati za simulaciju BSM danas predstavljaju najšire rasprostranjene alate u procesu projektovanja, proučavanja, testiranja aplikacija i protokola BSM. U radovima [Ammari14, Sundani11, Siraj12, Weingärt09, Sharma14, Mishra14, Gupta13, Chhimwal13, Abuarqoub12, López05] na osnovu različitih aspekata je izvršena detaljna analiza komercijalnih i slobodno

dostupnih alata za simulaciju BSM, pa su tako najčešće korišteni simulatori: *NS-2*, *NS-3*, *TOSIM*, *GloMoSim*, *UWSim*, *Avrora*, *SENS*, *COOJA*, *Castalia*, *Shawn*, *EmStar*, *J-Sim*, *SENSE*, *VisualSense*, *OPNET*, *NetSim*, *OMNeT++*, *QualNet*. Navedeni simulatori se razlikuju po funkcionalnosti, ograničenjima i implementacionom programskom jeziku (C, C++, Java, nesC). Oslanjajući se na korisnički orijentisan pristup, u radu [Lee11] je predstavljen sistem za automatizovano konfigurisanje SM, baziran na principu pitanja i odgovora, dok je u radovima [Maissa12, Maissa13] predloženo korištenje DSL-a u procesu modelovanja i formalne verifikacije BSM pod nazivom *VeriSensor*.

Pozicioniranjem ciljeva disertacije u okviru predstavljenih alata za projektovanje senzorskih mreža moguće je zaključiti da je za unapređenje procesa razvoja arhitekture sistema baziranih na SW mrežama, uz oslonac na dinamičko generisanje servisnog sloja, neophodno razviti skup alata koji obezbjeđuju planiranje i dijagnosticiranje u okviru kreiranih SW mreža. Tri osnovne smjernice u razvoju predloženog skupa alata na kojima se zasniva praktični dio disertacije su:

1. Oslonac na principe MDA i DSM-a što zahtjeva izbor odgovarajuće razvojne platforme i razvojnog okvira koji podržava definisanje domenski specifičnog jezika ali i pratećeg skupa alata (grafičkog editora, generatora kôda, itd.).
2. Mogućnost formulisanja ograničenja koja projektovana arhitektura mora ispoštovati.
3. Obezbjeđenje adekvatne softverske podrške za kreiranje modela (tekstualni ili grafički alati) i njihovu automatsku transformaciju u izvršni oblik (automatsko generisanje kôda).

U daljem tekstu izvršena je elaboracija osnovnih referenci koje se odnose na izbor platforme za modelovanje, mehanizme ograničenja, podloge za automatsko generisanje kôda i grafičkih okvira za kreiranje podrške MDA procesu.

2.3.1. Eclipse Modeling Framework (EMF)

Na osnovu postavljenih zahtjeva, proširenih sa dostupnošću, proširljivošću i jednostavnošću korištenja, primarni izbor predstavlja *Eclipse* [Eclipse] razvojna platforma koja je posljednjih godina postala *de facto* standard u modelom upravljanom razvoju softvera [Kolovos10]. *Eclipse* je univerzalna višenamjenska platforma, koja predstavlja proširivo, slobodno dostupno, *integrisano razvojno okruženje – Integrated*

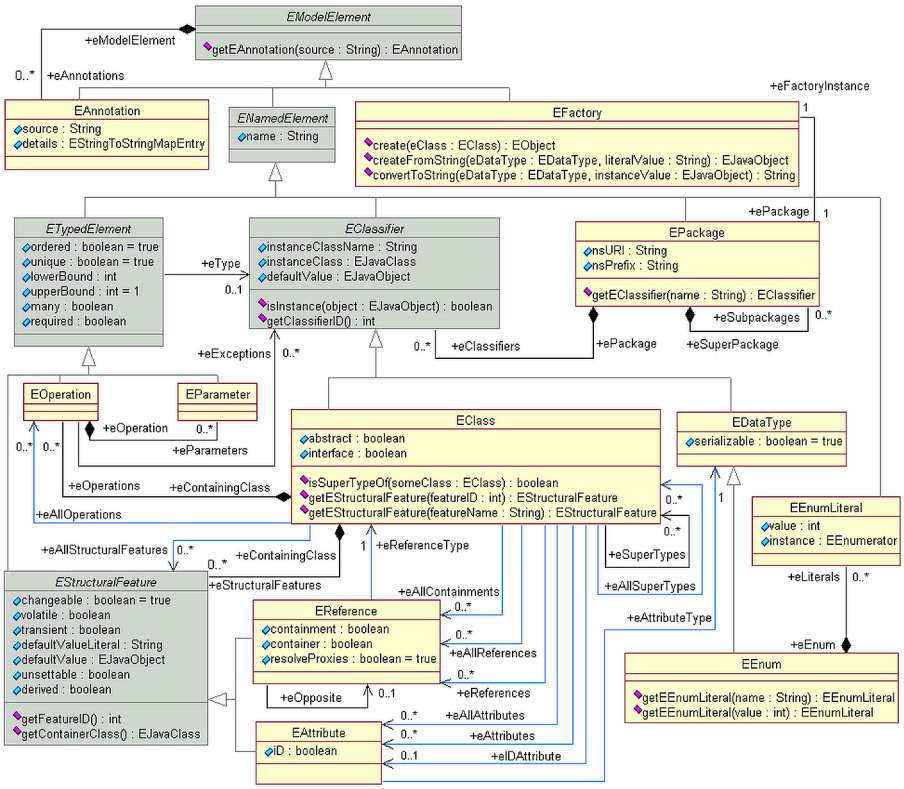
Development Environment (IDE) [Steinberg08, Rivieres04] i kao takva se može smatrati okvirom za kreiranje drugih integrisanih razvojnih okruženja. Definišući osnovnu strukturu IDE-a, *Eclipse* posjeduje veliki broj ugrađenih generičkih funkcionalnosti, dok sve ostale uvodi preko modula koji se nazivaju dodaci (*plug-ins*). *Eclipse Modeling Project* (EMP) [EclipseEMP, Steinberg08] predstavlja ključni element u *Eclipse* razvojnim tehnologijama zasnovanim na modelima, pružajući jedinstven skup okvira za modelovanje, alata i implementacionih standarda. Jezgru EMP-a definiše *Eclipse Modeling Framework* (EMF) koji pruža okvire za modelovanje i generisanje kôda za *Eclipse* aplikacije, bazirane na strukturiranim modelima podataka [Eclipse, Steinberg08, Moore04, Biermann06, Ma13, Jörges13]. EMF je započet kao implementacija MOF standarda od strane OMG-a, a danas podržava *Essential* MOF (EMOF) standard koji je dio OMG MOF 2.0 specifikacije [Moore04]. Jezgro ili metamodel koji se koristi za predstavljanje modela u EMF-u jeste *ECore* model koji odgovara metamodelu u EMOF standardu [Steinberg08, Biermann06, Ma13]. *ECore* predstavlja refleksivan metamodel [Jörges13] koji se često naziva i meta-metamodelom [Steinberg08]. Refleksivnost predstavlja bitan element za fleksibilnost *ECore* modela i dozvoljava mu primjenu na višestrukim MOF meta-nivoima (M2, M3, M4) [Jörges13]. Oslanjajući se na UML klasne dijagrame i Java koncepte, na slici 2.3.1.1. je prikazan dijagram klasa *ECore* meta-metamodela.

Na osnovu *ECore* modela, EMF može generisati četiri različita elementa u Java kôdu koji obezbjeđuju podršku za rad sa kreiranim metamodelom [Jörges13]:

- *Model code* – kreira Java implementaciju modela oslanjajući se na *ECore* API,
- *Edit code* – obezbjeđuje API-je za editovanje instanci modela koji su nezavisni od korisničkog interfejsa,
- *Editor code* – kreira proširenje *Eclipse* platforme sa jednostavnim editorom modela koji je baziran na hijerarhijskom prikazu,
- *Test code* – obezbjeđuje skup testova za testiranje modela.

Analizirajući odnos *ECore*-a i EMF-a, može se reći da [Steinberg08]:

- *ECore* sa XMI serijalizacijom modela predstavlja jezgro EMF-a,
- *ECore* model može biti kreiran na osnovu UML modela, XML šeme ili interfejsa sa Java anotacijama,
- da je na osnovu *ECore* modela moguće generisati implementaciju u Java programskom jeziku.



Slika. 2.3.1.1. Dijagram klasa ECore met-metamodela [EclipseEMF]

Kelly je u radu [Kelly04] izvršio uporednu analizu *Eclipse* IDE okruženja, uz oslonac na EMF, sa komercijalnim MetaEdit+ okruženjem, dok je u radu [Pelechano06] izvršena analiza sa Microsoft-ovim "*DSL Tools*" alatom. U oba slučaja se može zaključiti da *Eclipse* IDE sa EMF-om predstavlja ozbiljnu konkurenciju komercijalnim proizvodima (u slučaju "*DSL Tools*"-a *Eclipse* IDE sa EMF-om se pokazao i kao znatno bolji). Razvojem proširenja za *Eclipse* IDE i nadogradnje EMP-a, danas se ova slika znatno promijenila, pružajući korisnicima kompletno razvojno okruženje za kreiranje DSM rješenja.

U radu [Ma13] su predstavljene matematičke definicije svih elemenata koji participiraju u EMF-u i njegovim modelima i metamodelima, dok je primjena EMF-a kroz različite projekte, data u radovima [Steinberg08, Bull05, Pedro08, Shani10, Bender14].

2.3.2. Object Constraint Language (OCL)

Modeli se danas koriste u skoro svim fazama razvoja softvera, ali oni sami najčešće nisu dovoljni da bi opisali sve relevantne elemente problema, kao što su dodatne informacije, integritet i/ili ograničenja između elemenata modela [OCL14, Tan10]. Ovaj problem proizilazi iz ograničenog broja grafičkih elemenata i njihove semantike, koje mogu predstaviti samo određene podskupove informacija iz domena problema [Cabot12]. Koristeći prirodni jezik za proširenje opisa modela, kroz praksu se pokazalo kao veoma neefikasno rješenje radi unošenja dodatnih nejasnoća, dok je primjena tradicionalnih formalnih jezika bila uslovljena odličnim poznavanjem matematike. Da bi se riješio nametnuti problem, bilo je potrebno razviti formalne jezike koji su imali snagu tradicionalnih formalnih jezika, ali i lakoću korištenja od strane korisnika koji ne posjeduju ekspertska znanja iz matematike [OCL14]. 1995. godine IBM uvodi *Object Constraint Language* (OCL) [Warmer03], koji 2000 godine postaje dijelom UML specifikacije, a *predstavlja jezik za formalnu specifikaciju koji je korišten da obezbijedi dodatne opise i ograničenja UML modela* [Tan10]. Inicijalno, OCL se koristio samo kao proširenje UML jezika, međutim, danas on predstavlja ključnu komponentu za svaki MDE pristup razvoja softvera kao jezik za specifikaciju svih vrsta upita, manipulacije i zahtjeva nad modelima i metamodelima [Cabot12, Warmer03], a često i za opisivanje pravila za transformaciju modela [Cabot12]. Bitno je pomenuti tri osnovne karakteristike OCL jezika, a to su [OCL14]:

- OCL predstavlja jezik za specifikaciju i njegovi izrazi ne mogu imati dodatne efekte – izvršavanjem OCL izraza ne može se izvršiti promjena modela, već samo dohvatanje vrijednosti kao rezultat njegovog izvršavanja.
- OCL nije programski jezik, pa tako uz pomoć njega nije moguće pisati programsku logiku ili kontrolu programa – kako OCL, na prvom mjestu, predstavlja jezik za modelovanje, njegovi izrazi se ne mogu direktno izvršavati.
- OCL predstavlja tipizirani jezik, pa tako svaki OCL izraz mora imati definisani tip podataka – da bi se ispunio ovaj uslov, OCL izrazi moraju biti u skladu sa pravilima i predefinisanim tipovima podataka.

Zbog svoje raširenosti, primjena OCL-a je znatno evoluirala od svog nastanka 1995. godine, pa se danas koristi i za/kao [OCL14, Cabot12]:

- jezik za postavljenje upita,
- specifikaciju invarijanti (nepromjenjivosti) klasa i tipova u klasnom modelu,
- specifikaciju invarijanti (nepromjenjivosti) tipova stereotipova,
- opis post- i pred- uslova za operacije i metode,
- opis ograničenja/validaciju,
- specifikaciju poruka i akcija,
- specifikaciju ograničenja na operacijama i
- specifikaciju izvedenih pravila atributa modela.

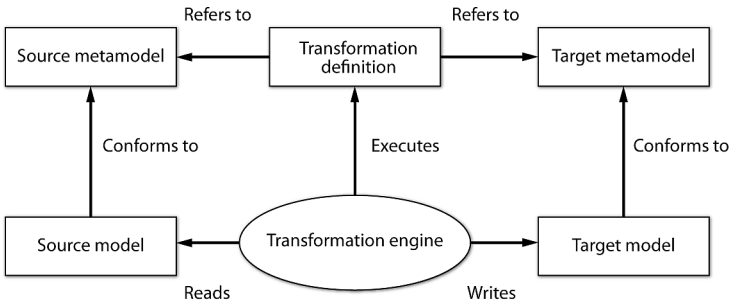
U radovima [OCL14, Richters98, Warmer03, OCL03, Cabot12] je prikazana formalana i matematička definicija OCL jezika, kao i njegova detaljna specifikacija kroz tekstualne i matematičke opise sintakse, dok je sama primjena OCL-a u modelima razmatrana u radovima [Heidenreich11, Damus14] i predstavljena je kao interna (OCL pravila su ugrađena u sam model) ili eksterna (OCL pravila su definisana u dodatnom fajlu). Alati za rad sa OCL-om su podijeljeni na one koji postoje kao samostalne aplikacije i one koji su dijelom integrisanih razvojnih okruženja [Cabot12, Damus14, Richters01, Heidenreich11]. U oba slučaja alati pružaju podršku korisnicima i ubrzanje pisanja OCL pravila, kroz sintaksne analizatore kao i alate za verifikaciju i validaciju kreiranih izraza [Cabot12, Damus14, Richters01, Heidenreich11]. Značajno je pomenuti integraciju OCL editora i *OCLinECore* jezika sa Eclipse razvojnom platformom, koja uz EMF obezbjeđuje kompletno okruženje za kreiranje DSL-ova [Damus14].

2.3.3. Acceleo

Automatsko generisanje kôda i transformacija modela predstavlja jedan od osnovnih elemenata, kako MDA tako i DSM pristupa razvoja softvera [Kelly08, Czarnecki06, Völter06]. Sve većom primjenom pomenutih paradigmi u svakodnevnom procesu razvoja, neophodno je osigurati adekvatan pristup, metode i alate koji mogu obezbijediti programsku podršku za ispunjenje ovog cilja. Analizom radova [Kelly08, Ledo10, Rose12, Jörges13, Herrington03] može se doći do zaključka da se transformacija primarno može koristiti za generisanje programskog kôda i dokumentacije, serijalizaciju modela (konvertovanje u oblik pogodan za razmjenu) i vizuelizaciju modela. Također se primjenom transformacije modela u procesu razvoja softvera širi i područje djelovanja na [Czarnecki06]:

- generisanje modela nižih nivoa apstrakcije na osnovu modela viših nivoa apstrakcije,
- preslikavanje i sinhronizacija modela na istom ili različitom nivou apstrakcije,
- kreiranje pogleda na sistem koji su bazirani na upitima,
- evolucija modela, kao što je refaktorisanje,
- reverzni inženjering modela višeg nivoa u modele nižeg nivoa apstrakcije.

Oslanjajući se na modele i metamodele koji definišu njihovu apstraktnu sintaksu, u [Czarnecki06] je predložen osnovni koncept transformacije (Slika 2.3.3.1) baziran na pravilima za preslikavanje i mašini za transformaciju.



Slika. 2.3.3.1. Osnovni koncepti transformacije modela [Czarnecki06]

Radovi [Czarnecki06, Jörges13, Herrington03] daju detaljne analize tipova generatora i tehnike generisanja kôda koje su bazirane na poznatim tehnikama i dostupnim alatima za generisanje kôda. Tako se u [Jörges13] tehnike generisanja kôda dijele na:

- *"Programirane" generatore kôda – Programming the Code Generator* – predstavljaju generatore kôda koji su nastali direktnom primjenom jezika opšte namjene na izvorni model, pri čemu se eksplicitno implementira preslikavanje iz izvornog u odredišni model/jezik.
- *Šablonski bazirane generatore kôda – Template-Based Code Generator* – predstavljaju generatore kôda koji su bazirani na primjeni šablona koji sadrže statičke dijelove sa ugrađenim dinamičkim elementima. Najveći broj generatora danas je baziran na ovoj tehnici generisanja kôda.
- *Generatore kôda bazirane na skupovima transformacionih pravila – Rule-Based Code Generator* – definišu se skupovi pravila na osnovu

kojih će biti izvršeno preslikavanje elemenata (iz izvornog modela/jezika u krajnji model/jezik)

Ledo i dr. u svom radu [Ledo10] konstatuju da je za uspješnu primjenu transformacije u MDA procesu potrebno utvrditi stepen transformacije, odnosno dvije osnovne transformacije:

- model-u-model transformaciju – *model-to-model* (M2M) koja transformiše model iz jednog domena, u model u nekom drugom domenu,
- model-u-tekst transformaciju – *model-to-text* (M2T) koja transformiše model u skup tekstualnih elemenata (programski kôd, dokumentaciju ili specifikaciju).

U radovima [Ledo10, Rose12, Ogunyomi14, Czarnecki06, André14, Völter06] su analizirane M2T tehnike za transformaciju izvornog modela/jezika u izvršni kôd sistema ili njegove konkretne dijelove, ali kao najbitnija činjenica navodi se da za M2T transformaciju nije potrebno poznavati odredišni metamodel. Ovakav pristup znatno pojednostavljuje proces transformacije i povećava opseg primjene M2T tehnike. 2008. godine u svrhu izbjegavanja neodređenosti i postizanja uniformnosti alata i tehnika za transformaciju (generisanje) modela/jezika, OMG definiše standard M2T (MOF *Model-to-Text Transformation Language*, MOFM2T) koji predstavlja specifikaciju jezika za transformaciju modela [Rose12, Seo14, Ogunyomi14], a osigurava smjernice za kreiranje alata za transformaciju (generatora) koji su bazirani na šablonima.

Danas je na tržištu prisutan veliki broj alaza za M2T transformaciju, ali jedan od vodećih je *Acceleo* [Acceleo], koji za razliku od drugih alata nastoji u potpunosti ispuniti sve postavljene ciljeve MOFM2T standarda [Rose12, Völter06, López13, Buchmann13]. *Acceleo* predstavlja alat za M2T transformaciju sa primarnim ciljem kreiranja generatora kôda. Za generisanje kôda, *Acceleo* zahtjeva izvorni metamodel i model koji je usklađen sa pomenutim metamodelom, dok odredišni metamodel nije potreban [André14]. Kao osnovna prednost *Accelea* može se navesti njegova integracija sa *Eclipse* platformom, što obezbjeđuje jednostavnost instalacije, primjene i dodatnu podršku postojećih alata [André14, Juliot06], kao i korištenje *Acceleo* sintakse za kreiranje šablona za generisanje kôda [André14]. Kao ulazne podatke *Acceleo* najčešće koristi metamodele i modele koji su definisani koristeći EMF, ali to mogu biti i bilo koji metamodeli i modeli nastali korištenjem jezika za transformaciju Atlas - *Atlas Transformation Language* (ATL) [André14], dok kao izlaz

mogu biti generisane bilo koje tehnologije (Hibernate, Struts, Spring, .Net, Php, itd.) [Juliot06] ili obične tekstualne datoteke.

Acceleo je podijeljen u tri osnovna elemenata [Juliot06]:

- *Čitanje – Acceleo* ne predstavlja alat za modelovanje, ali može čitati model iz bilo kojeg alata koji ispunjava XMI standard.
- *Editovanje – Acceleo* obezbjeđuje okruženje za editovanje (*Editor šablona – Eclipse* orijentisano okruženje sa: markiranjem sintakse, predikcijom kôda i dr.; *Reflektivni editor* – editor koji obezbjeđuje pregled generisanog kôda u realnom vremenu.).
- *Izvršavanje* – predstavlja osnovni dio *Accelea*, a služi za generisanje kôda na osnovu kreiranog šablona.

U radovima [André14, Juliot06] su izvršene određene analize *Accelea* (okruženje, sintaksa, primjena), pa se kao pozitivno može izdvojiti: integracija sa *Eclipse* okruženjem, korištenje metamodela čime se izbjegavaju kompleksni mehanizmi parsiranja i jednostavna sintaksa koja osigurava čitljivost šablona za generisanje, dok se kao negativno, najviše ističe nedostatak podrške, te manjak dokumentacije i određenih funkcionalnosti. Kroz radove [Son13, Buchmann13, Seo14, Völter06, Guana14, López13] je prikazana primjena *Accelea* u različitim oblastima, pa se može donijeti zaključak da *Acceleo* predstavlja jedan od vodećih M2T alata zbog svoje fleksibilnost i jednostavnosti korištenja.

2.3.4. Graphical Modeling Framework (GMF)

Graphical Modeling Framework (GMF) se prvi put pojavljuje 2006. godine, a danas predstavlja jedan od osnovnih podprojekata EMP-a [Shatalin06]. GMF osigurava generativnu komponentu (*GMF Tooling*) i izvršnu infrastrukturu (*GMF Runtime*) za razvoj grafičkih editora na osnovu EMF i GEF okvira [Herrm11, Seehusen11, Ruscio10, Shatalin06, Gouyette06] i predstavlja najbolji primjer MDA arhitekture, striktno razdvajajući PIM, PSM i implementacioni kôd [Herrm11, Herrm10]. U kontekstu GMF-a, EMF se koristi za definisanje metamodela ili apstraktne sintakse jezika (opisane *ECore* modelom) te za generisanje kôda neophodnog za kreiranje, uređivanje i pristupanje modelu, dok GEF pruža podršku za implementaciju i uređivanjue konkretne grafičke sintakse jezika [Seehusen11].

Da bi se razvio potpuno funkcionalan grafički editor u GMF-u, neophodno je kreirati određeni broj modela [Gronback09, Ruscio10, Shatalin06, Gouyette06, Pedro08, Madhuram07]:

- *Domenski model* (*Domain model*) – predstavlja *ECore* bazirani metamodel (apstraktnu sintaksu) jezika za čiju reprezentaciju i editovanje se kreira grafički editor. Sadrži sve koncepte i relacije koje trebaju biti implementirane unutar editora.
- *Model grafičkih elemenata* (*Graphical definition model*) – sadrži informacije o grafičkim elementima koji će biti prikazani u editoru, ali ne i vezu sa elementima u domenskom modelu koje predstavljaju.
- *Model alata* (*Tooling definition model*) – definiše alate potrebne za rad sa editorom, kao što su: alatne trake, meni, palete i dr.
- *Model preslikavanja* (*Mapping model*) – predstavlja model koji sadrži veze između grafičkih elemenata i definicija alata sa domenskim modelom.

Nakon što se definišu svi modeli preslikavanja, GMF kreira model generisanja (*generator model*) koji obuhvata sve informacije neophodne za kreiranje potpuno funkcionalnog grafičkog editora [Pedro08].

U radovima [Buchmann07, Gronback09, Seehusen11, Gouyette06, Pedro08, Almend09, Refsdal11] je kroz primjenu GMF-a predstavljeno kreiranje grafičkih editora baziranih na metamodelima, dok je u [Kolovos10, Kolovos15] dat prijedlog za smenjenje kompleksnosti i značajno poboljšanje produktivnosti, kvaliteta i održavanja rješenja baziranih na EMF i GMF tehnologiji.

2.3.5. Graphiti

Od 2010. godine, *Graphiti* okvir postaje dijelom *Eclipse* projekta i osigurava infrastrukturu za kreiranje grafičkih editora [Brand07, Buchmann14, Stritzke13, Refsdal11]. Poput GMF-a, i *Graphiti* okvir je zasnovan na GEF i *Draw2D* okvirima, ali se njihovo postojanje u potpunosti skriva iza *Graphiti* API-ja [Stritzke13, Filippelli12]. Za razliku od GMF-a, čiji je pristup zasnovan na modelima, *Graphiti* se oslanja na Java API-je za kreiranje grafičkih editora [Buchmann14, Stritzke13, Filippelli12] uvodeći novu paradigmu zvanu *funkcije* (*features*) [Refsdal11]. Za predstavljanje grafičkih elemenata, *Graphiti* okvir kreira *Pictogram model* koji sadrži grafičke reprezentacije svih objekata iz domenskog modela i za svaki od njih osigurava metode za [Stritzke13]:

- *dodavanje* (*add*) – određuje šta se dešava prilikom dodavanja novog elementa. Najčešće se iscrtava novi grafički objekat i povezuje sa domenskim objektom.

- *kreiranje* (*create*) – određuje šta se dešava prilikom kreiranja novog domenskog objekta koristeći paletu alata. Najčešće se kreira i domenski objekat i njegov grafički prikaz.
- *brisanje* (*delete*) – određuje šta se dešava kada se domenski objekat ili grafički prikaz objekta obrišu,
- *prikaz* (*layout*) – određuje kako se ponaša grafički prikaz objekta prilikom pomjeranja ili promjene veličine,
- *prilagođene opcije* (*custom*) – dodaje prilagođene opcije u kontekst meni grafičkog objekta.

Korištenjem *Graphiti* okvira ostvaruju se značajne prednosti prilikom kreiranja grafičkih editora [Brand07]:

- početne prepreke: platformski zavisne tehnologije (GEF/*Draw2D*) su skrivene,
- inkrementalan razvoj: brza početna realizacija pomoću zadanih implementacija, kratak razvojni ciklus (razvoj i provjera),
- homogeni izgled editora: svi editori kreirani u *Graphiti* okviru izgledaju i ponašaju se slično,
- opciona podrška različitim platformama: dijagrami su definisani kao platformski nezavisni i mogu biti prikazani na bilo kojoj platformi.

Poredeći GMF i *Graphiti* okvir, autori radova [Refsdal11, Brand07, Stritzke13] su došli do zaključka da se *Graphiti* okvir za većinu slučajeva pokazao kao optimalno, ali zbog velikog broja metoda koje je potrebno implementirati za svaki od domenskih objekata, također i kompleksno rješenje. Oslanjajući se na *Spray* [Spray] okvir, koji predstavlja tekstualni DSL za opis grafičkih reprezentacija objekata i generisanjem neophodnog kôda za implementaciju, znatno se pojednostavljuje i ubrzava proces razvoja *Graphiti* editora [Stritzke13, Filippelli12]. *Spray* okvir osigurava tri različita DSL-a za opis: oblika i veza (*Shape DSL*), stilova dijagrama i pojedinih elemenata (*Style DSL*) te odnosa između domenskog modela i grafičkih elemenata (*Spray Core DSL*) [Stritzke13, Filippelli12, Warmer13].

U radovima [Porten12, Ramas14] je ilustrovana primjena *Graphiti* okvira za kreiranje realnih sistema, dok je primjena *Spray*-a prikazana u radu [Filippelli12].

2.3.6. Sirius

Sirius okvir, koji enkapsulira GMF, koristi se za kreiranje, vizuelizaciju i editovanje modela koristeći interaktivni editor pod nazivom "*modelers*" [Sirius, Vujović14, Vujović14a, Vujović14b]. Za razliku od GEF-a, GMF-a i *Graphiti* okvira, za kreiranje osnovnog editora koristeći *Sirius* okvir, nije potrebno prethodno znanje Java programskog jezika, *Eclipse* platforme ili GMF-a, što *Sirius* okvir čini optimalnim za razvoj od strane velikog broja korisnika [Juliot10]. U zavisnosti od vizuelne reprezentacije modela, Sirius podržava pet različitih dijalekta (tipova reprezentacije) [Sirius, Vujović14a]:

- *dijagrame* (grafičke modele),
- *dijagrame sekvence,*
- *tabele*,
- *matrice*, i
- *stabla* (hijerarhijski prikaz).

Novi dijalekti mogu biti kreirani programskom implementacijom [Sirius].

Oslanjajući se na *poglede* (*viewpoint*), *Sirius* okvir, kroz alate za specifikaciju pogleda, pruža mogućnost višestruke analize domenskog modela. Tako, zavisno od potrebe korisnika, moguće je za isti domenski model kreirati više reprezentacija organizovanih u poglede, koje naglašavaju različite elemente domena [Vujović14, Vujović14a, Vujović14b]. Za definisanje upita i ograničenja, *Sirius* okvir koristi *Acceleo* i *OCL* notaciju, ali zavisno od potrebe, također je dozvoljena i primjena Java programskog jezika, kao i prilagođenje implementacije na nivou GMF okvira.

Sumirajući primjenu, u radovima [Buchmann14, Vujović14, Vujović14a, Vujović14b, Juliot10] moguće je zaključiti da *Sirius* okvir pojednostavljuje proizvodnju, skraćuje vrijeme dizajna i povećava ukupnu produktivnost izgradnje domenski specifičnih grafičkih editora.

U [Sirius] su navedene osnovne prednosti *Sirius* okvira koje obuhvataju:

- oslonac na otvoreni i najčešće korišćeni EMF standard,
- mogućnost adaptacije sa proizvoljnim, EMF kompatibilnim, DSL-om,
- odvajanje semantike od reprezentacije modela,
- podršku za različite reprezentacije domenski specifičnih modela,

- jednostavnu upotrebu i brz razvoj,
- veliku mogućnost proširenja.

Poglavlje 3

Aspekti primjene MDA koncepta na razvoj arhitekture Senzor Web mreža

Oslanjajući se na tri osnovne cjeline koje su prepoznate u pogavlju 2. i servis orijentisane senzorske mreže, koje uvode nove koncepte i dobro definisane Web tehnologije za integraciju hardverskih elemenata i Interneta, otvara se problem izgradnje adekvatne infrastrukture koja obezbjeđuje unapređenje procesa razvoja arhitekture sistema baziranih na SW mrežama.

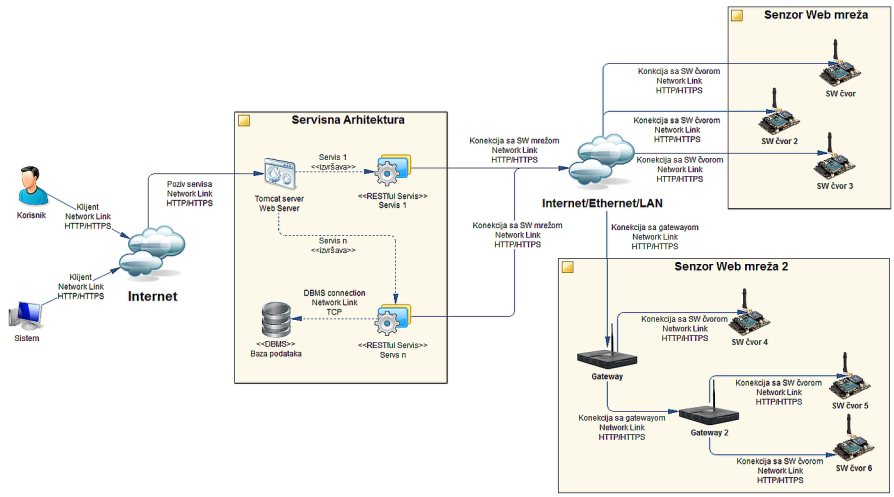
SW mreža se, kao surogat Interneta, u opštem slučaju može posmatrati kao nestrukturirana kolekcija gradivnih elemenata (senzorskih čvorova) koji su u stanju da uspostave korespondenciju sa fenomenima niskog nivoa apstrakcije (npr. fizikalni parametri) bez eksplicitno nametnute topologije. Mogućnost upotrebe SW mreža ograničena je jedino repertoarom i stepenom sofisticiranosti servisa koji dinamički formiraju virtuelne klastere pojedinačnih elemenata mreže. Sa tog aspekta, servisna arhitektura predstavlja ključ za praktičnu upotrebu SW mreža.

Projektovanje predložene arhitekture SW mreža, predstavlja kompleksan problem čije rješenje zavisi od tehnologije i raspoložive implementacione infrastrukture. Jedan od mogućih pravaca rješavanja predstavlja primjena MDA metodologije razvoja softvera koja se oslanja na princip podizanja nivoa apstrakcije i uvođenje novih implementacionih okvira, čime se obezbjeđuje podloga za automatizaciju procesa razvoja. Primjena MDA za rješavanje problema u različitim domenima podrazumijeva definisanje koncepta na različitim nivoima apstrakcije koji služe za izgradnju domenski specifičnih modela. Na osnovu toga, neophodno je izvršiti detaljnu analizu i ispitati mogućnost sinteze MDA pristupa i sistema za projektovanje SW mreža, odnosno prepozanti dijelove koji će predstavljati elemente za definisanje metamodela sistema. U nastavku

poglavlja je predstavljena infrastruktura SW mreža, i prikaz odabranih publikovanih rješenja.

3.1. Infrastruktura Senzor Web mreža

Na osnovu postavljenog cilja istraživanja, na slici (Slika 3.1.1.) je prikazana jedna od mogućih arhitektura SW mreža u formi infrastrukturnog dijagrama.



Slika 3.1.1. Infrastrukturni dijagram SW mreže

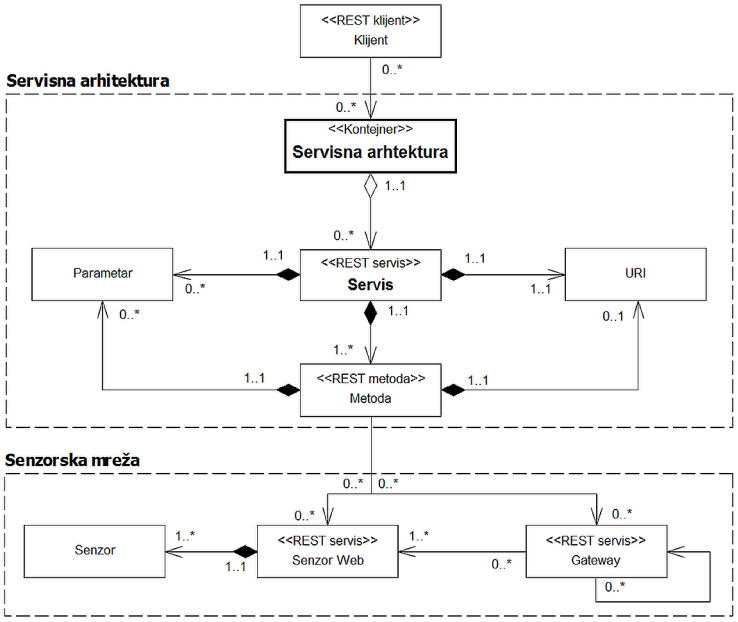
Infrastruktru predložene SW mreže čine:

- ***Klijentska strana*** – obuhvata klijente koji pristupaju servisima putem Interneta. Klijenti mogu biti nezavisni korisnici, ali isto tako i drugi softverski sistemi, čija priroda i implementacija sa aspekta arhitekture SW mreža mora biti u potpunosti transparentna.
- ***Servisna arhitektura*** – predstavlja implementaciju centralnog sistema za kreiranje virtuelnih klastera i prikupljanje podataka sa SW mreže, kao i njihovo skladištenje te obradu. Također osigurava dinamičku raspodjelu i rukovođenje SW elementima, te njihovo registrovanje, pristupanje i uklanjanje iz vrtuelnih klastera na zahtjev korisnika.
- ***Senzor Web mreža*** – predstavlja skup SW i *gateway* elemenata koji su povezani u određene logičke cjeline. Cjeline mogu predstavljati

pojedini SW elementi, ali i skupine pojedinih elemenata povezanih u hijerarhiju putem gateway uređaja.

- ***Gateway* i SW čvorovi** – predstavljaju osnovne gradivne elemente SW mreža kojima se pristupa putem IP adrese i implementiranog skupa servisa.

Predstavljeni elemenati mogu biti implementiran nezavisno korištenjem heterogenih tehnologija, a interakcija između njih se obezbjeđuje korištenjem unaprijed definisanih protokola i interfejsa. Oslanjajući se na zaključke koji su doneseni u poglavlju 2. optimalan izbor tehnologija za implementaciju SW mreža predstavlja ROA arhitektura, odnosno RESTful servisi. Na slici (Slika 3.1.2.) prikazan je model arhitekture predloženog rješenja uz oslonac na navedene tehnološke koncepte.



Slika 3.1.2. Arhitektura SW mreže

Zbog zahtjevane transparentnosti implementacije klijentske strane, fokus disertacije predstavljaju samo elementi neophodni za implementaciju SW mreže.

Predloženo rješenje spada u kategoriju kompleksnih sistema u skladu sa kriterijuma za ocjenu kompleksnosti softverskih sistema navedenim u [Alagar11], a koji obuhvataju:

- ***veličinu*** (*Size Complexity*) – veliki softverski sistemi se grade sa velikim brojem modula. Veličina takvog sistema se definiše u odnosu na broj modula, zahtjeve koji ih opisuju, njihovim interakcijama i ograničenjima u njihovom ponašanju,
- ***strukturu*** (*Structural Complexity*) – bazira se na dva aspekta: menadžment (rukovanje procesom izrade) i tehnička složenost. Kao mjera kompleksnosti strukture često se uzima i nivo interakcije (povezivanja) između modula sistema,
- ***okruženje*** (*Environmental Complexity*) – kompleksnost objekata i njihove interakcije sa okruženjem u kojemu se softverski sistem koristi,
- ***domen primjene*** (*Application Domain Complexity*) – kompleksnost znanja u domenskoj oblasti, i
- ***komunikaciju*** (*Communication Complexity*) – razmjena znanja i informacija između učesnika kao i njihove komunikacije u cilju obavljanja zadataka,

Podizanje nivoa apstrakcije, ako se može primjeniti, predstavlja najčešću i najefikasniju tehniku za smanjenje kompleksnosti softverskih sistema [Alagar11]. U situacijama kada to nije moguće, na primjer u slučaju okruženja i domena primjene, modelovanje predstavlja adekvatan odgovor. Uvođenjem modela domena koji enkapsulira znanje o domenu, njegovim primjenama, strukturi i mogućnosti ponovne upotrebe, moguće je znatno pojednostaviti domen.

Zbog kompleksne prirode predložene arhitekture SW mreža, adekvatan odgovor predstavlja primjena MDA pristupa na servisnu arhitekturu, SW mrežu, *gateway* i SW čvorove.

3.2. Analiza trenutno dostupnih rješenja

Servisna arhitektura i *senzorska mreža* (Slika 3.1.2.) predstavljaju ključne segmente predloženog rješenja za koje je neophodno obezbijediti podršku za modelovanje. Oba segmenta je potrebno predstaviti adekvatnim riječnikom domenskih pojmova metamodela koji opusuje REST arhitekturu i IoT koncepte. U nastavku su prikazane odabrane publikacije zasnovane na konceptima bliskim domenu teze.

3.2.1. Modelovanje REST koncepta

Danas su dostupni mnogi okviri za implementaciju RESTful servisa, ali pored toga i dalje postoji nedostatak podrške u ranim fazama procesa razvoja, posebno u fazama analize i dizajna [Schreier11]. Može se reći, da je za ovo dijelom odgovorno ne postojanje prihvaćenog standarda koji opisuje REST srvise, pa se u publikacijama najčešće mogu pronaći neformalni modeli koji su prilagođeni određenim situacijama. Generalno, metamodel za dizajniranje i implementaciju REST servisa još uvijek nedostaje.

U radu [Rodríguez13] su definisani koncepti i arhitektura za podršku REST servisima kao proširenje *Struts v1.3* okvira za razvoj Web aplikacija. Predloženi koncepti, koriste reverzni inženjering za dobijanje MVC arhitekture Web aplikacija nad kojima se naknadno dodaje sloj za podršku REST servisima. Predloženi metamodel predstavlja proširenje metamodela *Struts* okvira sa elementima REST arhitekture, a cjelokupno rješenje je nazvano MIGRARIA.

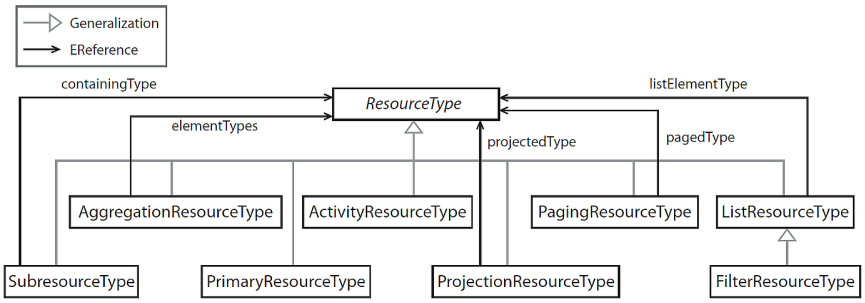
Schreibmann u svom radu [Schreib14] koristi UML kao meta-metamodel za kreiranje metamodela REST servisa, sa ciljem boljeg razumijevanja i uključivanja elemenata UML-a u fazi modelovanja. Autor predlaže, da se korištenjem definisanog metamodel-a, *Xtext* i *Xtend* okvira može generisati izvorni Java kôd za REST implementacione okvire (*Jersey*).

U [Valverde09] autori koriste MDA pristup u procesu integracije *Web 2.0* funkcionalnosti implementirane kao REST servisi. Formulisan je REST metamodel koji specificira REST servise kao konceptualne modele. Autori definišu REST servise kao agregaciju različitih resursa kojima se može pristupiti putem URI identifikatora. Na osnovu toga ROA predstavlja pogodnu arhitekturu za modelovanje REST servisa jer se fokusira na funkcionalnosti ili podatke koje pružaju servisi (resurs), a ne na njihove komunikacione interfejse [Valverde09]. Predloženi metamodel obezbjeđuje generički mehanizam za kreiranje formalne specifikacije, koja se može koristiti za automatsku transformaciju u drugu specifikaciju ili izvršni kôd.

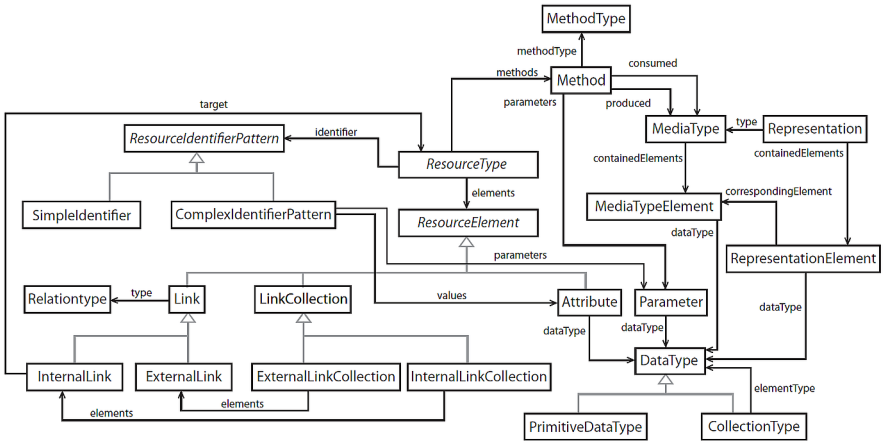
Oslanjajući se na tehnički aspekt i ROA arhitekturu, Schreier [Schreier11, Schreier12] predstavlja metamodel za dizajniranje i implementaciju RESTful servisa, koji je korišten kao referentni metamodel, i pomoć pri razumijevanju i poboljšanju RESTful servisa. Predloženi metamodel omogućava tipizaciju resursa (Slika 3.2.1.1.),

modelovanje statičke strukture i ponašanja REST servisa [Schreier11, Porten12]. Metamodel obuhvata:

- ***model strukture*** (*structural model*) – opisuje tipove resursa, njihove atribute, relacije, interfejse i reprezentacije (Slika 3.2.1.2.), i
- ***model ponašanja*** (*behavioral model*) – opisuje ponašanje u formi konačnog automata (*state machine*) (Slika 3.2.1.3.).

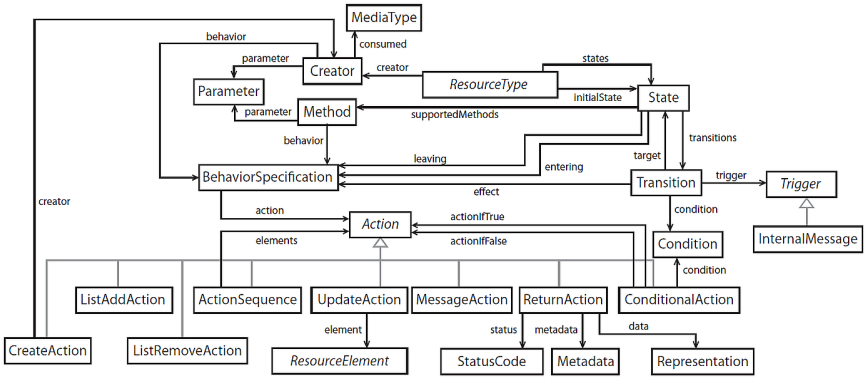


Slika 3.2.1.1. Hijerarhijski prikaz resursa metamodela [Schreier11]



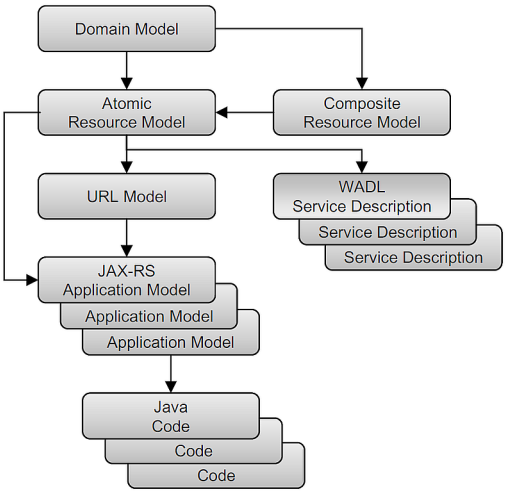
Slika 3.2.1.2. Metamodel strukture [Schreier11]

Schreier također predlaže, da se nakon verifikacije metamodela kreiraju tekstualni i grafički jezici kao i generatori kôda, dokumentacije i klijentskih aplikacija u različitim jezicima, okvirima i formatima [Schreier11, Schreier12]. Opisani metamodel predstavlja okosnicu za daljnji razvoj REST MDA pristupa.



Slika 3.2.1.3. Metamodel ponašanja [Schreier11]

U radu [Haupt14], oslanjajući se na Schreier-ine metamodele, predložen je višeslojni MDA pristup za modelovanje REST servisa koji odvaja pojedinačne nadležnosti slojeva u zasebne modele. Predloženo rješenje definiše šest modela (Slika 3.2.1.4.), odnosno: ***domenski model*** (*Domain Model*), ***model resursa*** (*Composite and Atomic Resource Model*), ***opis servisa*** (*Service Description*), ***URL model*** (URL Model) i ***model aplikacije i kôda*** (Application Models and Code) .



Slika 3.2.1.4. Slojeviti metamodel REST servisa [Haupt14]

Autori su implementaciju modela i odgovarajuće transformacije zasnivali na *Eclipse Epsilon* projektu, dok je metamodel definisan u *Emfatic* jeziku za definisanje EMF modela. M2M transformacije su

implementirane korištenjem *Epsilon Transformation jezika – Epsilon Transformation Language* (ETL), dok je za izradu grafičkog editora za modelovanje korišten GMF. M2T transformacija je izvršena korištenjem *Java Emitter Templaes* (JET) okvira.

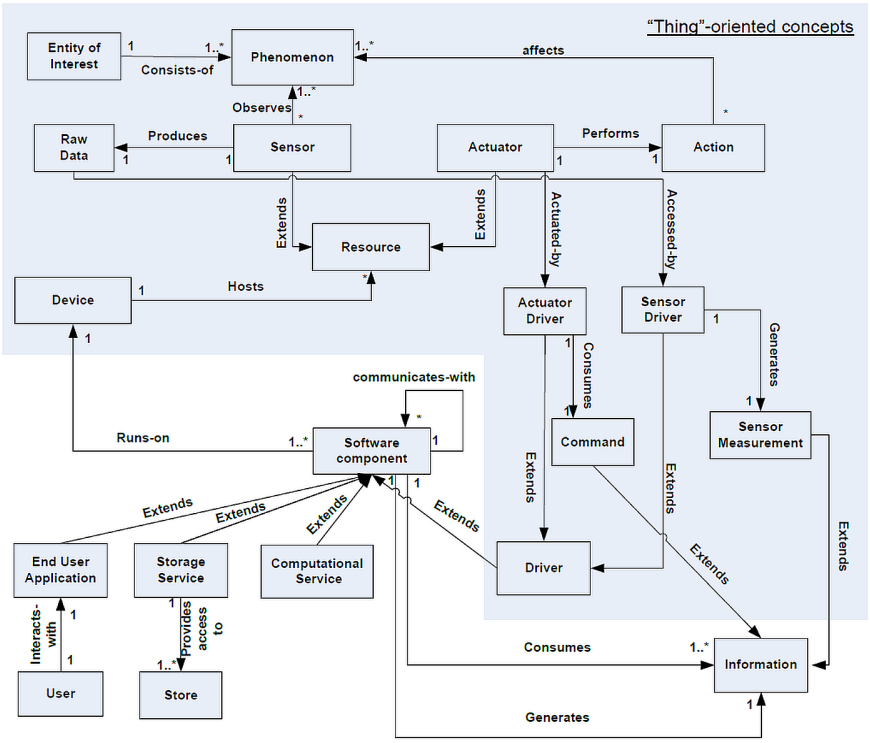
U radu [Porten12], koristeći *Graphiti* okvir, kreiran je vizuelni jezik koji dozvoljava modelovanje ROA arhitekture, odnosno grafički editor za modelovanje RESTful servisa na osnovu Schreier-inog metamodela.

3.2.2. Modelovanje IoT koncepta

IoT kombinuje različite elemente "tradicionalnog" Interneta (Web i baze podataka) i modernih koncepta (senzorske i aktuatorske mreže), što dovodi do povečanja obima i heterogenosti prouzrokovane velikim brojem uređaja različitih karakteristika. Tako razvoj aplikacija za IoT predstavlja izazov jer obuhvata širok spektar problema koji su obično proizvod nedostatka adekvatnih tehnologija ili standardizovane arhitekture. Može se reći da problem tehnologije uveliko zavisi od primjene IoT aplikacija, a isti se može otkloniti pravljenjem specijalizovanog hardvera za domen problema koji se rješava. Međutim, analiziom dostupne literature je utvrđeno da se veoma mali broj autora bavio standardizovanjem arhitekture i definisanjem njenih metamodela, koji bi podigli nivo apstrakcije i definisali podlogu za MDA arhitekturu.

Patel je u radu [Patel12] postavio cilj da se u proces razvoja specifikacija za automatsko generisanje kôda ili IoT baziranih aplikacija, koje uključuju veliki broj heterogenih uređaja, aktivno uključe i eksperti iz domena problema, odnosno budući neposredni korisnici. Vodeći se postavljenim ciljem, Patel je predstavio semantički model IoT-a (Slika 3.2.2.1.) na osnovu kojeg su identifikovane uloge svih zainteresovanih strana (*domenski ekspert*, *projektant softvera*, *projektant aplikacija*, *projektant hardvera* i *menadžer mreže*), njihove vještine i odgovornosti. Predložen je MDA pristup koji koristi poseban domenski jezik za svaku fazu projektovanja:

- *Srijan Vocabulary Language* (SVL) – za specifikaciju domenskog rječnika,
- *Srijan Architecture Language* (SAL) – za specifikaciju arhitekture aplikacije, i
- *Srijan Network Language* (SNL) – za definisanje osobina mreže na kojima će aplikacija da se izvršava.

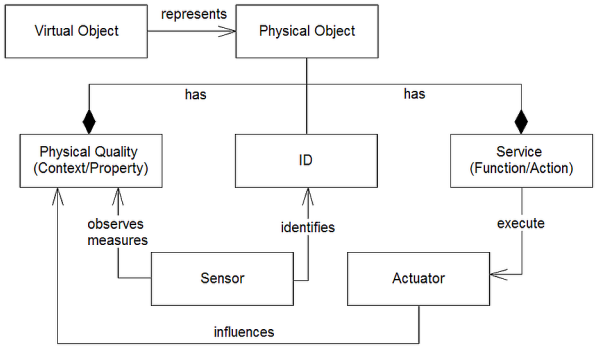


Slika 3.2.2.1. Semantički model IoT-a [Patel12]

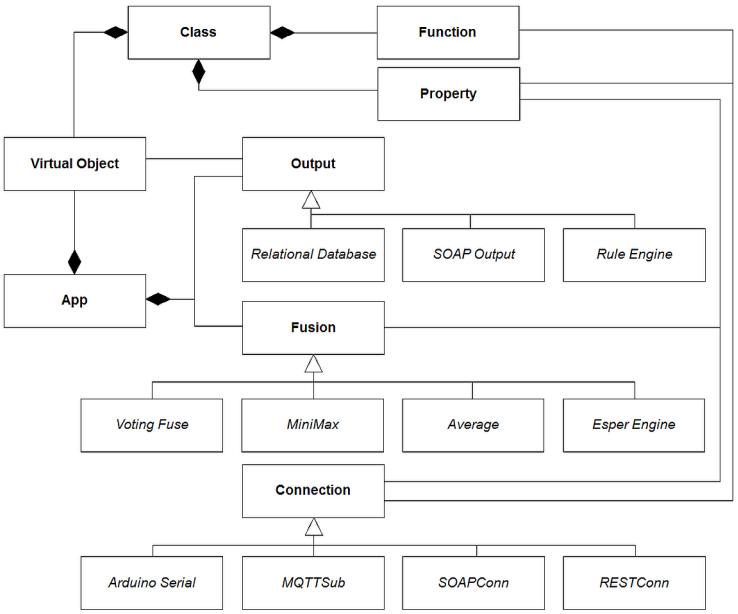
Oslanjajući se na *IoT-A*, evropski istraživački projekat koji za cilj ima standardizaciju IoT arhitekture, Pramudianto u svojoj disertaciji [Pramud15] i nizu radova [Pramud14, Patel12] predlaže razvojni alat baziran na MDA pristupu nazvan IoTLink. IoTLink omogućava neiskusnim projektantima da kombinuju distribuirane uređaje i servise u mrežu, koristeći grafički domenski specifični jezik za njeno modelovanje. Kreirani modeli se mogu transformisati u Java kôd.

Analizom predloženih koncepta u Pramudiantovim radovima, koji su nastali kao dio evropskog istraživačkog projekta u saradnji sa Brazilom (EBBITS i BEMO-COFRA), može se reći da su hipotezom postavljeni koncepti opravdani i da predstavljaju podlogu za budući razvoj ove obalsti.

Za kreiranje razvojnog alata, Pramudianto je koristio EMF okvir (za definisanje metamodela), GMF i *Extended Editing Framework* (EEF) okvire za kreiranje grafičkog editora, te *Acceleo* okvir za transformaciju modela u izvršni Java kôd. Na slikama (Slika 3.2.2.2., Slika 3.2.2.3.) su prikazani metamodeli predloženog razvojnog alata.



Slika 3.2.2.2. IoT metamodel baziran na IoT-A projektu [Pramud15, Pramud14]



Slika 3.2.2.3. Logički prikaz IoTLink Metamodel-a [Pramud15, Pramud14]

Corredor i dr. u svom istraživanju [Corredor12] predlažu MDA zasnovanu, resurs orijentisanu i ontološki upravljanu (ROOD) metodologiju za razvoj IoT i WoT baziranih aplikacija. Kao ključno navode činjenicu da se MDA pristup vrlo dobro uklapa sa IoT i WoT paradigmama, olakšavajući:

- apstrahiranje dijelova sistema modelima na visokom nivou apstrakcije koji su nezavisni od softverskih i hardverskih tehnologija,

- razdvajanje korisnika i pružalaca resursa (senzora, aktuatora, logičkih procesa), što omogućava ponovnu upotrebu modela artefakata i softverskih komponenti,
- pružanje okvira za modelovanje u svrhu olakšanja brze i agilne izrade prototipova složenih sistema, čak i za nestručne projektante.

Predložena ROOD metodologija zasniva se na *Smart Space Modeling Language* (SsML) DSM-u koji je baziran na UML profilima, a uključuje interakcije, učesnike, resurse i platfome. ROOD metodologija na osnovu SsML-a definiše dva modela:

- *Smart Object Model* (SOM) – definiše aspekte obrade vezane za mogućnosti zapažanja i aktiviranja "*pametnih objekata*"/"*stvari*", kao i modela konteksta kojim upravljaju
- *Environment Context Model* (ECM) – na visokoj razini apstrakcije opisuje ponašanje, interakciju i kontekstne informacije.

Autori za implementaciju *ROOD Visual Editor*-a koriste *Obeo Designer*, a za transformaciju modela ATL i *Acceleo* okvire.

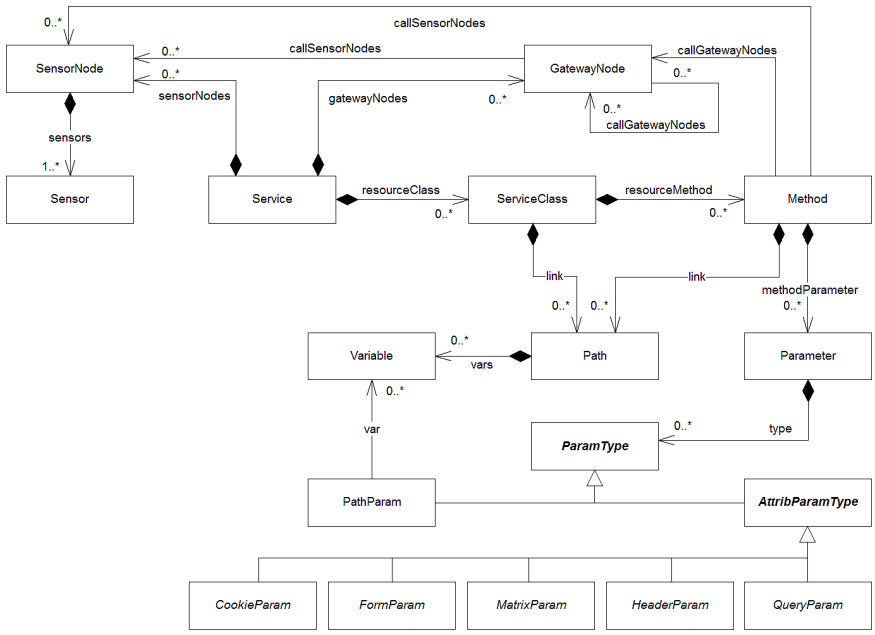
3.3. Prijedlog koncepta MDA rješenja

Analizom prethodno obrađenih publikacija, koje su zasnovane na konceptima bliskim domenu teze i predložene arhitekture SW mreža, može se doći do osnovnih koncepta elemenata dizajna metamodela koji predstavlja podlogu za kreiranje DSM rješenja.

Prateći Schreier-in koncept dizajna, za opis arhitekture SW mreža, kreiran je model strukture koji opisuje elemente SW mreže, kao i servisni sloj za kreiranje virtuelnih klastera SW elemenata. Na osnovu prikazanih rješenja i analize domena problema, metamodel se bazira na ROA arhitekturi, odnosno REST servisima. Kao implementacioni okvir izabran je JAX-RS [JaxRS] API za implementaciju RESTful Web Servisa u Java programskom jeziku [Jersey].

Oslanjajući se na tehnologiju i njene elemente, predloženi metamodel predstavlja rješenje, koje za razliku od drugih prkazanih rješenja, nije na visokom nivou apstrakcije, čime se smanjuje neodređenost i pojednostavljuje proces kreiranja arhitekture SW mreža. Osnovni koncepti modela REST servisa proizilaze iz JAX-RS standarda, dok se modelovanje senzora, senzorskih i *gateway* čvorova oslanja na ranije prikazane SW

koncepte. Na slici (Slika 3.3.1.) je prikazan pojednostavljeni metamodel SW arhitekture.



Slika 3.3.1. Pojednostavljeni metamodel SW arhitekture

Zavisno od predložene infrastrukture sistema, potrebno je definisati i uspostaviti veze između elemenata modela i definisanih parametara infrastrukture, kao i njihovo preslikavanje na metamodel koji će opisivati domen problema. Predloženi metamodel otvara put ka kreiranju DSM rješenja koje obuhvata:

- domenski specifični jezik za opis arhitekture SW mreža i servisnog sloja. Jezik treba da obezbijedi jednostavno kreiranje arhitekture bez suštinskog poznavanja problema SW arhitekture, kao i dinamičkih servisnih elemenata,
- interaktivno grafičko radno okruženje i aplikativnu podršku za automatizaciju procesa kreiranja arhitekture SW mreža i servisnog sloja na osnovu predloženih modela i pravila transformacije. Set alata mora obezbijediti integraciju sa postojećim razvojnim okruženjem – *Eclipse* platformom, i
- generator kôda – alati i pravila za automatsku transformaciju modela baziranih na kreiranom jeziku za opis arhitekture SW mreža i servisnog sloja u implementacione klase. Mehanizam transformacije

modela u implementacione klase mora biti bez gubitaka, čime se osigurava konzistentnost samog postupka.

Implementacija metamodela i ograničenja implementirana su uz oslonac na ECore [ECoreTools] model, EMF i OCLinECore okvire. Za kreiranje interaktivnog grafičkog editora korišten je *Sirius* okvir, dok je za transformaciju modela u implementacione klase primjenjen *Acceleo* generator kôda.

Poglavlje 4. sadrži detaljnu elaboraciju DSM razvojnog okruženja baziranog na definisanom metamodelu i izabranim implementacionim tehnologijama.

Poglavlje 4

DSM razvojno okruženje za modelovanje arhitekture Senzor Web mreže

Oslanjajući se na dobro definisane koncepte DSM metodologije i enkapsulaciju detalja niskog nivoa, te mogućnost da se ponašanje SW mreža opiše sa konceptima bliskim domenu problema, moguće je specificirati razvojno okruženje za modelovanje koje fokus razvoja pomjera sa implementacije na dizajn. Transformacijom modelovane arhitekture u implementacione klase servisnog sloja SW mreže obezbjeđuje se pristup servisima i integracija u jedinstvenu servisnu arhitekturu.

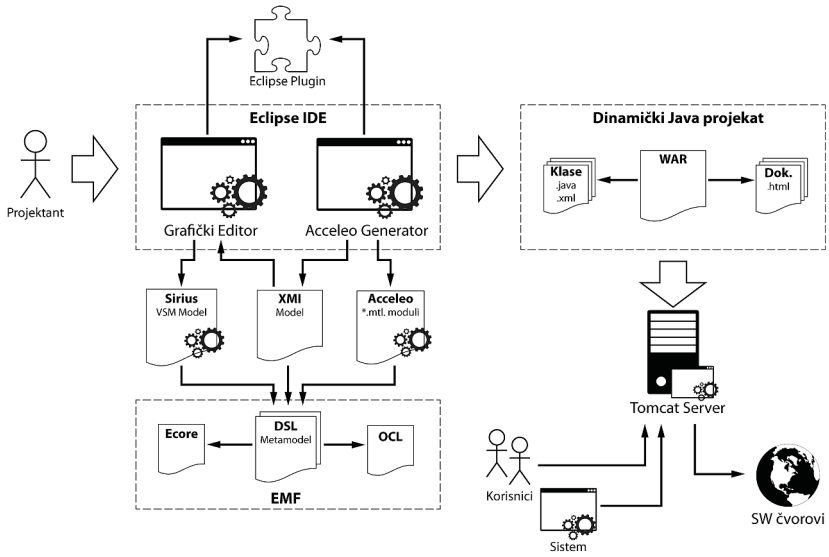
Na slici (Slika 4.1.) je prikazan model arhitekture DSM razvojnog okruženja izveden na osnovu zaključnih razmatranja prikazanih u poglavlju 3.

Arhitekturu predloženog DSM-a čine:

- domenski specifičan jezik – domen problema koji je opisan ECore modelom i ograničenja definisana OCL pravilima,
- interaktivno grafičko okruženje – proširenje *Eclipse* razvojnog okrueženja sa grafičkim editorom. Izlaz iz grafičkog editora predstavlja XMI specifikacija modela,
- generator kôda – proširenje Eclipse razvojnog okruženja sa generatorom kôda. Izlaz iz generatora predstavljaju implementacione klase (.java, .xml i .html fajlovi).

Predloženi metamodel sistema opisuje njegovu strukturu sa stanovišta gradivnih elemenata na visokom nivou apstrakcije i njihovih međusobnih odnosa. Na ovaj način, zavisno od izabrane platforme, projektovane arhitekture servisa i implementacije generatora za transformaciju modela u izvršni kôd, jedna ili više klasa iz modela može biti preslikana u jednu ili više implementacionih klasa aplikacije. U nekim slučajevima,

preslikavanje se neće izvršiti na klasu, već na svojstva postojećih klasa i na elemente postojećeg programskog kôda.



Slika 4.1. Model arhitekture DSM razvojnog okruženja

Verifikacija upotrebljivosti implementiranog DSM razvojnog okruženja je izvršena uz oslonac na niz eksperimenata koji uključuju realne reprezentativne primjere detaljno opisane u poglavlju 5.

4.1. Definicija domenski specifičnog jezika

Na osnovu modela arhitekture (Slika 4.1.) predloženi DSL obuhvata:

- ECore metamodel – definiše elemente domena problema predstavljene entitetima i relacijama ECore modela, a tvori apstraktnu sintaksu DSL-a, i
- OCL pravila – definiše formalna ograničenja i zajedno sa ECore metamodelom predstavlja gramatiku DSL-a.

Spisak raspoloživih metaklasa definisanih *ECore* jezikom sa kratkim opisom i roditeljskim metaklasama dat je u tabeli (Tabela 4.1.1.) u alfabetskom redoslijedu, dok je detaljan opis svake metaklase sa svojstvima, formalnim ograničenjima i inicijalnim vrijednostima dat u nastavku rada. Formalna ograničenja, pomoćni atributi, inicijalne vrijednosti i operacije su definisani OCL-om, u skladu sa sintaksom OCL-

a verzije 2.4 datom u okviru [OCL14] i strukturom ECore-a definisanom u EMF okviru [EMF, ECoreTools]

4.1.1. Implementacija domenski specifičnog jezika

Implementacija DSL-a bazira se na upotrebi metaklasa i pripadajućih pobrojanih tipova podataka, koji su u metamodelu organizovani u sljedeće logičke cjeline:

- *Senzorska mreža* – predstavlja metaklase koje opisuju elemente za prikupljanje podataka iz okoline, njihovu komunikaciju sa sistemom i razmjenu podataka.
- *Servisna arhitektura* – predstavlja metaklase koje opisuju elemente za kreiranje REST servisa, klasa, metoda i parametara.
- *Prilagođeni tipovi podataka* – predstavljaju tipove podataka koji odgovaraju potrebama prethodnih logičkih cjelina, a koji se ne nalaze u stadardnom setu EMF tipova podataka.

U tabelama (Tabela 4.1.1.1., Tabela 4.1.1.2. i Tabela 4.1.1.3.) je dat kratak pregled definisanih metaklasa i tipova podataka iz svih navedenih cjelina radi kompletnog uvida u metamodel, dok se detalji za svaku metaklasu mogu pronaći u okviru opisa logičke cjeline kojoj pripadaju.

Tabela 4.1.1.1. Metaklase definisane u okviru metamodela

Naziv metaklase	Roditeljska metaklasa	Logička cjelina	Opis
`AttribParamType`	`ParamType`	`Servisna arhitektura`	Apstraktna metaklasa koja proširuje elemente ParamType apstraktne metaklase dodatnim obilježjima predefinisanih parametara REST servisa
`CookieParam`	`AttribParamType`	`Servisna arhitektura`	Metaklasa koja predstavlja predefinisani parametar REST servisa za pristup ***cookie*** parametrima.
`FormParam`	`AttribParamType`	`Servisna arhitektura`	Metaklasa koja predstavlja predefinisani parametar REST servisa za pristup ***form*** parametrima.
`GatewayNode`		`Senzorska mreža`	Čvorište senzorske mreže koje omogućava hijerarhijsku raspodjelu senzorskih čvorova. Posjeduje identifikatore neophodne za komunikaciju putem Interneta i transparentna je za sistem. Također posjeduje opseg adresa koje sadrži u svojoj hijerarhiji, kao i broj dozvoljenih čvorova.
`HeaderParam`	`AttribParamType`	`Servisna arhitektura`	Metaklasa koja predstavlja predefinisani parametar REST servisa za pristup ***header*** parametrima.

MatrixParam	AttribParamType	Servisna arhitektura	Metaklasa koja predstavlja predefinisani parametar REST servisa za pristup varijablama koje su sadržane u ***URI matrix*** parametrima.
Method		Servisna arhitektura	Predstavlja metaklasu koja opisuje metode unutar klasa servisa, sa parametrima i obilježjima REST servisa - ***resursa***. *Metoda* se oslanja na predefinisane parametre klase, ali također definiše i samostalne parametre koji pružaju rad sa senzorskim i gateway čvorovima.
Parameter		Servisna arhitektura	Metaklasa koja predstavlja parametre metode, a može biti jednostavnog ili složenog tipa. Vrijednosti parametara mogu biti proslijeđene preko predefinisanih parametara REST servisa.
ParamType		Servisna arhitektura	Apstraktna metaklasa koja predstavlja predefinisane parametre REST servisa.
Path		Servisna arhitektura	Metaklasa koja predstavlja putanju do metoda – resursa servisa. Putanja može biti definisana na nivou klase servisa ili na nivou metode, a pored putanje može sadržati i varijable.
PathParam	ParamType	Servisna arhitektura	Metaklasa koja predstavlja predefinisani parametar REST servisa za pristup parametrima definisanim u varijablama putanje.
QueryParam	AttribParamType	Servisna arhitektura	Metaklasa koja predstavlja predefinisani parametar REST servisa za pristup varijablama koje su sadržane u ***URI query*** parametrima.
Sensor		Senzorska mreža	Metaklasa koja predstavlja senzorski element koji se nalazi na senzorskom čvoru i prikuplja informacije iz okoline. Senzori se razlikuju po tipu senzora, pojave i načina mjerenja odnosno prikupljanja podataka.
SensorNode		Senzorska mreža	Metaklasa koja predstavlja posrednika između senzora i ostatka sistema, odnosno senzorski čvor. Metaklasa posjeduje identifikatore neophodne za komunikaciju putem Interneta, čime se obezbjeđuje *Senzor Web* priroda senzorskog čvora.
ServiceClass		Servisna arhitektura	Omogućava kreiranje klasa servisa koje sadrže metode. Definiše osnovne parametre neophodne za kreiranje REST servisa, kao i predefinisanih parametara na koje se oslanjaju sadržane metode sistema.
Service		Servisna arhitektura	Metaklasa koja predstavlja osnovnu klasu sistema i koja sadrži sve elemente neophodne za kreiranje servisa i senzorske mreže.

`Variable`		`Servisna arhitektura`	Metaklasa koja definiše varijable koje su deklarisane u `Path` metaklasi. Varijable omogućavaju kreiranje složenih putanja.

Tabela 4.1.1.2. Nabrojani tipovi definisani u okviru metamodela

Naziv tipa	Značenje
`ContextType`	Skup podržanih resursa koji se definišu na nivou komponente.
`ImplementationType`	Skup parametara koji određuje implementaciju mjesta primjene senzorskih elemenata.
`MediaTypeElement`	Skup mogućih vrijednosti podržanih internet media tipova - MIME.
`MethodType`	Skup mogućih vrijednosti za tip metode u okviru REST tipa pristupa.
`MethodVisibility`	Skup mogućih vrijednosti za tip vidljivosti metode unutar klase.
`ResourceScopes`	Skup mogućih parametara koji obilježavaju životni vijek instance klase koja sadrži resurse.
`SensorType`	Skup mogućih vrijednosti za tip senzora koji se može pridružiti senzorskom elementu.

Tabela 4.1.1.3. Prilagođeni tipovi podataka definisani u okviru metamodela

Naziv prilagođenog tipa podataka	Opis
`GatewayName`	Prilagođeni tip podataka koji predstavlja ime gateway čvorova. Ime je definisano u obliku "**GAT_000**"
`GPS`	Geopozicione kordinate predstavljene u obliku longitude i latitude. GPS koordinate su definisane u obliku latitude (-90 – 90 , sjever-jug) i longitude (-180 – 180, zapad-istok)
`IP4`	Predstavlja adresu definisanu po Internet Protokolu v4 u brojevnom formatu sa tačkom: **0000.0000.0000.0000**
`IP6`	Predstavlja adresu definisanu po Internet Protokolu v6 u brojevnom formatu sa dvotačkom: **0000:0000:0000:0000:0000:0000:0000:0000**
`JavaDataType`	Predstavlja tip podataka koji određuje validan Java tip, primitivan: (***var, byte, short, int, long, float, double, boolean, char, String***), ***JSONObject*** ili složeni: ***paket.paket.ImeKlase***.
`MAC`	Predstavlja *Media Access Control* (MAC) adresu, odnosno jedinstveni identifikator hardverskog uređaja. Adresa može imati dva formata: sa linijom (**00-00-00-00-00-00**) ili sa dvotačkom (**00:00:00:00:00:00**)
`SensorName`	Prilagođeni tip podataka koji predstavlja ime senzorkog čvora. Ime je definisano u obliku "**SEN_000**"

Radi jednostavnijeg definisanja OCL ograničenja uvedene su pomoćne operacije u okviru metaklasa: `GatewayNode` i `Path`.

`context GatewayNode`

1. `checkSubnodes(subnodeIP EString)` – ispituje da li se IP adrese gateway-a i senzorskih čvorova nalaze unutar zadatog opsega podmreže.

```
public boolean checkSubnodes(final String subnodeIP)
{
    try
    {
        return new SubnetUtils(this.subnetIP.trim(),
        this.subnetMask.trim()).getInfo().isInRange(subnodeIP.trim());
    }
    catch(IllegalArgumentException iae)
    {
        iae.printStackTrace();
    }

    return false;
}
```

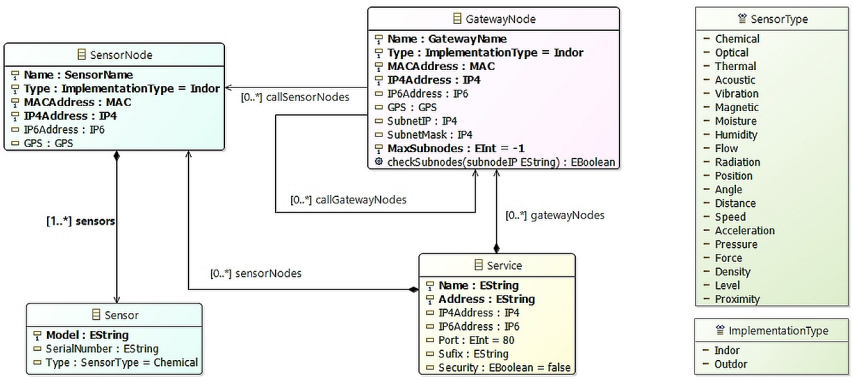
`context Path`

1. `checkVar(path EString)` – pronalazi sve deklarisane varijable u zadatoj putanji (dijelove putanje obilježene sa "$\{n\}$" za $n \in N^{+}$) i provjerava postojanje varijable sa zadatim indeksom n.

```
body: if(path.indexOf('{') = 0) then true else ((self.vars->
exists(var:Variable|(var.VarID = path.substring((path.indexOf('{')+1),
(path.indexOf('}')-1)).toInteger()))) and (if((path.indexOf('}') >=
path.size())) then true else checkVar(path.substring((path.indexOf('}')+1),
path.size())) endif )) endif;
```

4.1.1.1. Elementi za prikupljanje podataka − Senzorska mreža

Na slici (Slika 4.1.1.1.1.) je prikazan osnovni metamodel senzorske mreže koji se oslanja na arhitekturu bežičnih senzorskih mreža baziranoj na gateway i senzorskim čvorovima.



Slika 4.1.1.1.1. Metamodel – logička cjelina: senzorska mreža

Metamodel definiše elemente senzorske mreže koji opisuju fizičke reprezente sa dodjeljenim karakteristikama opisanim kroz svojstva elemenata. Odnos među elementima je kontrolisan OCL izrazima koji

definišu ograničenja i početne uslove koji svaki od predloženih elemenata može posjedovati. Određeni parametri zavise od nabrojanih tipova koji obezbjeđuju jednakost svih elemenata.

4.1.1.1.1. *GatewayNode*

Metaklasa `GatewayNode` predstavlja jedan od osnovnih gradivnih elemenata senzorske mreže, odnosno čvorište koje osigurava hijerarhijsko raslojavanje mreže. Pristup čvorovima koji su hijerarhijski u nižem poretku od čvora roditelja je moguć transparentno ali mora biti u zadatom opsegu (osigurava se smanjenje pretrage elemenata i kreiranje senzorskih klastera). Gateway čvor također osigurava identifikatore za pristup na hardverskom nivou kao i na *Internet Protocol* nivou verzije 4 i 6. U zavisnosti od postavljenog tipa, elementi mogu biti korišteni za indoor (unutrašnje) ili outdoor (vanjske) senzorske mreže od čega zavisi implementacija hardverskih jedinica. Kod izbora outdoor tipa, mora se definisati GPS lokacija senzorskog čvora.

Naziv meta-atributa	Uloga
`Name`	Ime `GatewayNode` elementa koje je definisano prilagođenim tipom podataka `GatewayName`.
`Type`	Definiše tip `GatewayNode` elementa, opisan nabrojanim tipom `ImplementationType`.
`MACAddress`	MAC adresa hardverskog uređaja koji implementira `GatewayNode` element. MAC adresa je opisana prilagođenim tipom podataka MAC.
`IP4Address`	IP4 adresa `GatewayNode` elementa na osnovu koje se pristupa elementu. IP4 adresa je opisana prilagođenim tipom podataka IP4.
`IP6Address`	IP6 adresa `GatewayNode` elementa na osnovu koje se pristupa elementu. IP6 adresa je opisana prilagođenim tipom podataka IP6.
`GPS`	Geopoziciona loakcija `GatewayNode` elementa. GPS je opisan prilagođenim tipom podataka GPS.
`CallGatewayNodes`	Spisak svih `GatewayNode` elemenata koji su direktni potomci elementa.
`CallSensorNodes`	Spisak svih `SensorNode` elemenata koji se povezuju na `GatewayNode` element.
`SubnetIP`	IP adresa podmreže na osnovu koje se vrši adresiranje svih elemenata (`GatewayNodes` i `SensorNodes`) koji se nalaze u podmreži. `SubnetIP` je opisana prilagođenim tipom podataka IP4.
`SubnetMask`	Maska podmreže na osnovu koje se vrši određivanje opsega adresa podmreže. `SubnetMask` je opisana prilagođenim tipom podataka IP4.
`MaxSubnodes`	Maksimalan broj `GatewayNode` i `SensorNode` elemenata koji se nalaze u podmreži.

Ograničenja:

1. Gateway čvor mora imati definisano ime.

```
invariant GatewayNodeEmptyName: self.Name->notEmpty() and self.Name <> '';
```

2. Gateway čvor mora imati definisanu IP4 adresu.

```
invariant GatewayNodeEmptyIP4Address: self.IP4Address->notEmpty() and
self.IP4Address <> '';
```

3. Gateway čvor mora imati definisanu MAC adresu.

```
invariant GatewayNodeEmptyMACAddress: self.MACAddress->notEmpty() and
self.MACAddress <> '';
```

4. Gateway čvor mora posjedovati barem jedan drugi gateway čvor ili barem jedan senzorski čvor.

```
invariant GatewayNodeMissingElements: self.callGatewayNodes->size() > 0 or
self.callSensorNodes->size() > 0;
```

5. Maska podmreže je definisana ali ne i adresa podmreže.

```
invariant GatewayNodeNotDefinedSubnetAddress: if (self.SubnetMask->notEmpty()
and self.SubnetMask <> '') then (self.SubnetIP->notEmpty() and self.SubnetIP <>
'') else true endif;
```

6. Adresa podmreže je definisana ali ne i maska podmreže.

```
invariant GatewayNodesNotDefinedSubnetMask: if (self.SubnetIP->notEmpty() and
self.SubnetIP <> '') then (self.SubnetMask->notEmpty() and self.SubnetMask <>
'') else true endif;
```

7. Gateway čvor ne može posjedovati sam sebe.

```
invariant GatewayNodeSelfContain: self.callGatewayNodes->excludes(self);
```

8. Gateway čvorovi ne smiju sadržavati petlju unutar bilo kojeg svog podstabla.

```
invariant GatewayNodeLoop: self.callGatewayNodes>closure(g:GatewayNode|
g.callGatewayNodes)->asSet()->excludes(self);
```

9. Ako je tip gateway čvora postavljen za outdoor (vanjski), potrebno je da ima definisane GPS koordinate.

```
invariant GatewayNodeGPSMissing: if (self.Type=ImplementationType::Outdor) then
if(self.GPS->isEmpty() or self.GPS = '') then false else true endif else true
endif;
```

10. Broj čvorova u podmreži je veći od definisanog broja čvorova.

```
invariant GatewayNodeExceededNumberOfSubnodes: if (self.MaxSubnodes > 0) then
(self.callSensorNodes->size()+self.callGatewayNodes->size()) <=
self.MaxSubnodes else true endif;
```

11. Adrese senzorskih čvorova nisu unutar opsega postavljene podmreže.

```
invariant GatewayNodeInvalidSensorNodesIP: if ((self.SubnetIP->notEmpty() and
self.SubnetIP <> '') and (self.SubnetMask->notEmpty() and self.SubnetMask <>
'') and self.callSensorNodes->notEmpty()) then self.callSensorNodes->
forAll(sensornode: SensorNode|checkSubnodes(sensornode.IP4Address)) else true
endif;
```

12. Adrese gateway čvorova nisu unutar opsega postavljene podmreže.

```
invariant GatewayNodeInvalidGatewayNodesIP: if ((self.SubnetIP->notEmpty() and
self.SubnetIP <> '') and (self.SubnetMask->notEmpty() and self.SubnetMask <>
'') and self.callGatewayNodes->notEmpty()) then self.callGatewayNodes->
forAll(gateway: GatewayNode|checkSubnodes(gateway.IP4Address)) else true endif;
```

Inicijalne vrijednosti:

1. Podrazumijevani tip `GatewayNode`-a je za indoor (unutrašnju) namjenu:

```
attribute Type : ImplementationType = 'Indoor';
```

4.1.1.1.2. *SensorNode*

Metaklasa `SensorNode` predstavlja jedan od osnovnih gradivnih elemenata senzorske mreže, odnosno senzorski čvor (Senzor Web čvor) koji obezbjeđuje pristup senzorskim elementima, prikupljanje, obradu i slanje podataka prema servisu. Senzorski čvor osigurava identifikatore za pristup na hardverskom nivou kao i na *Internet Protocol* nivou verzije 4 i 6, pri čemu se sama komunikacija obavlja preko ugrađenog servisa. U zavisnosti od postavljenog tipa, senzorski čvorovi mogu biti korišteni za indoor (unutrašnje) ili outdoor (vanjske) senzorske mreže od čega zavisi implementacija hardverskih jedinica. Kod izbora outdoor tipa, mora se definisati GPS lokacija senzorskog čvora.

Naziv meta-atributa	Uloga
`Name`	Ime `SensorNode` elementa koje je definisano prilagođenim tipom podataka `SensorName`.
`Type`	Definiše tip `SensorNode` elementa, opisan nabrojanim tipom `ImplementationType`.
`MACAddress`	MAC adresa hardverskog uređaja koji implementira `SensorNode` element. MAC adresa je opisana prilagođenim tipom podataka MAC.
`IP4Address`	IP4 adresa `SensorNode` elementa na osnovu koje se pristupa elementu. IP4 adresa je opisana prilagođenim tipom podataka IP4.
`IP6Address`	IP6 adresa `SensorNode` elementa na osnovu koje se pristupa elementu. IP6 adresa je opisana prilagođenim tipom podataka IP6.
`GPS`	Geopoziciona lokacija `SensorNode` elementa. GPS je opisan prilagođenim tipom podataka GPS.

Ograničenja:

1. Senzorski čvor mora imati definisano ime.

```
invariant SensorNodeEmptyName: self.Name->notEmpty() and self.Name <> '';
```

2. Senzorski čvor mora imati definisanu IP4 adresu.

```
invariant SensorNodeEmptyIP4Address: self.IP4Address->notEmpty() and
self.IP4Address <> '';
```

3. Senzorski čvor mora imati definisanu MAC adresu.

```
invariant SensorNodeEmptyMACAddress: self.MACAddress->notEmpty() and
self.MACAddress <> '';
```

4. Senzorski čvor mora imati barem jedan senzorski element.

```
invariant SensorNodeSensorMissing: self.sensors->size() > 0;
```

5. Ako je tip senzorskog čvora postavljen za outdoor (vanjski), potrebno je da ima definisane GPS koordinate.

```
invariant SensorNodeGPSMissing: if (self.Type=ImplementationType::Outdor) then
if(self.GPS->isEmpty() or self.GPS = '') then false else true endif else true
endif;
```

Inicijalne vrijednosti:

1. Podrazumjevani tip `SensorNode`-a je za indoor (unutrašnju) namjenu:

```
attribute Type : ImplementationType = 'Indoor';
```

4.1.1.1.3. *Sensor*

Metaklasa `Sensor` predstavlja hardversku realizaciju senzorskog elementa koji služi za mjerenje pojava iz okoline. Senzor je definisan modelom koji je propisan od proizvođača i bitan je kako bi se mogla utvrditi karakteristika koja opisuje senzor, dok serijski broj dopunjuje oznaku modela. Bitan faktor u opisu senzora čini tip senzora, od kojeg zavisi mjesto primjene, prenos i obrada podataka.

Naziv meta-atributa	Uloga
`Model`	Oznaka proizvođača senzora.
`SerialNumber`	Serijski broj proizvođača senzora (ako postoji).
`Type`	Tip senzora opisan nabrojanim tipom `SensorType`.

Ograničenja:

1. Senzor mora imati definisano ime.

```
invariant SensorEmptyModel: self.Model->notEmpty() and self.Model <> '';
```

4.1.1.1.4. *ImplementationType*

Nabrojani tip `ImplementationType` definiše skup mogućih vrijednosti za tip primjene gateway ili senzorskog čvora. Tip primjene se dijeli na

mjerenje indoor (unutrašnjih) ili outdoor (vanjskih) pojava. U zavisnosti od izabranog tipa, implementacija hardverskih jedinica može da varira i da se razlikuje za određene elemente (kućište, komunikacioni elementi, itd.)

Naziv vrijednosti	Značenje
`Indoor`	Obilježava gateway i senzorski čvor za unutrašnju primjenu.
`Outdoor`	Obilježava gateway i senzorski čvor za vanjsku primjenu.

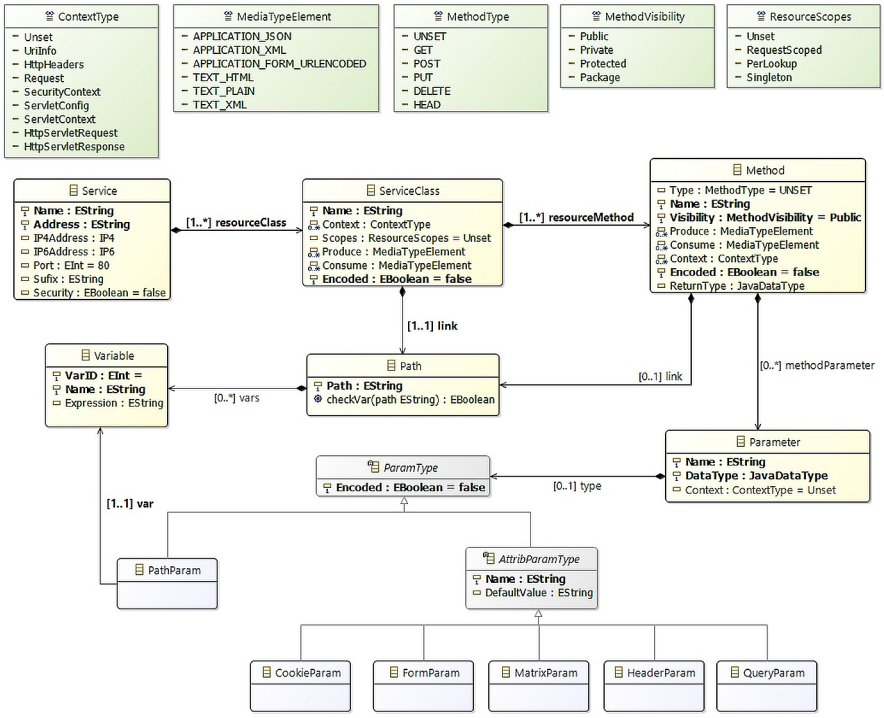
4.1.1.1.5. *SensorType*

Nabrojani tip `SensorType` definiše skup mogućih tipova senzora koji se mogu pridružiti senzorskom elementu. Tip senzora opisuje koja se prirodna veličina mjeri, odnosno koja je primarna namjena senzora za koji je napravljen. Tip senzora također služi za kreiranje klastera po tipu i načinu mjerenja, što omogućava lakšu pretragu hijerarhije i kreiranje slojeva za prikupljanje informacija.

Naziv vrijednosti	Značenje
`Chemical`	Tip senzora namjenjen za mjerenje hemijskih pojava.
`Optical`	Tip senzora namjenjen za mjerenje optičkih pojava.
`Thermal`	Tip senzora namjenjen za mjerenje temperature.
`Acoustic`	Tip senzora namjenjen za mjerenje zvuka.
`Vibration`	Tip senzora namjenjen za mjerenje vibracija.
`Magnetic`	Tip senzora namjenjen za mjerenje magnetnog polja.
`Moisture`	Tip senzora namjenjen za mjerenje vlažnosti tla.
`Humidity`	Tip senzora namjenjen za mjerenje vlažnosti vazduha.
`Flow`	Tip senzora namjenjen za mjerenje protoka tečnosti.
`Radiation`	Tip senzora namjenjen za mjerenje radijacije.
`Position`	Tip senzora namjenjen za mjerenje pozicije.
`Angle`	Tip senzora namjenjen za mjerenje ugla.
`Distance`	Tip senzora namjenjen za mjerenje udaljenosti.
`Speed`	Tip senzora namjenjen za mjerenje brzine.
`Acceleration`	Tip senzora namjenjen za mjerenje ubrzanja.
`Pressure`	Tip senzora namjenjen za mjerenje pritiska.
`Force`	Tip senzora namjenjen za mjerenje sile.
`Density`	Tip senzora namjenjen za mjerenje gustoče tečnosti.
`Level`	Tip senzora namjenjen za mjerenje nivoa tečnosti.
`Proximity`	Tip senzora namjenjen za mjerenje razdaljine.

4.1.1.2. Obrada podataka i komunikacija sa korisnikom − Servisna arhitektura

Slika (Slika 4.1.1.2.1.) prikazuje metamodel servisne arhitekture predloženog sistema koji je baziran na konceptima i implementaciji REST servisa. Predloženi metamodel prikazuje strukturu servisa koja omogućava kreiranje odnosa i zavisnosti između servisa, klasa i metoda servisa. Zbog uvođenja određenih ograničenja neophodnih prilikom implementacije i transformacije u izvršni kôd, potrebno je uvođenje konkretizacije i detalja okvira za implemtaciju REST servisa. U ovom slučaju je izabran JAX-RS okvir, koji predstavlja specifikaciju za implementaciju REST servisa u Java okruženju.



Slika 4.1.1.2.1. Metamodel – logička cjelina: servisna arhitektura

Uvodeći elemente JAX-RS okvira, predloženi metamodel definiše nabrojane tipove podataka koji sadrže elemente definisane u sklopu okvira, kao i relacije između elemenata metamodela. Odnos među elementima je kontrolisan OCL izrazima koji definišu ograničenja i početne uslove koji svaki od predloženih elemenata može posjedovati.

4.1.1.2.1. *Service*

Metaklasa `Service` predstavlja logički element koji opisuje servis, a sadrži podatke o klasama servisa, gateway i senzorskim čvorovima. Klasa obezbjeđuje neophodne informacije o URL adresi na kojoj se sistem nalazi (kao i informacije o IP4 i IP6 adresama), te o portu na kojem servis prima zahtjeve korisnika. Indikatorom sigurnosti po potrebi se može dodati i sigurnosni protokol − *Secure Socket Layer* (SSL) na komunikacioni kanal koji obezbjeđuje kriptovanje podataka. Servis je implementiran pomoću JAX-RS API-ja.

Naziv meta-atributa	Uloga
`Name`	Ime `servisa` definisano znakovnim nizom.
`Address`	URL adresa servisa.
`IP4Address`	IP4 adresa na osnovu koje se omogućava pristup servisu.
`IP6Address`	IP6 adresa na osnovu koje se omogućava pristup serivsu.
`Port`	Port na kojem se izvršava servis.
`Sufix`	Sufiks adrese na kojoj se nalazi servis.
`Security`	Indikator o korištenju SSL protokola u komunikaciji sa servisom.

Ograničenja:

1. Servis mora imati definisano ime.

```
invariant ServiceEmptyName: self.Name->notEmpty() and self.Name <> '';
```

2. Adresa servisa mora biti definisana.

```
invariant ServiceEmptyAddress: self.Address->notEmpty() and self.Address <> '';
```

3. Port servisa mora biti definisan.

```
invariant ServiceEmptyPort: self.Port->notEmpty() and self.Port <> '';
```

4. Mora postojati barem jedna klasa servisa.

```
invariant ServiceResourceClass: self.resourceClass->size() > 0;
```

5. Mora postojati barem jedan gateway ili senzorski čvor.

```
invariant ServiceMissingElements: self.gatewayNodes->size() > 0 or
self.sensorNodes->size() > 0;
```

Inicijalne vrijednosti:

1. Podrazumijevani port servisa je port 80.

```
attribute Port : ecore::EInt[?] = '80';
```

2. Podrazumijevano korištenje sigurnosnog protokola je `false`.

```
attribute Security : Boolean[?] = 'false';
```

4.1.1.2.2. *ServiceClass*

Metaklasa `ServiceClass` definiše klase servisa i njihova obilježja. Klase predstavljaju implementaciju JAX-RS klasa i sadrže elemente i parametre neophodne za njihovo kreiranje. Definisani parametri obuhvataju ime klase, kao i kontekstne elemente, te životni ciklus klase unutar framework-a. `ServiceClass` također definiše i MIME podržane tipove prilikom odgovora i prilikom zahtjeva. Definisani MIME tipovi na nivou klase, postaju predodređeni tipovi koji će biti korišteni u svim metodama klase, ako naknadno nisu predefinisani. Indikatorom o korištenju automatskog URI dekodiranja moguće je uključiti ili isključiti ovu opciju na nivou klase.

Naziv meta-atributa	Uloga
`Name`	Ime klase definisano znakovnim nizom.
`Context`	Definiše kontekstne elemente klase opisane nabrojanim tipom `Context-Type`.
`Scopes`	Definiše životni ciklus klase u zavisnoti od zahtjeva, opisan `Resource-Scope` nabrojanim tipom podataka.
`Produce`	Definiše MIME podržane tipove prilikom odgovora, a opisani su nabrojanim tipom `MediaTypeElement`.
`Consume`	Definiše MIME podržane tipove prilikom zahtjeva, a opisani su nabrojanim tipom `MediaTypeElement`.
`Encoded`	Indikator o korištenju automatskog URI dekodiranja u parametrima.

Ograničenja:

1. Klasa servisa mora imati definisano ime.

```
invariant ServiceClassEmptyName: self.Name->notEmpty() and self.Name <> '';
```

2. Mora postojati barem jedna metoda unutar klase servisa.

```
invariant ServiceClassResourceMethod: self.resourceMethod->size() > 0;
```

3. Mora biti definisana relativna putanja klase servisa.

```
invariant ServiceClassPath: self.link->size() = 1 and self.link.Path <> '';
```

Inicijalne vrijednosti:

1. Podrazumijevani životni ciklus klase servisa nije definisan (prepušta se kompajleru).

```
attribute Scopes : ResourceScopes[?]  = 'Unset';
```

2. Podrazumijevani indikator o korištenju URI dekodiranja unutar parametara je `false`.

```
attribute Encoded : Boolean = 'false';
```

4.1.1.2.3. *Method*

Metaklasa `Method` predstavlja osnovni gradivni element servisa koji je odgovoran za prihvatanje, obradu i odgovor na postavljene zahtjeve klijenta. Klasa definiše obavezna svojstva kao što su: ime metode, vidljivost unutar klase, te povratnu vrijednost podataka. Metode mogu mijenjati svoju vidljivost unutar klase, ali samo one metode koje su okarakterisane sa `Public` mogu biti korištene kao metode servisa, dok sve ostale metode mogu biti korištene kao pomoćne metode klase. Tip metode, u zavisnosti od načina pristupa servisu, definisan je parametrom `Type` koji je opisan nabrojanim tipom `MethodType`. `Method` također definiše i MIME podržane tipove prilikom odgovora i prilikom zahtjeva pri čemu se na taj način predefinišu MIME tipovi definisani na nivou klase. Indikatorom o korištenju automatskog URI dekodiranja moguće je uključiti ili isključiti ovu opciju na nivou metode.

Naziv meta-atributa	Uloga
`Type`	Definiše tip metode koji je opisan nabrojanim tipom `MethodType`.
`Name`	Ime metode definisano znakovnim nizom.
`Visibility`	Definiše vidljivost metode unutar klase koja je opisana nabrojanim tipom `MethodVisibility`.
`Produce`	Definiše MIME podržane tipove prilikom odgovora, a opisani su nabrojanim tipom `MediaTypeElement`.
`Consume`	Definiše MIME podržane tipove prilikom zahtjeva, a opisani su nabrojanim tipom `MediaTypeElement`.
`Context`	Definiše kontekstne elemente metode opisane nabrojanim tipom `ContextType`.
`Encoded`	Indikator o korištenju automatskog URI dekodiranja u parametrima.
`ReturnType`	Povratni tip podataka metode. Može biti `void`, `Response`, `GenericEntity` ili bilo koji drugi validan Java tip podataka.
`CallGatewayNodes`	Spisak svih `GatewayNode` elemenata koji su referencirani i kojima se pristupa iz metode.
`CallSensorNodes`	Spisak svih `SensorNode` elemenata koji su referencirani i kojima se pristupa iz metode.

Ograničenja:

1. Metoda mora imati defnisano ime.

```
invariant MethodEmptyName: self.Name->notEmpty() and self.Name <> '';
```

2. Povratni tip metode mora biti definisan.

```
invariant MethodEmptyReturnType: self.ReturnType->notEmpty() and
self.ReturnType <> '';
```

3. Za javne metode, sa postavljenim tipom metode, mora biti definisan MIME tip za odgovor.

```
invariant MethodMissingProduce: if (self.Visibility = MethodVisibility::Public
and self.Type <> MethodType::UNSET) then (self.Produce->notEmpty() or
self.ServiceClass.Produce->notEmpty()) else true endif;
```

4. Za javne metode, sa postavljenim tipom metode, mora biti definisan MIME tip za zahtjev.

```
invariant MethodMissingConsume: if (self.Visibility = MethodVisibility::Public
and self.Type <> MethodType::UNSET) then (self.Consume->notEmpty() or
self.ServiceClass.Consume->notEmpty()) else true endif;
```

5. Ako je definisan bilo koji parametar metode kao resursa, tada metoda mora biti javna.

```
invariant MethodWrongVisibilityOfResourceMethod: (self.Visibility =
MethodVisibility::Public and (self.Type <> MethodType::UNSET or self.Consume-
>notEmpty() or self.Produce->notEmpty() or self.Context->notEmpty() or
self.Encoded = true)) or (self.Type = MethodType::UNSET and self.Consume->
isEmpty() and self.Produce->isEmpty() and self.Context->isEmpty() and
self.Encoded = false);
```

6. Ime metode već postoji unutar klase.

```
invariant MethodNameAlreadyExist: self.ServiceClass.resourceMethod->
excluding(self)->forAll(method:Method|method.Name <> self.Name);
```

7. Metoda kao resurs već postoji.

```
invariant MethodResourceAlreadyExist: self.ServiceClass.resourceMethod-
>excluding(self)->forAll(method:Method|(if(((method.link->notEmpty() and
self.link->notEmpty()) and  (method.link.Path = self.link.Path)) or
(method.link->isEmpty() and self.link->isEmpty())) then if ((method.Type =
self.Type) and (method.Type <> MethodType::UNSET and self.Type <>
MethodType::UNSET)) then if (method.Produce->notEmpty() and self.Produce->
notEmpty()) then (method.Produce->intersection(self.Produce)->isEmpty()) else
if(method.Produce->notEmpty() and self.Produce->isEmpty()) then
(method.Produce-> intersection(self.ServiceClass.Produce)-> isEmpty()) else
if(method.Produce-> isEmpty() and self.Produce->notEmpty()) then
(self.ServiceClass.Produce-> intersection(self.Produce)->isEmpty()) else false
endif endif endif else true endif else true endif));
```

Inicijalne vrijednosti:

1. Podrazumijevana vidljivost metode.

```
attribute Visibility : MethodVisibility = 'Public';
```

2. Podrazumijevani indikator o korištenju URI dekodiranja unutar parametara je `false`.

```
attribute Encoded : Boolean = 'false';
```

4.1.1.2.4. *Path*

Metaklasa `Path` definiše relativnu putanju do klase servisa ili metode (resursa). Sačinjena je od znakovnih nizova, a može sadržavati i deklaracije varijabli koje se prosljeđuju prilikom poziva od strane klijenta. Putanja mora biti jedinstvena na nivou servisa (za putanje definisane u

klasama servisa) odnosno na nivou klase (za putanje definisane u metodama).

Naziv meta-atributa	Uloga
`Path`	Relativna putanja resursa koja može sadržavati deklaracije varijabli kroz "{n}" indekse. (npr. "data/{0}/user/{1}")

Ograničenja:

1. Relativna putanja nije definisana.

```
invariant PathInvalidPath: self.Path->notEmpty() and self.Path <> '';
```

2. Kreirana varijabla nije definisana u putanji.

```
invariant PathVariableNotUsed: self.vars-> forAll(var:Variable|
self.Path.indexOf('{'+var.VarID.toString()+'}')<>0);
```

3. Definisana varijabla ne postoji.

```
invariant PathVariableMissing: checkVar(self.Path);
```

4.1.1.2.5. *Variable*

Predstavlja varijable definisane u elementima `Path`. Svaka varijabla je opisana jedinstvenim identifikatorom koji odgovara identifikatoru korištenom pri definisanju putanje, imenom varijable, kao i regularnim izrazom koji obezbjeđuje validnost unosa podataka (očekivani unos podataka).

Naziv meta-atributa	Uloga
`VarID`	Identifikator varijable koji odgovara deklarisanom indeksu u `Path` putanji.
`Name`	Ime varijable definisano znakovnim nizom.
`Expression`	Regularni izraz koji definiše format podataka koji se može proslijediti kroz varijablu.

Ograničenja:

1. Varijabla mora imati definisano ime.

```
invariant VariableEmptyName: self.Name->notEmpty() and self.Name <> '';
```

2. Ne smiju postojati dvije varijable sa istim imenom u okviru putanje.

```
invariant VariableNameAlreadyExist: self.Path.vars->excluding(self)->
forAll(var:Variable|var.Name <> self.Name);
```

3. Ne smiju postojati dvije varijable sa istim ID-om u okviru putanje.

```
invariant VariableIDAlreadyExist: self.Path.vars->excluding(self)->
forAll(var:Variable|var.VarID <> self.VarID);
```

4.1.1.2.6. *Parameter*

Metaklasa `Parameter` opisuje parametre metode koji su definisani imenom i tipom podataka. Tip podataka predstavlja validan Java tip podataka, koji može biti primitivni ili korisnički definisan, pri čemu se mora koristiti kvalifikovano ime tipa.

Naziv meta-atributa	Uloga
`Name`	Ime parametra definisano znakovnim nizom.
`DataType`	Tip podataka parametra zadato prilagođenim tipom podataka `JavaDataType`.
`Context`	Definiše kontekstne elemente klase opisane nabrojanim tipom `ContextType`.

Ograničenja:

1. Parametar mora imati definisano ime.

```
invariant ParameterEmptyName: self.Name->notEmpty() and self.Name <> '';
```

2. Nije definisan tip podataka za parametar.

```
invariant ParameterEmptyDataType: self.DataType->notEmpty() and self.DataType
<> '';
```

4.1.1.2.7. *ParamType*

`ParamType` je apstraktna metaklasa koja definiše podgrupu JAX-RS parametara metode koji su neophodni za pristup podacima unutar putanje, forme ili objekata. Indikatorom o korištenju automatskog URI dekodiranja moguće je uključiti ili isključiti ovu opciju na nivou parametra.

Naziv meta-atributa	Uloga
`Encoded`	Indikator o korištenju automatskog URI dekodiranja u parametrima.

Inicijalne vrijednosti:

1. Podrazumijevani indikator o korištenju URI dekodiranja unutar parametara je `false`.

```
attribute Encoded : Boolean = 'false';
```

4.1.1.2.8. *AttribParamType*

Apstraktna metaklasa koja nasljeđuje klasu `ParamType`, te je proširuje parametrima koji deklarišu naziv i predefinisanu vrijednost.

Naziv meta-atributa	Uloga
`Name`	Ime parametra definisano znakovnim nizom.
`DefaultValue`	Predefinisana vrijednost parametra.

4.1.1.2.9. *PathParam*

Metaklasa `PathParam` nasljeđuje klasu `ParamType`, i obezbjeđuje pristup varijablama definisanim u putanji klase servisa ili metode kroz parametar `Var`.

Naziv meta-atributa	Uloga
`Var`	Referenciranje na varijablu deklarisanu u `Path` elementu.
`Encoded`	Indikator o korištenju automatskog URI dekodiranja u parametrima.

Ograničenja:

1. Parametar je povezan sa pogrešnom varijablom.

```
invariant PathParamWrongVariable: (self.Parameter.Method.link->size() > 0 and
self.Parameter.Method.link.vars->size() > 0 and
self.Parameter.Method.link.vars->includes(self.var)) or
(self.Parameter.Method.ServiceClass.link.vars->size() > 0 and
self.Parameter.Method.ServiceClass.link.vars->includes(self.var));
```

4.1.1.2.10. *CookieParam*

Obezbjeđuje pristup informacijama definisanim u Cookie objektu. Ime parametra mora biti jednako imenu Cookie parametra.

Naziv meta-atributa	Uloga
`Name`	Ime parametra definisano znakovnim nizom.
`DefaultValue`	Predefinisana vrijednost parametra.
`Encoded`	Indikator o korištenju automatskog URI dekodiranja u parametrima.

Ograničenja:

1. Ime parametra mora biti definisano.

```
invariant CookieParamEmptyName: self.Name->notEmpty() and self.Name <> '';
```

4.1.1.2.11. *FormParam*

Obezbjeđuje pristup informacijama definisanim form parametrima. Ime parametra mora biti jednako imenu form parametra.

Naziv meta-atributa	Uloga
`Name`	Ime parametra definisano znakovnim nizom.
`DefaultValue`	Predefinisana vrijednost parametra.
`Encoded`	Indikator o korištenju automatskog URI dekodiranja u parametrima.

Ograničenja:

1. Ime parametra mora biti definisano.

```
invariant FormParamEmptyName: self.Name->notEmpty() and self.Name <> '';
```

2. Metoda ili klasa ne sadrže odgovarajući MIME tip.

```
invariant FormParamWrongMediaType: self.Parameter.Method.Produce->notEmpty()
and self.Parameter.Method.Produce-> includes(
MediaTypeElement::APPLICATION_FORM_URLENCODED);
```

4.1.1.2.12. *MatrixParam*

Obezbjeđuje pristup parametrima definisanim u URI matrix patametrima. Ime parametra mora biti jednako imenu matrix patametra.

Naziv meta-atributa	Uloga
`Name`	Ime parametra definisano znakovnim nizom.
`DefaultValue`	Predefinisana vrijednost parametra.
`Encoded`	Indikator o korištenju automatskog URI dekodiranja u parametrima.

Ograničenja:

1. Ime parametra mora biti definisano.

```
invariant MatrixParamEmptyName: self.Name->notEmpty() and self.Name <> '';
```

4.1.1.2.13. *HeaderParam*

Obezbjeđuje pristup parametrima definisanim u HTTP zaglavlju. Ime parametra mora biti jednako imenu parametra HTTP zaglavlja.

Naziv meta-atributa	Uloga
`Name`	Ime parametra definisano znakovnim nizom.
`DefaultValue`	Predefinisana vrijednost parametra.
`Encoded`	Indikator o korištenju automatskog URI dekodiranja u parametrima.

Ograničenja:

1. Ime parametra mora biti definisano.

```
invariant HeaderParamEmptyName: self.Name->notEmpty() and self.Name <> '';
```

4.1.1.2.14. *QueryParam*

Obezbjeđuje pristup parametrima definisanim u URI query parametrima. Ime parametra mora biti jednako imenu query parametra.

Naziv meta-atributa	Uloga
`Name`	Ime parametra definisano znakovnim nizom.
`DefaultValue`	Predefinisana vrijednost parametra.
`Encoded`	Indikator o korištenju automatskog URI dekodiranja u parametrima.

Ograničenja:

1. Ime parametra mora biti definisano.

```
invariant QueryParamEmptyName: self.Name->notEmpty() and self.Name <> '';
```

4.1.1.2.15. *ContextType*

Nabrojani tip `ContextType` definiše skup mogućih instanci objekata koji se mogu ugraditi unutar klase ili metode na serverskoj ili klijentskoj strani. Ovako definisane instance objekata omogućavaju pristup određenim svojstvima koje sadrže informacije o aplikaciji, servisu ili o kontejneru na kojem se izvršava servis. Razlikuju se dvije vrste instanci objekata: objekti definisani na nivou aplikacije i objekti definisani na nivou `Servleta`.

Naziv vrijednosti	Značenje
`Unset`	Kontekst nije definisan.
`UriInfo`	URI adresa.
`HttpHeaders`	HTTP zaglavlje
`Request`	Informacije o reprezentacija resursa i preduslovima zahtjeva.
`SecurityContext`	Pristup sigurnosnim elementima servisa i zahtjeva.
`ServletConfig`	Informacije o konfiguraciji "kontejnera" na kojem se izvršava servis.
`ServletContext`	Informacije o komunikacionim elementima sa "kontejnerom" i pristup MIME tipovima učitanih fajlova, upisivanje u log fajlove i dr.
`HttpServletRequest`	`HttpServletRequest` objekat koji se prosljeđuje "kontejneru" kao parametar doGet, doPost, itd. metodama
`HttpServletResponse`	`HttpServletResponse` objekat koji se prosljeđuje "kontejneru" kao parametar doGet, doPost, itd. metodama

4.1.1.2.16. *MediaTypeElement*

Nabrojani tip `MediaTypeElement` definiše skup mogućih vrijednosti podržanih internet media tipova – *Multi-purpose Internet Mail Extensions* (MIME) tipovi. `MediaTypeElement` predstavlja podršku za media tipove prilikom zahtjeva i odgovora koji se prosljeđuje od korisnika prema

servisu (resursu) i obratno. U zavisnosti od definisanog media tipa, metoda će klijentu proslijediti očekivani format podataka prilikom odgovora. Kada se postavi zahtjev servisu, i proslijedi odgovarajući sadržaj, potrebno je odgovoriti adekvatnom metodom koja podržava prosljeđeni tip medija. Ovim je osigurano da se podaci ispravno prime, obrade i pošalju klijentu.

Naziv vrijednosti	Značenje
APPLICATION_JSON	*JavaScript Object Notation* (JSON) - application/json.
APPLICATION_XML	*Extensible Markup Language* (XML) - application/xml.
APPLICATION_FORM_URLENCODED	Form URL Encoded - application/x-www-form-urlencoded.
TEXT_HTML	*HyperText Markup Language* (HTML) - text/html.
TEXT_PLAIN	Text File - text/plain.
TEXT_XML	*Extensible Markup Language* (XML) - text/plain.

4.1.1.2.17. *MethodType*

Nabrojani tip `MethodType` definiše skup mogućih vrijednosti za tip metode u okviru REST pristupa, odnosno načina na koji se klijent obraća servisu ili resursu. UNSET vrijednost definiše da metoda servisa ne može biti deklarisana kao resurs, te služi kao pomoćna metoda klase. Sve ostale metode koje imaju postavljen tip različit od UNSET, predstavljaju resurse unutar servisa.

Naziv vrijednosti	Značenje
UNSET	Tip metode nije postavljen.
GET	GET – odgovara na HTTP get metodu – CRUD operacija za pristup sadržaju elementa – retrive/read.
POST	POST – odgovara na HTTP post metodu – CRUD operacija za kreiranje elementa - create.
PUT	PUT – odgovara na HTTP put metodu – CRUD operacija za mijenjanje sadržaja elementa - update.
DELETE	DELETE – odgovara na HTTP delete metodu – CRUD operacija za brisanje elementa - delete.
HEAD	HEAD – odgovara na HTTP head metodu.

4.1.1.2.18. *MethodVisibility*

Nabrojani tip `MethodVisibility` definiše skup mogućih vrijednosti za vidljivost metoda unutar klase. Vrijednosti su definisane implementacijom JAX-RS API-ja za Java programski jezik. U zavisnosti od selektovane vrijednosti, metoda može biti vidljiva unutar projekta, paketa, klasa nasljednica ili može biti privatna. Samo one metode koje imaju tip

vidljivosti postavljen na `Public` (javna) mogu predstavljati resurs. Sve ostale služe kao pomoćne metode u klasi.

Naziv vrijednosti	Značenje
`Public`	Metoda je javna i dostupna iz svih dijelova programa.
`Private`	Metoda je privatna i dostupna je samo unutar klase.
`Protected`	Metoda je zaštičena i dostupna je samo unutar klase i klase nasljednice.
`Package`	Metoda je privatna na nivou paketa (unutar paketa dostupna svim članovima)

4.1.1.2.19. *ResourceScopes*

Nabrojani tip `ResourceScopes` definiše skup mogućih vrijednosti životnog vijeka klase u zavisnosti od poziva "po zahtjevu", odnosno za svako obraćanje klijenata. Po predefinisanoj vrijednosti, klasa se kreira svaki put kada zahtjev za metodom (resursom) bude upućen na servis. U zavisnosti od potrebe, mijenjajući parametar definisan sa `ResourceScopes`, priroda kreiranja klase može biti promijenjena – kreiranje na svaki poziv ili kreiranje na nivou instance.

Naziv vrijednosti	Značenje
`Unset`	Životni vijek klase nije definisan – koristi se predefinisani životni vijek.
`RequestScoped`	Životni vijek klase je definisan "po zahtjevu".
`PerLookup`	Životni vijek klase je definisan po svakom obraćanju klasi.
`Singleton`	Životni vijek klase je definisan kao jedinstveni – jedno kreiranje klase za sve zahtjeve na nivou servisa.

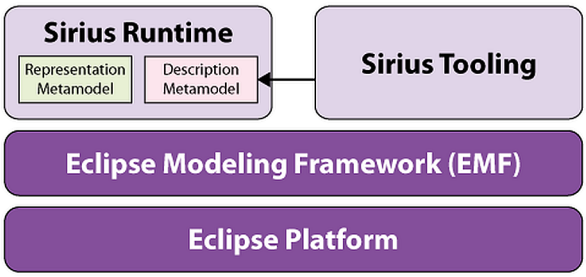
4.2. Alat za grafičko modelovanje servisnog sloja

Analiza vodećih slobodno dostupnih okvira za kreiranje grafičkh alata baziranih na DSL-u izvršena je u poglavlju 2. Rezultat ove analize, zajedno sa činjenicama publikovanim u radovima [Vujović14, Vujović14a], definišu *Sirius* okvir kao pogodan izbor za implementaciju grafičkog alata. *Sirius* okvir nudi rješenje za brz razvoj grafičkih editora bez potrebe za razumijevanjem pozadinskih procesa (GMF-a) [Vujović14b], pri čemu pojednostavljuje proizvod, smanjuje vrijeme dizajna i znatno povećava ukupnu produktivnost kreiranja domenski specifičnih grafičkih editora.

Sirius okvir predstavlja sveobuhvatnu *Eclipse* baziranu platformu za razvoj i korištenje grafičkih, interaktivnih editora za bilo koji domen primjene, koji je moguće opisati korištenjem EMF modela. Omogućava definiciju dvije kategorije korisnika:

- *Arhitekte* – koji učestvuju u razvoju domenski specifičnog interaktivnog editora, pod uslovom da je on definisan uz pomoć EMF *ECore* modela.
- *Krajnje korisnike* – koji koriste kreirane domenski specifične interaktivne editore za kreiranje, pregled i editovanje domenskih modela.

Na slici (Slika. 4.2.1.) prikazana je arhitektura *Sirius* okvra koju čine [Sirius]:



Slika 4.2.1. Arhitektura *Sirius* okvira [Sirius]

- *Sirius Tooling* (ST) – dio koji se koristi za definisanje specifikacije interaktivnih editora. Specifikacija se kreira na deklarativan način, odnosno konfiguracijom modela opisa (*description model*). Za svaki interaktivni model potrebno je navesti:
  - šta će biti predstavljeno modelom, odnosno koji elementi domena će biti vidljivi,
  - kako će ti elementi izgledati, odnosno njihov vizuelni prikaz, i
  - kako će se elementi ponašati, odnosno načini interakcije elemenata i alata sa krajnjim korisnicima.
- *Sirius Runtime* (SR) – dio koji upotrebljavaju krajnji korisnici za kreiranje, pregled i editovanje modela uz oslonac na interaktivni editor. Također je odgovoran za interpretiranje modela opisa i njihovu integraciju sa interaktivnim editorom. Za prikaz modela koristi se model domena koji opisuje konkretan domenski proces i njegov odgovarajući metamodel. Koristeći definisana pravila u modelu opisa, SR preslikava model u reprezentacioni domen, odnosno u njegov grafički prikaz.

Oslanjajući se na koncept kolekcije pogleda (*viewpoints*) *Sirius* omogućava definisanje različitih nivoa analize domena i podataka uz oslonac na pojedinačne poglede (*viewpoint*). Za kreiranje interaktivnih editora moguće je koristiti spoljni *ECore* model domena, ili interno definisani metamodel koji obezbjeđuje potrebne apstraktne koncepte i njihove odnose. Nakon definisanja DSM modela, *Sirius* pruža mogućnost jednostavnog kreiranja konkretnih reprezentacija modela (dijalekata).

Oslanjajući se na specifikacioni model pogleda – *Viewpoint Specification Model* (VSM), koji opisuje strukturu, izgled i ponašanje, *Sirius* može predstaviti bilo koji model kompatibilan sa EMF-om. VSM je specificiran unutar specifikacionog projekta pogleda – *Vewpoint Specification Projects* (VSP) koji predstavlja *Eclipse Plug-in* projekat i sadrži jedan ili više **.odesign* datoteka koji predstavljaju VSM model. Pet osnovnih koncepta definisani su u VSM modelu [Sirius, Vujović14d]:

- ***Pogled*** (*viewpoint*) – osnovni element koji predstavlja skup specifikacija reprezentacije i njihovih proširenja.
- ***Reprezentacija*** (*representation*) – grupa grafičkih elemenata koja predstavlja podatke domena. Također opisuje strukturu, izgled i ponašanje modela. Trenutno postoji pet reprezentacija (dijalekata): dijagrami (grafički modeli), dijagrami sekvence, tabele, matrice, i stabla (hijerarhijski prikaz).
- ***Mapiranje*** (*mapping*) – identifikuje podskup elemenata semantičkog modela koji bi trebalo da se pojavi u reprezentaciji. Također definiše kako bi elementi trebali biti prikazani, što zavisi od izabranog dijalekta.
- ***Stil*** (*style*) – svako mapiranje može imati jedan ili više stilova koji služe za konfigurisanje vizuelnog izgleda elemenata.
- ***Alat*** (*tool*) – opisuje ponašanje mapiranja.

Elementi VSM modela obično zahtijevaju "jezičke opise" za definisanje izraza čijim izvršavanjem se obezbjeđuje podrška implementaciji ponašanja karakterističnog za domen i reprezentaciju. Za definisanje izraza, *Sirius* može koristi jedan od tri jezika: *Acceleo Query Language* (AQL) [Acceleo] kao preporučeni jezik, OCL i Java OO jezik. Primjenom Java klasa i *Acceleo* upita definisanih u **.mtl* datotekama, moguće je izvršiti prilagođavanje *Siriusa* prema potrebama korisnika. Obzirom da Sirius enkapsulira GMF okvir, prilagođavanje je moguće izvršiti i na GMF nivou, međutim ovo predstavlja naprednu funkcionalnost za koju je potrebno detaljno poznavanje GMF okvira.

Validacija modela u *Sirius* okviru moguće je izvršiti sa dva ugrađena mehanizma. Prvi se oslanja na *Sirius Validate* koristeći pravila definisana u metamodelu kreiranog modela. Drugi mehanizam predstavlja primjenu prilagođenih validacionih pravila koja će biti primjenjena samo kada se pozove validacija za određenu instancu elementa.

Pogledi predstavljaju osnovni dio VSM modela i sadrže sve podatke o elementima koji su najčešće opisani atributima (svojstvima). Mnogi od elemenata imaju zajedničke atribute kao što su: identifikator (*ID*), naslov (*Label*), dokumentacija (*Documentation*) i korisnička dokumentacija (*End-user documentation*), dok se određeni broj atributa oslanja na tipove podataka definisane metamodelom. Pogledi također definišu i fajl ekstenziju modela – *Model File Extensions*, koji ograničavaju pogled na projekte sa određenim tipom modela.

Unutar pogleda mogu biti kreirani [Vujović14b]:

- ***Opis reprezentacije*** (*Representation Description*) – za dijagrame, dijagrame sekvenci, tabele, matrice i stabla (hijerarhijski prikaz).
- ***Proširenje reprezentacije*** (*Representation Extensions*) – za sada podržana proširenja samo za dijagrame.
- ***Validaciona pravila*** (*Validation Rules*) – pravila koja će biti primjenjena na sve reprezentacije unutar pogleda.
- ***Java proširenje*** (*Java Extensions*) – definiše Java klase koje sadrže definiciju servisa koji se mogu pozvati iz svih reprezentacija definisanih u pogledu.

Validaciju VSM modela je moguće uraditi za specifični element ili kompletan model, a obuhvata provjeru ispravnosti specificiranih reprezentacija, mapiranja i alata.

Na slici (Slika 4.2.2.) je prikazan primjer **.odesign* datoteke otvorene u *Sirius* editoru specifikacija – *Sirius Specification Editor*.

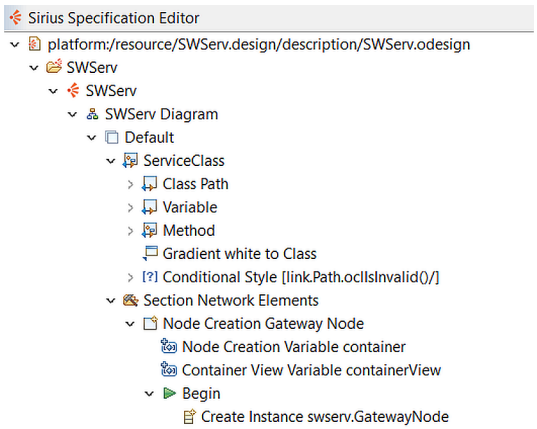
Element VSM-a na najvišem nivou hijerarhije je uvijek element grupe i služi samo kao kontejner za:

- ***poglede*** (*Viewpoint*) – definiše elemente i pravila mapiranja, i
- ***korisničke palete boja*** (*User Colors Palette*) – osiguravaju korisnički definisan skup boja za modele.

U zavisnosti od metamodela, unutar pogleda, *Sirius* može prikazati dvije grupe elemenata [Sirius]:

- ***slojeve*** (*Layers*) – svi grafički elementi i alati dijagrama su dio samo jednog sloja. Zbog toga, svaki dijagram mora imati barem jedan

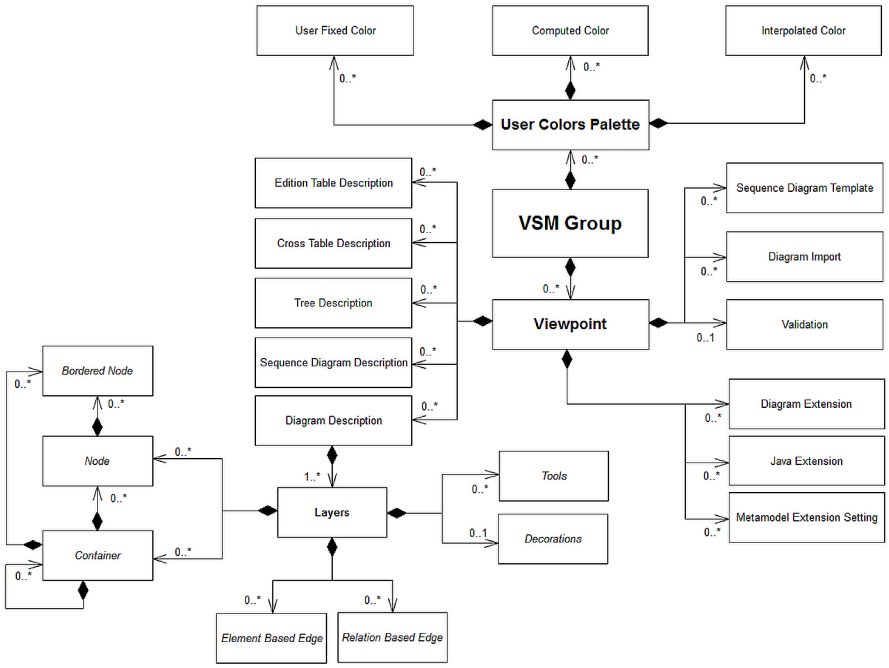
podrazumijevani sloj (*defaut layer*) koji je uvijek omogućen, te nijedan, jedan ili više dodatnih slojeva (koje je moguće omogućiti/onemogućiti u zavisnosti od potrebe), i



Slika 4.2.2. Primjer VSM-a otvorenog u *Sirius* editoru specifikacija

- ***grafičke elemente*** (*Graphical Vocabulary*) – različiti tipovi grafičkih objekata koji su opisani konceptom mapiranja (selektovanje, pridruživanje semantičkim elementima metamodela, definisanje stilova i ponašanja). Grafički elementi se mogu podijeliti na:
  - ***čvorove*** (*Nodes*) – elementi koji ne sadrže druge elemente,
  - ***kontejnere*** (*Containers*) – element koji može da sadrži druge elemente, uključujući i druge kontejnere,
  - ***list kontejnere*** (*List containers*) – posebna vrsta kontejnera koji predstavlja sadržane elemente kao vertikalnu listu elemenata. List kontejneri nisu rekurzivni,
  - ***region kontejneri*** (*Region containers*) – posebna vrsta kontejnera koji predstavlja sadržane elemente kao vertikalni ili horizontalni niz pregrada (odjeljaka). Region kontejneri nisu rekurzivni i ne mogu sadržavati čvorove kao direktne sadržane elemente,
  - ***regioni*** (*Region*) – posebna vrsta kontejnera koja predstavlja direktan element za region kontejnere,
  - ***čvorovi bordure*** (*Bordered nodes*) – elementi koji se ponašaju kao normalni čvorovi, ali se mogu pojavljivati samo na bordurama (granicama) drugog elementa, i
  - ***veze*** (*Edges*) – element koji predstavlja vezu između čvorova i elemenata.

Na slici (Slika 4.2.3.) prikazana je djelimična arhitektura VSM modela i njegovih gradivnih elemenata.



Slika 4.2.3. Djelimična arhitektura VSM modela

Za tri objekta u SVM-u mogu biti definisani grafički stilovi, odnosno grafička reprezentacija na dijagramu:

- ***Stilovi čvorova*** (*Node Styles*) – definišu kako će čvorovi biti prikazani na dijagramu. Sirius podržava osnovne i složene grafičke objekte za prikaz: *pravougaonik* (*square*), *paralelopiped* (*diamond*), *elipsa* (*ellipse*), *tačka* (*dot*), *stiker* (*note*), *slika* (*image*), *proizvoljno definisan stil* (*custom style*).
- ***Stilovi kontejnera*** (*Container Styles*) – definišu kako će kontejneri biti prikazani na dijagramu. Tri su osnovna stila za vizuelizaciju kontejnera: *gradijent* (*gradient*), *paralelogram* (*parallelogram*) i *slika* (*image*).
- ***Stilovi veze*** (*Edge Style*) – definišu kako će veze biti prikazane na dijagramu. Tri osnovna stila za vizuelizaciju veza su: *direktne linije* (*straight*), *linije sa pravougaonim skretanjima* (*manhattan*) i *stabla* (*tree*).

4.2.1. Kreiranje interaktivnog grafičkog editora

Da bi se kreirao interaktivni grafički editor pomoću *Sirius* okvira, potrebno je za predstavljeni metamodel definisati VSP unutar kojeg će biti kreiran VSM sa objektima i pravilima mapiranja elemenata metamodela. Prilikom kreiranja grafičkog editora pomoću *Sirius* okvira potrebno je razmotriti dva ključna elementa:

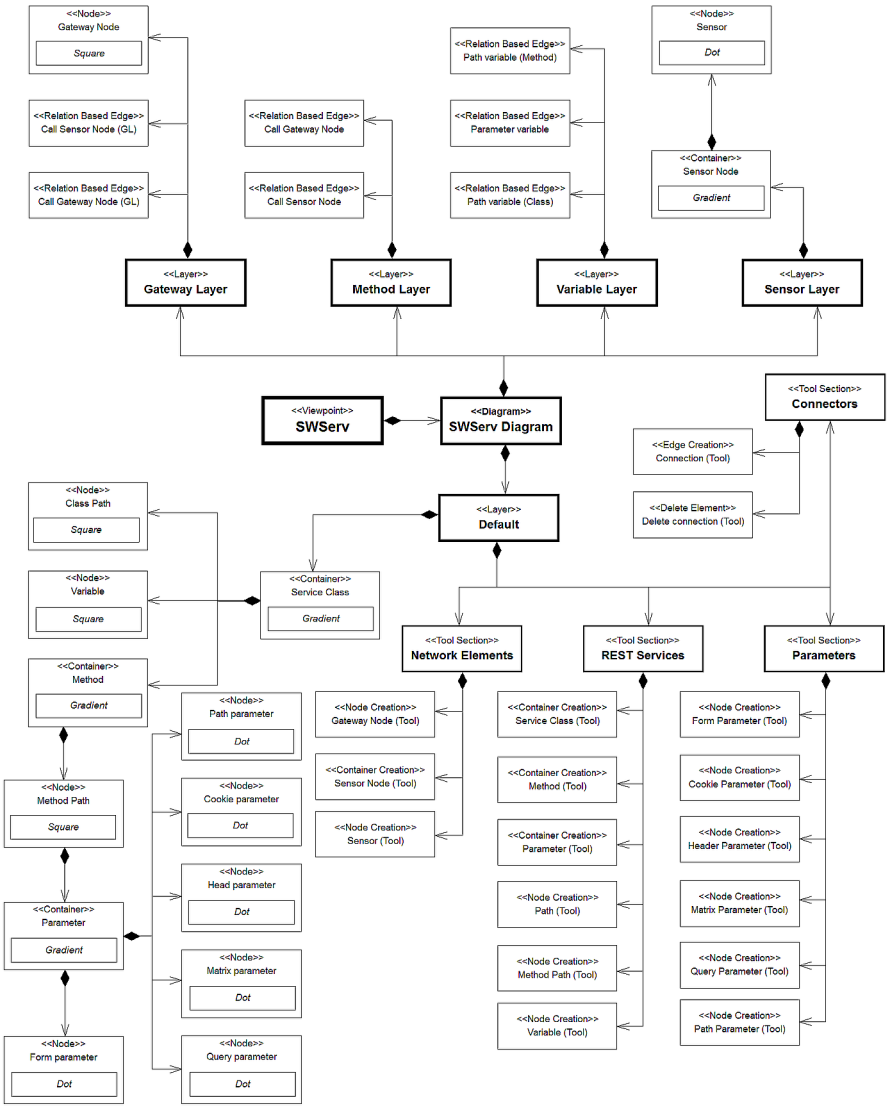
- dizajn objekata – izbor odgovarajućeg semantičkog stila prikaza elemenata i veza između elemenata na dijagramu, i
- funkcionalnost editora – ponašanje editora, alata i kreiranih objekata.

U VSM-u interaktivnog grafičkog editora kreiran je pogled (*viewpoint*) koji služi kao okvir za druge grafičke elemente i definiše semantički tip resursa koji će biti podržan ovim modelom, kao i dijagram *SWServ*, koji obezbjeđuje kreiranje modela i prikaz elemenata arhitekture servisnog sloja. Na slici (Slika 4.2.1.1.) je prikazana arhitektura predloženog VSM modela.

U sklopu dijagrama *SWServ* definisano je pet slojeva prikaza, od kojih je jedan unaprijed predodređen (default). Predloženi slojevi su podijeljeni u zavisnosti od semantike elemenata prikaza, što u korisničkoj verziji interaktivnog editora može obezbijediti lakšu čitljivost modela i bolji prikaz objekata dijagrama. Pet definisanih slojeva su:

- ***Unaprijed predodređeni*** sloj (*Default*) – obezbjeđuje prikaz klasa servisa, metoda, putanja, putanja metoda, varijabli i parametara. Također definiše sekcije alata:
  - ***mrežni elementi*** (*Network Elements*) – alati za kreiranje senzora, gateway i senzorskih čvorova.
  - ***REST servisi*** (*REST Service*) – alati za kreiranje klasa servisa, metoda, parametara, putanja, putanja metoda i varijabli.
  - ***parametri*** (*Parameters*) – alati za kreiranje parametara.
  - ***konektori*** (*Connectors*) – alati za kreiranje i brisanje veza između objekata (metoda, gateway i senzorskih čvorova).
- ***Gateway sloj*** (*Gateway Layer*) – objedinjuje elemente za vuzuelizaciju *gateway* čvorova i veza između: *gateway* – *gateway* čvorova i *gateway* – senzorskih čvorova.
- ***Sloj metoda*** (*Method Layer*) – obezbjeđuje elemente za prikaz veza između metod – *gateway* čvorova, te metod – senzorskih čvorova.
- ***Sloj varijabli*** (*Variable Layer*) – obezbjeđuje prikaz veza putanja – varijabla, kao i veze parametar – varijabla.

- ***Senzorski sloj*** (*Sensor Layer*) – definiše prikaz senzora i senzorskih čvorova.



Slika 4.2.1.1. Arhitektura predloženog VSM modela

U tabeli (Tabela 4.2.1.1.) prikazano je preslikavanje metaklasa definisanog metamodela na elemente i slojeve VSM modela u kojima je preslikavanje izvršeno.

Tabela 4.2.1.1. Preslikavanje metaklasa na elemente VSM modela

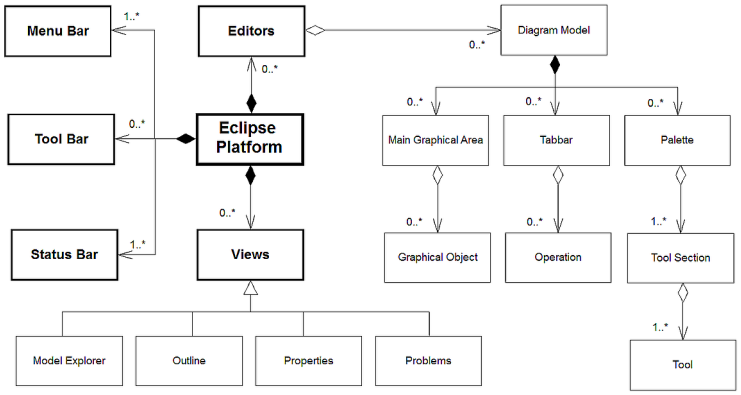
Naziv metaklase	Preslikavanje		
	Tip objekta	Sloj	Prikaz
AttribParamType	-	-	-
CookieParam	Node	Default	Prikazuje metaklasu CookieParam kao elementarni čvor sa *Dot* grafičkim prikazom bez slike.
FormParam	Node	Default	Prikazuje metaklasu FormParam kao elementarni čvor sa *Dot* grafičkim prikazom bez slike.
GatewayNode	Node	Gateway	Prikaz GatewayNode metaklase kao elementarni čvor sa *Square* grafičkim prikazom sa slikom.
HeaderParam	Node	Default	Prikazuje metaklasu HeaderParam kao elementarni čvor sa *Dot* grafičkim prikazom bez slike.
MatrixParam	Node	Default	Prikazuje metaklasu MatrixParam kao elementarni čvor sa *Dot* grafičkim prikazom bez slike.
Method	Container	Default	Prikaz Method metaklase kao kontejnerskog elementa sa *Gradient* stilom prikaza.
Parameter	-	-	-
ParamType	-	-	-
Path	Bordered Node	Default	Prikaz Path metaklase kao elementarni čvor sa *Square* grafičkim prikazom sa slikom.
PathParam	Node	Default	Prikazuje metaklasu PathParam kao elementarni čvor sa *Dot* grafičkim prikazom bez slike.
QueryParam	Node	Default	Prikazuje metaklasu QueryParam kao elementarni čvor sa *Dot* grafičkim prikazom bez slike.
Sensor	Node	Sensor	Prikaz Sensor metaklasa kao elemetarnih čvorova sa *Dot* grafičkim prikazom sa slikom.
SensorNode	Container	Sensor	Prikaz ServiceNode metaklase kao kontejnerskog elementa sa uslovnim *Gradient* stilom prikaza.
ServiceClass	Container	Default	Prikaz ServiceClass metaklase kao kontejnerskog elementa sa uslovnim *Gradient* stilom prikaza.
Service	Diagram	-	Definiše dijagram za prikaz modela.
Variable	Node	Default	Prikaz Variable metaklase kao elemetarni čvor sa *Square* grafičkim prikazom sa slikom.

4.2.2. Interaktivni grafički editor

Interaktivni grafički editor kreiran sa *Sirius* okvirom značajno zavisi od *Eclipse* platforme i sastoji se od niza alata (editora i pogleda) koji omogućavaju korisnicima da kreiraju, uređuju i vizuelizuju EMF metamodele. Na slici (Slika 4.2.2.1.) prikazana je arhitektura *Sirius* baziranog editora koja se oslanja na *Eclipse* platformu i obuhvata sve pomenute elemente.

Korisnici na dva načina mogu pokrenuti kreirani editor [Vujović14b]: kao proširenje *Eclipse* platforme ili kao samostalni editor (samostalno Eclipse okruženje), međutim u oba slučaja editor se pokreće u perspektivi za modelovanje (*Modeling Perspective*) koja pruža sve potrebne poglede,

čarobnjake i menije. Na slici (Slika 4.2.2.2.) prikazan je primjer perspektive za modelovanje koja predstavlja podrazumijevane poglede i editor dijagrama.



Slika 4.2.2.1. Arhitektura *Sirius* baziranog editora

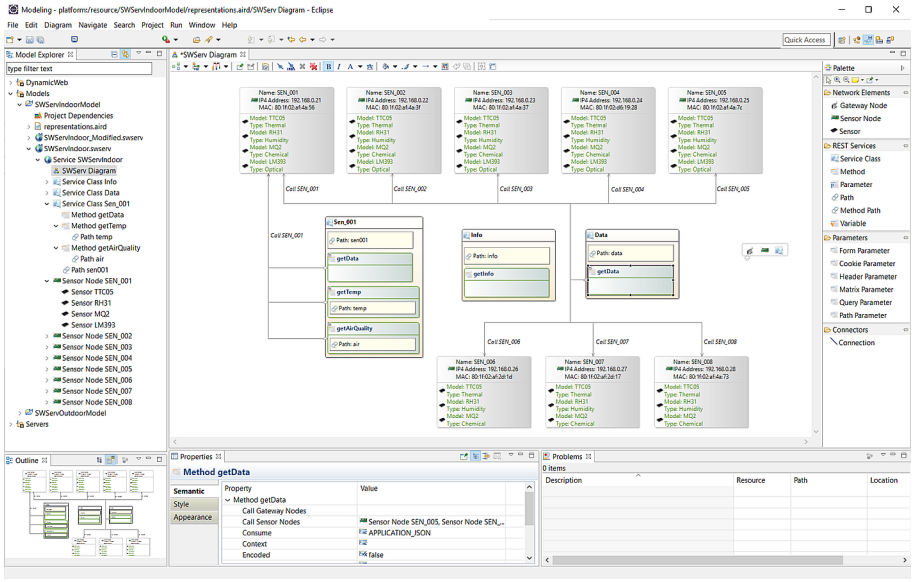
Podrazumijevani pogledi u perspektivi za modelovanje su [Sirius, Vujović14a, Vujović14b]:

- ***Pretraživač modela*** (*Model Explorer*) – glavni pogled za interakciju sa modelima koji prikazuje projekte i objekte koje sadrže. Također dozvoljava direktan pregled i manipulaciju semantičkih modela i njihovih prikaza.
- ***Slikovni prikaz*** (*Outline*) – omogućava strukturni pogled na trenutno otvoreni dokument ili model (minijaturni prikaz dijagrama sa mogućnošću navigacije).
- ***Svojstva*** (*Properties*) – detaljne informacije trenutno izabranog elementa (opcije i uređivanje zavise od izabranog elementa).
- ***Problemi*** (*Problems*) – prikazuje informacije različitog stepena ozbiljnosti (informacije, upozorenja ili greške) koje se koriste za validaciju modela.

Editor dijagrama se sastoji od tri oblasti [*Sirius*]:

- ***canvas*** (*Main graphical area*) – služi za prikaz i interakciju sa elementima dijagrama,
- ***paleta alata*** (*Palette*) – obezbjeđuje pristup dodatnim alatima (pristup alatima za kreiranje novih elemenata dijagrama), i
- ***traka operacija*** (*Tabbar*) – obezbjeđuje dodatne, uopštene operacije.

Interakciju sa dijagramom i njegovim elementima također je moguće ostvariti koristeći pogled svojstava i kontekstualni meni, dostupni za dijagram i selektovane elemente, kao i dijagram meni (*Diagram menu*) koji obezbjeđuje pristup svim opcijama i alatima editora.

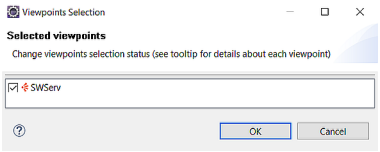


Slika 4.2.2.2. *Sirius* perspektiva za modelovanje

Kako su *Sirius* editori bazirani na *Eclipse* platformi, tako se perspektive mogu prilagođavati dodavanjem, pomjeranjem ili uklanjanjem pogleda, prečica, itd. Puna kontrola nad editorom se može postići korištenjem poznatih *Eclipse* funkcija, svojstava i prečica na tastaturi. Za kreiranje ili uređivanje dijagrama, pored *Sirius* editora dijagrama, moguće je koristiti i standardni EMF editor. U ovom slučaju sinhronizacija se vrši automatski.

U *Sirius*-u, projekat za modelovanje (*Modeling Project*) je odgovoran za organizaciju i upravljanje modelima i reprezentacijama, kao i za čuvanje podataka (dijagrama, tabela, objekata, itd.) u posebnim datotekama sa **.aird* ekstenzijom. **.aird* datoteke sadrže podatke potrebne za prikaz dijagrama (ili drugih reprezentacija) ali ne i semantičke podatke koji se čuvaju u samim modelima [Sirius]. Svaki projekat za modelovanje ima niz pogleda koji su omogućeni i koji kontrolišu koja reprezentacija može biti kreirana nad semantičkim modelom. Ovi pogledi definišu prirodu projekta i mogu jednostavno biti promijenjeni izborom drugog pogleda.

Na slici (Slika 4.2.2.3.) je prikazan dijalog za izbor pogleda u projektu koji se poziva iz kontekst menija koristeći akciju *Izbor pogleda* (*Viewpoint Selection*), a prikazuje sve poglede koji su kompatibilni sa projektom, te dozvoljava njihovo omogućavanje ili onemogućavanje.



Slika 4.2.2.3. Dijalog za izbor pogleda

Interaktivni editor kreiran u *Sirius*-u podržava različite rukovaoce rasporeda komponenti (*Layout manager*) koji obezbjeđuju pozicioniranje i prikaz komponenti i relacija, a po potrebi mogu biti prilagođeni. Koristeći pojavne menije (*Popup Bars*), u zavisnosti od konteksta selekcije, elementi na dijagramu mogu biti brzo dodati, što znatno ubrzava proces kreiranja i dizajna. Operacijom prikaza slojeva, lako se mogu sakriti ili prikazati svi elementi dijagrama koji pripadaju samo jednom izabranom sloju, što je korisno u slučaju dijagrama sa velikim brojem elemenata.

4.3. Generisanje kôda servisnog sloja

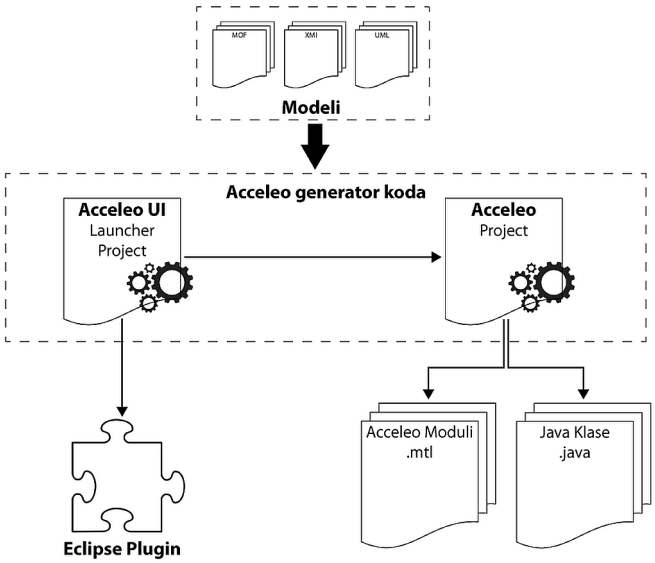
U poglavlju 2. je predstavljen značaj generisanja kôda u MDA i DSM procesu razvoja softvera, odnosno procesu transformacije M2M ili M2T, kao i predlog alata i okruženja za izvršenje ovog zadatka. Osnovni zahtjev koji se može postaviti pred generator kôda je jednostavno inkorporiranje generisanih klasa, uz minimalnu programsku modifikaciju, u dinamički JAVA projekat (postojeći ili novi). Na ovaj način se obezbjeđuje brzo kreiranje servisa koji su sposobni za integraciju sa postojećim servisnim arhitekturama.

Pored ranije navedenih prednosti *Acceleo* generatora, potrebno je dodatno istaći i integraciju kreiranog rješenja sa *Eclipse* platformom, odnosno njegovo uključivanje u strukturu razvojnog okruženja.

Kako *Acceleo* predstavlja šablonski baziran generator kôda, neophodno je definisati šablone koji će služiti za preslikavanje modela u procesu transformacije. U tu svrhu *Acceleo* može koristiti dvije vrste datoteka koji sadrže opis šablona:

- *Acceleo Module File* (.mtl) – predstavlja modul koji sadrži definisanu strukturu šablona i pravila preslikavanja, a ujedno obezbjeđuje i pristup elementima metamodela. Preslikavanje metamodela se izvršava na osnovu *Acceleo* definisane sintakse. U svakom *Acceleo* projektu mora postojati jedan *main* modul.
- *Java File* (.java) – predstavlja standardnu Java klasu koja može pristupiti elementima modela i vršiti naprednu obradu. Java klase se pozivaju iz *Acceleo* modula.

Arhitektura rješenja baziranog na *Acceleo* generatoru, oslanja se na dva elementa (Slika 4.3.1.), odnosno:



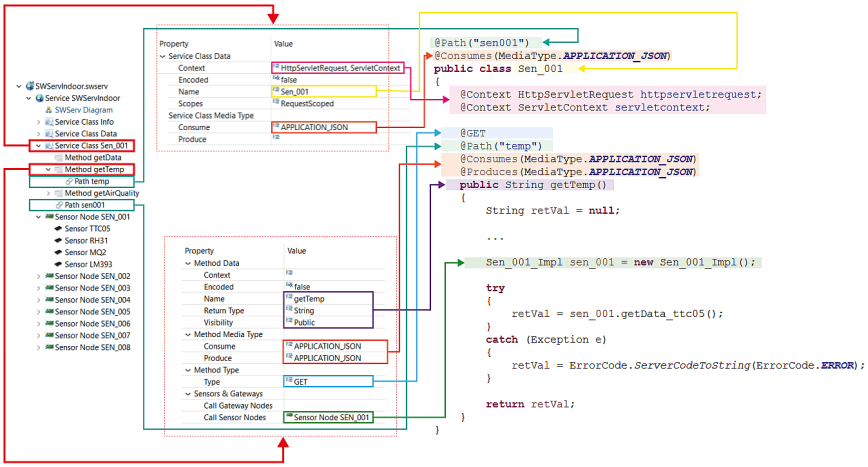
Slika 4.3.1. Arhitektura *Acceleo* generatora

- *Acceleo Project* – projekat koji definiše šablone za transformaciju i pomoćne Java klase,
- *Acceleo UI Launcher Project* – projekat koji definiše integraciju *Acceleo Project-a* sa *Eclipse* razvojnim okruženjem (osigurava proširenje *Eclipse* platforme sa kreiranim generatorom kôda).

4.3.1. Implementacija šablona za generisanje

Oslanjajući se na definisani metamodel i izabrani implementacioni okvir (JAX-RS) servisnog sloja, potrebno je prepoznati adekvatne

elemente transformacije i obezbijediti način na koji se kreirani model može prevesti u implementacione klase (Slika 4.3.1.1.).



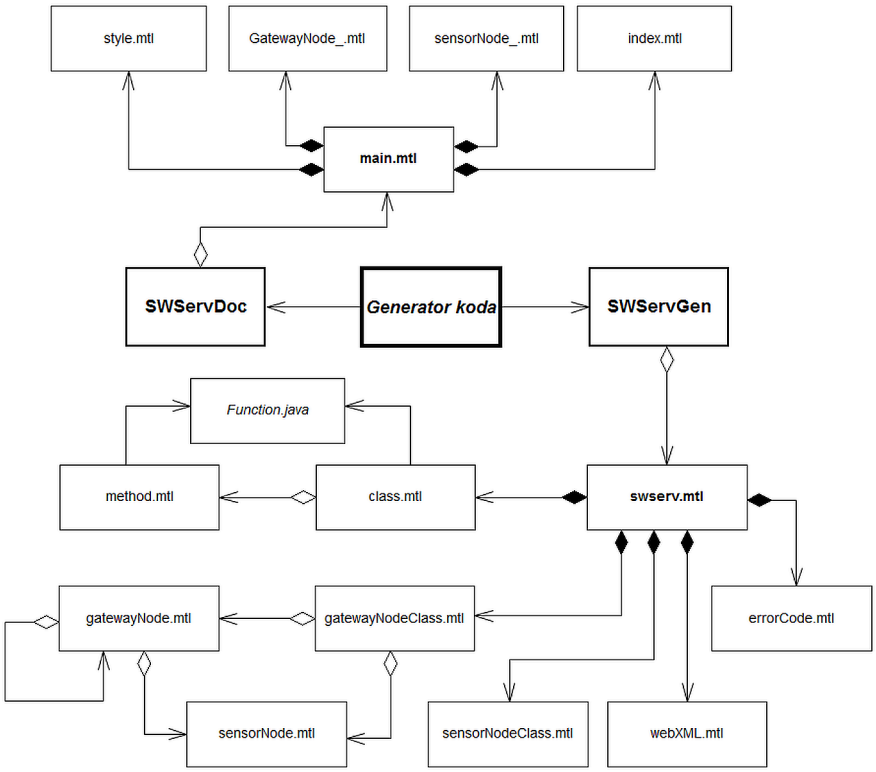
Slika 4.3.1.1. Elementi transformacije

Na osnovu modela neophodno je generisati tri grupe datoteka:

- datoteke koji obezbjeđuju postavku servisa (*web.xml*) – obezbjeđuju parametre neophodne za postavljanje i pokretanje servisa na serveru,
- datoteke koji predstavljaju opis sistema (*index.html*, *style.css*, *SEN_XXX.html*, *GAT_XXX.html*) – statički elementi servisa koji opisuju kreirani servisni sloj na osnovu parametara definisanih u modelu,
- implementacione datoteke servisa (*ErrorCode.java*, *Sen_XXX_-Impl.java*, klase servisa, npr.: *Info.java*, *Data.java*, *Temp.java* i dr.) – Java klase koje implementiraju servisni sloj u JAX-RS implemetacionom okviru.

Predloženo rješenje sadrži podršku: generisanju dokumentacije, i generisanju postavki i implementacionog kôda servisnog sloja. Ovako predloženi segmenti programskog rješenja impliciraju formulisanje dva *Acceleo* projekta koji sadrže definicije šablona za transformaciju (*SWServDoc* i *SWServGen*).

Na slici (Slika 4.3.1.2.) je prikazan model arhitekture *Acceleo* generatora, dok je u tabeli (Tabela 4.3.1.1.) dat pregled definisanih šablona na osnovu kojih se vrši transformacija kôda u implementacione klase.



Slika 4.3.1.2. Arhitektura predloženog generatora kôda

Tabela 4.3.1.1. Definisani šabloni u predloženom generatoru kôda

Naziv šablona	Segment	Opis
`gatewayNode.mtl`	`Dokumentacija`	Šablon za generisanje .html stranica sa osnovnim podacima za geteway čvorove.
`index.mtl`	`Dokumentacija`	Šablon za generisanje indeksne .html stranice sa osnovnim podacima o servisu, senzorskim i gateway čvorovima.
`main.mtl`	`Dokumentacija`	Šablon koji predstavlja main modul za *SWServDoc* projekat. Osigurava uključivanje i pozive svih ostalih modula generatora.
`sensorNode.mtl`	`Dokumentacija`	Šablon za generisanje .html stranica sa osnovnim podacima za senzorske čvorove.
`style.mtl`	`Dokumentacija`	Šablon koji generiše statički dio kôda namijenjen formatiranju prikaza .html stranica.
`class.mtl`	`Implementacija`	Šablon za generisanje klasa servisa na osnovu JAX-RS specifikacije. Također odgovoran za pozivanje ostalih modula i java kôda.
`errorCode.mtl`	`Implementacija`	Šablon koji generiše statički dio kôda namijenjen obradi izuzetaka.

Function.java	Implementacija	Java implementacija nadovezivanja elemenata sa formatiranim tekstom (operacija nije podržana *Acceleo* sintaksom).
gatewayNode.mtl	Implementacija	Pomoćni šablon za generisanje gateway čvorova u hijerarhiji sa gateway čvorovima (rekurzivni pozivi).
gatewayNodeClass.mtl	Implementacija	Šablon za preslikavanje metaklasa GatewayNode na implementacionu klasu koja predstavlja gateway čvor. Također je odgovoran za kreiranje hijerarhije senzorskih i gateway čvorova.
method.mtl	Implementacija	Šablon za generisanje metoda klasa na osnovu JAX-RS specifikacije. Također odgovoran za pozivanje java kôda.
sensorNode.mtl	Implementacija	Pomoćni šablon za generisanje senzorskih čvorova u hijerarhiji sa gateway čvorovima.
sensorNodeClass.mtl	Implementacija	Šablon za preslikavnje metaklasa SensorNode na implementacionu klasu koja predstavlja senzorski čvor.
swserv.mtl	Implementacija	Šablon koji predstavlja main modul za *SWServGen* projekat. Osigurava uključivanje i pozive svih ostalih modula generatora.
webXML.mtl	Implementacija	Šablon za generisanje parametara servisa.

Preslikavanje metaklasa iz metamodela na implementacione klase može biti u omjeru 1:1 (jedan-na-jedan), odnosno da se jedna metaklasa preslikava u jedan implementacioni objekat. Međutim, ponekad, zbog kompleksnosti problema omjer preslikavanja može biti i drugačiji. Za definisani metamodel, u tabeli (Tabela 4.3.1.2.) je prikazano preslikavanje metaklasa na implementacione objekte, što ujedno diktira i njihov omjer preslikavanja.

Tabela 4.3.1.2. Preslikavanje metaklasa na implementacione objekte

Naziv metaklase	Preslikavanje		
	Implement.	Dokument.	Šablon
AttribParamType	-	-	-
CookieParam	Preslikava se na parametre metode definisane u Java klasama.	-	method.mtl
FormParam	Preslikava se na parametre metode definisane u Java klasama.	-	method.mtl
GatewayNode	Preslikava se na Java klasu koja obezbjeđuje metode za pristup gateway uređajima i njihovim metodama.	Preslikava se u Web stranicu koja opisuje gateway čvor.	gatewayNodeClass.mtl, gatewayNode.mtl, class.mtl, method.mtl, gatewayNode_.mtl, index.mtl
HeaderParam	Preslikava se na parametre metode definisane u Java klasama.	-	method.mtl
MatrixParam	Preslikava se na parametre metode definisane u Java klasama.	-	method.mtl
Method	Preslikava se na metode servisa definisane u klasama servisa.	-	class.mtl, method.mtl,
Parameter	Preslikava se na parametre me-	-	method.mtl

	tode definisane u Java klasama.		
ParamType	-	-	-
Path	Preslikava se na putanju zadatu anotacijama na nivou klasa ili metoda.	-	Function.java, class.mtl, method.mtl
PathParam	Preslikava se na parametre metode definisane u Java klasama.	-	method.mtl
QueryParam	Preslikava se na parametre metode definisane u Java klasama.	-	method.mtl
Sensor	Preslikava se u metode senzorske klase i obezbjeđuje pristup senzorskim elementima.	Preslikava se u elemente Web stranice koja opisuje senzorski čvor.	sensorNodeClass.mtl, sensorNode.mtl, sensorNode_.mtl
SensorNode	Preslikava se na Java klasu koja obezbjeđuje metode za pristup senzorskim čvorovima i njihovim senzorskim elementima.	Preslikava se u Web stranicu koja opisuje senzorski čvor.	sensorNodeClass.mtl, gatewayNodeClass.mtl, sensorNode.mtl, gatewayNode.mtl, class.mtl, method.mtl, sensorNode_.mtl, gatewayNode_.mtl, index.mtl
ServiceClass	Preslikava se na Java klasu koja obezbjeđuje pristup i metode servisima.	-	class.mtl,
Service	Preslikava se u postavke servisa.	Preslikava se u opis pristupa servisu na indeksnoj Web stranici.	webXML.mtl, swserv.mtl, main.mtl, index.mtl
Variable	Preslikava se na varijable zadate unutar putanja koje su zadate anotacijama na nivou klasa ili metoda.	-	Function.java,

4.3.1.1. Primjeri implementacije šablona

U cilju ilustrovanja mehanizama transformacije u daljenjm tekstu su prikazani odabrani primjeri *Acceleo* šablona.

sensorodeClass.mtl pripada projektu *SWServGen* i predstavlja *Acceleo* modul koji transformiše `SensorNode` metaklase modela u Java klase. Prilikom kreiranja šablona, Acceleo sintaksa se oslanja na metamodel modela, čime se osigurava direktna kontrola elemenata modela i pristup njegovim svojstvima:

```
[comment encoding = UTF-8 /]
[module sensorNodeClass('http://swserv/1.0')]

[template public generateSensorNodeClass(aSensorNode : SensorNode)]

[file (aSensorNode.Name.toLower().toUpperFirst().concat('_Impl.java'), false,
'UTF-8')]
package impl;

import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.UriBuilder;
```

```
public class [aSensorNode.Name.toLower().toUpperFirst().concat('_Impl')/]
{
    private static String uriAddress =
            "http://[aSensorNode.IP4Address/]:8080/RPISensorWeb";

    public String getData()
    {
        WebTarget service = ClientBuilder.newClient()
                .target(UriBuilder.fromUri(uriAddress).build());
        return service.path("sw").path("info")
                .request(MediaType.APPLICATION_JSON).get(String.class);
    }

[for (aSensor : Sensor | aSensorNode.sensors) separator ('\r')]
    public String getData_[aSensor.Model.toLowerCase()/]()
    {
        WebTarget service = ClientBuilder.newClient()
                .target(UriBuilder.fromUri(uriAddress).build());
        return service.path("sw").path("[aSensor.Model.toLowerCase()/]")
                .request(MediaType.APPLICATION_JSON).get(String.class);
    }

    public String getData_[aSensor.Model.toLowerCase()/]_data()
    {
        WebTarget service = ClientBuilder.newClient()
                .target(UriBuilder.fromUri(uriAddress).build());
        return service.path("sw").path("[aSensor.Model.toLowerCase()/]")
                .path("data").request(MediaType.APPLICATION_JSON)
                .get(String.class);
    }
[/for]
}
[/file]

[/template]
```

Za razliku od prethodog modula, *sensorNode.mtl* pripada projektu *SWServDoc* i predstavlja *Acceleo* modul koji transformiše `SensorNode` metaklase modela u HTML dokumentaciju koja će biti dostupna na servisu, a opisuje senzorski čvor, njegove parametre i ugrađene senzorske elemente:

```
[comment encoding = UTF-8 /]
[module sensorNode('http://swserv/1.0')]

[template public generateSensorNode(aSensorNode : SensorNode)]

[file (aSensorNode.Name.toLower().toUpperFirst().concat('.html'), false, 'UTF-
8')]
<!DOCTYPE html>
<html><head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta charset="UTF-8">
<link rel="stylesheet" type="text/css" href="style.css">

<title>[aSensorNode.Name/]</title>
</head>
<body>
<h1>[aSensorNode.Name/]</h1>
<h2>Informacije o servisu</h2>
<div class="info">
    <div class="holder">
        <div class="label">Tip cvora:</div>
        <div class="data">[aSensorNode.Type/]</div>
```

```
    </div>
    <div class="holder">
        <div class="label">IP4 adresa:</div>
        <div class="data">[aSensorNode.IP4Address/]</div>
    </div>
    <div class="holder">
        <div class="label">IP6 adresa:</div>
        <div class="data">[aSensorNode.IP6Address/]</div>
    </div>
    <div class="holder">
        <div class="label">MAC adresa:</div>
        <div class="data">[aSensorNode.MACAddress/]</div>
    </div>
    <div class="holder">
        <div class="label">GPS koordiate:</div>
        <div class="data">[aSensorNode.GPS/]</div>
    </div>
</div>
<hr>
<h2>Senzori</h2>
[for (aSensor : Sensor | aSensorNode.sensors)]
<div class="sensor">
    <div class="holder">
        <div class="label">Model senzora:</div>
        <div class="data">[aSensor.Model/]</div>
    </div>
    <div class="holder">
        <div class="label">Serijski broj:</div>
        <div class="data">[aSensor.SerialNumber/]</div>
    </div>
    <div class="holder">
        <div class="label">Tip senzora:</div>
        <div class="data">[aSensor.Type/]</div>
    </div>
</div>
[/for]
</body></html>
[/file]
[/template]
```

webXML.mtl iz projekta *SWServGen* predstavlja *Acceleo* modul koji generiše XML fajl sa parametrima koji su potrebni za integraciju servisa u servisnu arhitekturu:

```
[comment encoding = UTF-8 /]
[module webXML('http://swserv/1.0')]

[template public generateWebXML(aService : Service)]

[file ('web.xml', false, 'UTF-8')]
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID"
version="3.1">
    <display-name>[aService.Name/]</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.htm</welcome-file>
        <welcome-file>default.jsp</welcome-file>
    </welcome-file-list>
```

```
    <servlet>
        <servlet-name>[aService.Name.replaceAll(' ', '')/]</servlet-name>
        <servlet-class>
            org.glassfish.jersey.servlet.ServletContainer
        </servlet-class>
        <init-param>
            <param-name>jersey.config.server.provider.packages</param-name>
            <param-value>service</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>[aService.Name.replaceAll(' ', '')/]</servlet-name>
        <url-pattern>/[aService.Sufix.toLower()/]/*</url-pattern>
    </servlet-mapping>
</web-app>
[/file]
[/template]
```

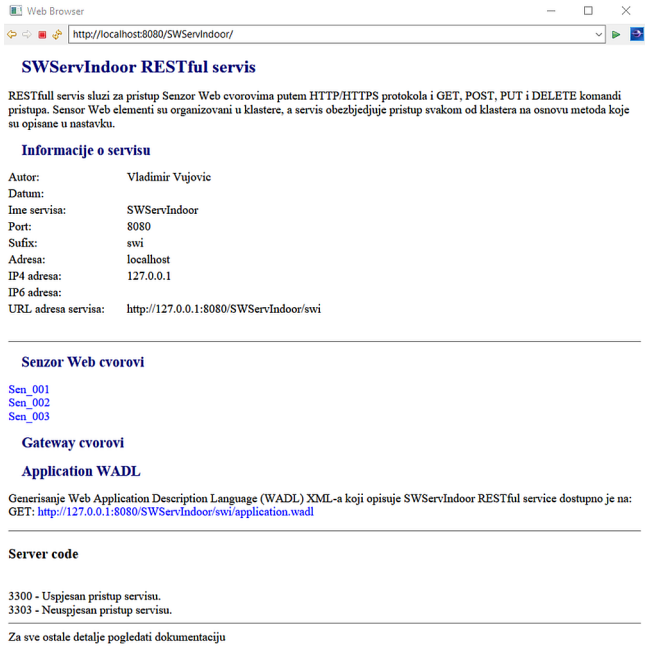
U projektu *SWServGen* definisana je Java klasa kao proširenje funkcionalnosti *Acceleo* modula, koja radi spajanje i izmjenu tekstualnih elemenata koji su definisani u modelu:

```
package SWServGen.files;

import org.eclipse.emf.common.util.EList;

import swserv.Path;
import swserv.Variable;

public class Function
{
    public String path(Path path)
    {
        Path tmpPath = path;
        String retVal = tmpPath.getPath().trim();

        EList<Variable> var = tmpPath.getVars();

        if(var.size() != 0)
        {
            for(Variable v: var)
            {
                String expression = v.getName().trim();

                if(v.getExpression() != null)
                {
                    if(!v.getExpression().equals(""))
                    {
                        expression = expression + ":" + v.getExpression().trim();
                    }
                }

                retVal = retVal.replaceAll(String.valueOf(v.getVarID()),
                                           expression);
            }
        }

        return retVal;
    }
}
```

U nastavku su prikazani rezultati generisanja, kao i elementi koje je neophodno manuelno prilagoditi.

4.3.2. Generisani elementi

Za *Acceleo* projekte, koji obezbjeđuju šablone za generisanje kôda, kreirana su dva projekta koja omogućavaju njihovu integraciju sa *Eclipse* razvojnim okruženjem. Generatori kôda se pozivaju iz kontekstnog menija modela i korisnicima pružaju izbor projekta u kojemu će biti generisani datoteke sistema. Inkorporiranjem generisanih datoteka u dinamički JAVA prjekat, njegovim postavljanjem i pokretanjem na serveru, pozivom servisa se dobija HTML stranica sa osnovnom dokumentacijom (Slika 4.3.2.1.).



Slika 4.3.2.1. Generisana HTML dokumentacija servisa prikazana u Web pregledniku

Na slici (Slika 4.3.2.1.) se može vidjeti indeksna stranica testnog servisa koja je kreirana generatorom kôda bez dodatnih modifikacija, a sadrži osnovne podatke o servisu, njegovim senzorskim i gateway čvorovima. Za generisanje je korišten `index.mtl` šablon.

Za razliku od dokumentacije (HTML stranica), implementacionei datoteke generišu Java kôd. U nastavku je prikazana implementaciona Java

klasa za senzorski čvor koji ima četiri senzorska elementa (zbog veličin kôda prikazan je segment klase):

```
package impl;

import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.UriBuilder;

public class Sen_001_Impl
{
    private static String uriAddress = "http://192.168.0.21:8080/RPISensorWeb";

    public String getData()
    {
        WebTarget service = ClientBuilder.newClient()
                            .target(UriBuilder.fromUri(uriAddress).build());
        return service.path("sw").path("info")
                      .request(MediaType.APPLICATION_JSON).get(String.class);
    }

    public String getData_ttc05()
    {
        WebTarget service = ClientBuilder.newClient()
                            .target(UriBuilder.fromUri(uriAddress).build());
        return service.path("sw").path("ttc05")
                      .request(MediaType.APPLICATION_JSON).get(String.class);
    }

    public String getData_ttc05_data()
    {
        WebTarget service = ClientBuilder.newClient()
                            .target(UriBuilder.fromUri(uriAddress).build());
        return service.path("sw").path("ttc05").path("data")
                      .request(MediaType.APPLICATION_JSON).get(String.class);
    }

    ...

    /*
    *po dvije generisane metode za svaki senzor:
    *getData_IME-SENZORA()
    *getData_IME-SENZORA_data()
    */

}
```

Za generisanje predstavljenog kôda korišten je `sensorNodeClass.mtl` šablon.

Predloženo rješenje automatski transformiše kreirani model u imlementacione klase, međutim, zbog prirode problema, određene klase je potrebno prilagoditi i manuelno modifikovati prije postavljanja na server. Dijelovi kôda koje treba implementirati unutar klasa su obilježeni `//TODO` markerom koji je vidljiv u *Eclipse* razvojnom okruženju. U nastavku je prikazana klasa koja definiše servis, a generisana je na osnovu `class.mtl` šablona:

```java
package service;

import javax.ws.rs.*;
import javax.ws.rs.core.*;

import helper.ErrorCode;

import impl.Sen_001_Impl;
import impl.Sen_002_Impl;

import net.sf.json.*;

@Path("data")
public class Data
{
    @GET
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public String getData()
    {
        String retVal = null;
        JSONObject retValJSON = new JSONObject();

        /*
         *JSONObject jsonReverse = null;
         *jsonReverse = (JSONObject) JSONSerializer.toJSON();
         */

        Sen_001_Impl sen_001 = new Sen_001_Impl();
        Sen_002_Impl sen_002 = new Sen_002_Impl();
        ...

        try
        {
            //TODO Implement function!
        }
        catch (Exception e)
        {
            retVal = ErrorCode.ServerCodeToString(ErrorCode.ERROR);
        }

        return retVal;
    }
}
```

Poglavlje 5

Eksperimentalna verifikacija razvojnog okruženja

Kako je postavljeni cilj disertacije vezan za unapređenje procesa razvoja arhitekture sistema baziranih na Senzor Web mrežama, uz oslonac na dinamičko generisanje servisnog sloja, potrebno je izvršiti eksperimentalnu provjeru odabranog pristupa i karakteristika razvijenog DSM okruženja za podršku projektovanju arhitekture SW mreža.

Na osnovu analize domena problema, koja je prikazana u poglavlju 2. ove disertacije, moguće je prepoznati dvije osnovne kategorije okruženja u kojima se primjenjuju SM/SW mreže: unutrašnja (*Indoor*) i vanjska (*Outdoor*).

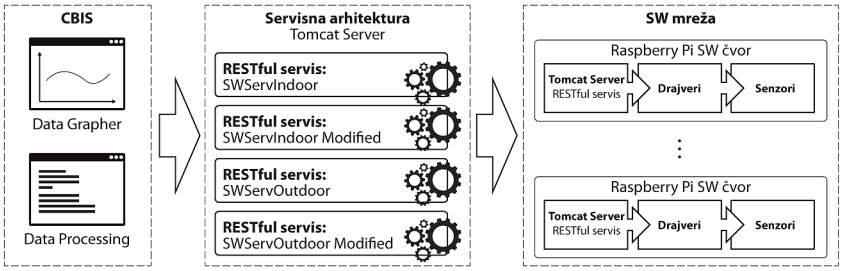
Prepoznate kategorije direktno diktiraju dva slučaja koja su odabrana za eksperimentalnu verifikaciju:

- *monitoring parametara životne sredine u zatvorenom prostoru* (*Indoor*), i
- *monitoring parametara životne sredine na ograničenom geografskom području* (*Outdoor*).

Na slici (Slika 5.1.) prikazan je model arhitekture okruženja koji je korišten za eksperimentalnu validaciju projektovanog programskog sistema za podršku modelom upravljanog razvoja arhitekture SW mreža. U skladu sa standardnim modelom SM, arhitekturu eksperimentalnog okruženja čine: SW mreža, Servisna arhitektura i softverski sistem (CBIS) koji ilustruje dva osnovna vida upotrebe prikupljenih podataka; obradu (*Data Processing*) i prezentacija (*Data Grapher*).

Sa stanovišta monitoringa, skup parametara koji definišu stanje životne sredine može značajno varirati u zavisnosti od njegove namjene. Bez obzira na fizikalnu prirodu parametara, koji su predmet praćenja u sklopu

eksperimentalne verifikacije, ključni problem predstavlja izbor tehnologije kojom će biti implementirani gradivni elementi na fizičkom nivou – SW čvorovi.



Slika 5.1. Arhitektura okruženja za eksperimentalnu validaciju projektovanog programskog sistema

Prilikom projektovanja SW čvorova neophodno je naći odgovor za sljedeća ključna pitanja: koji su to osnovni zahtjevi u pogledu hardvera i softvera, i kako izabrati adekvatnu implementaciju uz oslonac na raspoložive koncepte i tehnologije? Prethodno definisani ciljevi IoT sistema, koncepti SW, i multifunkcionalne programabilne platforme za *samostalni razvoj* - *Do It Yourself* (DIY) daju dovoljno prostora za pronalaženje odgovora na formulisana ključna pitanja.

Sprovođenje odabranih eksperimenata u realnim uslovima zahtjeva oslonac na konkretne implementacije SW čvorova. Proces kreiranja SW čvorova i širok repertoar mogućih tehnologija za njihovu izgradnju predstavljao je poseban izazov pri formiranju podloga za sprovedene eksperimente i zbog toga je detaljno elaboriran u sljedećem paragrafu.

5.1. Proces kreiranja Senzor Web čvorova

Razvojem IKT industrije i mikro-elektro-mehaničkih tehnologija, cijena hardverskih komponenti znatno opada, što rezultuje pojavljivanjem velikog broja različitih platformi baziranih na mikroprocesorima male potrošnje i malih dimenzija. Za razliku od mikrokontrolera koji su bili atraktivni do sredine '90-tih godina i koji su obavljali samo određenu, specifičnu funkciju, današnji mikroprocesori predstavljaju spoj više različitih elemenata i nazivaju se *sistemi na čipu – System on a chip* (SoC). Mikroprocesori najčešće obuhvataju jezgro (mikrokontroler, mikroprocesor), memoriju, oscilator (*clock*), eksterne interfejse (USB, Ethernet, SPI,

I^2C, UART), analogne interfejse i dr. U radu [Qian09] je izvršena komparativna analiza mikrokontrolera/mikroprocesora koji su danas najviše zastupljeni na tržištu, a neki od njih su: *Programmable Intelligent Computer* (PIC), Rabbit, AVR, *Advanced RISC Machine* (ARM) i ColdFire. Sa sigurnošću se može reći, da posljednjih godina ARM mikroprocesori postaju masovno rasprostranjeni jer se ugrađuju u jeftine DIY, koje zbog svoje relativno niske cijene i jednostavnosti programiranja uvode nove trendove i pružaju mogućnost da osobe koje imaju malo predznanja iz razvoja aplikacija, prilagode uređaje u skladu sa sopstvenim potrebama [Maksimović14]. Neke od najpoznatijih DIY platformi su Arduino, Raspberry Pi, Udoo, Phidgets i BeagleBone.

Za proces krairanja SW čvorova, neophodno je prepoznati minimalne hardverske i softverske zahtjeve koji bi osigurali implementaciju SW paradigme. Tako su u poglavlju 2. predstavljeni osnovni koncepti implementacije SW-a i njihove veze sa IoT paradigmom, što direktno utiče na izbor hardverske i softverske arhitekture, a na osnovu čega SW čvorovi treba da osiguraju:

- procesor i operativnu memoriju dovoljnu za obradu podataka i izvršavanje RESTful servisa,
- operativni sistem koji pruža jednostavno proširenje i implementaciju novih funkcionalnosti,
- jedinstvenu IP adresu i vezu sa Internetom,
- mogućnost povezivanja, prikupljanja i obrade podataka sa različitih hardverskih i senzorskih uređaja,
- analizu i agregaciju prikupljenih informacija,
- primjenu XML i JSON formata podataka,
- relativno malu potrošnju energije.

Izbor adekvatne platforme za implementaciju SW-a ne predstavlja trivijalan zadatak. Kako bi se utvrdile prednosti i mane postojećih platformi, potrebno je izvršiti detaljnu komparativnu analizu komercijalno dostupnih mikroprocesorskih platformi kao i aktuelnih bežičnih senzorskih čvorova koji će biti korišteni za formiranje SW elemenata. U radu [Maksimović14] izvršena je komparativna analiza pet DIY mikroprocesorskih platformi koje su danas dostupne na tržištu: Raspberry Pi, Arduino, BeagleBone Black, Phidgets i Udoo (Tabela 5.1.1.), dok su autori u [Vujović14c] dali komparativnu analizu performansi i ograničenja između popularnih bežičnih senzorskih čvorova (Tabela 5.1.2.).

Tabela 5.1.1. Komparativna analiza mikroprocesorskih platform

Naziv	Raspberry Pi (Model B)	Arduino	BeagleBone Black	Phidgets	Udoo (quad)
Procesor	ARM BCM2835	ATMEGA8, ATMEGA168, ATMEGA328, ATMEGA1280	AM335x 1GHz ARM® Cortex-A8	PhidgetSBC	Freescale i.MX6Quad, 4 x ARM® Cortex™-A9 core Atmel SAM3X8E ARM Cortex-M3 CPU
RAM	256-512 MB	16-32 KB	512 MB	64 MB	1 GB
Napajanje	5V USB	7-12V USB	5V	6-15V	6-15V
Operativni sistem	Raspbian, Ubuntu, Android, ArchLinux, FreeBSD, Fedora, RISC OS	/	Linux Angstrom	Linux	Ubuntu, Android, Linux, ArchLinux
Programiranje	C, C++, Java, Phyton	Arduino	Arduino	Visual Basic, VB.NET, C#, C/C++, Flash/Flex, Java, Labview, Matlab, ActionScript 3.0, Cocoa	Arduino, C, C++, Java
Analogni ulazi	0	6	6	8	14
Digitalni I/O pinovi	8-27	14	14	8I+8O	62+14
USB portovi	1-4	1	1	6	5
LAN (MBit)	10/100	–	10/100	–	10/100 /1000
WiFi Modul	–	–	–	–	+
Veličina (mm)	85.6×53.98×17 - 85×56×17	75×53×15	86.3×53.3	81.3×53,3	110×85
Težina (g)	42-45	~30	39.68	60	120-170
Cijena US$	25-35	30	45	50-200	99-135

Tabela 5.1.2. Komparativna analiza bežičnih senzorskih čvorova

Naziv	Procesor	RAM	Eksterna memorija	Operativni sistem	Veličina (mm)	Težina (g)	Cijena US$
Raspberry Pi (Model B)	ARM BCM2835	256-512 M	2-64 G	Raspbian	85.6×53.98 ×17	45	25-35
MicaZ	ATMEGA128	4 K	128 K	Tiny os, Mote runner	58×32×7	18	99
TelosB	TI MSP430	10 K	48 K	Tiny os, Sos, Mantisos	65×31×6	23	99

Iris	ATMEGA1281	8 K	128 K	Tiny os, Mote runner	58×32×7	18	115
Cricket	ATMEL128L	4 K	512 K	Tiny os	58×32×7	18	225
Lotus	ARM NXP LPC1758	64 K	512 K	Rtos, Tiny os	76×34×7	18	300

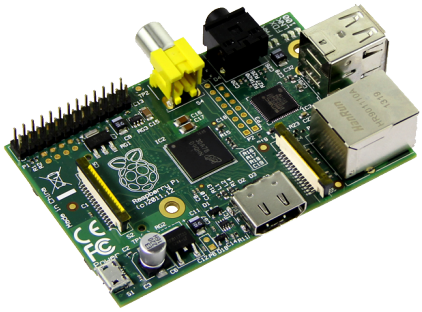
Na osnovu prikazanih analiza, moguće je zaključiti da Raspberry Pi (RPi), za razliku od drugih analiziranih platformi, ima optimalne hardverske i softverske karakteristike (cijena, težina, memorija, procesor, napajanje, proširenje, operativni sistem, programski jezici) koje su potrebne za izgradnju SW čvorova. RPi predstavlja hardverski moćnu platformu koja se po potrebi može koristiti i kao samostalni računar, pri čemu korisnici ne moraju imati napredna znanja iz hardvera, sistemskog softvera ili razvoja aplikacija. Jedan od glavnih razloga, osim ranije navedenih prednosti RPi-a, jeste i činjenica da je RPi najpogodniji za projekte koji zahtijevaju grafički interfejs ili Internet i kvalifikuju ga kao idealan izbor za implementaciju IoT.

Njegove prednosti predstavljaju: procesorska snaga, operativna memorija, mogućnost povezivanja, višenamjensko korištenje (USB), Linux OS, programiranje u objektno orijentisanim jezicima visokog nivoa, podrška kroz Internet zajednice, a osnovni nedostaci: potrošnja energije i težina.

U nastavku rada je prikazana implementacija SW čvora sa izabranom RPi platformom.

5.1.1. Raspberry Pi platforma

Raspberry Pi predstavlja jeftin, prilagodljiv i programabilan mali računar koji podržava veliki broj ulazno/izlaznih uređaja i mrežnih komunikacija što ga čini savršenom platformom za realizaciju IoT vizije [Dennis13] (Slika 5.1.1.1.).



Slika 5.1.1.1. Raspberry Pi Model B rev 2

RPi sadrži procesor, grafički čip, RAM memoriju i različite interfejse i konektore za eksterne uređaje u zavisnosti od verzije [Richardson13]. Neke od ovih komponenti predstavljaju osnovne, a neke opcione, međutim svaki RPi model ima ARM baziran procesor (BCM2835 i BCM2836) koji predstavlja jeftin SoC sa integrisanom memorijom, podrškom za grafiku i razne periferne uređaje. Na tržištu danas postoji više modela RPi uređaja, a razlike između njih su predstavljene u tabeli (Tabela 5.1.1.1.)

Tabela 5.1.1.1. Komparativna analiza Raspberry Pi modela

	Raspberry Pi 2 Model B	**Raspberry Pi 1 Model B+**	**Raspberry Pi 1 Model B**	**Raspberry Pi 1 Model A+**	**Raspberry Pi 1 Model A**
Procesor Chip	Broadcom BCM2836 ARMv7 Quad Core Processor multimedia applications processor	Broadcom BCM2835 ARMv6 SoC full HD multimedia applications processor	Broadcom BCM2835 ARMv6 SoC full HD multimedia applications processor	Broadcom BCM2835 ARMv6 SoC full HD multimedia applications processor	Broadcom BCM2835 ARMv6 SoC full HD multimedia applications processor
GPU	Videocore IV Quad Core @ 900MHz	Videocore IV Single Core @ 700MHz	Videocore IV Single Core @ 700MHz	Videocore IV Single Core @ 700MHz	Videocore IV Single Core @ 700MHz
RAM	1 GB SDRAM @ 450 MHz	512 MB SDRAM @ 400 MHz	512 MB SDRAM @ 400 MHz	256 MB SDRAM @ 400 MHz	256 MB SDRAM @ 400 MHz
Storage	MicroSD	MicroSD	SDCard	MicroSD	SDCard
USB 2.0	4× USB Ports	4× USB Ports	2× USB Ports	1× USB Ports	1× USB Ports
Power Draw/ voltage	800mA up to 1.8A @ 5V	600mA up to 1.8A @ 5V	700mA up to 1.2A @ 5V	200mA up to 1.8A @ 5V	300mA up to 1.2A @ 5V
GPIO	40	40	26	40	26
Ethernet port	Da	Da	Da	Ne	Ne

Napajanje RPi jedinica se može izvršiti sa različitim izvorima (pod uslovom da mogu obezbijediti napon od 5V i struju od 200mA-800mA), pa se tako najčešće koriste [Gay14a]:

- namjenski mrežni adapter ili adapter sa USB portovima,
- računarski USB port ili USB hub sa samostalnim izvorom napajanja,
- rezervna eksterna baterija za mobilne uređaje,
- solarni punjač za mobilne uređaje,
- punjive baterije (AA baterije i regulator napona) [Monk14].

Da bi se smanjila potrošnja energije, RPi ima četiri režima rada [Horan13]:

- *"run" režim* – centralna procesorska jedinica – *Central Processing Unit* (CPU) i sve funkcionalnosti ARM jezgra su na raspolaganju i uključene,
- *"standby" režim* – dijelovi CPU-a koji procesiraju instrukcije nisu aktivni. U ovom režimu, poznatom kao "očekivanje prekida" – *Wait for Interupt* (WFI), jezgro se može brzo aktivirati slanjem posebnog poziva "*interupt*" CPU-u, koji će zaustaviti bilo koji trenutni proces i izvršiti zahtjev koji je pozivajući proces tražio.
- *"shutdown" režim* – napajanje nije prisutno na RPi jedinici.
- *"dormant" režim* – jezgro je isključeno dok je memorija i dalje aktivna.

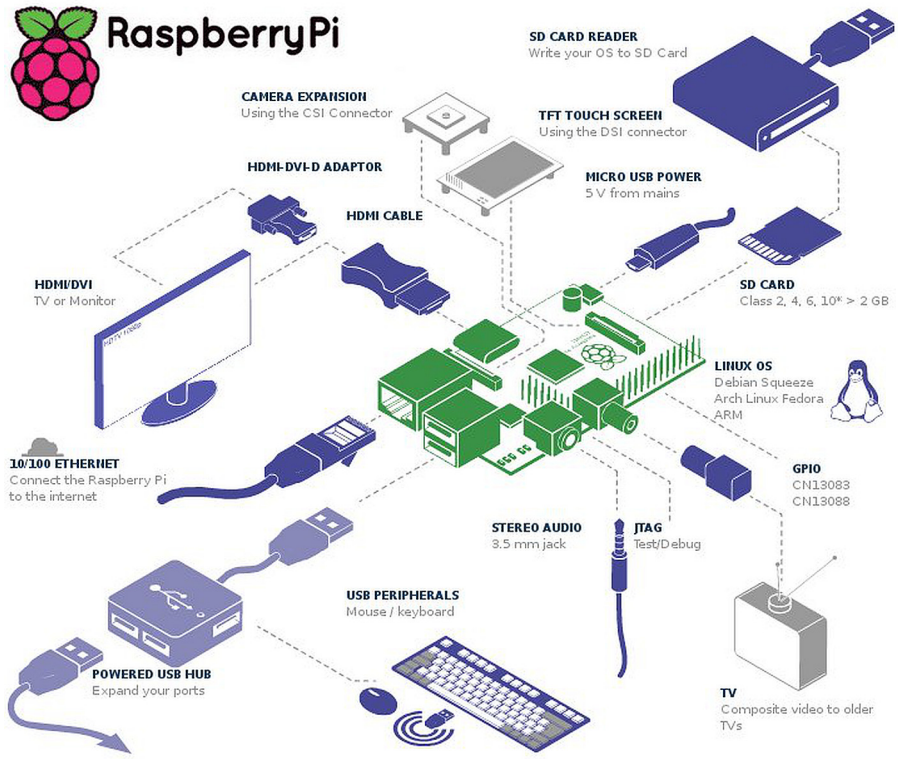
Raspberry Pi može da radi u dva načina (moda): kao standardni računar koji zahtijeva upotrebu tastature za unos komandi, korištenje ekrana (displeja) za prikaz i jedinice za napajanje ili kao Web server kada je potrebna samo jedinica za napajanje. Važno je napomenuti da RPi umjesto tvrdog diska koristi SD kartice koje su dostupne u veličinama od 2 do 64GB. SD kartice velikog kapaciteta su često skupe, pa minimalna potrebna veličina, u zavisnosti od distribucije operativnog sistema, iznosi 2GB i ako je moguće, zbog što kraćeg vremena upisa i čitanja, potrebno je da bude klase 10. Prostor za skladištenje podataka može biti značajno proširen korištenjem dodatnih eksternih tvrdih diskova spojenih preko USB portova. Ovi eksterni diskovi su poznati kao *USB Mass Storage* (UMS) uređaji i fizički mogu biti tvrdi diskovi, *Solid-State Drive* (SSD) diskovi ili prenosni *flash* diskovi [Upton12].

Konekcija RPi-ja se ostvaruje preko Ethernet/LAN kabla (modeli B i B+ imaju standardan RJ45 ethernet port, dok modeli A i A+ mogu biti povezan sa žičanom mrežom preko USB ethernet adaptera) koji može biti povezan sa ruterom, drugim računarom ili Internetom. WiFi konekcija se najčešće ostvaruje pomoću USB primopredajnika [Richardson13] i na taj način se RPi može koristiti za kreiranje *ad-hoc* mreža ili za povezivanje na 802.11 a/b/g/n bežične mreže [Upton12].

RPi koristi OS koji predstavlja distribuciju Linux-a pod nazivom Raspbian (optimizovana distribucija Debiana). Linux predstavlja odličan izbor za RPi platformu zbog svoje raširenosti, podrške, prilagodljivosti i činjenice da je besplatan i otvorenog kôda, što dodatno održava nisku cijenu platforme. Postoje i drugi OS koji se mogu izvršavati na RPi platformi a koji nisu bazirani na Linuxu [Richardson13].

Najveća prednost RPi platforme predstavlja njena fleksibilnost što pruža širok spektar primjene (od primjene kao računara opšte namjene,

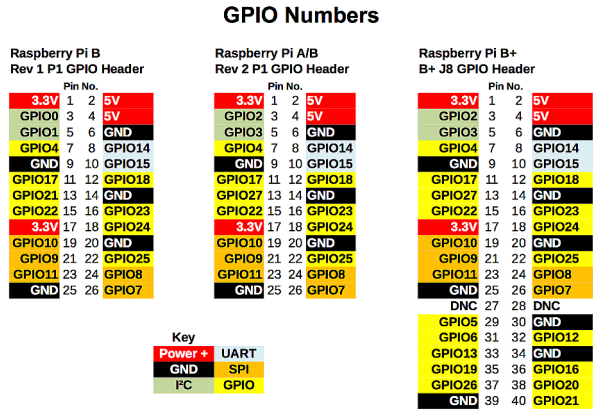
učenja programiranja, do integrisanja u elektronskim projektima) [Richardson13]. Na slici (Slika 5.1.1.2.) je prikazana primjena osnovnih komponenti RPi 1 u modelu B [Gay14a]:



Slika 5.1.1.2. Raspberry Pi 1 Model B osnovne komponente

- Dva USB 2.0 porta koja omogućavaju povezivanje perifernih uređaja i uređaja za skladištenje, dok jedan micro USB služi za napajanje uređaja.
- 3,5 mm analogni audio priključak koji omogućava povezivanje slušalica i zvučnika na RPi, što je posebno korisno za multimedijalne projekte.
- Kompozitni RCA priključak koji omogućava korištenje TV-a kao monitora.
- *High Definition Multi-Media Interface* (HDMI) – omogućava povezivanje RPi-ja sa HD televizorima i monitorima. Također se koristi za *streaming* multimedijalnog sadržaja sa Weba na TV.

- Podrška za *Display Serial Interface* (DSI) – pruža podršku za proširenje sa ekranom (displejom).
- Podrška za *Camera Serial Interface* (CSI) – pruža podršku za proširenje sa kamerom.
- Integrisani *General Purpose Input and Output* (GPIO) – predstavljaju glavni način za povezivanje RPi-ja sa drugim elektronskim komponentama. RPi 1 Model A, B rev 1 i B rev 2 se sastoje od 26 konektora raspoređenih u dva reda sa po 13 pinova, ali između njihovog rasporeda GPIO pinova postoji specifična razlika. RPi 1 Model A+, B+ i RPi 2 Model B imaju čak 40 GPIO konektora raspoređenih u dva reda sa po 20 pinova (Slika 5.1.1.3.).



Slika 5.1.1.3. Raspored GPIO konektora Raspberry Pi modela A, B, B+

GPIO imaju mogućnost prihvatanja ulazno/izlaznih komandi koje mogu biti upravljane programskim putem, pa se najčešće koriste za kontrolisanje hardvera (LED dioda, motora, releja) ili za očitavanje vrijednosti sa hardverskih komponenti (prekidači, dugmad, vrijednosti temperature, detekcija pokreta, itd.) [Schmidt13]. Neki od GPIO konektora se mogu koristiti kao digitalni ulazi/izlazi i kao interfejsi za ugrađene protokole, od kojih se dva posebno ističu zbog njihove široke upotrebe:

- I^2C interfejs male brzine – *Inter-Integrated Circuit* (I^2C) omogućava dobru podršku za komunikaciju sa različitim perifernim jedinicama male brzine u sistemima gdje se potreba za njihovom upotrebom javlja povremeno. Za komunikaciju su potrebne dvije linije: *Serial Data* (SDA), kojom se prenose podaci i *Serial Clock* (SCL) kojom se prenosi takt.

    - SPI - *Serial Peripheral Interface Bus* (SPI) je sinhrona *full-duplex* serijska veza.
- Proširenи GPIO konektori – pored standardnih GPIO konektora RPi 1 Model B Rev 2 ima prošireni skup konektora od kojih je važno spomenuti *P5 konektor* koji se sastoji od 8 pinova (3,3V, 5V, dva uzemljena konektora i četiri GPIO konektora koji mogu omogućiti sekundarni I^2C protokol) (Slika 5.1.1.3.) i *P6 konektor* sa 2 pina koji spajanjem omogućavaju resetovanje BCM2835 čipa.

Primjena RPi platforme se može pronaći u mnogim projektima, pa tako autori radova [Monk14, Gay14, Horan13] identifikuju širok spektar primjena RPi-ja za očitanje vrijednosti sa senzora, njihovo skladištenje i preduzimanje određenih akcija kao i kontrolu displeja, motora i drugih hardverskih komponenti, dok je u radu [Dennis13] predstavljena automatizacija kuće oslanjajući se na RPi i Arduino platformu. Autor rada [Bell13] ilustruje primjenu RPi i Arduino platformi za kreiranje SM, pri čemu naglašava da SM predstavljaju samo jedan primjer kako se mogu koristiti jeftine DIY mikroprocesorske platforme. Na primjeru XBee-ZigBee mreže, autor je pokazao kako je moguće izgraditi vlastitu senzorsku mrežu sa relativno malim programerskim znanjem. U radovima [Dennis13, Monk14, Bell13, Warner13, Goodwin13] se, u cilju povećanja opsega i kontrole hardverskih komponenti, predlaže kombinacija RPi sa Arduino platformom, te se u tu svrhu koriste različiti tehnološki pristupi i programski jezici. Autori u [Monk14, Warner13] za implementaciju softverskih rješenja koriste programski jezik Phyton, dok je mnoštvo tehnologija, uključujući X10, C-Bus, ZWave, ZigBee, i Hue kao i korištenje Perl, PHP, C ++ i Bash predstavljeno u radu [Goodwin13]. Pored informativnih aspekata projekata, autori navedenih radova također daju i mnogobrojne ideje za nove projekte.

5.1.2. Kreiranje Senzor Web čvora pomoću Raspberry Pi platforme

Za kreiranje SW čvora koristeći RPi, neophodno je ispuniti postavljene zahtjeve koje nameće SW arhitektura, pa su tako prepoznata tri osnovna gradivna elementa:

- povezivanje RPi platforme sa senzorima koji prikupljaju informacije iz okruženja,
- kreiranje drajvera koji će olakšati pristup i rad sa senzorima, i

- konverzija RPi-ja u RESTful servis, koji će biti dostupan preko Interneta i omogućiti pristup podacima sa senzora,

koja su u nastavku rada detaljno prikazana.

5.1.2.1. Raspberry Pi kao senzorska jedinica

Kao što je ranije pomenuto, zbog posjedovanja GPIO ulaza i izlaza, RPi platforma se može direktno koristiti u projektima iz elektronike. RPi GPIO pored digitalnih ulazno/izlaznih funkcija podržavaju i ugrađene I^2C te SPI protokole koji su implementirani u većini modernih elektronskih uređaja, logičkih kola i mikrokontrolera.

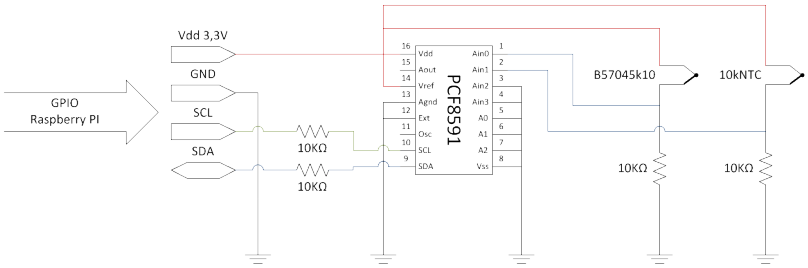
Postoji više različitih načina za očitavanje vrijednosti sa senzorskih jedinica, koje u zavisnosti od implementacije mogu biti analogne ili digitalne. Tako RPi u slučaju digitalnih senzorskih jedinica, može direktno pristupiti senzorima preko ugrađenih protokola i pročitati željene parametre, dok je u slučaju analognih senzora, potrebno izvršiti njihovu konverziju u digitalne vrijednosti. Najveća mana RPi platforme jeste u tome što nema integrisani AD/DA konvertor, pa je za rad sa analognim veličinama potrebno koristiti eksterna logička kola koja bi obezbijedila prevođenje analognih vrijednosti u digitalne. U ovu svrhu se mogu koristiti 8-bitni AD konvertori koji sa RPi platformom komuniciraju preko I^2C protokola (PCF5891) ili SPI protokola (MCP3004/3008). Bitno je napomenuti da ako je potrebna veća preciznost sistema, AD konvertori mogu biti zamijenjeni sa bilo kojim modelom veće preciznosti, odnosno 10-bitnim, 16-bitnim, itd.

U svrhu podrške eksperimentalnoj verifikaciji kreirana je prototipska senzorska jedinica sačinjena od dva analogna senzora temperature (termistor modeli: B57045k10 [ntc06] i 10kNTC [10kntc] koji su izabrani na osnovu cijene, dostupnosti i preciznosti. Karakteristike datih senzora su prikazane u tabeli (Tabela 5.1.2.1.1.)) i I^2C-bus integrisanim kolom 8-bitnog AD/DA konvertora (model: PCF8591 [PCF8591]). Šematski prikaz prototipa prikazan je na slici (Slika 5.1.2.1.1.)

Tabela 5.1.2.1.1. Karakteristike analognih senzora temperature

	B57045k10	**10kNTC**
$\mathbf{T_0}$	25 °C	25 °C
$\mathbf{R_{T0}}$	10 000 Ω	10 000 Ω
$\mathbf{B_{(25/100,25/50)}}$	4300 K	3950 K
$\mathbf{\Delta R_R/R_R}$	± 10%	± 1%

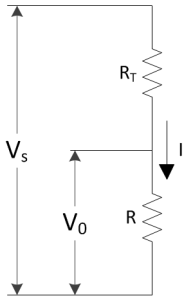
Opseg	-55 – 125 °C	-20 – 105 °C
Dodatne karakteristike		vodootporan



Slika 5.1.2.1.1. Šematski prikaz prototipa senzorske jedinice

Vrijednosti T_0 i R_{T0} predstavljaju referentnu temperaturu i otpornost, koje su potrebne za izračunavanje B parametra NTC termistora. Za oba senzora referentna temperatura je 25 °C, a otpornost 10 000 Ω, a parametri T_0, R_{T0} i B su najčešće dati u pratećim dokumentima navedenih senzora. $\Delta R_R/R_R$ predstavlja standardnu devijaciju.

Kako termistori rade na principu omjera temperature i otpornosti, a na ulazu u termistor poznata je samo vrijednost napajanja, neophodno je dovesti ove dvije vrijednosti u korelaciju. Iz poznate vrijednosti napajanja potrebno je izračunati vrijednost otpornosti termistora. U radu [Recktenwald13] je prikazano da električna otpornost termistora može dobiti korištenjem naponskog djelitelja sastavljenog od dva otpornika, od kojih je jedan sa promjenljivom i nepoznatom otpornošću, a drugi, sa fiksnom i poznatom otpornošću. U predloženom prototipu fiksna otpornost je 10 000 Ω a izvor napajanja senzora i AD konvertora je u opsegu 3,27V-3,3V (Slika 5.1.2.1.2.):



Slika 5.1.2.1.2. Naponski djelitelj

- V_s – napon napajanja (3,27-3,3 V)
- V_0 – mjereni napon
- R_T – otpornost termistora
- R – poznata otpornost (10 KΩ)
- I – struja

Strujno kolo sa slike (Slika 5.1.2.1.2.) opisano je sa jednačinama:

$$V_s = I\left(R + R_T\right) \tag{1}$$

$$V_0 = IR \tag{2}$$

iz kojih se dobija izraz za otpornost termistora:

$$R_T = \left(\frac{V_s}{V_0} - 1\right)R \tag{3}$$

Temperatura NTC termistora se izračunava pomoću *Steinhart-Hart* jednačine [Steinhart68] sa B parametrom:

$$\frac{1}{T} = a + b\ln\left(R\right) + c\ln^3\left(R\right) \tag{4}$$

Za

$$a = \left(\frac{1}{T_0}\right) - \frac{1}{B}\ln\left(R_0\right), \quad b = \frac{1}{B}, \quad c = 0 \tag{5}$$

Steinhart–Hart jednačina glasi:

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B}\ln\left(\frac{R}{R_0}\right) \tag{6}$$

Za poznate vrijednosti parametara T_0, R_{T0}, B i trenutnu otpornost termistora, može biti izračunata temperatura T koja je izražena u Kelvinima.

5.1.2.2. Razvoj drajvera za rad sa senzorima

Da bi Raspberry Pi mogao čitati podatke sa senzora, odnosno AD/DA konvertora, neophodno je razviti softver, koji može pristupiti logičkom kolu. Nakon čitanja podataka, u zavisnosti od tipa senzora, vrijednosti je neophodno konvertovati u odgovarajuće digitalne podatke (u slučaju termistora to je temperatura). Drajver za rad sa senzorima omogućava jednostavniji pristup informacijama sa senzora, pri čemu drajver preuzima odgovornost za komunikaciju, upis i čitanje podataka sa hardverskog uređaja.

Za razvoj drajvera, neophodnih za sprovođenje eksperimentalne verifikacije, korištena je *Pi4J* [Pi4J] biblioteka koja enkapsulira *Wiring Pi* biblioteku i predstavlja most između osnovnih GPIO i Java biblioteka čime se pruža potpuni pristup RPi-ju. *Pi4J* je također zavisna od Oracle JDK-a koji je potrebno dodatno instalirati na RPi operativni sistem, što je u prikazanom slučaju Raspbian (Linux - Debian "Wheezy").

Glavni problemi koji se javiljaju prilikom kreiranja drajvera su:

- Kako pristupiti sabirnici/magistrali i odabrati uređaj (svaki uređaj na sabirnici/magistrali ima jedinstvenu adresu)?
- Kako pristupiti odabranom uređaju i upisati/čitati informacije sa njega?

Za rešavanje prvog problema neophodno je poznavati adresu uređaja i broj sabirnice (Raspberry Pi 1 Model B rev 2 ima dvije I^2C sabirnice). Kako bi se dobila željena informacija, u Linux konzolu, je potrebno unijeti komandu:

```
sudo i2cdetect –y 1
```

Mogući rezultat izvršenja komande je prikazan na slici (Slika 5.1.2.2.1.).

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

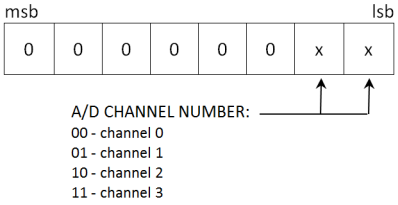
Slika 5.1.2.2.1. Prikaz izvršenja komende *i2cdetect*

Prikazani broj predstavlja adresu uređaja povezanog na I^2C sabirnicu (u ovom slučaju, adresa uređaja je 48 na sabirnici 1).

Za rješavanje drugog problema potrebno je poznavati arhitekturu hardverskog uređaja, što u konkretnom slučaju predstavlja I^2C 8-bitni AD/DA konvertor (model: PCF8591). PCF8591 koristi kontrolni registar, koji je neophodan za upravljanje funkcijama uređaja, a njegova struktura je prikazana na slici (Slika 5.1.2.2.2.).

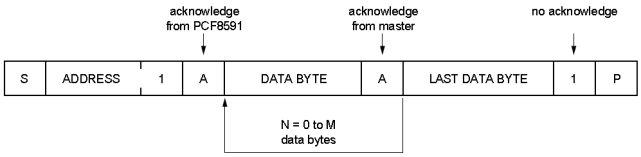
U konkretnom slučaju, dva LSB bita u kontrolnom registru predstavljaju broj AD kanala i omogućavaju selekciju jednog od četiri ulaza PCF8591 AD/DA konvertora. Kako je prikazano na slici (Slika 5.1.2.1.1.), termistor B57045k10 je povezan sa Ain0 ulazom AD konvertora dok je Ain1 ulaz povezan sa 10kNTC termistorom. Odabir

željenog ulaza vrši se unosom vrijednosti u posljednja dva bita kontrolnog registra, a kako su svi preostali bitovi 0, selekcija željenog ulaza može biti izvršena i upisivanjem bajtova 0x00, 0x01, 0x02, 0x03 u kontrolni registar.



Slika 5.1.2.2.2. Kontrolni registar PCF8591 AD/DA konvertora

Nakon što je vrijednost ulaznog napona očitana sa jednog ulaza AD konvertora, ona se pretvara u 8-bitni binarni kôd. Rezultat konverzije se nalazi u registru podataka AD konvertora i čeka prenos na procesorsku jedinicu. Prvi bajt prebačen u ciklusu čekanja sadrži rezultate prethodnog mjerenja, što znači da u prvom čitanju nema podataka, ali se šalje kontrolni bajt - za PCF8591 to je heksadecimalna vrijednost 80. Brzina čitanja sa PCF8591 je ograničena brzinom I^2C sabirnice, a na slici (Slika 5.1.2.2.3.) je prikazan protokol za mod čitanja AD konverzije.



Slika 5.1.2.2.2. Protokol za čitanje AD konverzije [PCF8591]

Koristeći Java objektno orijentisani programski jezik, kreirano je pet klasa pri čemu će zbog veličine kôda biti prikazani samo pojedini dijelovi.

`I2CDeviceClass` – klasa odgovorna za povezivanje i komunikaciju sa uređajima na I^2C sabirnici. Koristeći klase i metode iz *Pi4J* biblioteke, kreirana je instanca `I2CBus` klase koja se koristi za pristup I^2C sabirnici. `I2CDevice` objekat, koji je odgovoran za rad sa uređajem (pisanje i čitanje iz uređaja) je kreiran prosljeđivanjem adrese uređaja na sabirnici.

```
I2CDevice b57045k10 = I2CFactory.getInstance(1).getDevice
                      (Integer.decode("0x00").intValue());
```

`ADConversion` – posjeduje metodu za izračunavanje otpora termistora na osnovu izlaznog napon AD konvertora i jednačine (3). Potrebno je znati referentne vrijednosti napona i rezoluciju AD konvertora (8-bita = 256 stanja).

```
public double resistance(int adVal, int resistance)
 {
      vRef = 3.269;
      resolution = 255;
      refVoltage = vRef/resolution;

      return ((vRef/(refVoltage*adVal))-1)*resistance;
 }
```

`Thermistor` – posjeduje metodu za izračunavanje temperature u °C na osnovu jednačine (6). Za proračun je neophodno znati parametre termistora (Tabela 5.1.2.1.1.).

```
public double temperatureCelsius(double resistence)
{
      double t0 = 298.15;
      double r0 = 10000;
      double b = 4300;

      double resVal = resistence/r0;
      double tmpVal = (1/t0)+((1/b)*Math.log(resVal));

      return (1/tmpVal) - 273.15;
}
```

`B57045K10` i `NTC10K` (zbog analogije je prikazan samo `B57045K10` objekat) – klasa je odgovorna za kreiranje svih prethodno pomenutih objekata: kreiranje konekcije (`I2CDeviceClass`) i pomoćnih klasa (`ADConversion`, `Thermistor`). Klasa takođe obezbjeđuje metode za čitanje vrijednosti temperature sa senzora.

Prvi korak predstavlja odabir kanala na AD konvertoru podešavanjem bitova u kontrolnom registru:

```
b57045k10.write(Integer.decode("0x00").byteValue());
```

Naredni korak je čitanje vrijednosi iz registra podataka:

```
b57045k10.read(buffer, 0, 2);
...
byte[] buffer = new byte[2];
int buf1 = buffer[1] & 0xFF;
```

i konverzija vrijednosti:

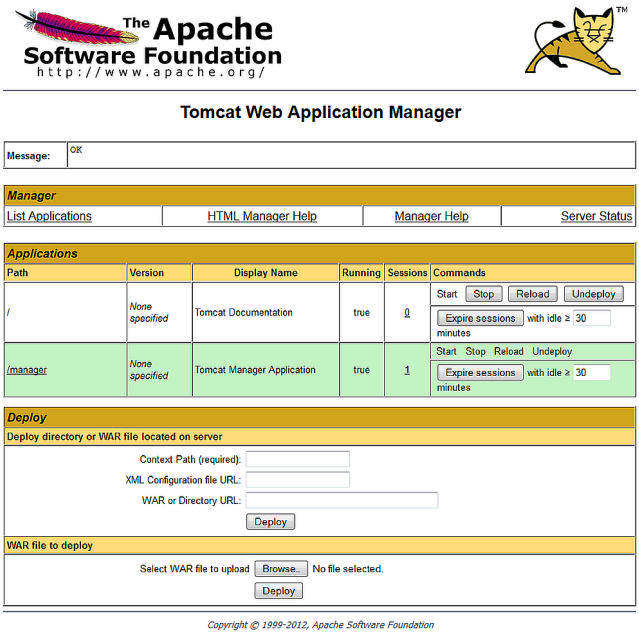
```
Thermistor B57045K10 = null;
ADConversion adconversion = null;
...
DecimalFormat format = new DecimalFormat("###.##");
...
String data  = format.format(B57045K10.temperatureCelsius
                            (adconversion.resistance(buf1, 10000)));
```

Pristup konvertovanim vrijednostima je omogućen pozivom metode `getTemperature()` iz klase `B57045K10`.

5.1.2.3. Raspberry Pi kao RESTful servis

U cilju ispunjenja osnovnih zahtjeva postavljenih za izradu prototipa SW čvora, potrebno je omogućiti distribuciju očitanih podataka putem Interneta. Na procesorskoj jedinici je implementiran RESTful servis baziran na JAVA programskom jeziku, koji predstavlja medijator između drajvera senzora i krajnjeg korisnika. U cilju pokretanje i izvršavanje RESTful servisa na procesorskoj jedinici je instaliran *Apache Tomcat* server [Tomcat] koji je izabran zbog jednostavne konfiguracije i optimalnog korištenja hardvera. Jednostavnost koju pruža menadžment konzola *Apache Tomcat* servera, odgovara velikom broju korisnika jer ne zahtjeva postojanje dodatnih administrativnih alata i znanja potrebnih za postavljanje i pokretanje Web aplikacija. Za implementaciju RESTful servisa korišten je Jersey Framework 2.22 [Jersey].

Nakon kreiranja RESTful Web servis, postavljanje na *Apache Tomcat* server se vrši putem Web interfejsa prikazanog na slici (Slika 5.1.2.3.1.). Web aplikacije, nakon kreiranja, su spremljene u *Web application ARchive* (WAR) datoteku, koji se nakon postavljanja na server, izvršava.

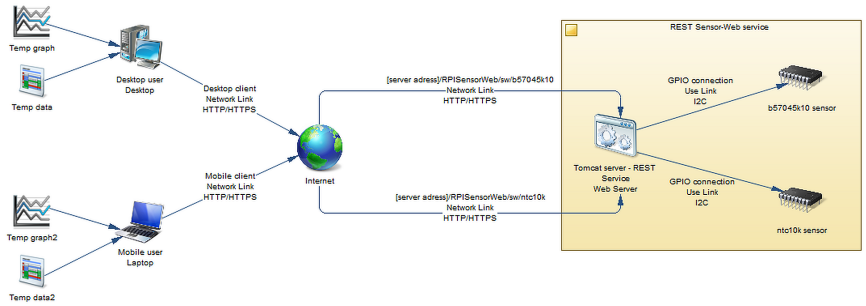


Slika 5.1.2.3.1. Administratorska konzola *Apache Tomcat* servera

Za komunikaciju između servisa i klijenta se, zbog svoje jednostavnosti, brzine procesiranja i smanjenja protoka podataka, koristi

JSON. Sama komunikacija je realizovana pomoću HTTP/HTTPS protokola.

Dijagram infrastrukture predloženog prototipa SW čvora prikazan je na slici (Slika 5.1.2.3.2.). Čine ga četiri osnovne cjeline: klijenti (mobilni ili PC korisnici), komunikacioni kanal (Internet ili Intranet), servisi (RPi kao SW element) i senzorske jedinice.



Slika 5.1.2.3.2. Infrastrukturni dijagram prototipa SW mreže

Klijenti mogu biti bilo koje SS ili aplikacije koje podržavaju izvršavanje poziva URI adrese u svrhu pristupanja SW čvoru. Većina današnjih sistema uglavnom posjeduje ovu mogućnost, ali u predloženom slučaju je korišten klijent implementiran u Java programskom jeziku. Komunikacioni kanal korišten za pristup SW čvoru predstavlja sam Internet, odnosno TCP/IP protokol, dok SW čvor uključuje sve preostale komponente: komunikacionu jedinicu koja posjeduje IP adresu, centralnu procesorsku jedinicu koja je u stanju da prikupi i obradi podatke, senzore i AD konvertor.

Za čitanje vrijednosti temperature sa senzora korištenih u kreiranom prototip modelu, obezbjeđene su po dvije URI adrese

```
GET: [server_address]/RPISensorWeb/sw/b57045k10
GET: [server_address]/RPISensorWeb/sw/b57045k10/data
```

i

```
GET: [server_address]/RPISensorWeb/sw/ntc10k
GET: [server_address]/RPISensorWeb/sw/ntc10k/data
```

Iako je za razmjenu informacija korišten JSON format, razlog za postojanje dvije adrese je smanjenje količine prenesenih podataka sa ciljem smanjenja potrošnje energije.

Pozivom prve adrese, generiše se JSON koji sadrži jedino informacije o mjerenoj veličini:

```
{"temp":"23.55"}
```

Pozivom druge adrese, generiše se JSON sa informacijom o tipu i modelu senzora, zajedno sa mjerenom vrijednošću:

```
{"type":"thermistor","model":"b57045k10","temp":"23.87"}
```

Objedinjene informacije o senzorima su dostupne na adresama:

```
GET: [server_address]/RPISensorWeb/sw/info
GET: [server_address]/RPISensorWeb/sw/info/data
GET: [server_address]/RPISensorWeb/sw/info/number
```

Prva adresa generiše informacije o broju senzora, podacima svih senzora i mjerenim vrijednostima,

```
{"sensor_number":2,"data":
[{"type":"thermistor","model":"b57045k10","temp":"23.23"},
{"type":"thermistor","model":"ntc10k","temp":"23.07"}]}
```

dok druga adresa generiše samo informacije o senzorima i mjerenim vrijednostima,

```
{"data":[{"type":"thermistor","model":"b57045k10","temp":"22.58"},
        {"type":"thermistor","model":"ntc10k","temp":"22.37"}]},
```

a treća adresa obezbjeđuje podatke o broju senzora u senzorskom čvoru:

```
{"sensor_number":2}.
```

Implementacija `b57045k10` klase RESTful Servisa je predstavljena sa:

```
@Path("/b57045k10")
public class b57045k10
{
      @Context
      private HttpServletRequest request;
      @Context
      private ServletContext context;

      @GET
      @Produces(MediaType.APPLICATION_JSON + ";charset=utf-8")
      public String getSensorData()
      {
               B57045K10 b57045k10 = new B57045K10("thermistor",
                                                   "b57045k10", "1", "0x48");

               return b57045k10.getTemperature().toString();
      }

      @GET
      @Path("/data")
      @Produces(MediaType.APPLICATION_JSON + ";charset=utf-8")
      public String getAllSensorData()
      {
```

```
            B57045K10 b57045k10 = new B57045K10("thermistor",
                                        "b57045k10", "1", "0x48");

            return b57045k10.getAllData().toString();
        }
}
```

Prikupljanje informacija sa senzorskih uređaja vrši se pozivom kreiranih drajvera pojedinačnih senzorskih jedinica. Nakon obrade podataka, informacije se šalju korisniku u JSON formi koja se veoma jednostavna generiše i parsira.

5.1.3. GSM/GPRS proširenje za Raspberry Pi

Analizirane primjene SW čvorova, ukazuju da često postoji potreba za njihovim postavljanjem na nedostupnom ili teško dostupnim mjestima, gdje ne postoji mogućnost za postavljanjem infrastrukture koja bi omogućila kreiranje konekcije sa Internetom (žičana ili WiFi Internet konekcija). Relativno mala pokrivenost signalom na decentralizovanom području (standardnim komunikacionim tehnologijama), značajno komplikuje infrastrukturu koja je potrebna za kreiranje mrežne topologije, što znatno podiže ukupne troškove implementacije SW mreže. Komunikaciona rješenja data u poglavlju 2, uvode i primjenu celularnih mreža koje predstavljaju adekvatno rješenje za povezivanje SW čvorova i Interneta.

U slučaju implementacije SW čvorova koristeći RPi platformu i celularne mreže, u publikovanoj literaturi su dostupna dva moguća rješenja [Vujović15]:

- korištenjem 3G/4G USB modema koji se RPi platformi dodaje preko USB konektora (npr. Hauvei E173 HSDPA 7,2Mbps GSM 3G USB Modem) – omogućava povezivanje sa *High-Speed Downlink Packet Access* (HSDPA)/UMTS/WCDMA mobilnim mrežama, i
- GSM/GPRS proširenja RPi platforme koja omogućavaju konekciju sa GPRS/GSM/3G/4G uslugama (npr. GPRS Sim900 Shield, 3G/GPRS Shield) – omogućava povezivanje sa HSPA i WCDMA mobilnim mrežama velikih brzina.

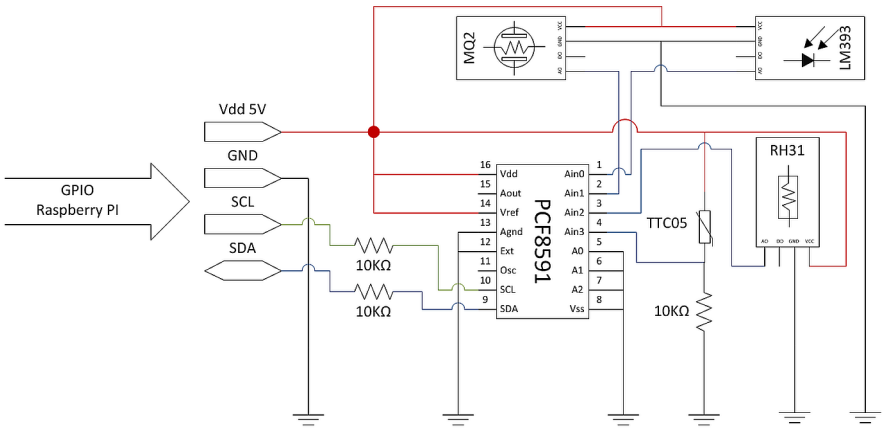
Koristeći predložena rješenja, moguće je izgraditi SW čvorove bazirane na GSM mobilnim mrežama koje se mogu kontrolisati putem Interneta na svakom mjestu gdje postoji pokrivenost signalom mobilne telefonije.

5.2. Eksperimentalna provjera okruženja za razvoj Senzor Web arhitekture

Sa stanovišta monitoringa, skup parametara koji definišu stanje životne sredine može značajno varirati u zavisnosti od njegove namjene. U sprovedenim eksperimentima, za dva navedena slučaja, kao parametri su odabrani temperatura, vlažnost, kvalitet zraka i osvjetljenje. Na osnovu toga, SW čvorovi koji su korišteni za akviziciju podataka, opremljeni su sa maksimalno četiri senzorska elementa:

- *temperaturski senzor* (TTC05) – detekcija temperature i njene promjene su od vitalnog značaja za prevenciju nastanka požara, kao i sigurnost od toplotnih udara osoba koje borave u prostorijama,
- *senzor vlažnosti* (RH31) – detekcija relativne vlažnosti zraka osigurava analizu okruženja na povećanu vlažnost, tj. stvaranje stakleničkog efekta, ili na smanjenu vlažnost čemu može biti uzrok nastanak požara u prostoriji,
- *senzor kvaliteta zraka* (MQ2) – detektuje prisustvo materija kao što su benzen, alkohol, dim, CO i CO_2. Neke od ovih materija su visoko kancerogene i njihovo prisustvo nije poželjno u prostorijama u kojima boravi čovjek,
- *senzor osvjetljenosti* (LM393) – služi za detektovanje jačine svjetlosti, kao i postojanje otvorenog plamena.

Šematski prikaz eksperimentalnog SW čvora sa četiri senzora prikazan je na slici (Slika 5.2.1.)



Slika 5.2.1. Šematski prizak akvizicionog SW čvora

Sa stanovišta SM, topologija predstavlja ključni element projektovanja [Maksimović14a, Maksimović14b], dok topologija SW mreža ima dinamičku prirodu koja je posljedica osnovnih principa Interneta. Na osnovu toga, SW mreže omogućavaju slobodno mijenjanje topologije u skladu sa zahtjevima i potrebama korisnika.

U daljnjem tekstu, dati su detaljni opisi i rezultati sprovedenih eksperimenata uz primjenu metodologijom predviđenih koraka koji obuhvataju: opis okruženja, formulaciju zahtjeva, modelovanje, generisanje i integraciju u arhitekturu servisa, a primjer upotrebe eksperimentom dobijenih podataka prikazan je u paragrafu 5.3.

5.2.1. Monitoring parametara životne sredine u zatvorenom prostoru

Monitoring parametara životne sredine u zatvorenom prostoru predstavlja jedan od ključnih elemenata IS koji se bave zaštitom i sigurnošću osoba, njihovog stambenog ili radnog okruženja, a najčešće predstavlja sastavni dio *sistema za upravljanje zgradom – Building Management System* (BMS) koji obuhvata i druge kontrolne elemente (sistem za grijanje, klimatizaciju i ventilaciju, električni sistemi, sistem rasvjete, protivpožarni sistem, sigurnosni sistem), pri čemu su neki od njih povezani sa Internetom [Maksimović13].

5.2.1.1. Osnovni slučaj monitoringa parametara životne sredine u zatvorenom prostoru

Opis okruženja:

U svrhu analize kreiranog sistema u zatvorenom prostoru, izabran je dio arhitekture poslovnog objekta koji se sastoji od osam povezanih prostorija različitih po dimenzijama i namjeni (Slika 5.2.1.1.1.).

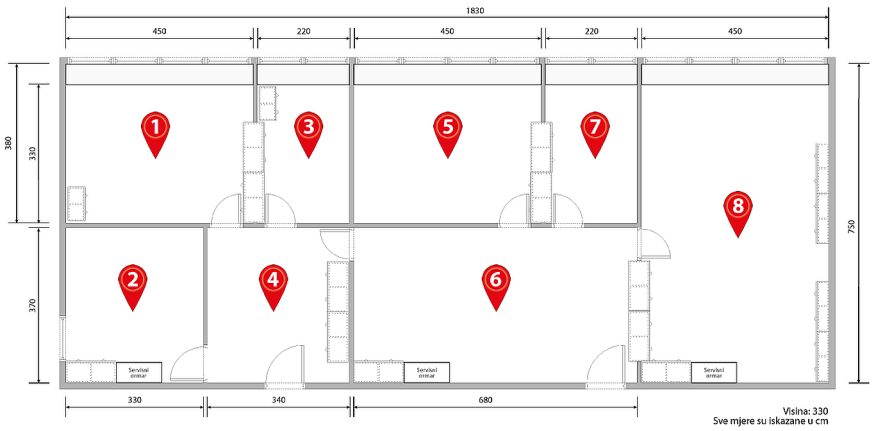
SW čvorovi, njihov ID i raspored u prostorijama su prikazani na slici (Slika 5.2.1.1.1.), dok su osnovni parametri svakog od čvorova dati u tabeli (Tabela 5.2.1.1.1.).

Pristup senzorskim podacima sa SW čvorova podržan je skupom RESTful servisa kojima se pristupa uz oslonac na sljedeće URI adrese:

```
GET: [server_address]/RPISensorWeb/sw/info
GET: [server_address]/RPISensorWeb/sw/info/data
GET: [server_address]/RPISensorWeb/sw/ttC05
GET: [server_address]/RPISensorWeb/sw/ttC05/data
```

```
GET: [server_address]/RPISensorWeb/sw/rh31
GET: [server_address]/RPISensorWeb/sw/rh31/data
GET: [server_address]/RPISensorWeb/sw/mq2
GET: [server_address]/RPISensorWeb/sw/mq2/data
GET: [server_address]/RPISensorWeb/sw/lsmlm393
GET: [server_address]/RPISensorWeb/sw/lsmlm393/data
```

a kao rezultat njihovog poziva klijentu se vraća JSON koji obezbjeđuje informacije sa senzora u skladu sa formatima prikazanim u poglavlju 5.1.2.3.



Slika 5.2.1.1.1. Tlocrt poslovnog objekta odabranog za mjerenje parametara životne sredine u zatvorenom prostoru

Tabela 5.2.1.1.1. Parametri SW čvorova

ID	IP adresa	MAC adresa	GPS koordinate	TTC05	RH31	MQ2	LM393
1	192.168.0.21	80:1F:02:AF:4A:56	-	+	+	+	+
2	192.168.0.22	80:1F:02:AF:4A:3F	-	+	+	+	+
3	192.168.0.23	80:1F:02:AF:4A:37	-	+	+	+	+
4	192.168.0.24	80:1F:02:D6:19:28	-	+	+	+	+
5	192.168.0.25	80:1F:02:AF:4A:7C	-	+	+	+	+
6	192.168.0.26	80:1F:02:AF:2D:1D	-	+	+	+	–
7	192.168.0.27	80:1F:02:AF:2D:17	-	+	+	+	–
8	192.168.0.28	80:1F:02:AF:4A:73	-	+	+	+	–

Formulacija zahtjeva:

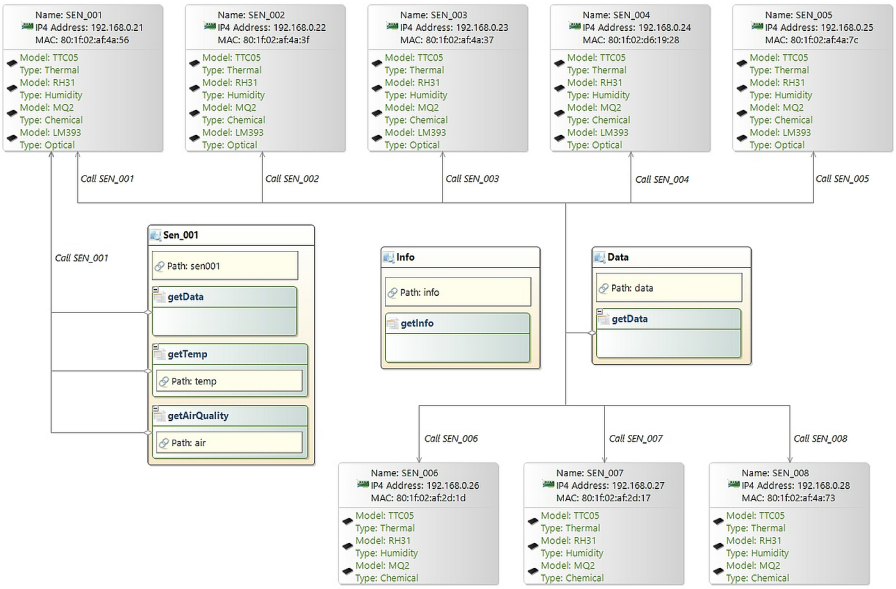
Za prikazani raspored i definisane parametre senzora (Slika 5.2.1.1.1.), neophodno je obezbijediti pristup podacima uz oslonac na servisni sloj za

pristup SW čvorovima, koji ujedno obezbjeđuje i enkapsulaciju SW čvorova. Na osnovu toga klijent nema svijest o postojanju pojedinih SW čvorova, već samo o postojanju interfejsa servisnog sloja i pojedinačnih servisa. U sklopu osnovnog eksperimenta formulisani su servisi koji trebaju da obezbijede informacije o:

- infrastrukturi senzorskih čvorova (broj SW čvorova koji se nalaze na servisu, broj senzora na svakom od SW čvorova kao i njihov tip),
- kumulativni prikaz svih podataka (očitavanje podataka sa svih SW čvorova i njihovih senzora), i
- pojedinačan pristup SW čvorovima (jednom čvoru i njegovim pojedinačnim senzorskim elementima).

Kreiranje modela:

Uz oslonac na alat *SWServ,* kreiran je model *SWServIndoor* koji opisuje predloženu arhitekturu SW mreže u zatvorenom prostoru, odnosno SW čvorove, senzore koje posjeduju SW čvorovi, klase, metode i pozive između metoda i SW čvorova (Slika 5.2.1.1.2.).



Slika 5.2.1.1.2. SWServIndoor model

Na osnovu opisa okruženja i formulisanih zahtjeva, u modelu je kreirano pet *Sensor Node* objekata (ID 1 − 5) sa po četiri senzorska elementa i tri *Sensor Node* objekta (ID 6 − 8) sa po tri senzorska elementa koji odgovaraju formulisanim SW čvorovima i tri *Service Class* klase:

- *Info* (*getInfo*) – klasa koja prikuplja informacije o SW čvorovima i servisu,
- *Data* (*getData*) – klasa koja obezbjeđuje akumulacioni prikaz informacija sa SW čvorova, i
- *SEN_001* (*getData*, *getTemp*, *getAirQuality*) – klasa koja obezbjeđuje podatke sa jednog senzorskog čvora (u kreiranom primjeru čvor *SEN_001*), odnosno podatke sa pojedinačnih senzora.

Na osnovu principa referenciranja u kreiranom modelu metoda *getData* iz klase *Data* ima svijest o svim *Sensor Node* objektima, dok metode *getData*, *getTemp* i *getAirQuality* iz klase *SEN_001* imaju svijest samo o jednom *Sensor Node* objektu.

Generisanje kôda:

Primjenom *SWServ* alata, na osnovu kreiranog *SWServIndoor* modela, generisan je skup datoteka prikazan u tabeli (Tabela 5.2.1.1.2.).

Tabela 5.2.1.1.2. Pregled generisanih datoteka za SWServIndoor model

SWServDoc	
Paket	**Generisane datoteke**
WebContent	index.html
WebContent	style.css
WebContent	Sen_001.html, Sen_002.html, Sen_003.html, Sen_004.html, Sen_005.html, Sen_006.html, Sen_007.html, Sen_008.html
WebContent/WEB-INF	web.xml
SWServGen	
Paket	**Generisane datoteke**
src/helper	ErrorCode.java
src/impl	Sen_001_Impl.java, Sen_002_Impl.java, Sen_003_Impl.java, Sen_004_Impl.java, Sen_005_Impl.java, Sen_006_Impl.java, Sen_007_Impl.java, Sen_008_Impl.java
src/service	Data.java, Info.java, Sen_001.java

Integracija u arhitekturu servisa:

Inkorporiranjem generisanih datoteka u dinamički JAVA projekat i implementacijom specifičnih detalja (dijelovi kôda koji obavljaju pozive i obradu informacija sa SW čvorova), kreira se servisni sloj za pristup SW čvorovima. Postavljanjem i pokretanjem kreiranog servisa na Tomcat serveru, vrši se integracija servisa u servisnu arhitekturu. Pristup servisnom sloju omogućen je uz oslonac na sljedeći skup URI adresa:

```
GET: [server_address]:8080/SWServIndoor/swi/info
GET: [server_address]:8080/SWServIndoor/swi/data
GET: [server_address]:8080/SWServIndoor/swi/sen001
```

```
GET: [server_address]:8080/SWServIndoor/swi/sen001/temp
GET: [server_address]:8080/SWServIndoor/swi/sen001/air
```

Definisane adrese klijentima pružaju informacije u JSON formatu, respektivno:


```
{"creator":"Vladimir Vujovic","date":"05.11.2015","sesnor":8,
"type":"indoor"}

{"data":[
{"sen_001":[
   {"type":"thermistor","model":"ttC05","temp":"22.7141"},
   {"type":"humidity","model":"rh31","percent":"75.2941"},
   {"type":"gas","model":"mq2","percent":"10.2254"},
   {"type":"photodetector","model":"lsmlm393","percent":"67.8431"}]},
{"sen_002":[
   {"type":"thermistor","model":"ttC05","temp":"23.1223"},
   {"type":"humidity","model":"rh31","percent":"60.3758"},
   {"type":"gas","model":"mq2","percent":"17.2549"},
   {"type":"photodetector","model":"lsmlm393","percent":"64.3137"}]},
{"sen_003":[
   {"type":"thermistor","model":"ttC05","temp":"22.3456"},
   {"type":"humidity","model":"rh31","percent":"40.2457"},
   {"type":"gas","model":"mq2","percent":"30.3874"},
   {"type":"photodetector","model":"lsmlm393","percent":"10.2345"}]},
{"sen_004":[
   {"type":"thermistor","model":"ttC05","temp":"22.9124"},
   {"type":"humidity","model":"rh31","percent":"80.7990"},
   {"type":"gas","model":"mq2","percent":"50.2455"},
   {"type":"photodetector","model":"lsmlm393","percent":"30.4510"}]},
{"sen_005":[
   {"type":"thermistor","model":"ttC05","temp":"21.2012"},
   {"type":"humidity","model":"rh31","percent":"74.9020"},
   {"type":"gas","model":"mq2","percent":"18.2541"},
   {"type":"photodetector","model":"lsmlm393","percent":"25.4902"}]},
{"sen_006":[
   {"type":"thermistor","model":"ttC05","temp":"22.1528"},
   {"type":"humidity","model":"rh31","percent":"78.8214"},
   {"type":"gas","model":"mq2","percent":"32.2579"}]},
{"sen_007":[
   {"type":"thermistor","model":"ttC05","temp":"23.6247"},
   {"type":"humidity","model":"rh31","percent":"41.2366"},
   {"type":"gas","model":"mq2","percent":"80.2477"}]},
{"sen_008":[
   {"type":"thermistor","model":"ttC05","temp":"22.2458"},
   {"type":"humidity","model":"rh31","percent":"75.2941"},
   {"type":"gas","model":"mq2","percent":"20.4505"}]}
 ]}

{"sensor_number":4,"data":[
   {"type":"thermistor","model":"ttc05","temp":"22.7141"},
   {"type":"humidity","model":"rh31","percent":"74.9020"},
   {"type":"gas","model":"mq2","percent":"17.2549"},
   {"type":"photodetector","model":"lsmlm393","percent":"68.2353"}]}
```

```
{"temp":"22.3033"}

{"percent":"17.2549"}
```

5.2.1.2. Modifikovani slučaj monitoringa parametara životne sredine u zatvorenom prostoru

Opis okruženja:

U sklopu postavljenog eksperimenta, osnovni model je restrukturiran promjenom servisnog sloja, koji podrazumijeva postavljanje novog okruženja i prateće formulacije zahtjeva. Oslanjajući se na dio arhitekture poslovnog objekata i ranije definisane SW čvorove koji su korišteni u prethodnom primjeru, izabrana su četiri SW čvora za koje je potrebno kreirati servisni sloj za pristup podacima (Slika 5.2.1.2.1.).

Na slici (Slika 5.2.1.2.1.) prikazani su SW čvorovi, njihov ID i raspored u prostorijama, dok su osnovni parametri svakog od čvorova i URI adrese za pristup senzorskim podacima isti kao i za osnovni slučaj.

Formulacija zahtjeva:

U sklopu modifikovanog slučaja eksperimenta, za prikazani raspored i definisane parametre senzora (Slika 5.2.1.2.1.) formulisani su servisi koji trebaju da obezbijede informacije o:

Slika 5.2.1.2.1 Tlocrt poslovnog objekta odabranog za mjerenje parametara životne sredine u zatvorenom prostoru sa izabranim SW čvorovima

- infrastrukturi senzorskih čvorova (broj SW čvorova koji se nalaze na servisu, broj senzora na svakom od SW čvorova kao i njihov tip),

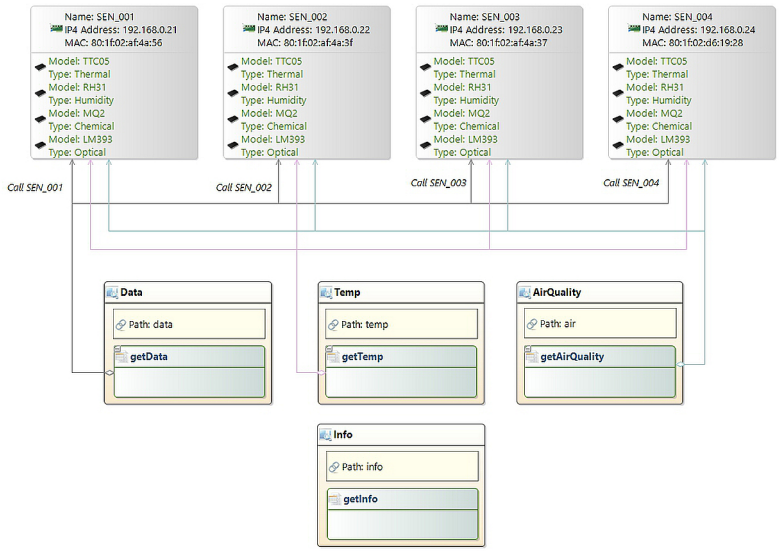
- kumulativni prikaz svih podataka (očitavanje podataka sa svih selektovanih SW čvorova i njihovih senzora),
- kumulativni prikaz informacija o temperaturi (očitavanje temperature sa svih selektovanih SW čvorova), i
- kumulativi prikaz informacija o kvalitetu zraka (očitavanje kvaliteta zraka sa svih selektovanih SW čvorova).

Kreiranje modela:

Ekvivalentno prethodnom slučaju, uz oslonac na alat *SWServ* kreiran je model *SWServIndoor Modified* koji opisuje predloženu arhitekturu SW mreže u zatvorenom prostoru za izabrani broj čvorova (Slika 5.2.1.2.2.).

Na osnovu opisa okruženja i formulisanih zahtjeva, u modelu su kreirana četiri *Sensor Node* objekta (ID 1 − 4) koji posjeduju po četiri senzorska elementa, kao i četiri *Service Class* klase:

- *Info* (*getInfo*) – klasa koja prikuplja informacije o SW čvorovima i servisu,
- *Data* (*getData*) – klasa koja obezbjeđuje akumulacioni prikaz informacija sa SW čvorova,
- *Temp* (*getTemp*) – klasa koja obezbjeđuje samo podatke o temperaturi sa svih SW čvorova, i
- *AirQuality* (*getAirQuality*) – klasa koja obezbjeđuje samo podatke o kvalitetu zraka sa svih SW čvorova.



Slika 5.2.1.2.2. SWServIndoor Modified model

Na osnovu principa referenciranja u kreiranom modelu metode *getData, getTemp* i *getAirQuality* iz klasa *Data, Temp* i *AirQuality* respektivno, imaju svijest o svim *Sensor Node* objektima.

Generisanje kôda:

Primjenom *SWServ* alata, na osnovu kreiranog *SWServIndoor Modified* modela, generisan je skup datoteka prikazan u tabeli (Tabela 5.2.1.2.1.).

Tabela 5.2.1.2.1. Pregled generisanih datoteka za SWServIndoor Modified model

SWServDoc	
Paket	**Generisane datoteke**
WebContent	index.html
WebContent	style.css
WebContent	Sen_001.html, Sen_002.html, Sen_003.html, Sen_004.html
WebContent/WEB-INF	web.xml
SWServGen	
Paket	**Generisane datoteke**
src/helper	ErrorCode.java
src/impl	Sen_001_Impl.java, Sen_002_Impl.java, Sen_003_Impl.java, Sen_004_Impl.java
src/service	AirQuality.java, Data.java, Info.java, Temp.java

Integracija u arhitekturu servisa:

Inkorporiranjem generisanih datoteka u dinamički JAVA projekat i implementacijom specifičnih detalja (dijelovi kôda koji obavljaju pozive i obradu informacija sa SW čvorova), kreira se servisni sloj za pristup SW čvorovima. Postavljanjem i pokretanjem kreiranog servisa na Tomcat serveru, vrši se integracija servisa u servisnu arhitekturu. Pristup servisnom sloju omogućen je uz oslonac na sljedeći skup URI adresa:

```
GET: [server_address]:8080/SWServIndoor/swim/info
GET: [server_address]:8080/SWServIndoor/swim/data
GET: [server_address]:8080/SWServIndoor/swim/temp
GET: [server_address]:8080/SWServIndoor/swim/air
```

Definisane adrese klijentima pružaju informacije u JSON formatu, respektivno:

```
{"creator":"Vladimir Vujovic","date":"05.11.2015","sesnor":4,
"type":"indoor"}

{"data":[
{"sen_001":[
   {"type":"thermistor","model":"ttC05","temp":"22.7141"},
   {"type":"humidity","model":"rh31","percent":"21.5876"},
   {"type":"gas","model":"mq2","percent":"16.8627"},
   {"type":"photodetector","model":"lsm1m393","percent":"24.4902"}]},
```

```
{"sen_002":[
   {"type":"thermistor","model":"ttC05","temp":"23.0045"},
   {"type":"humidity","model":"rh31","percent":"75.2941"},
   {"type":"gas","model":"mq2","percent":"30.2557"},
   {"type":"photodetector","model":"lsmlm393","percent":"35.6275"}]},
{"sen_003":[
   {"type":"thermistor","model":"ttC05","temp":"23.7357"},
   {"type":"humidity","model":"rh31","percent":"80.6554"},
   {"type":"gas","model":"mq2","percent":"45.4679"},
   {"type":"photodetector","model":"lsmlm393","percent":"66.6667"}]},
{"sen_004":[
   {"type":"thermistor","model":"ttC05","temp":"22.9821"},
   {"type":"humidity","model":"rh31","percent":"50.2443"},
   {"type":"gas","model":"mq2","percent":"19.3795"},
   {"type":"photodetector","model":"lsmlm393","percent":"6.4510"}]}
]}

{"data":[
   {"sen_001":{"temp":"22.7141"}},
   {"sen_002":{"temp":"23.0045"}},
   {"sen_003":{"temp":"23.7357"}},
   {"sen_004":{"temp":"22.9821"}}
 ]}

{"data":[
   {"sen_001":{"percent":"16.8627"}},
   {"sen_002":{"percent":"30.2557"}},
   {"sen_003":{"percent":"45.4679"}},
   {"sen_004":{"percent":"19.3795"}}
 ]}
```


5.2.2. Monitoring parametara životne sredine na ograničenom geografskom području

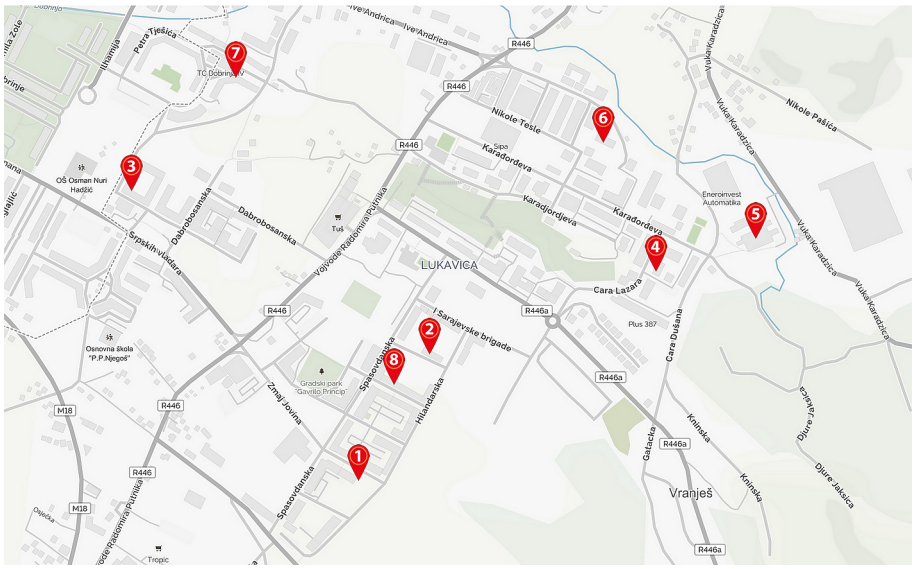
Monitoring parametara životne sredine na geografskom i ograničenom geografskom području predstavlja drugi ključni element IS koji se bave zaštitom osoba, životinja, njihovog životnog okruženja i staništa, prirode i prirodnih resursa. Ovakvi IS najčešće predstavljaju akvizicione sisteme koji sa decentralizovanih geografskih lokacija prikupljaju i obrađuju podatke iz okoline. Na ovaj način se obezbjeđuje preventivno djelovanje i korektivne akcije koje u budućnosti mogu otkloniti ili umanjiti uzroke nastajanja zagađenja i prirodnih katastrofa.

Povezivanje SW čvorova na distribuiranim geografskim lokacijama riješeno je korištenjem *Virtual Private Network* (VPN) mreže, koja obezbjeđuje povezivanje čvorova u podmrežu Interneta [Vujović15], a postavljena je u svrhu eksperimenta. Detalji kreiranja i postavljanja VPN-a izalze van okvira ove disertacije i neće biti detaljno razmatrani.

5.2.2.1. Osnovni slučaj monitoringa parametara životne sredine na ograničenom geografskom području

Opis okruženja:

U svrhu analize kreiranog sistema na otvorenom prostoru, izabran je dio geografskog područja koje se sastoji od stambenih četvrti i ukupne je površine oko 3km^2 (Slika 5.2.2.1.1.).



Slika 5.2.2.1.1. Kartografski prikaz geografskog područja odabranog za mjerenje parametara životne sredine na otvorenom prostoru

SW čvorovi, njihov ID i raspored na geografskom području su prikazani na slici (Slika 5.2.2.1.1.), dok su osnovni parametri svakog od čvorova dati u tabeli (Tabela 5.2.2.1.1.).

Tabela 5.2.2.1.1. Parametri SW čvorova

ID	IP adresa	MAC adresa	GPS koordinate	TTC05	RH31	MQ2	LM393
1	192.168.0.21	80:1F:02:AF:4A:56	43.818394, 18.362424	+	+	+	+
2	192.168.0.22	80:1F:02:AF:4A:3F	43.821212,18.364441	+	+	+	+
3	192.168.0.23	80:1F:02:AF:4A:37	43.824525,18.355859	+	+	+	+
4	192.168.0.24	80:1F:02:D6:19:28	43.822466,18.371243	+	+	+	+
5	192.168.0.25	80:1F:02:AF:4A:7C	43.823193,18.374483	+	+	+	+
6	192.168.0.26	80:1F:02:AF:2D:1D	43.825763,18.369462	+	+	+	–

7	192.168.0.27	80:1F:02:AF:2D:17	43.826970,18.358970	+	+	+	−
8	192.168.0.28	80:1F:02:AF:4A:73	43.831576,18.377263	+	+	+	−

Pristup senzorskim podacima sa SW čvorova podržan je skupom RESTful servisa kojima se pristupa uz oslonac na sljedeće URI adrese:

```
GET: [server_address]/RPISensorWeb/sw/info
GET: [server_address]/RPISensorWeb/sw/info/data
GET: [server_address]/RPISensorWeb/sw/ttC05
GET: [server_address]/RPISensorWeb/sw/ttC05/data
GET: [server_address]/RPISensorWeb/sw/rh31
GET: [server_address]/RPISensorWeb/sw/rh31/data
GET: [server_address]/RPISensorWeb/sw/mq2
GET: [server_address]/RPISensorWeb/sw/mq2/data
GET: [server_address]/RPISensorWeb/sw/lsmlm393
GET: [server_address]/RPISensorWeb/sw/lsmlm393/data
```

a kao rezultat njihovog poziva klijentu se vraća JSON koji obezbjeđuje informacije sa senzora u skladu sa formatima prikazanim u poglavlju 5.1.2.3.

Formulacija zahtjeva:

Za prikazani raspored i definisane parametre senzora (Slika 5.2.2.1.1.), neophodno je obezbijediti pristup podacima uz oslonac na servisni sloj za pristup SW čvorovima, koji ujedno obezbjeđuje i enkapsulaciju distribuiranih SW čvorova. Na osnovu toga klijent nema svijest o postojanju pojedinih SW čvorova i njihovom stvarnom geografskom položaju, već samo o postojanju interfejsa servisnog sloja i pojedinačnih servisa. U sklopu osnovnog eksperimenta formulisani su servisi koji trebaju da obezbijede informacije o:

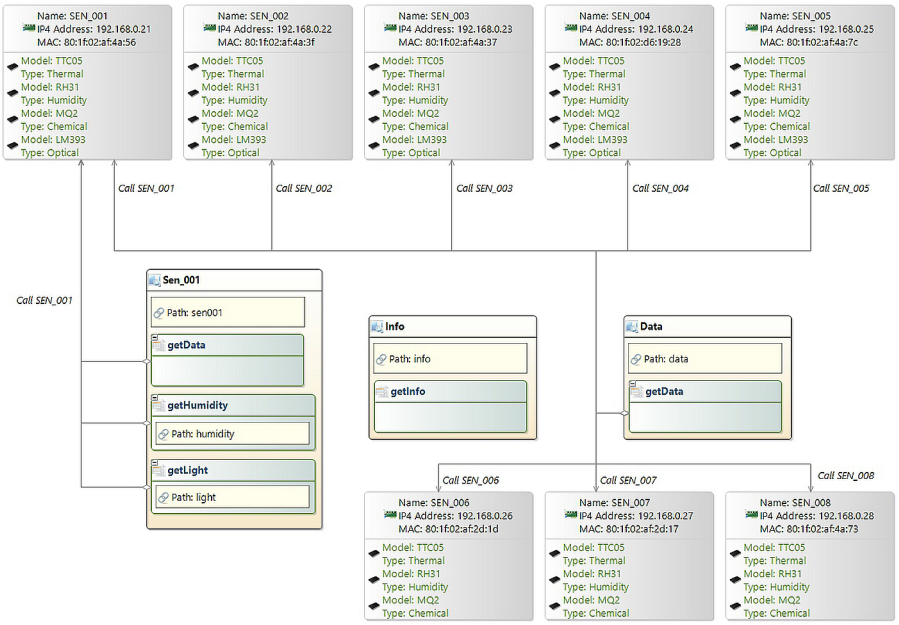
- infrastrukturi senzorskih čvorova (broj SW čvorova koji se nalaze na servisu, broj senzora na svakom od SW čvorova kao i njihov tip),
- kumulativni prikaz svih podataka (očitavanje podataka sa svih SW čvorova i njihovih senzora), i
- pojedinačan pristup SW čvorovima (jednom čvoru i njegovim pojedinačnim senzorskim elementima).

Kreiranje modela:

Uz oslonac na alat *SWServ,* kreiran je model *SWServOutdoor* koji opisuje predloženu arhitekturu SW mreže na otvorenom prostoru, odnosno SW čvorove, senzore koje posjeduju SW čvorovi, klase, metode i pozive između metoda i SW čvorova (Slika 5.2.2.1.2.).

Na osnovu opisa okruženja i formulisanih zahtjeva, u modelu je kreirano pet *Sensor Node* objekata (ID 1 − 5) sa po četiri senzorska

elementa i tri *Sensor Node* objekta (ID 6 − 8) sa po tri senzorska elementa koji odgovaraju formulisanim SW čvorovima i tri *Service Class* klase:



Slika 5.2.2.1.2. SWServOutdoor model

- *Info* (*getInfo*) – klasa koja prikuplja informacije o SW čvorovima i servisu,
- *Data* (*getData*) – klasa koja obezbjeđuje akumulacioni prikaz informacija sa SW čvorova, i
- *SEN_001* (*getData*, *getHumidity*, *getLight*) – klasa koja obezbjeđuje podatke sa jednog senzorskog čvora (u kreiranom primjeru čvor *SEN_001*), odnosno podatke sa pojedinačnih senzora.

Na osnovu principa referenciranja u kreiranom modelu metoda *getData* iz klase *Data* ima svijest o svim *Sensor Node* objektima, dok metode *getData*, *getHumidity* i *getLight* iz klase *SEN_001* imaju svijest samo o jednom *Sensor Node* objektu.

Generisanje kôda:

Primjenom *SWServ* alata, na osnovu kreiranog *SWServOutdoor* modela, generisan je skup fajlova prikazan u tabeli (Tabela 5.2.2.1.2.).

Tabela 5.2.2.1.2. Pregled generisanih datoteka za SWServOutdoor model

SWServDoc	
Paket	**Generisane datoteke**
WebContent	index.html
WebContent	style.css
WebContent	Sen_001.html, Sen_002.html, Sen_003.html, Sen_004.html, Sen_005.html, Sen_006.html, Sen_007.html, Sen_008.html
WebContent/WEB-INF	web.xml
SWServGen	
Paket	**Generisane datoteke**
src/helper	ErrorCode.java
src/impl	Sen_001_Impl.java, Sen_002_Impl.java, Sen_003_Impl.java, Sen_004_Impl.java, Sen_005_Impl.java, Sen_006_Impl.java, Sen_007_Impl.java, Sen_008_Impl.java
src/service	Data.java, Info.java, Sen_001.java

Integracija u arhitekturu servisa:

Inkorporiranjem generisanih datoteka u dinamički JAVA projekat i implementacijom specifičnih detalja (dijelovi kôda koji obavljaju pozive i obradu informacija sa SW čvorova), kreira se servisni sloj za pristup SW čvorovima. Postavljanjem i pokretanjem kreiranog servisa na Tomcat serveru, vrši se integracija servisa u servisnu arhitekturu. Pristup servisnom sloju omogućen je uz oslonac na sljedeći skup URI adresa:

```
GET: [server_address]:8080/SWServOutdoor/swo/info
GET: [server_address]:8080/SWServOutdoor/swo/data
GET: [server_address]:8080/SWServOutdoor/swo/sen001
GET: [server_address]:8080/SWServOutdoor/swo/sen001/temp
GET: [server_address]:8080/SWServOutdoor/swo/sen001/air
```

Definisane adrese klijentima pružaju informacije u JSON formatu, respektivno:

```
{"creator":"Vladimir Vujovic","date":"05.11.2015","sesnor":8,
"type":"outdoor"}

{"data":[
{"sen_001":[
   {"type":"thermistor","model":"ttC05","temp":"15.3215"},
   {"type":"humidity","model":"rh31","percent":"30.2941"},
   {"type":"gas","model":"mq2","percent":"17.2549"},
   {"type":"photodetector","model":"lsmlm393","percent":"10.8431"}]},
{"sen_002":[
   {"type":"thermistor","model":"ttC05","temp":"17.7141"},
   {"type":"humidity","model":"rh31","percent":"25.2941"},
   {"type":"gas","model":"mq2","percent":"11.2154"},
   {"type":"photodetector","model":"lsmlm393","percent":"8.3137"}]},
{"sen_003":[
   {"type":"thermistor","model":"ttC05","temp":"18.2458"},
```

```
   {"type":"humidity","model":"rh31","percent":"33.3335"},
   {"type":"gas","model":"mq2","percent":"5.9889"},
   {"type":"photodetector","model":"lsmlm393","percent":"13.3198"}]},
{"sen_004":[
   {"type":"thermistor","model":"ttC05","temp":"17.2546"},
   {"type":"humidity","model":"rh31","percent":"29.9246"},
   {"type":"gas","model":"mq2","percent":"25.1124"},
   {"type":"photodetector","model":"lsmlm393","percent":"20.4510"}]},
{"sen_005":[
   {"type":"thermistor","model":"ttC05","temp":"22.3345"},
   {"type":"humidity","model":"rh31","percent":"50.9248"},
   {"type":"gas","model":"mq2","percent":"40.2455"},
   {"type":"photodetector","model":"lsmlm393","percent":"11.4902"}]},
{"sen_006":[
   {"type":"thermistor","model":"ttC05","temp":"18.5432"},
   {"type":"humidity","model":"rh31","percent":"51.3576"},
   {"type":"gas","model":"mq2","percent":"27.3658"}]},
{"sen_007":[
   {"type":"thermistor","model":"ttC05","temp":"17.9899"},
   {"type":"humidity","model":"rh31","percent":"30.2467"},
   {"type":"gas","model":"mq2","percent":"18.2485"}]},
{"sen_008":[
   {"type":"thermistor","model":"ttC05","temp":"18.7678"},
   {"type":"humidity","model":"rh31","percent":"29.2248"},
   {"type":"gas","model":"mq2","percent":"20.9458"}]}
]}

{"sensor_number":4,"data":[
   {"type":"thermistor","model":"ttc05","temp":"15.3215"},
   {"type":"humidity","model":"rh31","percent":"30.2941"},
   {"type":"gas","model":"mq2","percent":"17.2549"},
   {"type":"photodetector","model":"lsmlm393","percent":"68.2353"}
]}

{"temp":"15.3215"}

{"percent":"17.2549"}
```

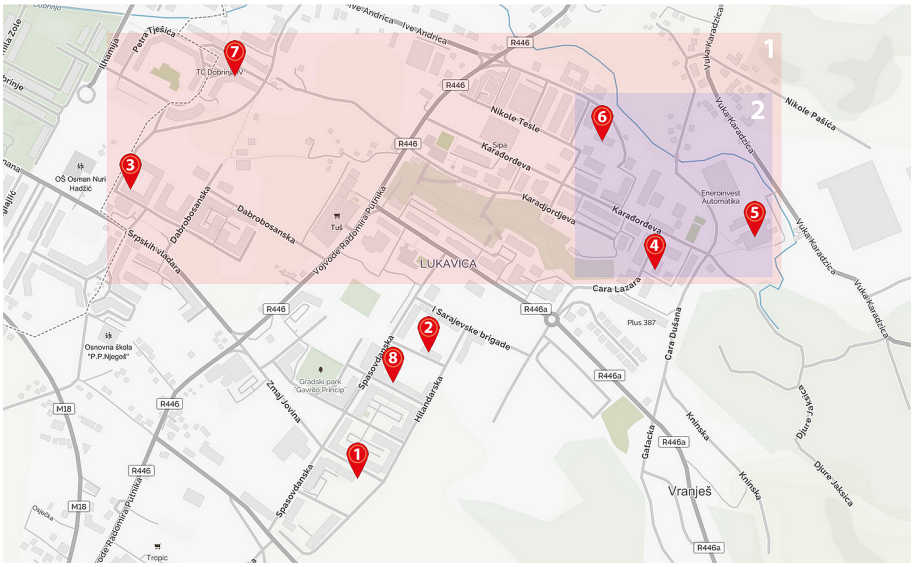

5.2.2.2. Modifikovani slučaj monitoringa parametara životne sredine na ograničenom geografskom području

Opis okruženja:

U sklopu postavljenog eksperimenta, osnovni model je restrukturiran promjenom servisnog sloja, koji podrazumijeva postavljanje novog okruženja i prateće formulacije zahtjeva. Oslanjajući se na izabrani dio geografskog područja koje je korišteno u osnovnom slučaju, izabrana su dva regiona (Slika 5.2.2.2.1.):

- region 1 – obuhvata pet SW čvorova i površine je približno 1km^2,
- region 2 – obuhvata tri SW čvora i površine je 0.2km^2.

Za oba regiona je potrebno kreirati servisni sloj za pristup podacima.



Slika 5.2.2.2.1. Kartografski prikaz geografskog područja odabranog za mjerenje parametara životne sredine na otvorenom prostoru

Na slici (Slika 5.2.2.2.1.) prikazani su SW čvorovi, njihov ID i raspored na geografskom području, dok su osnovni parametri svakog od čvorova i URI adrese za pristup senzorskim podacima isti kao i za osnovni slučaj.

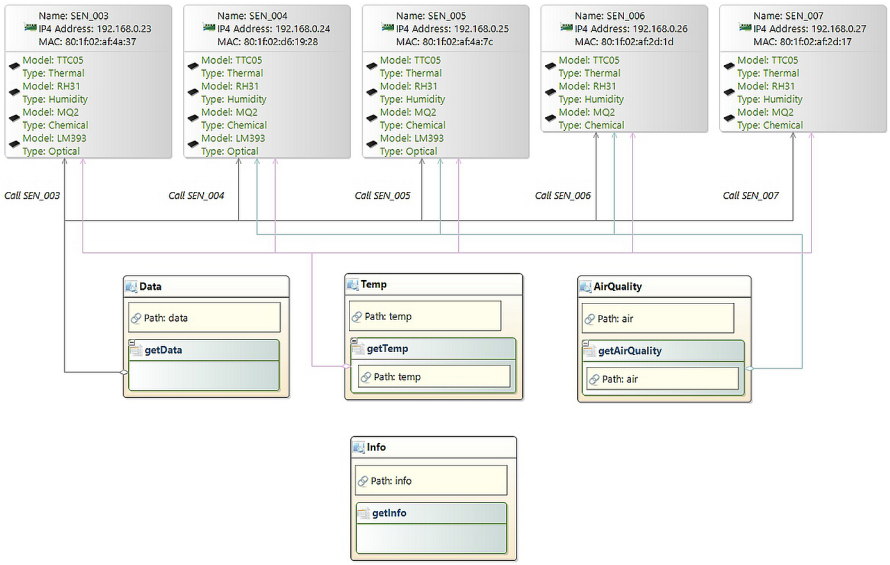
Formulacija zahtjeva:

U sklopu modifikovanog slučaja eksperimenta, za prikazani raspored i definisane parametre senzora (Slika 5.2.2.2.1.) formulisani su servisi koji trebaju da obezbijede informacije o:

- infrastrukturi senzorskih čvorova (broj SW čvorova koji se nalaze na servisu, broj senzora na svakom od SW čvorova kao i njihov tip),
- kumulativni prikaz svih podataka sa senzora iz regiona 1 (očitavanje podataka sa svih selektovanih SW čvorova i njihovih senzora koji se nalaze u regionu 1),
- kumulativni prikaz informacija o temperaturi sa svih senzora iz regiona 1 (očitavanje temperature sa svih selektovanih SW čvorova koji se nalaze u regionu 1), i
- kumulativni prikaz informacija o kvalitetu zraka sa svih senzora iz regiona 2 (očitavanje kvaliteta zraka sa svih selektovanih SW čvorova koji se nalaze u regionu 2).

Kreiranje modela:

Ekvivalentno prethodnom slučaju, uz oslonac na alat *SWServ* kreiran je model *SWServOutdoor Modified* koji opisuje predloženu arhitekturu SW mreže na otvorenom prostoru za izabrane regione i broj čvorova (Slika 5.2.2.2.2.).



Slika 5.2.2.2.2. SWServOutdoor Modified model

Na osnovu opisa okruženja i formulisanih zahtjeva, u modelu je kreirano tri *Sensor Node* objekta (ID 3 − 5) sa po četiri senzorska elementa i dva *Sensor Node* objekta (ID 6 − 7) sa po tri senzorska elementa koji odgovaraju formulisanim SW čvorovima i tri *Service Class* klase:

- *Info* (*getInfo*) – klasa koja prikuplja informacije o SW čvorovima i servisu,
- *Data* (*getData*) – klasa koja obezbjeđuje akumulacioni prikaz informacija sa SW čvorova koji se nalaze u regionu 1, i
- *Temp* (*getTemp*) – klasa koja obezbjeđuje samo podatke o temperaturi sa svih SW čvorova koji se nalaze u regionu 1, i
- *AirQuality* (*getAirQuality*) – klasa koja obezbjeđuje samo podatke o kvalitetu zraka sa svih SW čvorova koji se nalaze u regionu 2.

Na osnovu principa referenciranja u kreiranom modelu metoda *getData, getTemp* iz klasa *Data* i *Temp* respektivno, imaju svijest o svim *Sensor Node* objektima koji se nalaze u regionu 1, dok metoda

getAirQuality iz klase *AirQuality* ima svijest samo o *Sensor Node* objektima koji se nalaze u regionu 2.

Generisanje kôda:

Primjenom *SWServ* alata, na osnovu kreiranog *SWServOutdoor Modified* modela, generisan je skup datoteka prikazan u tabeli (Tabela 5.2.2.2.1.).

Tabela 5.2.2.2.1. Pregled generisanih datoteka za SWServOutdoor Modified model

SWServDoc	
Paket	**Generisane datoteke**
WebContent	index.html
WebContent	style.css
WebContent	Sen_003.html, Sen_004.html, Sen_005.html, Sen_006.html, Sen_007.html
WebContent/WEB-INF	web.xml
SWServGen	
Paket	**Generisane datoteke**
src/helper	ErrorCode.java
src/impl	Sen_003_Impl.java, Sen_004_Impl.java, Sen_005_Impl.java, Sen_006_Impl.java, Sen_007_Impl.java
src/service	AirQuality.java, Data.java, Info.java, Temp.java

Integracija u arhitekturu servisa:

Inkorporiranjem generisanih datoteka u dinamički JAVA projekat i implementacijom specifičnih detalja (dijelovi kôda koji obavljaju pozive i obradu informacija sa SW čvorova), kreira se servisni sloj za pristup SW čvorovima. Postavljanjem i pokretanjem kreiranog servisa na Tomcat serveru, vrši se integracija servisa u servisnu arhitekturu. Pristup servisnom sloju omogućen je uz oslonac na sljedeći skup URI adresa:

```
GET: [server_address]:8080/SWServOutdoor/swom/info
GET: [server_address]:8080/SWServOutdoor/swom/data
GET: [server_address]:8080/SWServOutdoor/swom/temp
GET: [server_address]:8080/SWServOutdoor/swom/air
```

Definisane adrese klijentima pružaju informacije u JSON formatu, respektivno:

```
{"creator":"Vladimir Vujovic","date":"05.11.2015","sesnor":5,
"type":"outdoor"}

{"data":[
{"sen_003":[
   {"type":"thermistor","model":"ttC05","temp":"22.7141"},
   {"type":"humidity","model":"rh31","percent":"20.2467"},
```

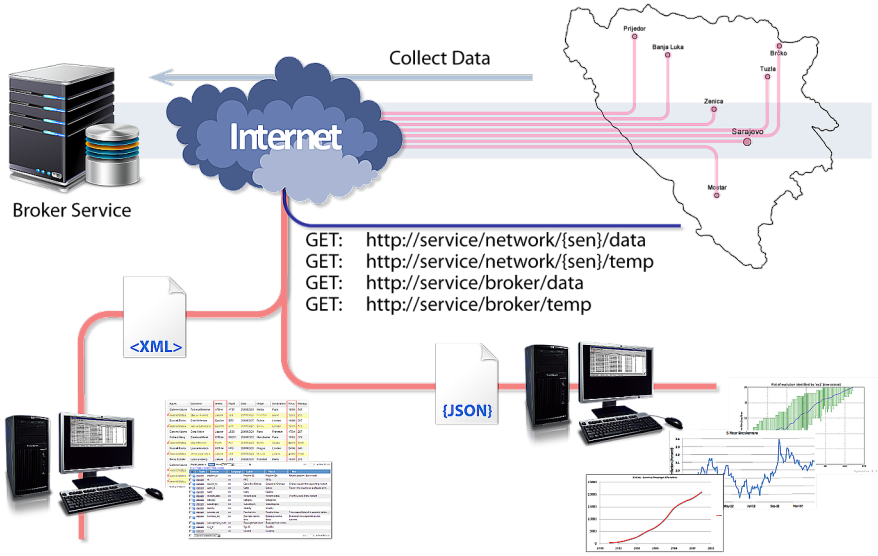
```
   {"type":"gas","model":"mq2","percent":"9.3122"},
   {"type":"photodetector","model":"lsm1m393","percent":"5.4902"}]},
{"sen_004":[
   {"type":"thermistor","model":"ttC05","temp":"20.2498"},
   {"type":"humidity","model":"rh31","percent":"30.9254"},
   {"type":"gas","model":"mq2","percent":"12.9425"},
   {"type":"photodetector","model":"lsm1m393","percent":"12.6275"}]},
{"sen_005":[
   {"type":"thermistor","model":"ttC05","temp":"18.3986"},
   {"type":"humidity","model":"rh31","percent":"24.289"},
   {"type":"gas","model":"mq2","percent":"16.2259"},
   {"type":"photodetector","model":"lsm1m393","percent":"8.6667"}]},
{"sen_006":[
   {"type":"thermistor","model":"ttC05","temp":"18.9758"},
   {"type":"humidity","model":"rh31","percent":"28.9861"},
   {"type":"gas","model":"mq2","percent":"25.2678"}]},
{"sen_007":[
   {"type":"thermistor","model":"ttC05","temp":"17.9102"},
   {"type":"humidity","model":"rh31","percent":"50.8241"},
   {"type":"gas","model":"mq2","percent":"57.8627"}]}
]}

{"data":[
   {"sen_003":{"temp":"22.7141"}},
   {"sen_004":{"temp":"20.2498"}},
   {"sen_005":{"temp":"18.3986"}},
   {"sen_006":{"temp":"18.9758"}},
   {"sen_007":{"temp":"17.9102"}}
]}

{"data":[
   {"sen_004":{"percent":"12.9425"}},
   {"sen_005":{"percent":"16.2259"}},
   {"sen_006":{"percent":"25.2678"}}
]}
```

5.3. Primjer upotrebe Senzor Web arhitekture

Ako rezultate pojedinačnih eksperimenata posmatramo kao nepovezane situacije u kojima je potrebno izvršiti nadzor nad lokalnim elementima, poseban izazov predstavlja integracija velikog broja izolovanih cjelina na širokom planu.

Moderni, na računarima bazirani sistemi (CBIS), koji se oslanjaju na SW mreže najčešće predstavljaju *Sisteme za akviziciju podataka – Data Aquisition Systems* (DAQ) [Vujović15a]. Danas je na tržištu dostupan veliki broj komercijalnih multifunkcionalnih DAQ sistema opšte namjene, koji imaju primjenu u industriji, zaštiti okoliša, procesu edukacije, monitoringa parametara životne sredine i prirodnih pojava. Oni najčešće

generišu velike količine podataka koji moraju biti visoke kvalitete, skalabilni i sa mogućnošću obrade u realnom vremenu. Oslanjajući se na nove tehnologije i koncepte kao što su IoT, WoT i SW, DAQ sistemi postaju globalno povezani sa senzorskim jedinicama, a korisnici podacima pristupaju putem servisa i definisanih interfejsa [Vujović15a] (Slika 5.3.1.)



Slika 5.3.1. IoT DAQ sistem za prikupljanje i analizu podataka [Vujović15a]

Četiri ključna elementa CBIS-a predstavljaju:

- *obrada* – predstavlja proces analize prikupljenih podataka. Algoritamski može biti veoma jednostavna ili kompleksna, pri čemu može uključivati druge kompleksne procese obrade kao što su: Fuzzy logika, neuronske mreže, algoritme predikcije, i sl.,
- *skladištenje* – predstavlja proces u kojem se prikupljeni podaci i rezultati obrada trajno čuvaju,
- *vizuelizacija podataka* – predstavlja proces u kojem se računarska i digitalna tehnologija koristi za prikazivanje kvantitativne i kvalitativne informacije, metematičkih ili naučnih modela, kao i mjerenih, statističkih ili izračunatih podataka [Cox04]. Prmjena vizuelizacije u sistemima koji rade u realnom vremenu imaju ulogu prevođenja uočenih vrijednosti u animirane i interaktivne grafičke 2D i 3D modele. Važnost primjene vizuelizacije podataka se može pronaći u mnogim različitim oblastima. Tako je u radovima [Cox04, Heer12, Heer10, Gershon94] analiziran opšti značaj vizuelizacije u

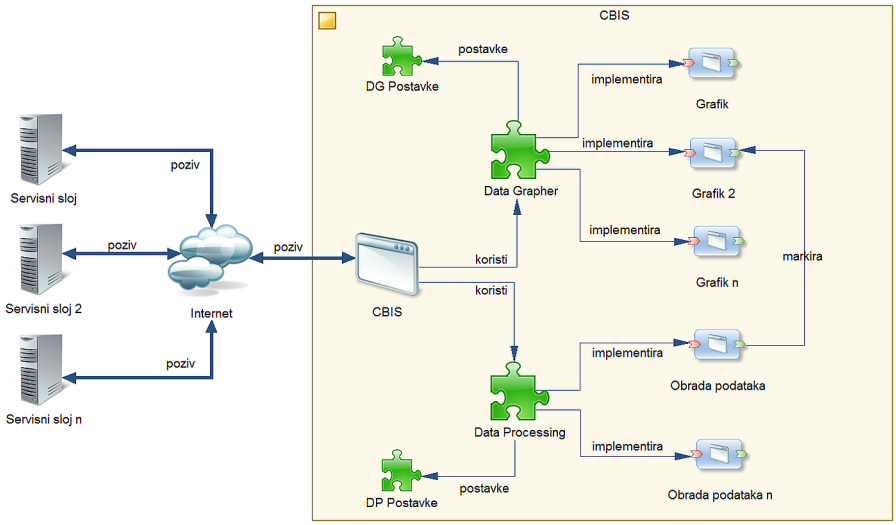
IS, dok je u [Cheng13, Chowdhry09] prikazana primjena vizuelizacije u sistemima koji rade u realnom vremenu, te SM, i

- *komunikacija* – predstavlja proces razmjene podataka između CBIS-ova.

U svrhu testiranja ranije generisanog servisnog sloja, kreiran je jednostavan prototip CBIS-a za obradu i vizuelizaciju podataka. Kreirani CBIS putem Interneta pristupa servisnom sloju u realnom vremenu i sa njega čita podatke preko definisanih URI adresa. Podaci se sa servisnog sloja dobijaju u JSON formatu, pa je u zavisnosti od prirode problema potrebno izvršiti njihovu obradu, analizu, vizuelizaciju i skladištenje. Kreirane su dvije komponente sistema [Maksimović15, Vujović13, Vujović15b]:

- *Data Grapher* – komponenta koja obavlja vizuelizaciju izmjerenih podataka u funkciji vremena, i
- *Data Processing* – komponenta koja obavlja obradu prikupljenih podataka sa senzorskog čvora.

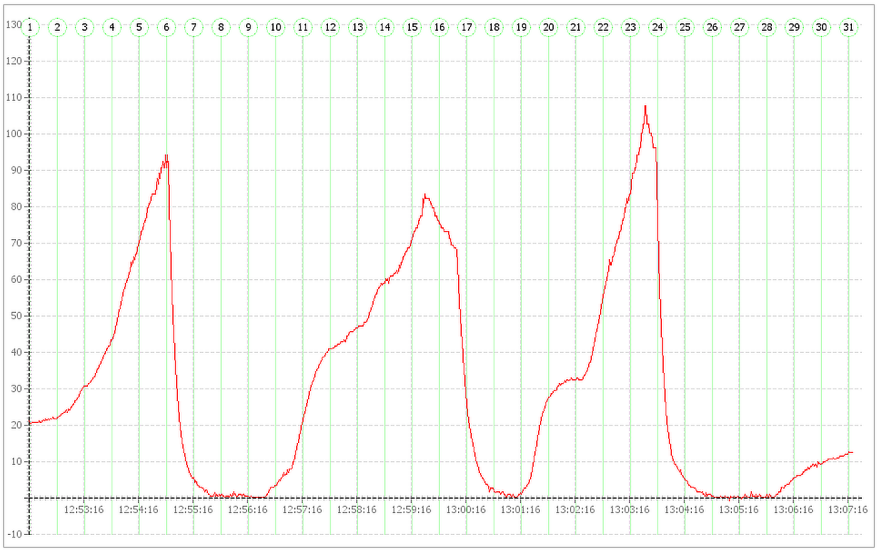
Arhitektura *Data Grapher* i *Data Processing* komponenti i njihova integracija sa CBIS sistemom prikazana je na slici (Slika 5.3.2.).



Slika 5.3.2. Arhitektura *Data Grapher* i *Data Processing* komponenti i integracija sa CBIS-om

5.3.1. Data Grapher

Vizuelizacija mjerenih podataka korištenjem *Data Grapher* komponente je prikazana na slici (Slika 5.3.1.1.).



Slika 5.3.1.1. Segment vizuelizacije mjerenih podataka za temperaturski senzor

Arhitektura ove komponente se temelji na RESTful klijentu i dizajnirana je za vizuelizaciju podataka dobijenih sa bilo kojeg servisnog sloja ili SW čvora koji obezbjeđuje podatke sa senzora u JSON formatu. Kako su prikupljeni podaci u JSON formatu, cijela obrada podataka vrši se na strani klijenta. Kao rezultat toga, klijent ima mogućnost manipulacije širokim opsegom postavki, a izlaz može biti prilagođen po potrebi korisnika. Budući da se podaci sa senzora prikupljaju u realnom vremenu, primarni cilj predstavlja određivanje vremenskog okvira u kojem se obavlja mjerenje, kao i podešavanje opsega mjerenja te intervala uzorkovanja.

Data Grapher komponenta, kroz postavke, omogućava izbor tipa senzora, kao i izbor odnosa prikaza mjerenih podataka i veličine grafika. Pored toga, *Data Grapher* komponenta omogućava i uvođenje vizuelnih indeksiranih identifikatora (markera), koji mogu biti proizvoljno postavljeni u određenim trenucima obrade.

5.3.2. Data Processing

Za razliku od *Data Grapher* komponente, koja ima samo jednu ulogu, tj. vizuelizaciju podataka sa senzora, *Data Processing* komponenta vrši obradu istih. Obrada podataka može biti bazirana na jednostavnim, unaprijed definisanim pravilima, ali isto tako može biti bazirana i na složenim matematičkim modelima i odlučivanju na temelju pravila u domenu vještačke inteligencije.

Prednost pristupa u kom se procesorska jedinica i servisni sloj nalaze na istom mjestu ili su direktno povezani putem Interneta, posebno se može vidjeti u procesu kreiranja klijenata sa različitim strategijama obrade podataka. Tako se, pristupom servisnom sloju preko REST servisa, klijentima pruža mogućnost izbora vlastite strategije obrade podataka.

Na slici (Slika 5.3.2.1.) prikazan je segment obrade podataka sa temperaturskog senzora od strane *Data Processing* komponente.

```
Legend:
##### - Very low critical
##### - Low critical
##### - Medium critical
##### - High critical
##### - Very high critical

Date of measurement: 19.10.15

No.:    Time:     PrevTemp: CurrTemp: Delta:    Appr.1:   Appr.2:   Deg.:
-------------------------------------------------------------------------------
Start at: 12:52:16
1.      12:52:16  20,80     20,80     + 00,00   21,27     21,27     A1( 1[00,28] 3[00,28] 11[00,28] 13[00,72] )       A2( 2[00,28] 7[00,72] )
2.      12:52:46  20,80     21,84     + 01,04   22,11     22,20     A1( 1[00,21] 3[00,28] 11[00,21] 13[00,72] )       A2( 2[00,21] 3[00,21] 7[00,79] 8[00,21] )
3.      12:53:16  21,84     30,33     + 08,49   34,97     40,82     A1( 3[00,21] 5[00,21] 13[00,64] 15[00,36] )       A2( 8[00,30] 9[00,64] 13[00,30] 14[00,36] )
4.      12:53:46  30,33     43,10     + 12,78   57,34     66,45     A1( 15[00,64] 17[00,21] 25[00,36] 27[00,21] )     A2( 14[00,44] 15[00,56] 19[00,21] 20[00,21] )
5.      12:54:16  43,10     69,55     + 26,45   90,82     92,14     A1( 27[00,03] 29[00,79] 37[00,03] 39[00,21] )     A2( 20[00,03] 25[00,97] )
6.      12:54:46  69,55     90,82     + 21,27   92,14     92,20     A1( 39[00,03] 49[00,97] )       A2( 25[01,00] )
7.      12:55:16  90,82     05,05     - 85,77   25,00     07,75     A1( 42[01,00] )       A2( 1[01,00] )
8.      12:55:46  05,05     01,84     - 03,21   07,75     08,47     A1( 1[01,00] )       A2( 1[00,64] 2[00,36] )
9.      12:56:16  01,84     00,16     - 01,68   07,75     08,43     A1( 1[01,00] )       A2( 1[00,34] 2[00,66] )
10.     12:56:46  00,16     03,47     + 03,31   07,75     20,48     A1( 1[01,00] )       A2( 2[00,34] 3[00,66] )
11.     12:57:16  03,47     20,80     + 17,33   21,27     50,00     A1( 1[00,28] 3[00,72] )       A2( 5[00,28] 10[00,72] )
12.     12:57:46  20,80     40,07     + 19,27   50,24     75,06     A1( 5[00,28] 7[00,00] 15[00,72] 17[00,00] )       A2( 15[01,00] 20[00,00] )
13.     12:58:16  40,07     46,81     + 06,74   61,72     66,99     A1( 25[00,55] 27[00,45] 35[00,00] 37[00,00] )     A2( 13[00,55] 14[00,35] 18[00,45] 19[00,35] )
14.     12:58:46  46,81     59,04     + 12,22   79,03     91,29     A1( 27[00,55] 29[00,27] 37[00,45] 39[00,27] )     A2( 19[00,56] 20[00,44] 24[00,27] 25[00,27] )
15.     12:59:16  59,04     70,44     + 11,40   91,67     91,64     A1( 39[00,73] 49[00,27] )       A2( 24[00,72] 25[00,28] )
16.     12:59:46  70,44     76,31     + 05,87   92,20     91,86     A1( 49[01,00] )       A2( 23[00,83] 24[00,17] )
17.     13:00:16  76,31     28,51     - 47,79   32,51     32,51     A1( 43[00,77] 45[00,23] )       A2( 6[00,77] 11[00,23] )
18.     13:00:46  28,51     01,84     - 26,67   08,21     07,75     A1( 11[00,77] 21[00,23] )       A2( 1[01,00] )
19.     13:01:16  01,84     01,84     + 00,00   07,75     07,75     A1( 1[01,00] )       A2( 2[01,00] )
20.     13:01:46  01,84     27,44     + 25,60   30,73     55,73     A1( 3[00,84] 5[00,16] )       A2( 10[00,84] 15[00,16] )
21.     13:02:16  27,44     32,17     + 04,73   37,13     37,13     A1( 13[00,52] 15[00,48] 23[00,16] 25[00,16] )     A2( 7[00,05] 8[00,52] 12[00,05] 13[00,48] )
22.     13:02:46  32,17     54,73     + 22,55   73,97     91,64     A1( 15[00,02] 17[00,52] 25[00,02] 27[00,48] )     A2( 15[00,02] 20[00,98] )
23.     13:03:16  54,73     83,43     + 28,70   92,16     92,20     A1( 29[00,02] 39[00,98] )       A2( 25[01,00] )
24.     13:03:46  83,43     86,19     + 02,76   92,20     91,28     A1( 49[01,00] )       A2( 22[00,45] 23[00,55] )
25.     13:04:16  86,19     05,44     - 80,75   25,00     07,75     A1( 42[01,00] )       A2( 1[01,00] )
26.     13:04:46  05,44     01,01     - 04,43   07,75     07,97     A1( 1[01,00] )       A2( 1[00,89] 2[00,11] )
27.     13:05:16  01,01     00,16     - 00,85   07,75     08,08     A1( 1[01,00] )       A2( 1[00,17] 2[00,83] )
28.     13:05:46  00,16     00,16     + 00,00   07,75     07,75     A1( 1[01,00] )       A2( 2[01,00] )
29.     13:06:16  00,16     04,66     + 04,50   07,75     23,67     A1( 1[01,00] )       A2( 2[00,10] 3[00,90] )
30.     13:06:46  04,66     09,57     + 04,91   07,75     24,76     A1( 1[01,00] )       A2( 2[00,02] 3[00,98] )
31.     13:07:16  09,57     12,45     + 02,88   12,13     19,27     A1( 1[00,84] 3[00,16] )       A2( 2[00,42] 3[00,58] 7[00,16] 8[00,16] )
Stop at: 13:07:20
```

Slika 5.3.2.1. Segment obrade podataka

Data Processing komponenta može raditi samostalno ili u skladu sa *Data Grapher* komponentom. Tako svaki uzorak na slici (Slika 5.3.2.1.) odgovara jednom od markera prikazanih na slici (Slika 5.3.1.1.) i sadrži informacije o vremenu očitanja, prethodnoj temperaturi, trenutnoj temperaturi i razlici između trenutne i prethodno očitane vrijednosti temperature (delta). Na osnovu predloženog pristupa u radu

[Maksimović15], prikazana je vjerovatnoća nastanka požara: različitim bojama na slici (Slika 5.3.2.1.) su označene mogućnosti postojanja požara koja je podijeljena na 5 segmenata sa korakom od 20%, a analiza je izvršena korištenjem fuzzy logike i definisanog skupa pravila.

Poglavlje 6

Zaključak i pravci daljnjih istraživanja

6.1. Zaključna razmatranja

Problem koji se danas susreće prilikom projektovanja sistema baziranih na *Bežičnim senzorskim mrežama* (BSM) više nije u nedostatku, već naprotiv, u izboru optimalne tehnologije. Razvoj savremenih *informacionih i komunikacionih tehnologija* (IKT) dovodi do pojave novih paradigmi koje obuhvataju arhitekturna i implementaciona rješenja, pri čemu se znatno poboljšavaju postojeći koncepti.

Primjenom *Internet protokola* (IP) u uređajima sa ograničenim resursima (kao što su senzorski čvorovi), dovodi do radikalne promjene Interneta i pojave potpuno novog koncepta pod nazivom *Internet stvâri – Internet of Things* (IoT). Jedan od osnovnih gradivnih elemenata IoT-a jeste *Senzor Web* (SW) čvor koji predstavlja elementarni "*resurs*" u SW mreži. U savremenoj praksi on se načešće implementira primjenom principa i tehnologija arhitekture oslonjene na resurse - *Resource Oriented Architecture* (ROA).

Kao surogat Interneta, SW mreža se u opštem slučaju može posmatrati kao nestrukturirana kolekcija gradivnih elemenata koji uspostavljaju korespondenciju sa fenomenima niskog nivoa apstrakcije, bez eksplicitno nametnute topologije. Mogućnost upotrebe SW mreža ograničena je jedino repertoarom i stepenom sofisticiranosti servisa koji dinamički formiraju virtuelne klastere pojedinačnih elemenata mreže. Sa tog aspekta, servisna arhitektura predstavlja ključ za praktičnu upotrebu SW mreža.

Cilj ove disertacije predstavlja unapređenje procesa razvoja arhitekture sistema baziranih na SW mrežama uz oslonac na dinamičko generisanje servisnog sloja u svrhu povećanja produktivnosti, održivosti i smanjenja troškova razvoja, gdje se pod unapređenjem procesa razvoja arhitekture

smatra analiza, integracija i prilagođavanje postojećih sistema i pristupa projektovanja arhitekture senzorskih mreža, kao i sistema baziranih na IoT konceptima.

U sklopu uvodnih razmatranja (poglavlje 1.) definisan je problemski okvir disertacije koji obuhvata osnovne koncepte razmatrane u sklopu rada i uvedene su precizne definicije osnovnih pojmova korištenih, kako u fazi analize, tako i u fazama dizajna i implementacije.

Sistematizacija i analiza dostupne literature iz predmetne oblasti predstavljena je u sklopu drugog poglavlja.

Polazni deo analize usmjeren je ka elaboraciji zahtjeva koji se postavljaju pred sisteme bazirane na SW mrežama. Izolovane su tri osnovne cjeline SW mreže: *softverski sistem* (SS), *komunikacioni sloj* (KS) i *senzorske mreže* (SM). Pojedinačnom analizom postojećeg stanja, utvrđena je pozicija i funkcionalnost svake od navedenih cjelina, pri čemu su formulisani ključni elemenati neophodni za izgradnju sistema baziranih na SW mrežama. SS mora osigurati globalnu interoperabilnu podršku za obradu, skladištenje i prikaz velike količine podataka koji nastaju u sklopu SM-a. KS obezbeđuje vezu između SS i SM-a, i predstavlja element čiji izbor zavisi od primjene i upotrebe projektovanog sistema. SM obuhvataju senzorske čvorove i *gateway* uređaje, Njihovom analizom je utvrđena: priroda senzorskih čvorova, namjena, klasifikacija, kao i hardverski/soft-verski zahtjevi koje je neophodno zadovoljiti u postupku projektovanja.

U drugom dijelu, posvećena je pažnja analizi servis orijentisanih senzorskih mreža, koje predstavljaju temelj za uvođenje novih tehnoloških koncepta kao što su IoT, *Web of Things* (WoT) i *Senzor Web* (SW). Uz oslonac na definisane Web tehnologije formulisan je savremeni pristup za realizaciju sistema za monitoring i prikupljanje podataka pomoću BSM. Uz oslonac na nove protokole ovi koncepti omogućavaju integraciju hardverskih uređaja sa Internetom. Detaljna analiza tehnologija i praktične primjene IoT i WoT-a, dovodi do zaključka da REST servisi i ROA arhitektura predstavljaju optimalni izbor za implementaciju navedenih paradigmi. Korak dalje predstavlja koncept i implementacija SW elemenata koji obuhvataju sve ranije pomenute paradigme i pristupe za integraciju senzorskih čvorova sa Internetom putem Web-a.

Na kraju poglavlja izvršena je analiza softverskih rješenja za podršku u procesu projektovanja, planiranja i analize BSM, kao i pozicioniranje ciljeva disertacije u okviru predstavljenih alata za projektovanje senzorskih

mreža. U skladu sa tim, izvršena je elaboracija alata (platformi i okvira) na kojima se zasniva praktičan dio disertacije.

Ključni dijelovi disertacije prezentovani su u trećem i četvrtom poglavlju.

U poglavlju 3. definisani su aspekti primjene MDA koncepta na razvoj arhitekture SW mreža. U prvom dijelu je, u zavisnosti od predložene infrastrukture sistema, definisana arhitektura SW mreže koja uspostavlja veze između elemenata modela i parametara infrastrukture, i njihovo preslikavanje na metamodel koji služi za opis domena problema. U sklopu predložene arhitekture prepoznate su dvije logičke cjeline koje predstavljaju ključne segmente rješenja: *senzorska mreža* i *servisna arhitektura*. Oba segmenta su razmatrana kroz odabrane publikacije zasnovane na konceptima bliskim domenu teze, odnosno kroz koncepte REST arhitekture i IoT-a, iz čega je proizašao prijedlog koncepta MDA rješenja koje je predstavljeno na kraju poglavlja 3.

Oslanjajući se na koncepte definisane u poglavlju 2. i prikazane aspekte primjene MDA na razvoj arhitekture SW mreža u poglavlju 3., definisan je niz koraka koji dovode do kreiranja DSM rješenja, odnosno kreiranja domenski specifičnog jezika, interaktivnog grafičkog editora i alata za automatsku transformaciju modela u implementacione klase.

Detaljna elaboracija predloženog DSM rješenja data je u poglavlju 4. Definisan je domenski specifičan jezik uz detaljan opis metaklasa, nabrojanih i prilagođenih tipova podataka. Formulisani metamodel je organizovani u tri logičke cjeline: *Senzorska mreža*, *Servisna arhitektura* i *Prilagođeni tipovi podataka*. Za svaku predloženu metaklasu dat je detaljan opis parametara, OCL ograničenja koja su definisana u sklopu metamodela i inicijalne vrijednosti koje parametri mogu imati.

Druga cjelina obuhvata analizu arhitekture i koncepta *Sirius* okvira koji je, nakon uporedne analize, odabran jer smanjuje vrijeme dizajna, pojednostavljuje arhitekturu i znatno povećava ukupnu produktivnost u procesu kreiranja DSM grafičkih editora. Uvođenjem VSM-a kao i njegovih pratećih elemenata koji opisuju preslikavanje objekata i stilova prikaza, formirana je podloga za kreiranje prototipa interaktivnog grafičkog editora. Definisana su preslikavanja metaklasa na objekte *Sirius*-a i arhitektura predloženog VSM modela. Kreirani interaktivni grafički editor, njegova arhitektura i mogućnosti su opisani na kraju druge cjeline poglavlja 4.

U završnom dijelu poglavlja 4. opisan je *Acceleo* šablonski generator kôda, arhitektura predloženog rješenja, i šabloni za generisanje kôda i preslikavanje metaklasa na implementacione objekte. Za kreirani prototip, prikazani su generisani elementi (izvorni Java kôd, HTML stranice i XML) koji predstavljaju podršku za unapređenje procesa razvoja arhitekture sistema baziranih na SW mrežama uz oslonac na dinamičko generisanje servisnog sloja.

U petom poglavlju izvršena je eksperimentalna verifikacija predloženog modela i razvojnog okruženja čime je dokazana njhova praktična primjena. Verifikacija prototipa platforme obavljena je primjenom na odabranim realnim primerima monitoringa parametara životne sredine u zatvorenom i otvorenom prostoru koji se danas najčešće oslanjaju na sisteme bazirane na SW mrežama.

Na osnovu formulisanog modela arhitekture eksperimentalnog okruženja, realizovana je cjelokupna hardversko/softverska podrška koja je upotrebljena u četiri pojedinačno sprovedena eksperimenta. Detaljno je elaboriran izbor tehnologije (*Raspberry Pi* – RPi) i način implementacije SW čvorova upotrebljenih u pojedinačnim eksperimentima. U procesu kreiranja SW čvorova primjenom RPi-ja prepoznata su tri osnovna implementaciona segmenta:

- povezivanje RPi platforme sa senzorima koji prikupljaju informacije iz okruženja,
- kreiranje drajvera koji će olakšati pristup i rad sa senzorima, i
- konverzija RPi-ja u RESTful servis dostupan preko Interneta uz mogućnost pristupa podacima očitanim sa senzora.

Sva tri segmenta su detaljno analizirana i ilustrovana kroz skup primjera. Na kraju uvodnog izlaganja predstavljena je integracija GSM/GPRS tehnologija sa RPi platformom.

U drugom dijelu poglavlja 5. opisan je cjelokupan postupak esperimentalne provjere okruženja za razvoj SW arhitekture. Na osnovu postavljenih polaznih uslova, predložena je realizovana hardverska implementacija eksperimentalnih SW čvorova. Izborom eksperimentalnog ambijenta i postavljanje polaznih parametara SW čvorova stvoreni su početni uslovi za kreiranje neophodnih modela uz oslonac na tezom specificirano razvojno okruženje. Generisanjem kôda na osnovu kreiranih modela i integracijom u arhitekturu servisa, proces kreiranja eksperimentalnih servisnih slojeva je okončan. Ovako kreirani servisni slojevi testirani su na realnim SW mrežama. Modifikacijom polaznih postavki i okruženja, te ponovnim kreiranjem servisnog sloja,

demonstrirana je jednostavnost i prilagodljivost kreiranog okruženja u uslovima promjene zahtjeva i arhitekture SW mreže.

Mogućnost praktične upotrebe generisane arhitekture SW mreže ilustrovana je integracijom kreiranog servisnog sloja sa eksperimentalnim CBIS-om koji posjeduje dvije komponente:

- *Data Grapher* – komponenta koja obavlja vizuelizaciju izmjerenih podataka u funkciji vremena, i
- *Data Processing* – komponenta koja obavlja obradu prikupljenih podataka sa senzorskog čvora.

Pojedinačni eksperimenti i primjer njihove integracije sa CBIS, u potpunosti potvrđuju ispravnost tezom formulisanog pristupa i praktičnu upotrebljivost razvojnog okruženja koje je nastalo kao posljedica primjene tezom formulisanog koncepta modelom upravljanog razvoja arhitekture SW mreža.

Na osnovu analize rezultata istraživanja moguće je izvesti sljedeće opšte zaključke:

- Projektovanje sistema baziranih na BSM predstavlja interdisciplinarnu oblast koja uključuje ekspertska znanja, metode, tehnike i tehnologije iz oblasti: razvoja softverskih sistema (SS), komunikacionih tehnika, tehnologija, protokola, hardverske i softverske implementacije senzorskih mreža (SM).
- Najčešće korišćeni standardi u domenu komunikacija obuhvataju: IEEE 802.11 a/b/g/n, IEEE 802.15.1, IEEE 802.15.4, IEEE 802.16 i celularne mreže (2G/3G/4G).
- Za povezivanje BSM i Interneta potrebno je implementirati *Internet Protokol* (IP) na senzorskim čvorovima ili *gateway* uređajima.
- Za implementaciju novih koncepta (kao što su IoT i WoT koji obuhvataju primjenu standardnih Web tehnologija), koriste se HTTP, TCP/IP, IPv6 protokoli, SOA i ROA tehnologije (SOAP i REST servisi), te XML, JSON, kao i novi protokoli i standardi namjenjeni isključivo za uređaje male procesorske snage sa autonomnim izvorima napajanja (6LoWPAN, CoAP).
- Koncept "*Senzor Web*" obuhvata IoT, WoT, SOA i ROA paradigme i pristupe za integraciju senzorskih čvorova sa Internetom putem Web-a.
- Ključnu ulogu predstavlja korištenje softverskih alata u procesu projektovanja, planiranja i analize BSM.
- Razvoja skupa alata za projektovanje, planiranje i analizu BSM je potrebno zasnivati na:

  - primjeni principa MDA i DSM-a, što zahtjeva izbor odgovarajuće razvojne platforme i razvojnog okvira koji podržava definisanje domenski specifičnog jezika ali i pratećeg skupa alata (grafičkog editora, generatora kôda, itd.),
  - mogućnosti formulisanja ograničenja koja projektovana arhitektura mora ispoštovati,
  - obezbjeđenju adekvatne softverske podrške za kreiranje modela (tekstualni ili grafički alati) i njihovu automatsku transformaciju u izvršni oblik (automatsko generisanje kôda).
- Neophodno je izabrati odgovarajući okvir za implementaciju servisnog sloja. U rješenju nastalom implementacijom tezom fomulisanih principa izabran je JAX-RS API.
- Za definisanje domenski specifičnog jezika najčešće se koristi ECore model, dok se formalna ograničenja, pomoćni atributi, inicijalne vrijednost i operacije definišu OCL-om.
- *Sirius* predstavlja dovoljno dobar (gotovo optimalan) okvir za kreiranje interaktivnog grafičkog editora na bazi DSL-a.
- *Acceleo* predstavlja dovoljno dobar (gotovo optimalan) generator kôda za transformaciju modela u implementacione klase na bazi DSL-a.
- *Raspberry Pi* mikroprocesor predstavlja, sa aspekta hardvera, softvera, performanse i cijene, dovoljno dobar (gotovo optimalan) izbor za kreiranje SW čvorova, koji je moguće proširiti sa GSM/GPRS modulima.
- Proces kreiranja SW čvorova se može provesti u tri koraka: povezivanje hardvera (senzora i RPi jedinice), kreiranje drajvera za rad sa povezanim senzorima i implementacija servisa za pristup servisima putem Interneta.
- Kreirano okruženje za unapređenje procesa razvoja arhitekture sistema baziranih na SW mrežama znatno ubrzava proces razvoja servisnog sloja SW mreža.
- Kreirani modeli se mogu modifikovati i mijenjati.
- Generisanjem kôda na osnovu kreiranog modela značajno se smanjuju greške i povećava produktivnost (neophodne male izmjene ili dopune kôda).
- Integracija servisnog sloja je moguća sa bilo kojim klijentima koji imaju mogućnost parsiranja JSON formata poruka.

6.2. Rezultati i doprinos istraživanja

Sumirajući osnovne principe i rezultate istraživanja koji su prezentovani u okviru ove disertacije, moguće je zaključiti da primjena DSM pristupa u procesu kreiranja alata za podršku projektovanja i implementacije aritekture SW mreža predstavlja ispravan pristup koji dovodi do podizanja efikasnosti u procesu projektovanja.

Uz oslonac na predloženo DSM rješenje, proces kreiranja SW mreža i infrastrukture koja podržava koncepte bazirane na IoT i WoT standardima, moguće je povećati brzinu projektovanja i isporuke softverskih sistema koji se oslanjaju na SW mreže, kao i kvalitet isporučenog rješenja sa aspekta funkcionalnosti i pouzdanosti.

Mogućnost dinamičkog formiranja virtuelnih klastera pojedinačnih elemenata mreže predstavljaju ključ za praktičnu upotrebu SW mreža sa ciljem adekvatnog i pravovremenog dobijanja informacija bez fiksiranja topologije mreže.

Analizom i razmatranjem koncepta arhitekture SW mreža, njenom komunikacijom i interakcijom sa okruženjem i obradom prikupljenih podataka, stvara se osnova za obrazovanje budućih projektanata, kao i povratne informacije koje mogu služiti za daljnju analizu kvaliteta arhitekture SW mreža.

Dobijeni rezultati i konkretni doprinosi nastali kao rezultat rada na ovoj disertaciji mogu se sumirati u sljedećem:

d) Kreiran je dobro definisan proces i jezik za opis arhitekture SW mreža i servisnog sloja. Jezik obezbjeđuje jednostavno kreiranje arhitekture bez suštinskog poznavanja problema arhitekture SW mreže, kao i dinamičkih servisnih elemenata.
e) Kreirana su pravila za transformaciju modela baziranih na jeziku za opis arhitekture SW mreža i servisnog sloja u implementaciju. Mehanizam transformacije modela u implementaciju je bez gubitaka semantike čime je osigurana konzistentnost samog postupka.
f) Kreirano je radno okruženje i aplikativna podrška za automatizaciju procesa kreiranja arhitekture SW mreža i servisnog sloja na osnovu predloženih modela i pravila transformacije. Integracijom sa postojećim razvojnim okruženjem *Eclipse* IDE, obezbjeđena je podrška za modelom-upravljan razvoj softvera.

U periodu rada na izradi ove doktorske disertacije objavljeno je 37 (tridesetsedam) naučnih i stručnih radova koji imaju direktnu ili indirektnu povezanost sa temom doktorske disertacije. Od objavljenih radova posebno se ističu radovi u časopisima sa SCI liste sa impakt faktorom, ukupno 3 (tri) M23 (svi iz oblasti teze). Pored toga objavljeno je i 6 (šest) radova u međunarodnim časopisima kategorije M24, 2 (dva) rada u domaćim časopisima kategorije M52, 23 (dvadesettri) rada na međunarodnim konferencijama kategorije M33 i 3 (tri) rada na domaćim skupovima kategorije M53.

Detaljan pegled svih objavljenih publikacija dat je u Prilogu A.

6.3. Pravci daljnjih istraživanja

Oslanjajući se na disertacijom dobijene rezultate, mogući pravci daljnjeg istraživanja i razvoja obuhvataju:

- Proširenje interaktivnog grafičkog editora sa različitim kolekcijama pogleda i dijalektima (dijagrami, tabele, matrice, i sl.).
- Modifikaciju definisanih koncepta u cilju usklađivanja sa novim standardima i pravilima.
- Modifikaciju implementiranog rješenja u cilju oslanjanja na informacije definisane u bazama podataka (informacije o senzorskim i *gateway* čvorovima).
- Proširenje definisanih koncepta i elemenata DSL-a sa ciljem ostvarenja podrške za:
  - kreiranje slojevitih prikaza podataka unutar virtuelnih klastera,
  - rad sa bazama podataka koje omogućavaju odloženi (naknadni) pristup i analizu podataka, njihovo sortiranje i pretragu,
  - generisanje implementacionih elemenata niskog nivoa, odnosno servisa na SW čvorovima u zavisnosti od njihove arhitekture i hardverske implementacije.
- Potpunu automatizaciju procesa transformacije modela u implementacione klase.

Prilog A

Publikovani radovi u okviru istraživanja

2015

1. V. Vujović, “Development of a custom Data Acquisition System based On Internet Of Things”, UNITECH’15 20/21 nov. 2015., pp. 339-343, 2015
2. V. Vujović, S. Jokić i M. Maksimović, “Power Efficiency analysis in Internet of Things Sensor nodes”, 2nd International Electronic Conference on Sensors and Applications, Vol. 2, pp. 1-6, DOI 10.3390/ecsa-2-D005, 2015
3. V. Vujović i M. Maksimović, “Data acquisition and analysis in educational research based on Internet of Things”, 11th International conference “Interactive Systems: Problems of Human-Computer Interactions", pp. 57-62, ISBN 978-5-9795-1412-3, Ulyanovsk, Russia, 2015
4. V. Vujović i M. Maksimović, “The Impact of the Internet of Things on Engineering Education”, The Second International Conference on Open and Flexible Education - ICOFE2015, pp. 135-144, ISBN 978-988-8238-08-8, Hong Kong, 2015
5. M. Maksimović, V. Vujović i E. Omanović-Mikličanin, “A Low Cost Internet of Things Solution for Traceability and Monitoring Food Safety During Transportation”, 7th International Conference on Information and Communication Technologies in Agriculture, Food and Environment (HAICTA 2015), pp. 583-593, 2015
6. V. Vujović i M. Maksimović, “Savremeni tehnološki pristup u procesu praćenja i kontrole hrane”, Glasnik Instituta za standardizaciju Bosne i Hercegovine, Vol. IX, No. 1-2, pp. 36-41, 2015
7. V. Vujović i M. Maksimović, “Raspberry Pi as a Sensor Web node for home automation”, Computers & Electrical Engineering, ISSN 0045-7906, DOI 10.1016/j.compeleceng.2015.01.019, 2015
8. M. Maksimović, V. Vujović, B. Perišić i V. Milošević, “Developing a fuzzy logic based system for monitoring and early detection of residential fire based on thermistor sensors”, Computer Science and Information Systems, Vol. 12, No. 1, pp. 63-89, ISSN 1820-0214 (Print) 2406-1018 (Online), DOI 10.2298/CSIS140330090M, 2015
9. V. Vujović, M. Maksimović, D. Kosmajac i B. Perišić, “Resource: A connection between Internet of Things and Resource-Oriented Architecture”, European

Conference on Smart Objects, Systems and Technologies - Smart SysTech 2015, pp. 1-7, ISBN 978-3-8007-3996-7, 2015

10. M. Maksimović, V. Vujović i B. Perišić, "A Custom Internet of Things Healthcare System", 10th Iberian Conference on Information Systems and Technologies - CISTI'2015, Vol. 1, pp. 653-658, ISBN 978-898-98434-5-5, 2015

11. V Vujović, M. Maksimović, G. Balotić i P. Mlinarević, "Internet stvari – tehnički i ekonomski aspekti primjene", INFOTEH-JAHORINA, Vol. 14, pp. 658-663, 2015.

12. E. Omanović-Mikličanin, M. Maksimović i V. Vujović, "The Future of Healthcare: Nanomedicine and Internet of Nano Things", Folia Medica Facultatis Medicinae Universitatis Saraeviensis, Vol. 50, No. 1, pp. 23-28, ISSN 0352-9630, 2015

13. E. Omanović-Mikličanin, M. Maksimović i V. Vujović, "The Future of Healthcare: Nanomedicine and Internet of Nano Things", 1st Conference on Medical and Biological Engineering in Bosnia and Herzegovina, 2015.

14. V. Vujović, M.Maksimović, B. Perišić i G. Milošević, "A Proposition of Low Cost Sensor Web Implementation Based on GSM/GPRS Services", 2015 IEEE 1st International Workshop on Consumer Electronics, 2015.

2014

1. M. Maksimović, V. Vujović i V. Milošević, "Fuzzy Logic and Wireless Sensor Networks – A Survey", Journal of Intelligent and Fuzzy System, Vol. 27, No. 2/2014, pp. 877–890, ISSN 1064-1246 (Print) 1875-8967 (Online), DOI 10.3233/IFS-131046, 2014

2. M. Maksimović i V. Vujović, "Sistemi za praćenje i kontrolu požara u zatvorenom prostoru", Glasnik Instituta za standardizaciju Bosne i Hercegovine, No. 3-4, pp. 4-8, ISSN 1840-2860, 2014

3. V. Vujović, M. Maksimović i B. Perišić, "Development of DSM Graphical Editor for RESTful Sensor Web Networks Modeling", Scientific Bulletin of the "Politehnica" University of Timisoara, Romania, Transactions on Automatic Control and Computer Science, Vol. 59(73), No. 2, pp. 131-140, ISSN 1224-600X , 2014

4. V. Vujović, M. Maksimović i B. Perišić, "A DSM for a Modeling RESTful Sensor Web Network", Athens Journal of Technology and Engineering, Vol. 1, No. 3, pp. 209-222, ISSN 2241 - 8237, 2014

5. M. Maksimović, V. Vujović i V. Milošević, "Applying Fuzzy Logic and Data Mining Techniques in Wireless Sensor Network for Determination Residential Fire Confidence", Journal of Engineering Science and Technology Review , Vol. 7, No. 4, pp. 89 - 96, ISSN 1791-2377 , 2014

6. V. Vujović, M. Maksimović i B. Perišić, "A DSM for a Modeling RESTful SensorWeb Network", 10th Annual International Conference on Information Technology & Computer Science, No. COM2014-0941, ISSN 2241 - 2891, 2014

7. V. Vujović, M. Maksimović, B. Perišić i V. Milošević, "A Graphical Editor for RESTful Sensor Web Networks Modeling", 9th IEEE International Symposium on Applied Computational Intelligence and Informatics, pp. 61-66, ISBN 978-1-4799-4694-5, 2014

8. M. Maksimović, V. Vujović i V. Milošević, "Analysis of various parameters

influence on heat detector response", International Scientific Conference "UNITECH 2014" , Vol. 1, pp. 226-230, ISSN 1313-230X, 2014

9. H. Hasić i V. Vujović, "Civil law protection of the elements comprising the "Internet of Things" from the perspective of the legal owner of the property in question", INFOTEH-JAHORINA, Vol. 13, pp. 1005-1010, 2014

10. V. Vujović, M. Maksimović i B. Perišić, "Comparative analysis of DSM Graphical Editor frameworks: Graphiti vs. Sirius", 23nd International Electrotechnical and Computer Science Conference ERK 2014, pp. 7-10 , 2014

11. M. Maksimović, V. Vujović, V. Milošević i B. Perišić, "Evaluating the optimal heat detector deployment for fire detection", International Conference "Engineering & Telecommunication En&T 2014", pp. 129 - 134, ISBN 978-1-4799-7011-7, UDK 004.82+621.39, DOI 10.1109/EnT.2014.31, 2014

12. M. Maksimović, V. Vujović, V. Milošević i B. Perišić, "Increasing the lifetime of hexagonal deployed Wireless Sensor Web Network" , ICIST 2014 , pp. 131-136, ISBN 978-86-85525-14-8, 2014

13. D. Kosmajac, V. Vujović, M. Maksimović, N. Davidović i B. Perišić, "MasterBroker: REST oriented Service Broker", IEEE 18 th International Conference on Intelligent Engineering Systems, pp. 227-232, ISBN 978-1-4799-4616-7, 2014

14. V. Vujović i M. Maksimović, "Raspberry Pi as a Wireless Sensor Node: Performances and Constraints", The 37th International ICT Convention – MIPRO 2014, pp. 1247-1252, ISSN 1847-3938, ISBN 978-953-233-078-6 , 2014

15. M. Maksimović, V. Vujović, N. Davidović, V. Milošević i B. Perišić, "Raspberry Pi as Internet of Things hardware: Performances and Constraints", 1st International Conference on Electrical, Electronic and Computing Engineering - IcETRAN 2014, 2014

16. V. Vujović, M. Maksimović i B. Perišić, "Sirius: A Rapid Development of DSM Graphical Editor", IEEE 18th International Conference on Intelligent Engineering Systems , pp. 233-238, ISBN 978-1-4799-4616-7, 2014

17. M. Maksimović, V. Vujović i V. Milošević, "The fire possibility prediction based on fuzzy logic generated dataset", YU INFO 2014, pp. 492-496, ISBN 978-86-85525-13-1 , 2014

2013

1. M. Maksimović, V. Vujović i V. Milošević, "Mining and predicting rate of rise heat detector data", Facta Universitatis, Series: Working and Living Environmental Protection, Vol. 10, No. 1, pp. 37 – 51, 2013

2. M. Maksimović, V. Vujović i D. Kosmajac, "Fuzzy rule reduction influence on system's accuracy", 21st Telecommunications forum TELFOR 2013 , pp. 920-923, 2013

3. M. Maksimović i V. Vujović, "Uloga Internet baziranih bežičnih senzorskih mreža u zaštiti od požara", INFOTEH-JAHORINA, Vol. 12, pp. 629-634, 2013

4. V. Vujović, M. Maksimović, D. Kosmajac, V. Milošević i B. Perišić, "Web Integration of REST Enabled Wireless Sensor Networks for Fire Detection", International conference on Applied Internet and Information Technologies AIIT ,

pp. 30-35, 2013

5. M. Maksimović i V. Vujović, “Comparative analysis of data mining techniques applied on wireless sensor network data for fire detection”, Journal of Information Technology and Applications - JITA, Vol. 2, pp. 65-77, 2013

2012

1. D. Kosmajac i V. Vujović, “Sigurnost informacionih sistema i proširenje sigurnosti u Jersey RESTful framework-u”, TELFOR 2012, pp. 1556-1559, ISBN 978-1-4673-2983-5, DOI 10.1109/TELFOR.2012.6419518, 2012

Bibliografija

[10kntc] Resistance table for 10kNTC Thermistor, online: http://coldtears.lin3.siteonlinetest.com/files/10kNTC.pdf [Dostupno: 07.11.2015]

[Aalst14] W. M. P. van der Aalst, "Data Scientist: The Engineer of the Future", Enterprise Interoperability VI, Proceedings of the I-ESA Conferences 7, K. Mertins et al. (Eds.), Springer, 2014, pp. 13-28.

[Abidin09] H. Z. Abidin i F. Y. A. Rahman, "Provisioning QoS in Wireless Sensor Networks using a Simple MaxMin Fair Biwidth Allocation", 2009 World Congress on Computer Science i Information Engineering, Vol.1, 2009, pp 44-48.

[Abuarqoub12] A. Abuarqoub, F. Al-Fayez, T. Alsboui, M. Hammoudeh i A. Nisbet, "Simulation Issues in Wireless Sensor Networks: A Survey", The Sixth International Conference on Sensor Technologies i Applications – SENSORCOMM 2012, 2012, pp. 222-228.

[Acceleo] Acceleo, online: https://eclipse.org/acceleo/ [Dostupno: 17.03.2015].

[Adama] V. R. Adama, "Wireless Sensor Network Architecture for Smart Buildings", online: http://82.130.102.95/misc/plag.pdf [Dostupno: 15.09.2015]

[Aggarwal13] C. C. Aggarwal, "An Introduction to Sensor Data Analytics", Managing i Mining Sensor Data, C. C. Aggarwal (Eds.), Springer US, 2013, pp. 1-8.

[Akyildiz02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam i E. Cayirci, "A Survey on Sensor Networks", IEEE Communications Magazine, 2002, pp 102-114.

[Akyildiz02a] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam i E. Cayirci, "Wireless sensor networks: a survey", Computer Networks, Vol. 38, 2002, pp 393-422.

[Alagar11] V. S. Alagar i K. Periyasamy, "Specification of Software Systems", Springer, 2011.

[Albert14] J. Albert, L. Barrère, S. Chaumette i D. Sauveron, "Wireless Sensors (Languages/Programming/Developments Tools/Exam-

ples)", Secure Smart Embedded Devices, Platforms i Applications, K. Markantonakis i K. Mayes, Springer, 2014, pp. 541-564.

[Ali10] F. Ali, "A Middleware to Connect Software Applications with Sensor Web", The International Journal of Technology, Knowledge i Society Vol. 6, No. 5, 2010, ISSN 1832-3669, pp. 27-35

[Almend09] J. M. Almendros-Jiménez, L. Iribarne, J. A. Asensio, N. Padilla i C. Vicente-Chicote, "An Eclipse GMF Tool for Modelling User Interaction", Visioning i Engineering the Knowledge Society: A Web Science Perspective, M. D. Lytras et al. (eds.), Springer, 2009, pp. 405-416.

[Alphi10] O. Alphi, A. Duda, M. Heusse, B. Ponsard, F. Rousseau i F. Theoleyre, "Towards the Future Internet of Sensors", The Internet of Things, D. Giusto et al. (eds.), Springer, 2010, pp. 309-318.

[Ammari14] H. M. Ammari, "The Art of Wireless Sensor Networks - Volume 1:Fundamentals", Springer, 2014.

[Amyot06] D. Amyot, H. Farah, i J.-F. Roy, "Evaluation of Development Tools for Domain-Specific Modeling Languages", System Analysis i Modeling: Language Profiles, R. Gotzhein i R. Reed (eds.), 2006, pp. 183-197.

[André14] E. André, M. M. Benmoussa i C. Choppy, "Translating UML State Machines to Coloured Petri Nets Using Acceleo: A Report", ESSS, 2014. pp. 1-7.

[Anliker04] U. Anliker, J.A. Ward, P. Lukowicz, G. Töster, F.Dolveck, M. Baer, F. Keita, E. Schenker,F. Catarsi, L. Coluccini, A. Belardinelli, D. Shklarski, M. Alon, E. Hirt, R. Schmid, i M.Vuskovic., "AMON: A Wearable Multiparameter Medical Monitoring i Alert System", IEEE Transactions on Information Technology in Biomedicine Vol.8.No.4, 2004, pp 415-427.

[Ausín13] J. L. Ausín, J.F. Duque-Carrillo, J. Ramos i G. Torelli, "From Hiheld Devices to Near-invisible Sensors: The Road to Pervasive e-Health", Pervasive & Mobile Sensing & Computting for Healthcare – SSMI2, S.C. Mukhopadhyay et al. (Eds.), Springer, 2013, pp. 135-156.

[Barka15] E. Barka, S. S. Mathew i Y. Atif, "Securing the Web of Things with Role-Based Access Control", Codes, Cryptology, i Information Security, S. El Hajji et al. (Eds.), Springer, 2015, pp. 14-26.

[Bassi13] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. van Kranenburg, S. Lange i S. Meissner, "Enabling Things to Talk", Springer, 2013.

[Bell13] C. A. Bell, "Beginning Sensor Networks with Arduino i Raspberry Pi", Apress Media, 2013.

[Bender14] A. Bender, S. Bozic i I. Kondov, “An EMF-based Toolkit for Creation of Domain-specific Data Services”, Proceedings of the 2nd International Conference on Model-Driven Engineering i Software Development, MODELSWARD 2014, L. F. Pires, S. Hammoudi, J. Filipe i R. C. das Neves (Eds.), 2014. pp. 30-40.

[Bender14] A. Bender, S. Bozic i I. Kondov, “An EMF-based Toolkit for Creation of Domain-specific Data Services”, Proceedings of the 2nd International Conference on Model-Driven Engineering i Software Development, MODELSWARD 2014, L. F. Pires, S. Hammoudi, J. Filipe i R. C. das Neves (Eds.), SCITEPRESS, 2014, pp. 30-40.

[Bharat01] A. Bharathidasan, V. Ani, S. Ponduru, “Sensor Networks: An Overview”, Department of Computer Science, University of California, 2001. Technical Report

[Bhiare08] T. Bhiare, “LTE i WiMAX Comparison”, Santa Clara University, 2008.

[Biermann06] E. Biermann, K. Ehrig, C. Köhler, G. Kuhns1, G. Taentzer, i E. Weiss, “Graphical Definition of In-Place Transformations in the Eclipse Modeling Framework”, Model Driven Engineering Languages i Systems, 2006, pp. 425-439.

[Blackburn08] M. Blackburn, “What’s Model Driven Engineering (MDE) i How Can it Impact Process”, People, Tools i Productivity, Systems i Software Consortium, Inc., 2008

[Botts06] M. Botts, A. Robin, J. Davidson i I. Simonis, “OpenGIS Sensor Web Enablement Architecture Document”, Open Geospatial Consortium, 2006.

[Botts07] M. Botts, G. Percivall, C. Reed i J. Davidson, “OGC Sensor Web Enablement: Overview I High Level Architecture”, OGC White Paper, Open Geospatial Consortium, 2007.

[Botts08] M. Botts, G. Percivall, C. Reed i J. Davidson, “OGC® Sensor Web Enablement: Overview i High Level Architecture”, GeoSensor Networks, S. Nittel, A. Labrinidis, i A. Stefanidis (Eds.), Springer, 2008, pp. 175-190.

[Botts14] M. Botts, “OGC® SensorML: Model i XML Encoding Stiard”, Open Geospatial Consortium, 2014.

[Brambilla12] M. Brambilla, J. Cabot i M. Wimmer, “Model-Driven Software Engineering in Practice”, Morgan & Claypool publishers, 2012

[Bri07] C. Bri, M. Gorning, T. Kaiser, J. Pasch i M. Wenz, “Graphiti - Development of High-Quality Graphical Model Editors”, 2007, online: http://www.eclipse.org/graphiti/documentation/files/EclipseMagazineGraphiti.pdf [Dostupno: 20.04.2015]

[Bröring11] A. Bröring, J. Echterhoff, S. Jirka, I. Simonis, T. Everding, C. Stasch, S. Liang i R. Lemmens, “New Generation Sensor Web

Enablement", Sensors, Vol. 11, 2011, pp. 2652-2699.

[Buchmann07] T. Buchmann, A. Dotor i B. Westfechtel, "Model-Driven Development of Graphical Tools: Fujaba Meets GMF", Proceedings of the 2nd International conference on Software i Data Technologies (ICSOFT 2007), INSTICC, 2007, pp. 425-430.

[Buchmann13] T. Buchmann, J. Baumgartl, D. Henrich i B. Westfechtel, "Towards A Domain-specific Language For Pick-I-Place Applications", Proceedings of the Fourth International Workshop on Domain-Specific Languages i Models for Robotic Systems (DSLRob 2013)., U. P. S. Christian Schlegel i S. Stinckwich, (eds.) arXiv.org, 2013.

[Buchmann14] T. Buchmann i P. Pezoldt, "A Lightweight Framework for Graphical Editors on Iroid Devices", Proceedings of the Ninth International Conference on Software Engineering i Applications, A. Holzinger, L. A. Maciaszek i S. J. Mellor (Eds.), SciTePress, 2014, pp. 81-89.

[Bull05] R. I. Bull i M. A. Storey, "Towards Visualization Support for the Eclipse Modeling Framework", A Research-Industry Technology Exchange at EclipseCon, 2005.

[Buratti09] C. Buratti, A. Conti, D. Dardari i R. Verdone, "An Overview onWireless Sensor Networks Technology i Evolution", Sensors,Vol. 9, 2009, pp: 6869-6896.

[Cabot12] J. Cabot i M. Gogolla, "Object Constraint Language (OCL): a Definitive Guide", Formal Methods for Model-Driven Engineering, M. Bernardo, V. Cortellessa i A. Pierantonio (Eds.), Vol. 7320, 2012, pp. 58-90.

[Camilo08] T. Camilo, P. Pinto, A. Rodrigues, J. Sa Silva i F. Boavida, "Mobility management in IP based Wireless Sensor Networks", International Symposium on World of Wireless Mobile i Multimedia Networks, Vol.23. No.26, 2008, pp1-8.

[Castillejo13] P. Castillejo, J.-F. Martínez, L. López, i G. Rubio, "An Internet of Things Approach for Managing Smart Services Provided by Wearable Devices", International Journal of Distributed Sensor Networks, Hindawi Publishing Corporation, 2013.

[Cecílio14] J. Cecílio i P. Furtado, "Wireless Sensors in Heterogeneous Networked Systems", Springer, 2014.

[Chen14] H. Chen-Da, L. Dong, Q. Jie-Fan, S. Hai-Long i C. Li, "SeaHttp: A resource-oriented protocol to extend REST style for Web of Things", Journal of Computer Science i Technology, Vol. 29, No. 2, 2014, pp. 205–215.

[Cheng13] T. Cheng, J. Teizer, "Real-time resource location data collection i visualization technology for construction safety i activity monitoring applications", Automation in Construction, Volume

34, September 2013, pp. 3–15.

[Chhimwal13] P. Chhimwal, D. S. Rai i D. Rawat, “Comparison between Different Wireless Sensor Simulation Tools”, IOSR Journal of Electronics i Communication Engineering (IOSR-JECE), Vol. 5, No. 2, 2013, pp. 54-60.

[Chinru06] J. Chinrungrueng, U. Sununtachaikul i S. Triamlumlerd, “A Vehicular Monitoring System with Power Efficient Wireless Sensor Networks”, ITS Telecommunications Proceedings, 6th International Conference on, 2006, pp 951-954.

[Chowdhry09] B. S. Chowdhry, N. M. White, J. K. Jeswani, K. Dayo, M. Rathi, “Visualization i Analysis of Wireless Sensor Network Data for Smart Civil Structure Applications Based On Spatial Correlation Technique”, AIP Conference Proceedings, Vol. 1146 Issue 1, 2009, pp. 113–122.

[Christin10] D. Christin, P. S. Mogre i M. Hollick, “Survey on Wireless Sensor Network Technologies for Industrial Automation: The Security i Quality of Service Perspectives”, Future Internet, Vol. 2, 2010, pp. 69-125.

[Chu07] X. Chu i R. Buyya, “Service Oriented Sensor Web”, Sensor Network i Configuration: Fundamentals, Techniques, Platforms, i Applications Experiments, N.P. Mahalik (Eds.), Springer, 2007, pp. 51–74.

[Chua11] T. E. Chua, M. Merlo i M. Bachman, “System of Systems for Sensor i Actuator Networks”, HCI International 2011 Posters' Extended Abstracts, C. Stephanidis (Eds.), Springer, 2011, pp. 13-17.

[Colitti14] W. Colitti, N. T. Long, N. De Caro i K. Steenhaut , “Embedded Web Technologies for the Internet of Things”, Internet of Things, S. C. Mukhopadhyay (ed.), 2014, pp. 55-74.

[Corredor12] I. Corredor, A.M. Bernardos, J. Iglesias i J. R. Casar, “Model-Driven Methodology for Rapid Deployment of Smart Spaces Based on Resource-Oriented Architectures”, Sensors, Vol. 12, No. 7, 2012, pp. 9286-9335.

[Cox04] D. J. Cox, “The Art i Science of Visualization: Metaphorical Maps i Cultural Models”, Technoetic Arts: A Journal of Speculative Research Vol. 2, No. 2, 2004, pp. 71–79.

[Cox10] S. Cox, “OGC Identifiers – the case for http URIs”, Open Geospatial Consortium, 2010.

[Culler04] D. Culler, D. Estrin i M. Srivastava, “Overview of Sensor Networks”, Computer, Vol. 37, No. 8, 2004, pp: 41-49.

[Czarnecki06] K. Czarnecki i S. Helsen, “Feature-Based Survey of Model Transformation Approaches”, IBM Systems Journal, Vol. 45, No. 3, 2006, pp. 621-645.

[daCosta13] F. daCosta, “Rethinking the Internet of Things: A Scalable Approach to Connecting Everything”, ApressOpen, 2013.

[Dai12] G. Dai i Y. Wang, “Design on Architecture of Internet of Things”, Advances in Computer Science i Information Engineering Vol. 1, D. Jin i S. Lin (Eds.), Springer, 2012, pp. 1-7.

[Damus14] C. Damus, A. S.-B. Herrera, A. Uhl, E. Willink i contributors, “OCL 5.0.0 Documentation”, Eclipse OCL 5.0, 2014.

[Dejanović11] I. Dejanović, “Prilog metodama brzog razvoja softvera na bazi proširivih jezičkih specifikacija”, Novi Sad, 2011;

[Delgado13] J. Delgado, “Service Interoperability in the Internet of Things”, Internet of Things & Inter-cooperative Computational Technologies for Collective Intelligence, N. Bessis et al. (Eds.), Springer, 2013, pp. 51-87.

[Delicato13] F. C. Delicato, P. F. Pires i T.Batista, “Middleware Solutions for the Internet of Things”, Springer, 2013.

[Delin02] K. A. Delin, “The Sensor Web: A Macro-Instrument for Coordinated Sensing”, Sensors, Vol. 2, 2002, pp. 270-285.

[Delin05] K. A. Delin, “Sensor Webs in the Wild”, Wireless Sensor Networks: A Systems Perspective, Artech House, 2005

[Delin05] K. A. Delin, S. P. Jackson, D. W. Johnson, S.C. Burleigh, R. R. Woodrow, J. M. McAuley, J. M. Dohm, F. Ip, T. P.A. Ferré, D. F. Rucker i V. R. Baker, “Environmental Studies with the Sensor Web: Principles i Practice”, Sensors, Vol. 5, 2005, pp. 103-117.

[Delin99] K. Delin, S. Jackson, R. Some, “Sensor Webs”, NASA Technical Briefs Vol 23. 1999.

[Dennis13] A. K. Dennis, “Raspberry Pi Home Automation with Arduino”, Packt Publishing, USA, 2013.

[Deshpie09] A. Deshpie, L. Getoor i P. Sen, “Graphical models for uncertain data”, Managing i Mining Uncertain Data, C. C. Aggarwal (eds.), Springer, 2009, pp. 77-112.

[Di07] L. Di, “Geospatial Sensor Web i Self-adaptive Earth Predictive Systems (SEPS), Proceedings of the Earth Science Technology Office (ESTO)/Advanced Information System Technology (AIST) Sensor Web Principal Investigator (PI) Meeting”, San Diego, USA, 2007

[Di07] L. Di, “Geospatial Sensor Web i Self-adaptive Earth Predictive Systems (SEPS)”, Proceedings of the Earth Science Technology Office (ESTO)/Advanced Information System Technology (AIST) Sensor Web Principal Investigator (PI) Meeting”, San Diego, USA, 2007

[Dickinson14] D. P. Dickinson, “So Many Wireless Technologies …Which Is the Right One for My Application?”, Phoenix Contact, 2014.

[Doumit02] S. S. Doumit i D. P. Agrawal, “Self Organizing i Energy Efficient Network of Sensors”, IEEE, 2002, pp 1-6.

[Draw2D] Draw2D, online: http://www.eclipse.org/gef/draw2d/ [Dostupno: 19.04.2015]

[Eclipse] Eclipse, online: https://eclipse.org/ [Dostupno: 07.04.2015].

[EclipseEMF] Eclipse Modeling Framework, online: https://eclipse.org/modeling/emf [Dostupno: 07.04.2015].

[EclipseEMP] Eclipse Modeling Project, online:https://eclipse.org/modeling [Dostupno: 07.04.2015].

[ECoreTools] ECoreTools – Graphical Modeling for Eclipse, online: https://www.eclipse.org/ecoretools/ [Dostupno: 07.11.2015]

[Eisenhauer10] M. Eisenhauer, P. Rosengren i P. Antolin, “HYDRA: A Development Platform for Integrating Wireless Devices i Sensors into Ambient Intelligence Systems”, The Internet of Things, D. Giusto et al. (eds.), Springer, 2010, pp. 367-373.

[Fairgrieve09] S. M. Fairgrieve, J. A. Makuch i S. R. Falke, “PULSENetTM: An Implementation of Sensor Web Stiards”, Proceedings of International Symposium on Collaborative Technologies i Systems, 2009, pp. 64–75.

[Fantana13] N. L. Fantana, T.Riedel, J. Schlick, S. Ferber, J. Hupp, S. Miles, F. Michahelles, i S. Svensson, “IoT Applications − Value Creation for Industry”, Internet of Things: Converging Technologies for Smart Environments i Integrated Ecosystem, River Publishers, 2013

[Farooq11] M. O. Farooq i T. Kunz, “Operating Systems for Wireless Sensor Networks: A Survey”, Sensors, 2011, pp. 5900-5930.

[Feiler06] P.Feiler, R. P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, L. Northrop, D. Schmidt, K. Sullivan i K. Wallnau, “Ultra-Large-Scale Systems - The Software Challenge of the Future”, Software Engineering Institute, Carnegie Mellon University, 2006.

[Fielding00] R. T. Fielding, “Architectural Styles i the Design of Network-based Software Architectures”, Dissertation on University of California, Irvine, 2000

[Filippelli12] F. Filippelli, S. Kollosche, M. Bauer, M. Gerhart, M. Boger, K. Thoms i J. Warmer, “Concepts for the model-driven generation of graphical editors in Eclipse by using the Graphiti framework”, 2012, online: http://spray.eclipselabs.org.codespot.com/files/SprayPaper.pdf [Dostupno: 20.04.2015]

[Fleisch10] E. Fleisch, “What is the Internet of Things? An Economic Perspective”, Business Processes & Applications, Auto-ID Labs

White Paper WP-BIZAPP-053, 2010

[Fleisch10] E. Fleisch, "What is the Internet of Things? An Economic Perspective", Business Processes & Applications, Auto-ID Labs White Paper WP-BIZAPP-053, 2010

[Fleisch14] E. Fleisch, M. Weinberger i F. Wortmann, "Business Models i the Internet of Things (Extended Abstract)", Interoperability i Open-Source Solutions for the Internet of Things, I. P. Žarko et al. (Eds.), Springer, 2014, pp. 6-10.

[Fowler10] M. Fowler, "Domain-Specific Languages", Addison Wasley, 2010

[Fröhlich08] A. A. Fröhlich i L. F. Wanner, "Operating System Support for Wireless Sensor Networks", Journal of Computer Science, Vol. 4, No. 4, 2008, pp. 272-281.

[Funk11] A. Funk, C. Busemann, C. Kuka, S. Boll i D. Nicklas, "Open Sensor Platforms: The Sensor Web Enablement Framework i Beyond", 6th Conference on Mobile i Ubiquitous Information Systems, 2011, pp. 39-52.

[Gay14] W. Gay, "Mastering the Raspberry Pi", Apress, 2014.

[Gay14a] W. Gay, "Raspberry Pi Hardware Reference", Apress, 2014.

[GEF] Graphical Editing Framework, online: https://eclipse.org/gef/ [Dostupno: 19.04.2015]

[Gershon94] N. D. Gershon, R. M. Friedhoff, J. Gass, R. Langridge, H.-P Meinzer i J. D. Pearlman, "Is visualization REALLY necessary? The role of visualization in science, engineering, i medicine", Paper presented at the International Conference on Computer Graphics i Interactive Techniques, New York, 1994, pp. 499-500.

[Ghosh11] D. Ghosh, "DSLs in Action", Manning, 2011

[Ghosh11a] D. Ghosh, "DSL for the Uninitiated", Communications of the ACM, Vol. 54, No. 7, 2011

[Gianni15] D. Gianni, A. D'Ambrogio i A. Tolk, "Modeling i Simulation-Based Systems Engineering Hibook", CRC Press Taylor & Francis Group, 2015

[Giordano14] A. Giordano i G. Spezzano, "Service-Oriented Middleware for the Cooperation of Smart Objects iWeb Services", Internet of Things Based on Smart Objects, G. Fortino i P. Trunfio (Eds.), 2014, pp. 49-68.

[GMF] Graphical Modeling Framework, online: http://eclipse.org/gmf-tooling/ [Dostupno: 19.04.2015]

[GMP] Graphical Modeling Project, online: http://eclipse.org/modeling/gmp/ [Dostupno: 19.04.2015]

[Gomaa96] H. Gomaa, D. Menascé, i L. Kerschberg, “A Software Architectural Design Method for Large-Scale Distributed Information Systems”, Journal of Distributed Systems Engineering, 1996.

[Goodwin13] S. Goodwin, “Smart Home Automation with Linux i Raspberry Pi”, 2nd Edition, Apress Media, 2013.

[Gouyette06] M. Gouyette, “How to create a FSM graphical editor with GMF?”, GMF graphical editor tutorial, 2006, online: http://www.kermeta.org/docs/fr.irisa.triskell.kermeta.samples.fsm.documentation/build/pdf.fop/KerMeta-Create-FSM-Graphical-Editor-With-GMF/KerMeta-Create-FSM-Graphical-Editor-With-GMF.pdf [Dostupno: 20.04.2015]

[Graphiti] Graphiti, online: https://eclipse.org/graphiti/ [Dostupno: 19.04.2015]

[Gronback09] R. C. Gronback, “Eclipse Modeling Project: A Domain-Specific Language Toolkit”, Addison-Wesley, 2009.

[Grønli13] T.-M. Grønli, G. Ghinea, i M. Younas, “A Lightweight Architecture for the Web-of-Things”, MobileWeb Information Systems, F. Daniel, G.A. Papadopoulos, P. Thiran (Eds.), Springer, 2013, pp. 248-259.

[Grzechca13] D. Grzechca, D. Komorowski i S. Pietraszek, “A Universal Wireless Device for Biomedical Signals Recording”, Pervasive & Mobile Sensing & Computting for Healthcare – SSMI2, S.C. Mukhopadhyay et al. (Eds.), Springer, 2013, pp. 157-174.

[Guana14] V. Guana i E. Stroulia, “ChainTracker, a Model-Transformation Trace Analysis Tool for Code-Generation Environments”, Theory i Practice of Model Transformations, D. D. Ruscio i D. Varró (eds.), 2014, pp. 146-153.

[Gubbi13] J. Gubbi, R. Buyya, S. Marusic i M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, i future directions”, Future Generation Computer Systems 29, 2013, pp.1645–1660

[Guinard09] D. Guinard i V. Trifa, “Towards the Web of Things: Web Mashups for Embedded Devices”, WWW (International World Wide Web Conferences), Enterprise Mashups i Lightweight Composition on the Web (MEM 2009) Workshop, 2009

[Guinard09a] A. Guinard, A. McGibney i D. Pesch, “A Wireless Sensor Network Design Tool to Support Building Energy Management”, Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, 2009, pp. 25-30.

[Guinard10] D. Guinard, V. Trifa i E. Wilde, “A Resource Oriented Architecture for the Web of Things.”, Internet of Things 2010 International Conference (IoT 2010), 2010

[Guinard11] D. Guinard, “A Web of Things Application Architecture – Integrating the Real-World into the Web (Ph.D.)”, ETH Zurich, 2011

[Guinard11a] D. Guinard, V.Trifa, F. Mattern i E. Wilde, “From the Internet of Things to the Web of Things: Resource Oriented Architecture i Best Practices”, Architecting the Internet of Things, D. Uckelmann et al. (Eds.), Springer, 2011, pp. 97–129.

[Gupta12] A. K. Gupta i S. S. Nair, “Environmental Legislation for Disaster Risk Management, Module-I. Environmental Knowledge for Disaster Risk Management Project”, National Institute of Disaster Management & Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH, New Delhi , pp. 68, 2012

[Gupta13] S. G. Gupta, M. M. Ghonge, P. D. Thakare i P. M. Jawihiya, “Open-Source Network Simulation Tools: An Overview”, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Vol. 2, No. 4, 2013, pp. 1629-1635.

[Gűrgen10] L. Gűrgen, C. Roncancio, C. Labbé i S. Honiden, “Data Management Solutions in Networked Sensing Systems”, WSN Technologies for the Information Explosion Era, T. Hara et al. (Eds.), Springer, 2010, pp. 111–137.

[Haller10] S. Haller, “The Things in the Internet of Things”, Internet of Things Conference, 2010

[Han12] Q. Han i J. Li, “An Authorization Management Approach in the Internet of Things”, Journal of Information & Computational Science, Vol. 9, No. 6, 2012, pp. 1705-1713.

[Hart06] J. K. Hart i K. Martinez, “Environmental Sensor Networks: A revolution in the earth system science?”, Earth-Science Reviews, Vol. 78, 2006, pp: 177–191.

[Hasić14] H. Hasić i V. Vujović, “Civil law protection of the elements comprising the “Internet of Things” from the perspective of the legal owner of the property in question”, Infoteh-Jahorina, Vol. 13, 2014, pp. 1005-1011.

[Haupt14] F. Haupt, D. Karastoyanova, F. Leymann i B. Schroth, “A model-driven approach for REST compliant services”, Proceedings of the IEEE International Conference on Web Services (ICWS 2014), 2014, pp. 129-136.

[He12] D. He, G. Mujica, J. Portilla i T. Riesgo, “Simulation Tool i Case Study for Planning Wireless Sensor Network”, IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society, 2012, pp. 6024-6028.

[Heer10] J. Heer, M. Bostock, V. Ogievetsky, “A Tour Through the Visualization Zoo”, Communications of the Acm, Vol. 53, No. 6, 2010, pp. 59–67.

[Heer12] J. Heer, B. Shneiderman, “Interactive Dynamics for Visual Analysis”, Communications of the Acm, Vol. 55, No. 4, 2012, pp. 45–55.

[Heidenreich11] F. Heidenreich, J. Johannes, S. Karol, M. Seifert, M. Thiele, C. Wende i C.Wilke, “Integrating OCL i Textual Modelling Languages”, Models in Software Engineering, J. Dingel i A. Solberg (eds.), Vol. 6627, 2011, pp. 349-363

[Herrington03] J. Herrington, “Code Generation in Action”, Manning, 2003.

[Herrm10] M. Herrmannsdoerfer, D. Ratiu i G. Wachsmuth, “Language Evolution in Practice: The History of GMF”, Software Language Engineering , M. Bri, D. Gašević i J. Gray (eds.), Springer, 2010, pp. 3-22.

[Herrm11] M. Herrmannsdoerfer, “GMF: A Model Migration Case for the Transformation Tool Contest”, Fifth Transformation Tool Contest (TTC 2011), V. Gorp, S. Mazanek i L. Rose (eds.), 2011, pp. 1-5.

[Hill05] J. Hill, R. Szewczyk, A. Woo, P. Levis, S. Madden, C. Whitehouse, J. Polastre, D. Gay, C. Sharp, M. Welsh, E. Brewer i D. Culler, “TinyOS: An Operating System for Sensor Networks”, Ambient Intelligence, W. Weber, J. M. Rabaey, i E.Aarts (Eds.), Springer, 2005 , pp. 115-148.

[Holger05] K. Holger i W. Ireas, “Protocols i Architectures for Wireles Sensor Networks”, Wiley, 2005

[Horan13] B. Horan, “Practical Raspberry Pi”, Apress, 2013.

[Hulshout07] A. Hulshout i J.-P. Tolvanen, “Modeling for full code generation”, Embedded Computing Design, 2007

[Hussain09] S. Hussain, S. Schaffner i D. Moseychuck, “Applications of Wireless Sensor Networks i RFID in a Smart Home Environment”, Proceedings of the 2009 Seventh Annual Communication Networks i Services Research Conference, 2009, pp 153-157.

[Hussmann04] H. Hussmann i S. Zschaler, “The Object Constraint Language for UML 2.0 – Overview i Assessment”, Upgrade, digital journal of CEPIS (Council of European Professional Informatics Societies), Vol. 5, No. 2, 2004.

[Ingelrest10] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach i M. Parlange, “SensorScope: Application-Specific Sensor Network for Environmental Monitoring”, ACM Transactions on Sensor Networks, Vol. 6, No. 2, 2010, pp.1-32.

[IoTIEC14] “Internet of Things: Wireless Sensor Networks”, International Electrotechnical Commission - White Paper, Geneva, Switzerli, 2014.

[Jardosh08] S. Jardosh i P. Ranjan, “A Survey: Topology Control for Wireless Sensor Networks”, ICSCN '08 - International Conference on

Signal Processing, Communications i Networking, 2008, pp 422-427.

[JaxRS] JAX-RS API, online: https://jax-rs-spec.java.net [Dostupno: 07.11.2015]

[Jersey] Jersey JAX-RS framework, online: https://jersey.java.net/ [Dostupno: 07.11.2015]

[Jolma15] A. Jolma, A.-M. Ventelä, M. Tarvainen i T. Kirkkala, "An Interactive Website for the River Eurajoki", Environmental Software Systems Infrastructures, Services i Applications, R. Denzer et al. (Eds.), Springer, 2015, pp. 81-90.

[Jörges13] S. Jörges, "Construction i Evolution of Code Generators, A Model-Driven i Service-Oriented Approach", Springer, 2013.

[Jörges13] S. Jörges, "Construction i Evolution of Code Generators - A Model-Driven i Service-Oriented Approach", Springer-Verlag Berlin Heidelberg, 2013, pp. 11-38.

[Juliot06] E. Juliot i S. Lacrampe, "MDE for large projects : today needs i tomorow stiards", Eclipse Modeling Symposium, 2006.

[Juliot10] E. Juliot i J. Benois, "Viewpoints creation using Obeo Designer or how to build Eclipse DSM without being an expert developer?", Obeo Designer Whitepaper, 2010, Online: http://www.obeo.fr

[Jurdak04] R. Jurdak, C. V. Lopes i P. Baldi, "A Framework for Modeling Sensor Networks", Proceedings of the Building Software for Pervasive Computing Workshop at OOPSLA, Vol. 4, 2004, pp. 1-5.

[Kalin09] M. Kalin, "Java Web Services: Up i Running", O'Reilly, 2009

[Kardoš10] M. Kardoš i M. Drozdová, "Analytical Method of CIM to PIM Transformation in Model Driven Architecture (MDA)", JIOS, Vol. 34, No. 1, 2010, pp: 89-99

[Katiyar10] V. Katiyar, N. Chi, N. Chauhan, "Recent advances i future trends in Wireless Sensor Networks", International journal of applied engineering research, Dindigul, Vol.1, No 3, 2010, pp 330-342

[Kelly00] S. Kelly i J.P. Tolvanen, "Visual domain-specific modelling: Benefits i experiences of using metaCASE tools", International Workshop on Model Engineering, ECOOP 2000, 2000

[Kelly04] S. Kelly, "Comparison of Eclipse EMF/GEF i MetaEdit+ for DSM", 19th annual ACM conference on object-oriented programming, systems, languages, i applications, 2004.

[Kelly08] S. Kelly i J.P. Tolvanen, "Domain-Specific Modeling: Enabling full code generation", Wiley-IEEE Computer Society Press, 2008.].

[Kelly09] S. Kelly i R. Pohjonen, “Worst Practices for Domain-Specific Modeling”, IEEE Software, Vol. 26, No. 4, 2009

[Kerschberg00] L. Kerschberg i D. Weishar, “Conceptual Models i Architectures for Advanced Information Systems”, Applied Intelligence, Vol. 13, 2000, pp. 149-164.

[Kerschberg90] L. Kerschberg, “Expert Database Systems: Knowledge/ Data Management Environments for Intelligent Information Systems”, Information Systems, Vol. 15, 1990, pp. 151-160

[Kerschberg96] L. Kerschberg, H. Gomaa, D. A. Menascé, i J. P. Yoon, “Data i Information Architectures for Large-Scale Distributed Data Intensive Information Systems”, Eighth IEEE International Conference on Scientific i Statistical Database Management, Stockholm, Sweden, 1996.

[Kleppe03] A. Kleppe, J. Warmer i W. Bast, “MDA Explained: The Model Driven Architectur: Practice i Promise”, Addison Wesley, 2003

[Kolovos10] D. S. Kolovos, L. M. Rose, S. Bin Abid, R. F. Paige, F. A.C Polack, i G. Botterweck, “Taming EMF i GMF Using Model Transformation”, Model Driven Engineering Languages i Systems, MODELS 2010, D. C. Petriu, N. Rouquette i O. Haugen (eds.), 2010, pp. 211-225.

[Kolovos10] D. S. Kolovos, L. M. Rose, S. B. Abid, R. F. Paige, F. A.C Polack, i G. Botterweck, “Taming EMF i GMF Using Model Transformation”, Model Driven Engineering Languages i Systems, D.C. Petriu, N. Rouquette i Ø. Haugen (eds.), Springer, 2010, pp. 211-225.

[Kolovos15] D. S. Kolovos, A. García-Domínguez, L. M. Rose i R. F. Paige, “Eugenia: towards disciplined i automated development of GMF-based graphical model editors”, Software & Systems Modeling, Springer, 2015.

[Kosar10] T. Kosar, N. Oliveira, M. Mernik, V. J. M. Pereira, M. Črepinšek, D. D. Cruz i R.P. Henriques, “Comparing General-Purpose i Domain-Specific Languages: An Empirical Study”, Computer Science i Information Systems, Vol. 7, No. 2, 2010, pp. 247-264.

[Larman04] C. Larman, “Applying UML i Patterns: An Introduction to Object-Oriented Analysis i Design i Iterative Development, Third Edition”, Addison Wesley Professional, 2004.

[Ledo10] A. Ledo, N. Melo i F Ramalho, “Guidelines for Improving Model To Text Transformations”, 1 Brazilian Workshop On Model-Driven Development, 2010.

[Lee11] J. C. Lee i L. J. Henschen, “A Framework for a User Friendly Wireless Sensor Network Configuration System”, HCI International 2011 Posters' Extended Abstracts, C. Stephanidis (Ed.), Springer, 2011, pp. 57-61.

[Li09] Na Li, N. Zhang, Sajal K. Das i B. Thuraisingham, “Privacy preservation in wireless sensor networks: A state of theart survey”, Ad Hoc Networks 7, 2009, pp 1501–1514.

[López05] E. Egea-López, J. Vales-Alonso, A. S. Martínez-Sala, P. Pavón-Mariño i J. García-Haro, “Simulation Tools for Wireless Sensor Networks”, Proceedings of the International Symposium on Performance Evaluation of Computer i Telecommunication Systems - SPECTS 2005, 2005.

[López13] T. R. L´opez, C. R. Dom´ınguez, M. J. Rodr´ıguez, S. F. Ochoa i J. L. Garrido1, “Context-Aware Self-adaptations: From Requirements Specification to Code Generation”, Ubiquitous Computing i Ambient Intelligence - Context-Awareness i Context-Driven Interaction, G. Urzaiz, S. F. Ochoa, J. Bravo, L. L. Chen, J. Oliveira (eds.), Springer, 2013, pp. 46-53.

[Lorincz04] K. Lorincz, DJ Malan, TRF Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainli, M. Welsh i S. Moulton, “Sensor networks for emergency response: challenges i opportunities”, Pervasive computing, Vol.12, 2004, pp 16-22.

[Ludovici10] A. Ludovici i A. Calveras, “Integration of Wireless Sensor Networks in IP-based networks trough Web Services”, Proceedings of 4th Symposium of Ubiquitous Computing i Ambient Intelligence, Valencia, Spain, 2010

[Luoma04] J. Luoma, S. Kelly i J.-P. Tolvanen, “Defining Domain-Specific Modeling Languages: Collected Experiences”, Proceedings of the 4th OOPSLA Workshop on Domain-Specific Modeling (DSM 2004), 2004

[Luzeaux13] D. Luzeaux, “SoS i Large-Scale Complex Systems Architecting”, Complex Systems Design & Management, M. Aiguier et al. (Eds.), Springer, 2013, pp. 39-50.

[Ma13] Q. Ma, P. Kelsen, C. Glodt, “A generic model decomposition technique i its application to the Eclipse modeling framework”, Journal of Software i System Modeling, 2013, pp. 1-32.

[Madhuram07] R. C. Madhuram, “Dynamic Wizard Modeling with GMF, Using GMF to Build a Dynamic Wizard Framework i a Graphical Editor”, Eclipse Magazine, Vol. 6, 2007, pp. 20-28.

[Maier98] M. W. Maier, “Architecting Principles for Systems-of-Systems”, Systems Engineering, Vol. 1, No. 4, 1998, pp. 267-284

[Maissa12] Y. B. Maissa, F. Kordon, S. Mouline i Y. Thierry-Mieg, “Modeling i Analyzing Wireless Sensor Networks with VeriSensor”, International Workshop on Petri Nets i Software Engineering, 2012.

[Maissa13] Y. B. Maissa, F. Kordon, S. Mouline i Y. Thierry-Mieg, “Modeling i Analyzing Wireless Sensor Networks with VeriSensor: An IntegratedWorkflow”, Transactions on Petri Nets

i Other Models of Concurrency VIII, M. Koutny et al. (Eds.), Springer, 2013, pp. 24-47.

[Maksimović13] M. Maksimović, V. Vujović, "Uloga Internet baziranih bežičnih senzorskih mreža u zaštiti od požara", INFOTEH-JAHORINA, Vol. 12, 2013, pp. 629-634.

[Maksimović14] M. Maksimović, V. Vujović, N. Davidović, V. Milošević, B. Perišić, "Raspberry Pi as Internet of Things hardware: Performances i Constraints", 1st International Conference on Electrical, Electronic i Computing Engineering - IcETRAN 2014, 2014, pp. ELI1.6.1-6.

[Maksimović14a] M. Maksimović, V. Vujović, V. Milošević, B. Perišić, "Increasing the lifetime of hexagonal deployed Wireless Sensor Web Network", ICIST 2014, 2014, pp. 131-136.

[Maksimović14b] M. Maksimović, V. Vujović, V. Milošević, B. Perišić, "Evaluating the optimal heat detector deployment for fire detection", International Conference "Engineering & Telecommunication En&T 2014", 2014, pp. 129-134.

[Maksimović15] M. Maksimović, V. Vujović, B. Perišić, V. Milošević, "Developing a fuzzy logic based system for monitoring i early detection of residential fire based on thermistor sensors", Computer Science i Information Systems, Vol. 12, No. 1, 2015, pp. 63-89.

[Mallika07] A. Mallikarjuna, V. Reddy, P. Kumar, D. Janakiram i G. A. Kumar, "Operating Systems for Wireless Sensor Networks: A Survey Technical Report", 2007.

[Marković] G. B. Marković, M. L. Dukić, "Bežične senzorske mreže, I deo: Osnovna arhitektura, karakteristike i primene", Telekomunikacije - Stručno-naučni časopis republičke agencije za elektronske komunikacije, treći broj.

[Martinez11] A.P. Martinez, "Implementation i documentation of Sensor Web Enablement at KNMI", De Bilt, Stageverslag, 2011

[Mathew13] S. S. Mathew, Y. Atif, Q. Z. Sheng i Z. Maamar, "The Web of Things – Challenges i Enabling Technologies", Internet of Things & Inter-cooperative Computational Technologies for Collective Intelligence, N. Bessis et al. (Eds.), Springer, 2013, pp. 1-23.

[Mendez13] G. R. Mendez i S. C. Mukhopadhyay, "A Wi-Fi Based Smart Wireless Sensor Network for an Agricultural Environment", Wireless Sensor Networks & Ecolcological Monitoring, S.C. Mukhopadhyay & J.A. Jiang (eds.), 2013, pp. 247-268.

[Mernik05] M. Mernik, J. Heering i A. M. Sloane, "When i how to develop domain-specific languages", ACM Computing Surveys, Vol. 37, No. 4, 2005, pp. 316-344.

[Miller03] J. Miller i J. Mukerji, "MDA Guide Version 1.0.1", OMG Inc.,

2003.

[Miorii12] D. Miorii, S. Sicari, F. De Pellegrini i I. Chlamtac, “Internet of things: Vision, applications i research hallenges”, Ad Hoc Networks, Vol. 10, 2012, pp. 1497-1516.

[Mishra14] V. Mishra i S. Jangale, “Analysis i comparison of different network simulators”, International Journal of Application or Innovation in Engineering & Management (IJAIEM) Special Issue for International Technological Conference - 2014, 2014.

[Modica09] T. Modica, E. Biermann i C.Ermel, “An ECLIPSE Framework for Rapid Development of Rich-featured GEF Editors based on EMF Models”, Proceedings of Informatik 2009: Im Focus das Leben, Beiträge der 39. Jahrestagung der Gesellschaft füur Informatik e.V (GI), Vol. 154, 2009, pp. 2972-2985.

[Mohamed11] N. Mohamed, J. Al-Jaroodi. “A survey on service-oriented middleware for wireless sensor networks”, Service Oriented Computing i Applications, Vol. 5, No. 2, 2011, pp. 71-85.

[Monk14] S. Monk, “Raspberry Pi Cookbox”, O Reilly Media, CA, 2014.

[Moore04] B. Moore, D. Dean, A. Gerber, G. Wagenknecht i P.Vierheyden, “Eclipse Development using the Graphical Editing Framework i the Eclipse Modeling Framework”, IBM Corp. 2004.

[Mukhop13] S. C.Mukhopadhyay i J.-A. Jiang, “Wireless Sensor Networks i Ecological Monitoring”, Springer, 2013.

[Mukhop14] S. C. Mukhopadhyay i N. K. Suryadevara, “Internet of Things: Challenges i Opportunities”, Internet of Things, S. C. Mukhopadhyay (ed.), 2014, pp. 1-18.

[Muracevic10] Dz. Muracevic, F. Orucevic, H. Kurtagic, “Method of integration of geospatial data with business intelligent systems based on services oriented architecture”, International Journal of Computer Applications, Vol. 7, No. 7, 2010, pp. 35-39.

[Nack09] F. Nack, “An Overview on Wireless Sensor Networks”, Institute of computer science (ICS), 2009.

[Nicolai07] M. J. Nicolai., “SOA in Practice”, O'Reilly, 2007

[Nie12] P. Nie i J. K. Nurminen, “Integrate WSN to the Web of Things by Using XMPP”, Sensor Systems i Software, F. Martins, L. Lopes, i H. Paulino (Eds.), Springer, 2012, pp. 105-120.

[Nittel08] S. Nittel, A. Labrinidis i A. Stefanidis, “Introduction to Advances in Geosensor Networks”, GeoSensor Networks, S. Nittel, A. Labrinidis, i A. Stefanidis (Eds.), Springer, 2008, pp. 1-6.

[Novkov07] B. Novkov i M. Jovanović, “Opis i primena softverskih alata u programiranju bežičnih senzorskih mreža”, 15. Telekomunikacioni forum TELFOR 2007, 2007, pp. 103-106.

[ntc06] NTC thermistors for temperature measurement, Type B57164, March 2006, online: http://eecs.oregonstate.edu/education/docs/datasheets/10kThermistor.pdf [Dostupno: 07.11.2015]

[OCL03] Object Management Group (OMG), "Object Constraint Language Specification", Unified Modeling Language Specification, Version 1.5, 2003.

[OCL14] Object Management Group (OMG), "Object Constraint Language", Specification Version 2.4, 2014.

[Ogunyomi14] B. Ogunyomi, L. M. Rose, i D. S. Kolovos, "On the Use of Signatures for Source Incremental Model-to-text Transformation", Model-Driven Engineering Languages i Systems, J. Dingel, W. Schulte, I. Ramos, S. Abrahão i E. Insfran (eds.), 2014, pp. 84-98.

[Oliveira09] N. Oliveira, M. J. V. Pereira, P. R. Henriquez i D. D. Cruz, "Domain-Specific Languages: A Theoretical Survey", Related Technologies i Applications (CoRTA 2009), 2009, pp. 35-46

[Ortega12] J. A. Ortega Ramírez, L.M. Soria Morillo, A. Kröner, i J.A. Álvarez García, "Application of Things: A Step beyond Web of Things", Modern Advances in Intelligent Systems i Tools, W. Ding et al. (Eds.), Springer, 2012, pp. 37-46.

[Othmana12] M. F. Othmana i K. Shazali, "Wireless Sensor Network Applications: A Study in Environment Monitoring System", 2nd International Symposium on Robotics i Intelligent Sensors 2012 (IRIS 2012), Procedia Engineering. Vol. 41, Elsevier Ltd, 2012, pp. 1204-1210.

[Patel12] P. Patel, "Enabling High-Level Application Development in Internet of Things", Research Report - hal-00732094, V.1, 2012.

[PCF8591] PCF8591 8-bit A/D i D/A converter data sheet, Jan 2003, online: http://www.aurel32.net/elec/pcf8591.pdf [Dostupno: 07.11.2015]

[Pedro08] L. Pedro i M. Risoldi, "Metamodeling with Eclipse", SMV technical report series, 2008.

[Pedro08] L. Pedro i M. Risoldi, "Metamodeling with Eclipse", Department of Computer Science University of Geneva, 2008.

[Pelechano06] V. Pelechano, M. Albert, J. Munoz i C. Cetina, "Building Tools for Model Driven Development, Comparing Microsoft DSL Tools i Eclipse Modeling Plug-Ins", The Conference on Software Engineering i Databases, JISBD 2006, 2006.

[Pérez14] I. C. Pérez i A. M. Bernardos Barbolla, "Exploring Major Architectural Aspects of the Web of Things", Internet of Things, Smart Sensors, Measurement i Instrumentation 9, Springer, 2014.

[Pérez14] I. C. Pérez i A. M. Bernardos Barbolla, "Exploring Major Architectural Aspects of theWeb of Things", Internet of Things,

S. C. Mukhopadhyay (ed.), 2014, pp. 19-55.

[Perisic94] B. Perisic, "Projektovanje informacionih sistema u ambijentu otvorene arhitekture", Doktroska disertacija, Univerzitet u Novom Sadu, Fakultet tehničkih nauka Novi Sad, 1994.

[Pi4J] The Pi4J Project, online: http://pi4j.com/ [Dostupno: 07.11.2015]

[Pie14] P. Pie i A. R. Padwalkar, "Internet of Things –A Future of Internet: A Survey", International Journal of Advance Research in Computer Science i Management Studies, Vol. 2, Issue 2, 2014 pp. 354-361

[Porten12] O. van Porten, "Development i Evaluation of a Graphical Notation for Modelling Resource-Oriented Applications", Master thesis, University of Hagen, 2012.

[Porten12] O. van Porten, "Development i Evaluation of a Graphical Notation for Modelling Resource-Oriented Applications", Research Report 1/2012, FernUniversität, Hagen, 2012.

[Potter93] W. D. Potter, K. J. Kochut, J. A. Miller, V. P. Giham, i R. V. Polamraju, "The Evolution of the Knowledge/Data Model", International Journal of Expert Systems, Vol. 6, 1993, pp. 39-81.

[Potti12] P.K. Potti et al., "Comparing Performance of Web Service Interaction Styles: SOAP vs. REST", Proceedings of the Conference on Information Systems Applied Research, New Orleans Louisiana, USA, 2012

[Pramud13] F. Pramudianto, I. R. Indra i M. Jarke, "Model Driven Development for Internet of Things Application Prototyping", Proceedings of the International Conference On Software Engineering i Knowledge Engineering, 2013, pp. 703-708.

[Pramud14] F. Pramudianto, C. A. Kamienski, E. Souto, F. Borelli, L. L. Gomes, D. Sadok i M. Jarke, "IoTLink: An Internet of Things Prototyping Toolkit", Proceedings of the Ubiquitous Intelligence i Computing, 2014 IEEE 11th Intl Conf on i IEEE 11th Intl Conf on i Autonomic i Trusted Computing, i IEEE 14th Intl Conf on Scalable Computing i Communications i Its Associated Workshops (UTC-ATC-ScalCom) , 2014, pp. 1-9.

[Pramud15] F. Pramudianto, "Rapid Application Development in the Internet of Things: A Model-Based Approach", Dissertation, Faculty of mathematics, computer science i natural sciences, RWTH Aachen University Germany, 2015.

[Priyantha08] N. B. Priyantha, A. Kansal, M.Goraczko i F. Zhao, "Tiny Web Services: Design i Implementation of Interoperable i Evolvable Sensor Networks", In Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys), 2008, pp. 253-266.

[Puliafito10] A. Puliafito, A. Cucinotta, A. L. Minnolo i A. Zaia, "Making the Internet of Things a Reality: TheWhereX Solution", The Internet

of Things, D. Giusto et al. (eds.), Springer, 2010, pp. 99-108.

[Purvis99] M. Purvis, S. Cranefield i M. Nowostawski, “Environmental Software Systems: Environmental Information I Decision Support”, 3rd International Symposium on Environmental Software Systems (ISESS'99), R.Denzer, D. A. Swayne, M. Purvis i G. Schimak (Eds.), Springer, 1999, pp. 49-57.

[Qian09] K. Qian, D. d. Haring, L. Cao, “Embedded Software Development with C”, Springer, 2009.

[Rajago06] R. Rajagopalan, P.K. Varshney, “Data-Aggregation Techniques in Sensor Networks: A Survey”, IEEE Comm. Surveys & Tutorials, Vol.8, No.4, 2006.

[Ramas14] A. Ramaswamy, B. Monsuez i Adriana Tapus, “SafeRobots: A Model-Driven Approach for Designing Robotic Software Architectures”, Proceedings of IEEE International Conference on Collaboration Technologies i Systems (CTS), 2014.

[Ray09] A. Ray, “Planning i Analysis Tool for Large Scale Deployment of Wireless Sensor Network”, Proceedings of the International Journal of Next-Generation Networks (IJNGN), Vol.1, No.1, 2009, pp. 29-36

[Ray09] A. Ray, “Planning i Analysis Tool for Large Scale Deployment of Wireless Sensor Network”, International Journal of Next-Generation Networks (IJNGN), Vol.1, No.1, 2009, pp. 29-36

[Recktenwald13] G. Recktenwald, “Temperature Measurement with a Thermistor i an Arduino”, 2013, online: http://web.cecs.pdx.edu/~eas199/B/howto/thermistorArduino/thermistorArduino.pdf [Dostupno: 07.11.2015]

[Reed13] C. Reed, M. Botts, G. Percivall i J. Davidson, “OGC: Sensor Web Enablement: Overview I High Level Architecture”, Open Geospatial Consortium, 2013.

[Refsdal11] I. Refsdal, “Comparison of GMF i Graphiti based on experiences from the development of the PREDIQT tool”, Master thesis, University of Oslo, 2011.

[Richardson10] L. Richardson i S. Ruby, “RESTful Web Services”, O'Rielly Media, 2007; Jim Webber, Savas Parastatidis, Ian Robinson “REST in Practice”, O'Reilly Media, 2010

[Richardson13] M. Richardson i S. Wallace, “Getting started with Raspberry Pi”, O’Reilly, USA, 2013.

[Richters01] M. Richters i M. Gogolla, “OCL: Syntax, Semantics i Tools”, Advances in Object Modelling with OCL, LNCS, Springer, 2001, pp. 38-63.

[Richters98] M. Richters i M. Gogolla, “On Formalizing the UML Object Constraint Language OCL”, 17th International Conference in

Conceptual Modeling, LNCS, Springer, 1998, pp. 156-171.

[Ririan12] T. Ririanarivelo, P. Lagarde i V. Heurteaux, "Sensor Web Enablement Stiards for Groundwater Monitoring", Geospatial Free i Open Source Software in the 21st Century, E. Bocher i M. Neteler (eds.), Springer, 2012, pp. 141-156.

[Rivieres04] J. des Rivieres i J. Wiegi, "Eclipse: A platform for integrating development tools", IBM Systems Journal, Vol. 43, No. 2, 2004.

[Rodríguez13] R. Rodríguez-Echeverría, F. Macías, V. M. Pavón, J. M. Conejero i F. Sánchez-Figueroa, "Model-Driven Generation of a REST API from a Legacy Web Application", Current Trends inWeb Engineering - ICWE 2013 Workshops, Q.Z. Sheng i J. Kjeldskov (Eds.), Springer, 2013, pp. 133-147.

[Romer04] K. Romer, F. Mattern, "The Design Space of Wireless Sensor Networks", IEEE Wireless Communication, 2004.

[Rose12] L. M. Rose, N. Matragkas, D. S. Kolovos i R. F. Paige, "A Feature Model for Model-to-Text Transformation Languages", Modeling in Software Engineering (MISE), 2012, pp. 57-63.

[Rouached10] M. Rouached, S. Chaudhry, "A. Koubaa. Lowpans meet service-oriented architecture", JUSPN, Vol. 1, No.1, 2010, pp. 39-48.

[Rouached12] M. Rouached, S. Baccar i M. Abid, "RESTful Sensor Web Enablement Services for Wireless Sensor Networks", IEEE Eighth World Congress on Services, pp. 65-72, 2012.

[Rubel12] D. Rubel, J. Wren i E. Clayberg, "The Eclipse Graphical Editor Framework (GEF)", Addison-Wesley, 2012.

[Ruscio10] D. D. Ruscio, R. Lammel, i A. Pierantonio, "Automated Co-evolution of GMF Editor Models", Software Language Engineering, B. Malloy, S. Staab i M. Bri (eds.), Springer, 2010, pp. 143-162.

[Sadilek08] D. A. Sadilek, "Domain-Specific Languages for Wireless Sensor Networks", Doctoral symposium of the Modellierung conference © Gesellschaft für Informatik, 2008.

[Santucci08] G. Santucci i S. Lange, "Internet of Things in 2020: A Roadmap for the future", European Technology Platform On Smart Systems Integration (EPOSS), 2008.

[Schaeffer12] B. Schaeffer, B. Baranski, T.Foerster i Johannes Brauner, "A Service-Oriented Framework for Real-Time i Distributed Geoprocessing", Geospatial Free i Open Source Software in the 21st Century, E. Bocher i M. Neteler (Eds.), Springer, 2012, pp. 3-20.

[Schmidt13] M. Schmidt, "Raspberry Pi – A Quick Start Guide", The Pragmatic Bookshelf, 2013.

[Schor09] L. Schor, P. Sommer, R. Wattenhofer, "Towards a zero

configuration wireless sensor network architecture for smart buildings", In Procee, dings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings - BuildSys '09, 2009, pp. 31-36.

[Schreib14] V. Schreibmann, "Design i Implementation of a Model-Driven Approach for Restful APIs", Fifth IEEE Germany Students Conference 2014 Passau, 2014.

[Schreier11] S. Schreier, "Modeling RESTful applications", Proceedings of the Second International Workshop on RESTful Design, 2011, pp. 15-21.

[Schreier12] S.Schreier, "Model-Driven Development of Resource-Oriented Applications", Service-Oriented Computing – ICSOC2011 Workshops, G. Pallis et al. (Eds.), Springer, 2012, pp. 201–206.

[Seehusen11] F. Seehusen i K. Stølen, "An Evaluation of the Graphical Modeling Framework (GMF) Based on the Development of the CORAS Tool", Theory i Practice of Model Transformations, J. Cabot i E. Visser (eds.), Springer, 2011, pp. 152-166.

[Seligman93] L. Seligman i L. Kerschberg, "An Active Database Approach to Consistency Management in Heterogeneous Data- i Knowledge-based Systems", International Journal of Cooperative i Intelligent Systems, Vol.2, 1993.

[Seo14] C. Y. Seo, H. S. Son i R. Y. C. Kim, "Model Transformation Rule for generating Automatic Database Schema of Business Process Framework", International Journal of Software Engineering i Its Applications, Vol. 8, No. 3, 2014, pp. 47-54.

[Shani10] U. Shani i A. Sela, "Integrating Domain-Specific Programming into Software Design", 2010 IEEE International Conference on Software Science, Technology & Engineering, 2010, pp. 1-6.

[Sharma10] K. Sharma i M.K. Ghose, "Wireless Sensor Networks: An Overview on its Security Threats", Mobile Ad-hoc Networks, 2010, pp: 42-45.

[Sharma14] M. Sharma, A. Kaushik i M. Shekhar, "Network Simulators for Next Generation Networks: an overview", International Journal of Mobile Network Communications & Telematics (IJMNCT), Vol. 4, No. 4, 2014, pp. 39-51.

[Shatalin06] A. Shatalin i A. Tikhomirov, "Graphical Modeling Framework Architecture Overview", Eclipse Modeling Symposium, 2006.

[Siraj12] S. Siraj, A. K. Gupta i R.-Badgujar, "Network Simulation Tools Survey", International Journal of Advanced Research in Computer i Communication Engineering, Vol. 1, No. 4, 2012, pp. 201-210.

[Sirius] Sirius, online: https://eclipse.org/sirius/ [Dostupno: 19.04.2015]

[Sohraby07] K. Sohraby, D. Minoli i T. Ynati, “Wireless Sensor Networks: Technology, Protocols, i Applications”, John Wiley & Sons, 2007.

[Son13] H. S. Son, W. Y. Kim i R. Y. C. Kim, “MOF based Code Generation Method for Iroid Platform”, International Journal of Software Engineering i Its Applications, Vol. 7, No. 3, 2013, pp. 415-426.

[Song14] Z. Song, M. T. Lazarescu, R. Tomasi, L. Lavagno i M. A. Spirito, “High-Level Internet of Things Applications Development Using Wireless Sensor Networks”, Internet of Things, S. C. Mukhopadhyay (ed.), 2014, pp. 75-110.

[Spray] Spray, online: https://code.google.com/a/eclipselabs.org/p/spray/ [Dostupno: 19.04.2015]

[Stair12] R. Stair i G.Reynolds., “Fundamentals of Information Systems”, Course Technology, 2012

[Stefani10] F. De Stefani, P. Gamba, E. Goldoni, A.Savioli, D. Silvestri, F. Toffalini, “A RESTful database for pervasive environmental wireless sensor networks”, In Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems Workshops - ICDCSW ’10, 2010, pp. 206-212.

[Steinberg08] D. Steinberg, F. Budinsky, M. Paternostro i E. Merks, “EMF: Eclipse Modeling Framework, Second Edition”, Addison-Wesley Proffesional, 2008.

[Steinhart68] J. S. Steinhart, S. R. Hart, “Calibration curves for thermistors”, Deep Sea Research i Oceanographic, Vo. 15, Issue 4, August 1968, pp. 497–503.

[Stocker15] M.Stocker, O. Kauhanen, M. Hiirsalmi, J. Saarela, P. Rossi, M. Rönkkö, H. Hytönen, V. Kotovirta i M. Kolehmainen, “A Software System for the Discovery of Situations Involving Drivers in Storms”, Environmental Software Systems Infrastructures, Services i Applications, R. Denzer et al. (Eds.), Springer, 2015, pp. 226-234.

[Stojčev05] M. K. Stojčev, “Računarske mreže i prenos podataka”, Elektronski faklutet, Niš, 2005.

[Striell02] C. Striell, A. Striell, N. Mayow, E. Leskinen i E. Needham, “Monitoring the living environment: shortened version of the working group report”, Carita Striell, 2002.

[Stritzke13] C. Stritzke i S. Lehrig, “Why i How We Should Use Graphiti to Implement PCM Editors”, Proceedings of the Symposium on Software Performance, S. Becker, W. Hasselbring, A. van Hoorn i R. Reussner (eds.), Vol. 1083, 2013, pp. 1-10.

[Sundani11] H. Sundani, H. Li, V. K. Devabhaktuni, M. Alam i P. Bhattacharya, “Wireless Sensor Network Simulators A Survey i Comparisons”, International Journal Of Computer Networks

(IJCN), Vol. 2, No. 5, 2011, pp. 249-256.

[Taleghan07] M. A. Taleghan, A. Taherkordi, M. Sharifi i T.-H. Kim, “A Survey of System Software for Wireless Sensor Networks”, Future Generation Communication i Networking, Vol. 2, 2007, pp. 402-407.

[Tan10] L. Tan, Z. Yang i J. Xie, “OCL Constraints Automatic Generation for UML Class Diagram”, Software Engineering i Service Sciences (ICSESS), 2010, pp. 392-395.

[Tanenbaum11] A. S. Tanenbaum i D. J. Wetherall, “Computer Networks 5th ed.”, Prentice Hall, 2011.

[Tidwell01] D. Tidwell, J. Snell i P. Kulchenko, “Programming Web services with SOAP”, O'Reilly, 2001

[Tomcat] Apache Tomcat server, online: http://tomcat.apache.org/ [Dostupno: 07.11.2015]

[Townsend05] C. Townsend i S. Arms, “Wireless Sensor Networks: Principles i Applications”, Sensor Technology, J.S. Wilson (eds.), 2005 , pp:575-589.

[Tridium09] "The Web of Things”, White Paper, Tridium Inc., 2009

[Trifa11] V. Trifa, “Building Blocks for a Participatory Web of Things: Devices, Infrastructures, i Programming Frameworks (Ph.D.)”, ETH Zurich, 2011.

[Ukil11] A. Ukil, J. Sen i S. Koilakonda, “Embedded Security for Internet of Things”, Emerging Trends i Applications in Computer Science (NCETACS), 2011, pp. 1-6.

[Ulrich10] F. Ulrich, “Outline of a method for designing Domain-Specific Modelling languages”, ICB-Research Report No. 42, 2010.

[Upton12] E. Upton, G. Halfacree, “Raspberry Pi User Guide”, Wiley, 2012.

[Valverde09] F. Valverde i O.Pastor, “Dealing with REST Services in Model-driven Web Engineering Methods”, V Jornadas Científico-Técnicas en Servicios Web y SOA, JSWEB 2009, 2009.

[Vermesan13] O. Vermesan, P.Friess, P. Guillemin, H. Sundmaeker, M. Eisenhauer, K. Moessner, F. L. Gall, i P. Cousin, “Internet of Things Strategic Research i Innovation Agenda”, Internet of Things: Converging Technologies for Smart Environments i Integrated Ecosystem, River Publishers, 2013.

[Vermesan14] O. Vermesan i P. Friess, “Internet of Things – From Research i Innovation to Market Deployment”, River Publishers, 2014.

[Vermesan15] O. Vermesan i P. Friess, “Building the Hyperconnected Society: IoT Research i Innovation Value Chains, Ecosystems i Markets”, River Publishers, 2015.

[Völter06] M. Völter i B. Kolb, “Best Practices for Model-to-Text Transformations”, Modeling Symposium at Eclipse Summit, 2006.

[Völter06] M. Völter i B. Kolb, “Best Practices for Model-to-Text Transformations”, Eclipse Summit Europe, Modeling Symposium, Vol. 2006, 2006.

[Vujović13] V. Vujović, M. Maksimović, D. Kosmajac, V. Milošević, B. Perišić, “Web Integration of REST Enabled Wireless Sensor Networks for Fire Detection”, International conference on Applied Internet i Information Technologies AIIT , 2013, pp. 30-35.

[Vujović14] V. Vujović, M. Maksimović, B. Perišić, “Comparative analysis of DSM Graphical Editor frameworks: Graphiti vs. Sirius”, 23nd International Electrotechnical i Computer Science Conference ERK 2014, 2014, pp. 7-10.

[Vujović14a] V. Vujović, M. Maksimović, B. Perišić, “Development of DSM Graphical Editor for RESTful Sensor Web Networks Modeling”, Scientific Bulletin of the “Politehnica” University of Timisoara, Romania, Transactions on Automatic Control i Computer Science, Vol. 59(73), No. 2, 2014, pp. 131-140.

[Vujović14b] V. Vujović, M. Maksimović, B. Perišić, “Sirius: A Rapid Development of DSM Graphical Editor”, IEEE 18th International Conference on Intelligent Engineering Systems , 2014, pp. 233-238.

[Vujović14c] V. Vujović, M. Maksimović, “Raspberry Pi as a Wireless Sensor Node: Performances i Constraints”, The 37th International ICT Convention – MIPRO 2014, 2014, pp. 1247-1252.

[Vujović14d] V. Vujović, M. Maksimović, B. Perišić, V. Milošević, “A Graphical Editor for RESTful Sensor Web Networks Modeling”, 9th IEEE International Symposium on Applied Computational Intelligence i Informatics, 2014, pp. 61-66.

[Vujović15] V. Vujović, M.Maksimović, B. Perišić, G. Milošević, “A Proposition of Low Cost Sensor Web Implementation Based on GSM/GPRS Services”, 2015 IEEE 1st International Workshop on Consumer Electronics, 2015.

[Vujović15a] V. Vujović, M. Maksimović, “Data acquisition i analysis in educational research based on Internet of Things”, 11th international conference “Interactive Systems: Problems of Human-Computer Interactions”, 2015, pp. 57-62.

[Vujović15b] V. Vujović, M. Maksimović, “Raspberry Pi as a Sensor Web node for home automation”, Computers & Electrical Engineering, Vol. 44, 2015, pp. 153-171.

[Vujović15c] V. Vujović, M. Maksimović, D. Kosmajac i B. Perišić, “Resource: a connection between Internet of Things i Resource-

Oriented Architecture", European Conference on Smart Objects, Systems i Technologies - Smart SysTech 2015, 2015, pp. 1-7.

[Vujović15d] V. Vujović, M. Maksimović, G. Balotić i P.Mlinarević, "Internet stvari – tehnički i ekonomski aspekti primjene", Infoteh-Jahorina, Vol. 14, 2015, pp. 658-663.

[Walter10] K. Walter, "Development of an early warning information infrastructure using spatial web services technology", International Journal of Digital Earth, Vol. 3, No. 4, 2010, pp. 384-394

[Wang06] Y. Wang at al., "A Survey of Security Issues in Sensor Networks: A Survey", IEEE Comm. Surveys & Tutorials, Vol. 8, No. 2, 2006.

[Warmer03] J. Warmer i A. Kleppe, "Object Constraint Language, The: Getting Your Models Ready for MDA, Second Edition", Addison Wesley, 2003.

[Warmer13] J. Warmer, K. Thoms i J. Reichert, "Spray – A quick way of creating Graphiti", 2013, online: https://spray.eclipselabs.org.codespot.com/files/SprayUserGuide-0.5.0.pdf [Dostupno: 20.04.2015]

[Warner13] T. L. Warner, "Hacking Raspberry Pi, Que Publishing", USA, 2013.

[Webber10] J. Webber, S. Parastatidis i I. Robinson "REST in Practice", O'Reilly Media, 2010

[Weber10] R. H. Weber i R. Weber, "Internet of Things: Legal Perspecives", Springer, 2010.

[Wegeler13] T.Wegeler, F. Gutzeit, A. Destailleur i B. Dock, "Evaluating the Benefits of Using Domain-Specific Modeling Languages - an Experience Report", 13th Workshop on Domain-Specific Modeling, SPLASH, 2013

[Wehrle10] K. Wehrle, M. Günes i J. Gross, "Modeling i Tools for Network Simulation", Springer, 2010.

[Weingärt09] E.Weingärtner, H. vom Lehn i K. Wehrle, "A performance comparison of recent network simulators", IEEE International Conference on Communications ICC'09, 2009. ICC'09, 2009, pp. 1-5.

[Wiederhold92] G. Wiederhold, "The Roles of Artificial Intelligence in Information Systems", Journal of Intelligent Information Systems, Vol. 1, 1992, pp. 35-56.

[Wood06] A. Wood, G. Virone, T. Doan, Q. Cao, L. Selvao, Y. Wu, L. Fang, Z. He, S. Lin i J. Stankovic, "ALARMNET: Wireless sensor networks for assistedliving i residential monitoring", SiteSeerX, 2006, pp 1-14.

[Wortmann15] F. Wortmann i K. Flüchter, “Internet of Things - Technology i Value Added”, Business & Information Systems Engineering, Vol. 57, No. 3, 2015, pp. 221-224.

[Xmlpsd] "XML Professional Skills Development”, Application Developers Training Company i AppDev Products Company

[Yang14] G. Z. Yang (eds.), “Body Sensor Networks, second edition”, 2014.

[Yang14] S.-H. Yang, “Wireless Sensor Networks”, Springer, 2014.

[Yazar09] D. Yazar, “RESTful Wireless Sensor Networks Domain”, Master Thesis, Uppsala Universitet, 2009

[Yazar09] D. Yazar, A. Dunkels, “Efficient application integration in ip-based sensor networks”, In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings - BuildSys ’09, 2009, pp. 43-48.

[Yick08] J. Yick, B. Mukherjee i D. Ghosal, “Wireless sensor network survey”, Computer Networks, Vol. 52, 2008, pp: 2292–2330

[Zeng11] D. Zeng, S. Guo i Z. Cheng, “The Web of Things: A Survey”, Journal of Communications, Vol. 6, No. 6, 2011

[Zest] Zest, online: http://eclipse.org/gef/zest/ [Dostupno: 19.04.2015]

[Zhang10] G. Zhang i J. Tian, “An Extended Role Based Access Control Model for the Internet of Things”, International Conference on Information, Networking i Automation (ICINA), 2010, pp. 319-323.

Biografija

Vladimir Vujović rođen je 31.01.1984. godine u Sarajevu, opština Centar. Osnovno obrazovanje je stekao u Kiseljaku (Bosna i Hercegovina) 1999. godine, a iste godine upisao je srednju strukovnu školu u Kreševu (Bosna i Hercegovina) te 2003. godine, po njenom završetku, dobio zvanje "*Tehničara za računarstvo*".

Elektrotehnički fakultet Univerziteta u Istočnom Sarajevu upisuje nakon završetka srednje škole. Studirajući po starom nastavnom planu i programu, isti završava 2010. godine na odsjeku za *Računarstvo i informatiku*, sa prosjekom 8.7.

Nakon završetka studija, 16.02.2010. godine, stupa u radni odnos na Elektrotehničkom fakultetu kao saradnik u nastavi, i u ovom radnom odnosu ostaje do 24.06.2010. godine, nakon čega dobija izbor u zvanje asistenta na odsjeku za *Računarstvo i informatiku*, užu naučnu oblast "*Informacione nauke i bioinformatika*".

Od izbora u zvanje asistenta na Elektrotehničkom fakultetu Univerziteta u Istočnom Sarajevu, učestvuje u izvođenju nastave na III i IV godini osnovnog ciklusa smjera *Računarstvo i informatika*, te Master studiju. Predmeti na kojima je od zaposlenja do danas bio asistent su: Algoritmi i strukture podataka, Praktična nastava, Upravljanje projektima, Projekat 1, Projektovanje softvera, Projektovanje informacionih sistema, Projekat 2, Softversko inženjerstvo i tehnologije, Kriptografija i zaštita podataka, Specifikacija i modelovanje softvera.

Doktorske studije upisao je 2010. godine na Fakultetu Tehničkih nauka u Novom Sadu, na katedri za *Računarstvo i automatiku*.

Za vrijeme studiranja i rada na fakultetu objavio je 50 stručnih radova u/na međunarodnim i nacionalnim časopisima i skupovima, od čega 3 (tri) ranga M23, 6 (šest) ranga M24, 2 (dva) ranga M52, 28 (dvadesetosam) ranga M33 i 11 (jedanaest) ranga M53. Također je učestvovao i u realizaciji tempus projekata i projekata finansiranim od strane Ministarstva nauke i tehnologije Republike Srpske.

Sipendista Ministarstva nauke i tehnologije Republike Srpske, *Fonda dr Milan Jelić*, postaje 2015. godine, a iste dobija i nagradu na "Festivalu nauke 2015" za najuspješniji naučnoistraživački rad na trećem ciklusu studija u Republici Srpskoj.

Od svog zapošljavanja na Elektrotehničkom fakultetu Univerziteta u Istočnom Sarajevu, stalni je član organizacionog odbora, te sekretarijata programskog odbora, kao i aktivni recenzent međunarodne konferencije iz informacionih tehnologija "*INFOTEH-Jahorina*". Od 2013. godine postaje član Tehničkog komiteta za Informacione tehnologije Instituta za Standardizaciju Bosne i Hercegovine.

Oblasti interesovanja su mu softversko inženjerstvo, jezici specifični za domen, modelom upravljan razvoj softvera, servis orijentisane arhitekture, web tehnologije, Internet of Things, senzorske mreže i elektronika.

Od stranih jezika govori engleski jezik.