SUPPORTING MOBILE PLATFORMS WITH MODEL-DRIVEN METHODOLOGIES

Hassan Charaf

Dissertation submitted
for the degree of Doctor of the Hungarian Academy of Sciences

Budapest, 2015

Contents

List of Figures

List of Tables

Summary

Over the past twenty years I have been working on the challenges of applied computer science. I have experienced that software artifacts, methodologies, and services are continuously changing and provide an incredibly wide range of solutions for different areas of computer science. Analyzing and systematically addressing the different solutions of the area have led me to several findings that outline the development trends, and surface such innovative research and development goals. This can be realized within the ambience of universities, establish the relevance of research groups, and provide practical application fields. I have found that significant results in this area can be achieved only by building up a team of young, talented and ambitioned colleagues, who consider both scientific and application approaches equally important, since the software industry focuses on applications and services with high degree of innovation. Therefore, my contributions as a researcher were threefold: (i) conducting research, (ii) application development to fulfill the industrial requirements, and (iii) building up my research & development team. The research results introduced in this thesis have been developed together with several of my students. The number of the papers, degree theses, master and PhD theses indicates the size of the collaborative work. Under my supervision, eleven PhD theses have been submitted. I declare that some parts of the research results provided in this thesis have been joint work with my PhD students.

Software-driven services and applications are quite significant both in the different sectors of the industry and in the everyday lives of almost every person. The wide-spread use of mobile devices emphasized the importance of mobile application development. Recently, it has become one of the most focused areas in software industry. Alongside the increasing number of different target mobile platforms, the necessity for reliable and efficient applications is also growing rapidly. I am proud of the fact that I was among the first educators who introduced the mobile software development in a university curriculum in Europe. In this thesis, I introduce methods that support multi-platform mobile application development. The approach utilizes the findings of our research team, which has been carried out in the past twenty years. There are five software areas in which we conduct active research, namely (i) software modeling and model processing, (ii) mobile platforms, (iii) distributed systems and cloud-based services, (iv) data technologies, and (v) Internet of Things (IoT).

The goal of my research was to establish a coherent methodology that facilitates the reduction the time-to-market, and increase the quality of application development for various mobile and IoT platforms. The methodology applies modeling and model processing solutions along with automated tool support. In my thesis I have worked out the following results: (i) a methodology to support domain-specific modeling and model processing, (ii) methods to increase the efficiency of mobile platforms, and (iii) a model-driven methodology to support multiple mobile platforms.

These methods significantly decrease the number of software errors and the development time, therefore increase the efficiency of development processes for multiple platforms.

Összefoglaló

Az elmúlt két évtizedben az alkalmazott informatika kihívásaival foglalkoztam. Azt tapasztaltam, hogy az informatikai termékek, módszertanok és szolgáltatások folyamatosan változó köre szinte követhetetlenül gazdag. Másrészről a tematika tudományos igényességgel rendszerezett analízise olyan felismerésekhez vezet, amelyek mentén világossá válnak a fejlesztési trendek, körvonalazódnak az egyetemi környezetben elvégezhető, kutató egyetemi létünket megalapozó, gyakorlati alkalmazásokkal is jó esélyekkel kecsegtető, innovatív alkalmazott informatikai iskolát megalapozó kutatások és fejlesztések célkitűzései. Markáns előrelépést a területen, minden irány lefedését, egy a tudományos és alkalmazói szemléletet egyaránt prioritásnak tekintő, ambiciózus és kiemelkedő képességű fiatalok csapattá kovácsolása útján láttam biztosítottnak. A kutatói munkásságom így három pillérre alapoztam: kutatómunka, alkalmazások fejlesztése és a kutató-fejlesztő csoport építése. Jelen értekezésben megfogalmazott tudományos eredmények kidolgozásában számos hallgatómmal dolgoztam együtt, TDK dolgozatok, szakdolgozatok, diplomamunkák és PhD dolgozatok számszerűségükben is jelzik az elvégzett közös munkát. A vezetésemmel készült és már megvédett tizenegy PhD dolgozatot illetően nyilatkozom afelől, hogy a munkámban szereplő eredmények részben közösen kerültek kidolgozásra az általam vezetett PhD hallgatókkal.

Tudjuk, hogy a szoftvertermékek igen nagy jelentőséggel bírnak mind az ipar különböző területein, mind pedig mindennapi életünkben. A mobil készülékek széles körű elterjedése olyannyira hangsúlyossá tette a mobil alkalmazásfejlesztést, hogy az elmúlt időszakban az egyik legjelentősebb ágává vált a szoftver iparnak. Amellett, hogy a mobil platformok száma folyamatosan nő, a megbízható és hatékony mobil alkalmazások iránti kereslet is növekszik. Büszke vagyok a tényre, hogy az elsők között voltam, akik a mobil alkalmazásfejlesztést bevezették az egyetemi oktatásba Európában. Jelen értekezésben olyan módszereket és módszertanokat mutatok be, amelyek hatékonyan támogatják a multi-platform mobil alkalmazásfejlesztést. A megközelítés felhasználja a kutatócsoportunk elmúlt két évtizedben kidolgozott eredményeit. Ezen eredmények öt szoftverterületre koncentrálnak: (i) szoftvermodellezés és modellfeldolgozás, (ii) mobil platformok, (iii) elosztott rendszerek és felhő alapú megoldások, (iv) adatkezelési technikák, valamint (v) a tárgyak internete (IoT).

Mindezek fényében a kutatómunkám célja olyan egységes módszertan kidolgozása volt, melynek alkalmazásával a mobil eszközökön futó szoftverek kifejlesztése felgyorsítható valamint az előálló termékek minősége növelhető azáltal, hogy a fejlesztési feladatokhoz modellező és modellfeldolgozó módszereket alkalmaz. A modellezés során a célplatformtól független modellek segítségével a mobil alkalmazások egyes rétegeinek működése jól definiálható, majd ez alapján az egyes konkrét mobil környezetekhez forráskód részletek, adott esetekben futó alkalmazások generálhatók. Értekezésemben három tézist fogalmaztam meg: (i) módszertant dolgoztam ki a szakterület-specifikus modellezés és modellfeldolgozás támogatására, (ii) megoldásokat dolgoztam ki a mobil platformok hatékonyságának növelésére, és (iii) módszertant dolgoztam ki a több mobil platform együttes támogatására.

A módszerek jelentősen csökkentik a programozási hibákból adódó szoftverhibák számát, lerövidítik a fejlesztés idejét, ezáltal hatékonyabbá teszik a szoftverfejlesztési folyamatokat.

Acknowledgements

My research was carried out at the Budapest University of Technology and Economics, Department of Automation and Applied Informatics. The goal of my research could not have been achieved without the help and support of several people.

First, I am grateful to Prof. István Vajk guiding me on my PhD studies and giving me a big freedom to work and challenge. I am deeply indebted to him. I am also grateful to Prof. Róbert Tuschák offering me the opportunity to work on his Department. I would like to express my grateful attitude to Jenő Hetthéssy who started supporting me since I was a PhD student and my thanks to László Lengyel being my right hand at the department and helping me in just everything.

Right next comes the core of the Applied Informatics Team, which consists of talented and enthusiastic researchers. I would like to emphasize my thanks to Tihamér Levendovszky, Bertalan Forstner, Péter Ekler, Bence Kővári, István Albert, Imre Kelényi, Gergely Mezeu, Márk Asztalos, Tamás Mészáros, Péter Vingelmann and the other members of my team, my colleagues and to all my students.

My deepest gratitude goes to my closest collaborator Prof. Frank Fitzek (Aalborg, Dresden).

Last but not least, I would like to thank my family. My wife Sawsan and my children Kamel, Klára and Ádám for their continuous patience and support.

1 Introduction

The software industry, including mobile application development, has significantly improved in the recent decades. Mobile devices are currently core part of our society. We continuously use them to access and consume digital data [Gartner, 2010] [Vision, 2012]. With the increasing variety of target devices, the need for mobile applications continues to grow steadily. Since platform providers prefer to distinguish themselves from their competitors, the same application must be developed and verified for each platform independently before it can be published. Another challenge in mobile application development is the limited availability of energy. A wasteful application or a poorly designed algorithm may significantly increase the energy consumption and therefore shorten the time to use the devices, also referred to as operational time.

Based on these facts, we have identified several challenges, which define the following research areas:

- A method to efficiently design and develop mobile applications for various mobile platforms. The solution should assist in designing an application once and then generate executable applications from the same common models for different target mobile platforms using the language preferred on each platform.
- A solution to allow the generated applications to be able to utilize energy efficient frameworks to conservatively utilize battery power.
- Applying cloud computing-based techniques to increase the availability of the provided services as well as further support for energy efficient solutions. Computation-intensive tasks can be delegated into the cloud and the results can be utilized by the mobile applications.

In order to decrease the efforts in maintaining similar functionality to support many different mobile environments, the main objective of the research activities is cross-fertilizing different research areas: model-based software development, developing energy efficient mobile applications, and cloud services.

A model-driven framework provides a way to efficiently support the technology of multi-platform mobile application development. This framework should facilitate the modeling of different aspects of mobile applications in a platform-independent way. In doing so, we are integrating cloud services as well as providing automated solutions that generate verified, ready-to-use, energy efficient mobile applications for different platforms from the same models and to integrate these applications with the cloud-based backend services.

Considering the above challenges, there are multiple objectives that should be addressed:

1. **Supporting software modeling and model processing**
   These objectives include the design and implementation of the core concepts and services of an integrated modeling and model-processing framework. This framework will provide the modeling capabilities and model processing features.

The framework facilitates the development of domain-specific modeling languages, software modeling, design model processors, and model processing.
The framework hosts the domain-specific languages that support the platform-independent mobile application design. Furthermore, the framework provides the tools necessary for the design and execution of different mobile platform-related model processors.

2. **Designing mobile, platform-related, domain-specific languages**
   This objective includes the examination of the most relevant mobile platforms. The goal is to find out common properties and specifics of the mobile platforms. The measurements and the comprehensive research facilitate the effective support for the development of mobile platform-specific programming libraries: software development kits (SDKs) and application programming interfaces (APIs).
   This objective incorporates the design of mobile platform-independent domain-specific languages (DSLs). These DSLs help to define different aspects of mobile applications.

3. **Developing energy-efficient software patterns for mobile platforms**
   This objective covers the elaboration and collection of software patterns that target energy efficient resource usage. Software patterns related to the usage of network modules and communication are good examples. The patterns will be implemented in mobile, platform-specific libraries that can be used by the generated source code.

4. **Providing a common platform framework**
   In order to keep the model processors (in our case mostly source code generators) as simple as possible, the generated source code should use pre-prepared, high-level services. Every mobile platform has commonly applied source code patterns, such as executing the network communication in a separate thread, applying push notification.
   The objective includes the design and development of the common platform framework. This framework provides scalable runtime services for applications, which either wrap commonly used operations and/or wrap external services (e.g. cloud, Web, GSM, or other services available locally on the device). Furthermore, the area provides code generators for the common platform framework. These generators target desktop, mobile, and IoT (Internet of Things) applications.

5. **Integrating the results with cloud-based services.**
   This objective prepares the necessary cloud platform on top of which all the cloud services can be realized. Besides the framework tools, we also integrate external services into the same framework.

We believe that providing solutions for the above goals will significantly support the innovation in both the model-driven software development and mobile application development.

1.1 Technological Motivations

Mobile devices are significant part of our daily life, we actively use them in communication, administration, to reach and consume digital data and even more importantly, we arrange our social life around it. The diversity of mobile platforms and the device capabilities necessitates developing the same functionality for each relevant mobile platform. One further issue of the mobile devices is their limited availability of battery power, i.e. mobile applications should strain for energy-efficient solutions. The goal of the research is to work out a methodology that addresses both of these issues and to apply the results in real world application development.

The role of the mobile devices is determining the present and also the close future of the software industry. The diversity of mobile platforms and the mobile device capabilities requires providing automatic application generation for different mobile platforms. Mobile device owners would like to utilize the special capabilities of their own devices. Developing the same mobile application for all relevant mobile platforms requires relevant development effort. The main motivation of this research is to provide a model-driven solution and also address the issue of the energy efficiency, because of the limited resources of mobile devices. The cross platform software design is already a challenging task for currently seven billion mobile devices, but it will become even more challenging when we move in the IoT area in 2022 with more than 500 billion devices, where heterogeneity among the devices will even larger than it is now among the commercial mobile smart phones.

In the mobile industry, the convergence of various types of devices and technologies resulted in very complex and powerful handheld computers, which are on par with personal computers with respect to both performance and functionality, or sometimes even outperform them. Compared to mobile environments, the development of a personal computer reflects less possible target platforms and the web-based thin clients (considering Flash or Java plugins) are also more accepted than on mobile phones. When programming traditional desktop, server or web applications, it is usually enough to specialize yourself and/or your team for one specific platform. However, in the case of mobile development, the IT market is much more diverse. There are numerous device vendors with a variety of different hardware capabilities and application programming frameworks. Web-clients are also not as popular and less widely accepted on mobile devices. In addition to the different frameworks, vendors also prefer to distinguish themselves from their competitors with unique user interfaces and distinct application design. Consequently, no convergence can be expected in this area in the near future.

The Internet of Things (IoT) is transforming the surrounding everyday physical objects into an ecosystem of information that enriches our everyday life. The IoT represents the convergence of advances in miniaturization, wireless connectivity and increased data storage and is driven by various sensors. Sensors detect and measure changes in position,

temperature, light, and many others, furthermore, they are necessary to turn billions of objects into data-generating “things” that can report on their status, and often interact with their environment. Application and service development methods and frameworks are required to support the realization of solutions covering data collection, transmission, data processing, data analysis, reporting and advanced querying.

The focus of my work was to provide a model-driven solution, where mobile applications and IoT solutions are designed in domain-specific languages, and the executable artifacts are generated by domain-specific model processors. These model processors are available for different mobile platforms. Therefore, from the same models the application can be generated for different mobile platforms. The generated source code utilizes both the cloud computing and platform-specific energy-efficient programing libraries developed by our team. In this way the methodology supports to effectively realize energy efficient mobile applications for different mobile platforms.

In order to decrease efforts in maintaining the same functionality in many different mobile environments, the main objective of the research activities is the cross-fertilization of three different R&D areas: (i) model-based software development, (ii) mobile application development, and (iii) cloud-based services. These issues raise several open research questions. The following list introduces these challenges and their nature.

1. Examine and compare the most popular mobile platforms (Android, iOS, Windows Phone, and further platforms), explore connection points and commonalities of the user interface and business logic capabilities that may be handled in a uniform way. These common parts provide the base of further modeling and code generation methods. Moreover, formalisms and languages are expanded and capable of describing the common parts in a more precise way. These languages are able to integrate the use of cloud services into the business logic.
2. Perform laboratory measurements to analyze the different aspects of energy consumption used by all the most frequently executed functions on each mobile platform. Based on the results, design optimal software development patterns for using these functionalities more efficiently. The patterns should encompass energy efficient network communication, location based services, and multimedia features.
3. Simplify the complexity of the automated code generation. Suggest a method that supports the efficient application development and improves the energy efficiency of the resulted applications.
4. Support multiplatform application development.

To summarize, the objectives target one of the most pressing problems of mobile software development, which derives mainly from the diversity of mobile platforms. To address this issue, mobile platforms should be analyzed from different perspectives. A modeling language family is required that enables modeling for all relevant platforms and supports code generation. The result is a complete methodology that allows designing mobile applications, generates source code and in certain cases complete applications for each major mobile platform.

1.2 Structure of the Thesis

The Thesis has seven regular chapters and four appendices. It is organized in the following way:

- Chapter 1 provides the introduction, motivations, and the main objectives related to the research activities.
- Chapter 2 is devoted to illustrate the areas in which we have been actively researching: software modeling and model processing, mobile platforms, distributed systems, and cloud-based services, along with data technologies.
- Chapter 3 discusses the research results related to domain-specific modeling and model processing. These results allow the modeling of the different aspects of software applications and the efficient model processing that generates different platform-specific software artifacts and therefore, increases the efficiency and the quality of the software products.
- Chapter 4 introduces the methods that increase the efficiency of mobile platforms. These methods include the investigation of mobile peer-to-peer networks, providing energy efficient (green) mobile peer-to-peer solutions and network coding-based information sharing in mobile peer-to-peer networks.
- Chapter 5 provides the model-driven methodology for effectively supporting multiple mobile platforms. The chapter introduces two different solutions we contributed during the last years. These solutions are referred as the *First Wave* and *Second Wave* cross-platform solutions.
- Chapter 6 discusses the application fields of the scientific results and introduces several applications that utilize the research results. Furthermore, three long-term research projects are also introduced that also utilized several elements of the results and the cross-platform solutions.
- Chapter 7 concludes the Thesis by summarizing the main scientific results.
- Appendix A shows a sample for supporting multiple mobile platforms.
- Appendix B, Appendix C, and Appendix D, respectively, provide three comprehensive case studies, which illustrate the capabilities of the provided methodology. The third case study focuses on showing the efficiency of model-based development comparing to the traditional one.

2 Backgrounds

This section provides the state-of-the-art overview of the research areas that significantly contribute to the overall goal of the thesis, i.e. the methodology to simultaneously support multiple mobile platforms.

2.1 Software Modeling and Model-based Engineering

The two key points in the evolution of application development are always invariant: optimize the development process, i.e. time and effort/cost, and provide high quality artifacts. Model-driven approaches address both issues.

The growing dimension and complexity of software systems have made software modeling technologies a very efficient tool in application development. Within the modeling approaches, there exists a clear trend to move from universal modeling languages towards domain-specific solutions. Domain-specific languages (DSLs) [Fowler, 2010] [Kelly et al, 2008] are strictly limited to a domain, but this limitation also makes it possible to become much more efficient than is possible with a universal language. Using domain-specific artifacts and enforcing the domain rules automatically makes DSLs useful not only for software developers, but for domain experts as well. By extending DSLs to support product families, product line development [Beuche et al, 2006] [Czarnecki et al., 2000] [Donohoe, 2012] can also be supported. One of our goals is to increase the effectiveness of the current development methods through the application of DSLs.

Domain-specific languages can be textual (e.g., a script language), graphical (e.g., a workflow language) or mixed. Both kinds have their role, while complex algorithms are often easier to express by text, explaining the relation between different concepts and giving an overview of the components are easier to understand via visual representations. Usually, developers prefer the textual form, while non-technical domain experts find the graphical representation more suitable. Modern frameworks addressing domain-specific modeling should provide solutions for both kinds of DSLs [Eclipse] [Visual Studio].

A popular way to define the DSL structure is through metamodeling. The tools following this approach are called metamodeling environments [Fowler, 2010]. They usually provide a metamodeling language that targets the definition of other languages, and the language designers may build arbitrary languages based on that. Well-known metamodeling environments are e.g. Generic Modeling Environment (GME) [Lédeczi et al, 2001], AtomPM [Mannadiar, 2012], AToM^3 [de Lara et al, 2002], AGG [Taentzer, 2004], MetaEdit+ [Tolvanen, 2006], VIATRA [Varró et al, 2003]. Visual Modeling and Transformation System [VMTS] or the open-source Eclipse Modeling Framework (EMF/GMF) [Eclipse]. Another approach is to hard-code the elements of the language into the tools, as is followed by e.g. Ptolemy [Brooks et al, 2008] and MATLAB Simulink [MATLAB]. A completely different approach is to define the language syntax through a context-free grammar in an appropriate metasyntax notation, e.g. EBNF [EBNF, 1996]. This approach is implemented in various environments like ANTLR [ANTLR], Bison [Levine, 2009] or Xtext [Xtext]. This solution is

appropriate rather for textual tools, however, there are attempts to apply them during the definition of graphical languages as well [Xia et al, 2003].

Domain-specific models are rarely the final product of a modeling scenario. We can generate reports, document templates, or statistics from models. Moreover, the specialization makes it possible to create a framework containing the base knowledge of the domain and generate code from models built using this framework. The final products are then automatically generated from these high-level specifications with domain-specific code generators. Therefore, there is no need to create error-prone manual mappings from domain concepts to design concepts, or to programming language concepts. Moreover, we can create more than one code generator for a domain allowing us to generate applications for multiple platforms, e.g. for multiple mobile platforms (Figure 2-1). In this scenario, platform-independent domain models are translated by the platform-dependent code generators to the software products.

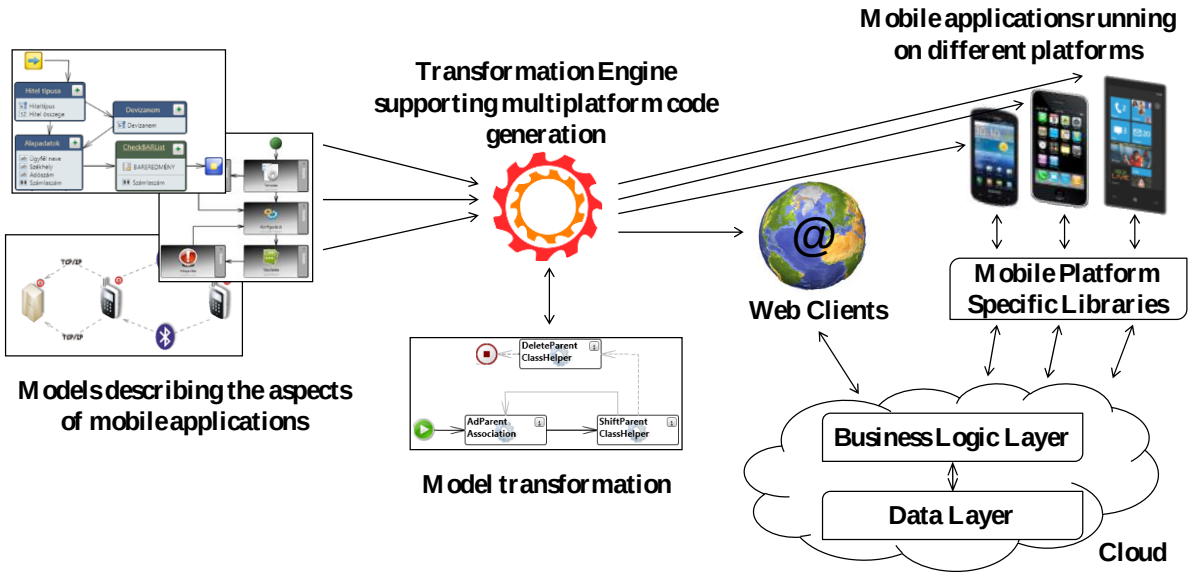

Figure 2-1 Generating applications for multiple platforms

An upcoming trend is to move both modeling and transformation into the cloud [Alajrami et al, 2014] [Vajk et al, 2012]. This grants accessibility/stability for models and processing power for the transformations. The cloud environment can also be used to host the domain-based functions and the generated applications can directly call the framework in the cloud, thus saving processing power and memory. This is especially important in the case of mobile or IoT devices.

Domain-specific modeling improves and accelerates the software development process [Kelly et al, 2008]. Due to its advantages, it now enjoys widespread use and is expected to grow even more popular in the near future [Fowler, 2010].

2.2 Model Processing

In general, three kinds of model processors are used: traversing processors, template-based processors, and graph transformations [Fowler, 2010]. Traversing processors use the model, visit the model elements and generate code from them (e.g. [Vajk et al, 2009]). Template-

based techniques extend this behavior by adding a template document as the base of code generation. Graph transformations [Ehrig et al, 1999] follow a whole different approach; they are basically highly configurable, smart search and replace operations. We specify a domain-specific pattern to search for and a replacement pattern for it. This technique is used in various tools e.g. in GReAT [GReAT] and VMTS [VMTS]. Graph transformations operate like production rules. Graph transformations are easy to use even for non-programmers due to the graphical representation and due to the fact that patterns are built from domain concepts.

In order to use the graph transformation in industrial environments, more scalable graph transformation techniques are required. Recent research directions include for example incremental overlapping rule application [Mészáros et al, 2010], defining declarative graph queries over EMF models, and executing them efficiently [Ujhelyi et al, 2015] and utilization of multi-core architectures [Imre et al, 2012].

Transformations appear in many different situations in a model-based development process. A few representative examples are as follows. (i) Refining the design to implementation; this is a basic case of platform-independent model to platform-specific model mapping [Kelly et al, 2008]. (ii) Aspect weaving; the integration of aspect models/code into functional artifacts is a transformation on the design [Gray et al, 2004]. (iii) Generative techniques; the implementation of generative paradigms is often realized with model transformation [Czarnecki et al., 2000]. (iv) Analysis and verification; analysis algorithms can be expressed as transformations on the design [Assmann, 1996].

We can conclude that transformations in general play an essential role in model-based development, thus, there is a need for model transformation solutions [Levendovszky et al, 2009a].

2.2.1 Model-Driven Architecture

Model-Driven Architecture (MDA) [OMG MDA] offers a standardized framework to separate the essential, platform-independent information from the platform-dependent constructs and assumptions. A complete MDA application consists of a definitive platform-independent model (PIM), one or more platform-specific models (PSM) including complete implementations, one on each platform that the application developer decides to support. The platform-independent artifacts are UML and different software models containing enough specification to generate the platform-dependent artifacts automatically by model compilers (Figure 2-2).

MDA development focuses first on the functionality and behavior of a distributed application or system, undistorted by characteristics of the technology platform or platforms on which it will be implemented. In this way, MDA separates implementation details from business functions. Thus, it is not necessary to repeat the process of defining a functionality of the application or a system behavior each time a new technology arises. With MDA, functionality and behavior are modeled once. Then, mapping from a PIM through a PSM to the supported MDA platforms is implemented by different model transformations, which ease the challenge of supporting new or different technologies and areas.

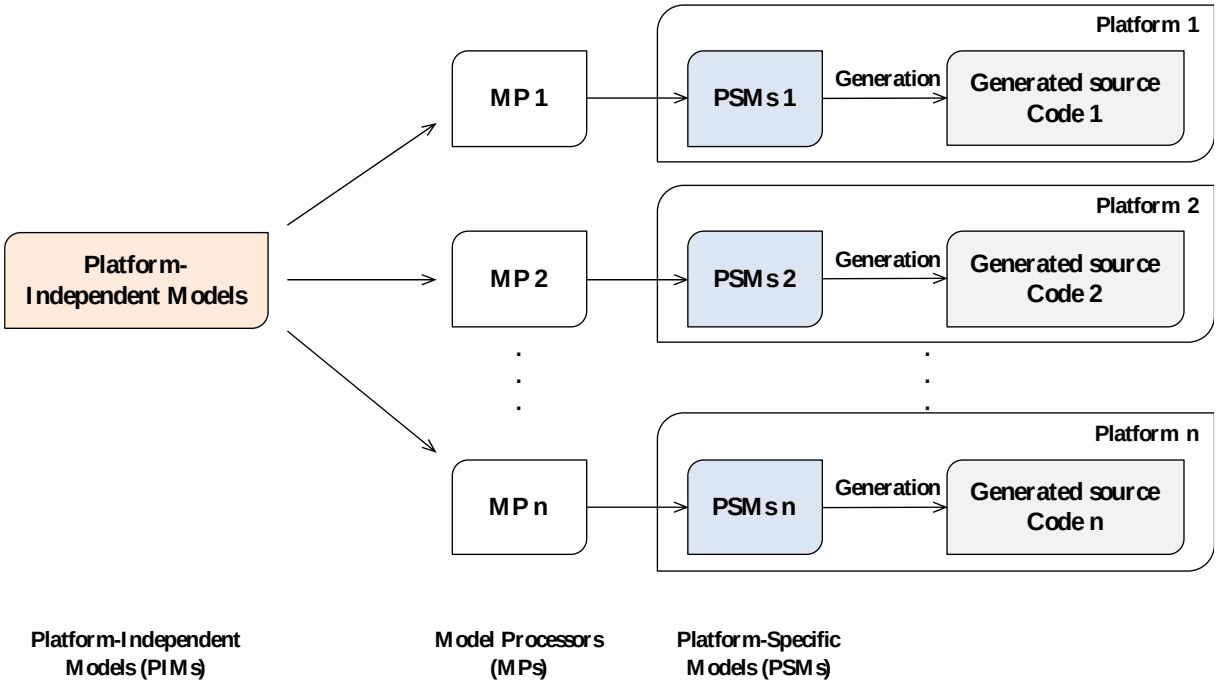

Figure 2-2 Overview of the Model-Driven Architecture

2.2.2 Model Integrated Computing

Model Integrated Computing (MIC) [Sztipanovits et al. 1997] is a model-based approach to software development. MIC facilitates the synthesis of application programs from models created using customized, domain-specific program synthesis environments. MIC focuses on models, supports the flexible creation of modeling environments, and helps following the changes of the models. At the same time it facilitates code generation and provides tool support for converting the created models into code artifacts. MIC was the first methodology, which requires metamodeling environments, model processors and provides a framework for them to cooperate and create computer-based systems in the practice [Sztipanovits, 1998].

The MIC development cycle (Figure 2-3) starts with the formal specification of a new application domain. The specification proceeds by identifying the domain concepts, together with their attributes and inter-relationships using metamodeling. After the domain has been defined, the metamodel of the domain is used to generate a domain-specific design environment. The domain-specific design environment can then be used to create domain-specific models. The next step to synthesize executable code, perform analysis or drive simulators. This is achieved by converting the models into another format such as executable code, input language of analysis tools, or configuration files for simulators. This mapping of the models to another useful form is achieved by model transformation. Model transformations are programs that convert models in a given domain into models of another domain. Note that the result of the transformation can be considered a model that conforms to a different metamodel.

MIC can be used as a methodology for domain-specific MDA where the focus is on developing the MDA process for specific domains. An implementation of domain-specific MDA should consist of a domain-specific modeling environment that allows users to describe

systems, using domain concepts. This environment is then used to develop domain-specific platform-independent models. These models represent the behavior and structure of the system with no implementation details. Such models then need to be converted to domain-specific platform-specific models. These models could either be based on the use of domain-specific libraries and frameworks or they do not have any domain-specific information.

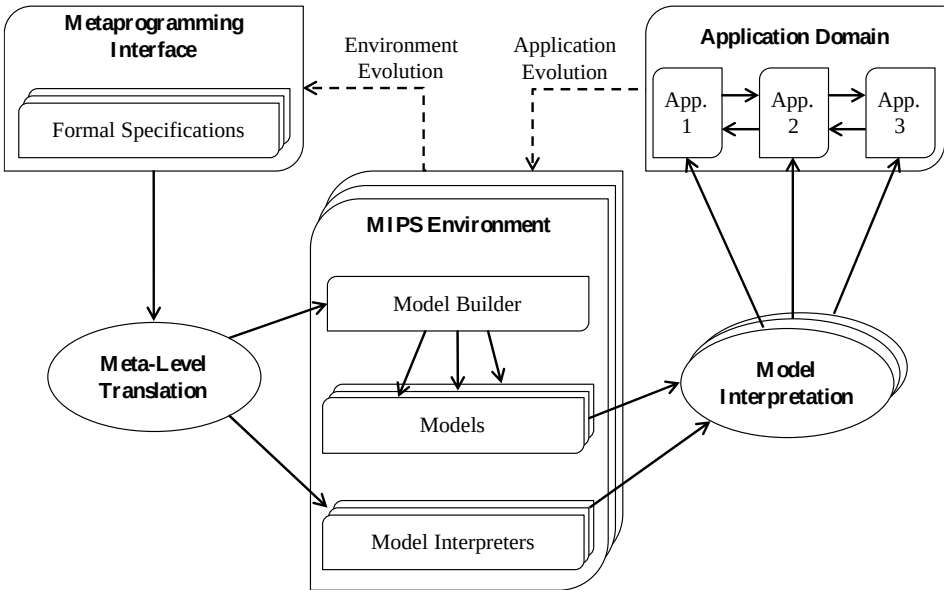

Figure 2-3 Overview of the Model Integrated Computing

2.2.3 Model Transformation in the Cloud

In order to use graph transformation in industrial environments, scalable graph transformation techniques are required. This leads to approaches to store [Vajk et al, 2012] and execute transformations in the cloud [Clasen et al, 2012]. Weaving of MDE and cloud technologies seems natural and may result in unprecedented performance boost in model-based tools [Alajrami et al, 2014]. This approach can be referred to as Modeling as a Service (MaaS) according to [Bruneliere et al, 2010]. The main advantage of in-cloud transformations is the possibility to parallelize their execution.

2.3 Mobile platforms

The mobile application industry has a significant role in the current software industry. In [Gartner, 2010] it was forecasted that worldwide mobile application store revenue would surpass $15 billion. According to another estimate [Vision, 2012], the global mobile applications market is expected to be worth $25 billion in 2015. This also reveals that software development is going to greatly cover the development of mobile applications.

Currently mobile devices are fully-featured computers that can effectively replace personal computers in regard to performance, functionality and network capabilities. However, the different mobile platforms challenge the mobile software industry. Therefore, the applications should typically be implemented for the three or four most popular platforms. In this case, only a small part can be reused from the code base (production and also test code) of one application in case the same application exists on another platform. Moreover, the same testing procedures should be performed in each particular case.

The most significant shortcoming of all portable electronics also applies to mobile devices: their operation time is limited by their battery capacity. In general, there are two ways to address this issue. The first of which is to create hardware components that consume less energy and design batteries with a higher energy density. The second is to write software that uses the resources of the system more efficiently [Pándi and Charaf, 2013] [Pándi and Charaf, 2015].

The main energy consuming components within mobile devices include CPU, display, I/O, and data transmission. Several researchers have proposed different system-level power models for mobile devices [Dong et al, 2011] [Xiao et al, 2010]. These models attempt to describe the instantaneous power draw of the entire device as a function of other measurable performance attributes such as CPU counters and display brightness level. Other studies have focused on measuring and modeling particular components of the device, such as the display and wireless communication [Huang et al, 2012a], or specific kinds of applications, such as peer-to-peer applications on mobile devices [Ekler et al, 2008a] [Kelényi et al, 2007].

These and other measurements as well as modeling efforts have led to important discoveries concerning how software-based techniques and optimizations can help in energy saving. For example, it has been shown that with wireless radio, including WLAN and 3G, the higher the bit rate, the more energy-efficient the communication is [Nurminen, 2010], which implies that smart scheduling is very important for energy efficiency. For instance, scheduling data traffic simultaneously with voice calls allows the data to be transmitted with little extra energy consumed.

2.4 Network Coding

Network Coding is a promising paradigm [Ahlswede et al, 2000], which provides several benefits in different networks and applications. Network coding is fundamentally different from the end-to-end approach of channel and source coding, because network coding makes possible coding at individual nodes in a communication network. With network coding packets are not treated as atomic entities as they can be combined and re-combined at any node in the network. This allows for a less restricted view on the flow of information in networks. This is significant help in realizing distribution systems for less structured networks, such as peer-to-peer networks.

The actual focus is on the Random Linear Network Coding approach to network coding, while deterministic coding approaches are neglected. The reason is that the primary interest is cooperative and highly mobile networks as for the IoT domain, which fit perfectly with the highly decentralized nature of random linear network coding. In particular, random linear network coding reduces the signaling overhead and increases robustness towards changing channel conditions in the network. Furthermore, it allows for the construction of simpler distribution systems. In practical applications of random linear network coding, the original data is divided into generations [Chou et al, 2003]. This facilitates that coding can be performed over data of any size. Furthermore, the performance of random linear network coding is independent of the data size. It can also be used to reduce the computational

complexity, which is often a significant problem in case of devices with low computational capabilities.

Some simplifications that can increase the coding throughput of random linear network coding are binary, systematic, and sparse variants [Li et al, 2011] [Heide et al, 2011]. Binary codes are in widespread use and can obtain a low code overhead. They are faster because operations in the binary field can be performed in parallel by all modern computers. However, both systematic and random sparse approaches suffer from the drawback that recoding becomes difficult or impractical.

Alternatively, the underlying code can be fundamentally modified or replaced to ensure a lower decoding complexity. A noteworthy suggestion is to use a convolutional code as the underlying code [Li et al, 2006a] as they have been used in communication systems.

Another direction in the search for improved trade-off between computational work and code overhead was suggested in [Silva et al, 2009]. Here the authors considered coding over several generations, called a random annex code [Li et al, 2010]. Each generation is extended to include symbols from other generations and thus when a generation is decoded these extra symbols are released. This reduces the problem of ensuring that all generations are decoded, and thus the overhead. At the same time it is less computational demanding as the decoding is performed in an inner and outer step. The approach is useful for file transfers.

Security and network coding is tightly coupled since the data that is transmitted is coded at the sender, and thus appears to be “random” to an adversary. However, this requires that the coding vectors that describe the operations performed during encoding remain secret to the adversary, and does not protect against the injection of malicious packets into the network. Homomorphic hashing introduced in [Krohn, 2004] allows nodes to check blocks on-the-fly in a system where the content is encoded at a source. This permits nodes in the network to check if a received symbol is malicious and thus offers a defense against misbehaving nodes that intentionally or unintentionally poisons the data. However, homomorphic signatures are known to be computationally demanding, which is particularly problematic in large complex topologies and for nodes with low computational capabilities. A more practical solution is proposed in [Gkantsidis et al, 2006] and is specifically designed for cooperative peer-to-peer like systems.

For the large and growing area of distributed storage system, network coding is a natural fit as it enables the possibility to code in between all nodes in the system [Dimakis et al, 2011]. One important application of coding in such systems is that of content reconstruction in case of node failure [Papailiopoulos et al, 2012]. In this case coding permits a scalable and efficient way of implementing redundancy, and the recoding possibility of network coding permits the system to be decentralized compared to solution build on traditional erasure codes. When network coding is utilized, care must be taken in order to avoid re-computation and re-distribution of one or more whole blocks [Zakerinasab et al, 2012].

User cooperation in ad hoc networks and user formed groups offers a long list of potential benefits [Fitzek et al, 2006]. However, such system though they are simple from a conceptual point of view entails many practical problems primarily due to node coordination and high

signaling overheads. Fortunately, network coding provides an elegant solution to exactly these problems [Chiu et al, 2006] [Pedersen et al, 2009].

In my work network coding is a handy tool to trade-off complexity of the CPU and the radio parts. Network coding is also a powerful tool to use distributed algorithms without overwhelming signaling, which in my work is especially interesting for the cloud solution space.

2.5 Distributed Systems and Cloud-Based Services

A distributed system is a software application ensuring that a collection of independent computers appears to its users as a single coherent system. In effect, a distributed system consists of a collection of autonomous computers, connected through a network and distribution middleware. The middleware enables the computers to coordinate their activities and to share the resources of the system. The users perceive the system as a single, integrated computing facility [Gregory et al, 2000] [Maurice et al, 2008] [Verdú et al, 2012].

There are many cases in which the use of a single computer would be possible, but the use of a distributed system is beneficial for practical reasons. For example, it may be more cost-efficient to obtain the desired level of performance by using a cluster of several computers, in comparison with a single, more powerful computer. A distributed system can be more reliable than a non-distributed system as well, as there is no single point of failure. Furthermore, a distributed system may be easier to expand.

The main goals of distributed systems are connecting resources and users, distribution transparency, resource sharing (the ability to use any hardware, software or data contained anywhere in the system), openness, concurrency, scalability and fault tolerance.

Openness requires being able to interact with services from other open systems, regardless of the underlying environment. Scalability denotes at least three components: the number of users and/or processes (size scalability), the maximum distance between nodes (geographical scalability) and the number of administrative domain (administrative scalability).

Components in distributed systems are executed in concurrent processes. These components access and update shared resources (e.g., variables, databases, device drivers). Distributed systems coordinate concurrent updates in order to ensure the integrity of the system.

Cloud computing has received significant attention recently, with regards to the above mentioned goals. Companies may store their data and perform their computations off-premise in a highly available and scalable environment where they only pay for the resources that they actually use. Compared to traditional infrastructures, that only use in-house resources, cloud computing has many advantages that have been transforming the computing solutions used at companies. Naturally, already existing on-premise resources can still be used as part of the infrastructure by connecting them to cloud services and creating a hybrid environment.

Cloud computing provides a cost-effective, highly-scalable solution for problems which include large amounts of data or computation. Both of these appear in model transformation, in which models can be arbitrarily large and the transformation may be complex. Thus, the weaving of model-driven engineering and cloud technologies seems natural and can result in

an unprecedented boost in the performance of model-based tools. Several architectural options arise for model-driven engineering tools: (i) the data can be stored in the cloud [Vajk et al, 2012] while transformations may be executed on the clients, (ii) on-premise data can be transformed in the cloud, (iii) both data and computation can be handled in the cloud, and finally, (iv) a hybrid solution where both on-premise and cloud resources are used for either storage or computation. All of these options excel in different scenarios. For instance, (i) can be considered if the transformations are simple, and thus, the communication overhead would be too great to run the transformations in the cloud. However, the data being available to all users in the cloud results in more easy collaboration. On the other hand (ii) this is relevant in transformations that can be easily distributed to several machines. Option (iii) provides complete Software as a Service (SaaS) solution for model transformations. And finally, the hybrid solution may be used in situations in which complex and simple transformations are mixed. In this case, complex tasks should be handled in the cloud, while simple ones should be executed locally to avoid the communication overhead. This is the Modeling as a Service (MaaS) approach [Bruneliere et al, 2010].

A completely different application area of cloud technologies regards integrating online, cloud-based services into the executable application during code generation. Here we have two possibilities: either the language and the model describing the system contains explicit references to the cloud services (what, when and how to do with a cloud service, e.g., login to the cloud), or the cloud layer is completely hidden by the code generators, and the language describes only the expected behavior (e.g. login), but not the means to realize it.

2.6 Data Technologies

In this work, we focus on application and service development. These applications and solutions are supported by various communication techniques, such as network coding, as well as by different data technologies and several data managing solutions. The data validation and schema definition are integrated parts of our work. Data technologies related solutions (business intelligence technologies, reporting, analytics, data mining [Fayyad et al, 2006], business performance management, and benchmarking [Rud, 2009]) are mainly utilized form existing results, but in certain cases, they are partly extended, therefore, they are involved into the research activities.

2.7 Internet of Things-Based Solutions

The Internet of Things (IoT) is the network of physical objects or "things" embedded with electronics, software, sensors, and connectivity to enable objects to exchange data with the manufacturer, operator and/or other connected devices [ITU, 2015]. The IoT is a network of physical *things* equipped with electronics, software, sensors and connectivity that provides greater value and better service by exchanging data with the manufacturer, operator and/or other connected devices. Each element of the network, i.e. each thing, is uniquely identifiable through its embedded computing system and is able to interoperate within the existing Internet infrastructure. [Zanella et al, 2014]

Things in the IoT can refer to a wide variety of devices such as biochips on farm animals, heart monitoring implants, production line sensors in factories, vehicles with built-in sensors,

or field operation devices that assist firefighters [Shixing et al, 2011]. These devices collect useful data with the help of various existing technologies, then autonomously flow the data between other devices and usually upload them into a data center environment for further processing.

Various sensors drive the IoT ecosystems and make the things active elements. Undoubtedly, the Internet of Things has reached and is about to dominate several domains. Top industries investing in sensors and utilizing data collected by them are energy, mining, transportation, vehicles, healthcare, production lines, power, technology, and financial services [Zanella et al, 2014].

Analysts expect that 50 to 100 billion devices will be connected to the Internet by 2020. According to a BBC Research report [BBC Research Report, 2015], the global market for sensors was valued at $79.5 billion in 2013 and is expected to increase to $86.3 billion in 2014, $95.3 billion in 2015, and to nearly $154.4 billion by 2020, a compound annual growth rate of 10.1% over the five-year period from 2015 through 2020.

The IoT is on the right way to be a major source of big data, contributing massive amounts of streamed information from billions of devices and sensors. Typical IoT applications that produce big data include vehicles and transportation, meteorology, experimental physics, astronomy, biology, and environmental science. For example, a Boeing jet generates 20 TBs of data every hour during a flight. Airliners have more than 300,000 sensors on board constantly generating data streams. Indeed, machine-to-machine (M2M) communication generates enormous amounts of Internet traffic. The availability of massive amounts of information streaming from billions of IoT devices inevitably and justly requires appropriate handling methods and techniques, furthermore, in certain cases, the sensitivity of the data brings up security and privacy concerns as well. [Farooq et al, 2015]

3 Domain-Specific Modeling and Model Processing

Software modeling and model processing related research work and results provide the basis of our R&D activities. A significant part of my team dedicated its time and efforts to this area.

3.1 Introduction

Model-driven software engineering is an actively researched field. The growing size and complexity of software systems made software modeling technologies essential in application development. Model-driven development approaches increase the development productivity and the quality of the produced software artifacts.

The most known and used software modeling language is the Unified Modeling Language (UML) [OMG UML] defined by the Object Management Group (OMG). UML is a graphic language to specify, visualize, build, and document the software systems artifacts. Moreover, it provides a standard way to write the models of a system, covering both conceptual elements, like business process and system functions, and the concrete aspects, like classes written in a specific programming language and software components. UML is a standard modeling language standardized in 1997 as UML 1.1, evolving till the 2.0 version, and nowadays there is a UML version 2.5. While main goal of UML 1.5 is the response to the classic needs of software industry, the UML 2.0 and 2.5 versions are greater evolution in visual modeling, where the new improvements allow to describe many of the new elements found in the software technology of today. The main contributions and goals of the UML are the followings:

- The primary value of UML models is in communication and understanding. Diagrams help communicate the understanding.
- UML models can help us understand either a software system or a business process.
- UML models are not a replacement for textual programming languages, but they are helpful assistants. For example, they are used for protocol verification.
- The UML's importance comes from its wide use and standardization within the object-oriented development community.

The Object Constraint Language [OMG OCL] is a formal language for analysis and design of software systems. It is a subset of the industry standard Unified Modeling Language [OMG UML] that allows software developers to write constraints and queries over object models. A constraint is a restriction on one or more values of an object-oriented model or system. There are four types of constraints: (i) An *invariant* is a constraint that states a condition that must always be met by all instances of the class, type, or interface. (ii) A *precondition* to an operation is a restriction that must be true at the moment that the operation is going to be executed. The obligations are specified by postconditions. (iii) A *postcondition* to an operation is a restriction that must be true at the moment that the operation has just ended its execution. (iv) A *guard* is a constraint that must be true before a state transition fires. Besides these applications, OCL can be used as a navigation language as well.

As constraints are restrictions on a model or system, they are always coupled to the items used in that model, usually a series of UML diagrams. In fact, the OCL can be used in any model as long as it supports the basic notions of class and instance, attributes, associations and operations.

Software modeling is a synonym for producing diagrams. Most models consist of a number of nodes and edges, pictures and some accompanying text. The information conveyed by such a model has a tendency to be incomplete and informal. A UML diagram, such as a class diagram, is typically not refined enough to provide all the relevant aspects of a specification. There is, among other things, a need to describe additional constraints about the objects in the model. Such constraints are often described in natural language. Practice has shown that this will always result in ambiguities. In order to write unambiguous constraints, formal languages have been developed. The disadvantage of traditional textual languages is that their use demands strong mathematical background, but difficult for the average business or system modeler to use. OCL is a formal language that remains easy to read and write.

Many of the flaws in the models are caused by the limitations of the diagram specification languages being used. A diagram simply cannot express the statements that should be part of a thorough specification. OCL expressions are unambiguous and make the model more precise and more detailed. These expressions can be checked by automated tools to ensure that they are correct and consistent with other elements of the model.

The combination of UML and OCL provides the best of both worlds to the software developer. A large number of different diagrams, together with expressions written in OCL, can be used to specify models. To obtain a complete model, both the diagrams and OCL expressions are necessary.

In contrast to the UML, a domain-specific language is strictly limited to a certain domain, but this limitation makes it possible to be more efficient than a universal language could be. Applying domain-specific models and enforcing of the domain rules makes domain-specific languages useful for software developers and also for domain experts. One of the most problematic phases in software development is the communication between the programmers and the domain experts. Domain experts do not understand any programming language nor do they want to learn these languages. However, they know the notations of their domain that can be textual or graphical symbols. Therefore, by using the domain notations, communication issues can be effectively reduced.

Models are usually not created just for documentation purposes [Schürr and Selic, 2009]. We apply models for the automatic generation of additional resources. These resources show a large versatility depending on the application domain. We may generate textual documentation, reports and presentations based on a model, but generation of source code and in certain cases complete applications is also possible. Models may also be transformed into other models. It may also be important to provide languages to define such model processing scenarios at a high abstraction level with the possibility to facilitate their verification.

Model processing programs are often applied in model-based software development in an automated way. Their formal analysis and the proof of their correctness are vital

requirements. Another important requirement of such programs is their understandability, which can be improved by using clear and high-level methods. Graph transformation-based model transformations [Ehrig et al, 2006] are special model processing programs that are based on the formal background of graph transformations (graph rewriting systems) [Sendall et al, 2003] [Mens et al, 2006].

Graphs provide an expressive and versatile data representation. Typically, nodes represent objects or concepts, and edges represent relationships among them. Hierarchical relationships can be depicted by nesting nodes into each other. Auxiliary information is expressed by adding attributes to nodes or edges. Given the widespread use of graphs as a data representation, it is natural that graph manipulations form the basis of many useful computations. Graph manipulations can be represented implicitly, embedded in a program that, among other things, constructs or modifies a graph. Alternatively, graph manipulations can be represented explicitly, using clearly-delineated graph rewriting rules that modify a host graph. The explicit use of graph-rewriting rules offers several advantages. Graph rewriting provides an abstract and high-level representation of a solution to a computational problem.

Graph rewriting has a strong mathematical background [Rozenberg, 1997] [Ehrig et al, 2006]. A graph rewriting rule is applied to a host graph to replace one subgraph by another. The atoms of the graph transformation are rewriting rules, each rewriting rule consists of a left-hand side graph (LHS) and a right-hand side graph (RHS). Applying a graph rewriting rule means finding an isomorphic occurrence (match) of LHS in the graph to which the rule is applied (host graph), and replacing this subgraph with RHS. Replacing means removing the elements that are in LHS but not in RHS, and gluing the elements that are in RHS but not in LHS.

Graph transformations have been recognized as a powerful technique for specifying complex transformations that can be used in various situations in a software development process [Assmann, 1996]. Many tasks in software development can be formulated using this approach.

3.2 Contributions

Model-based engineering approaches emphasize the use of models at all stages of system development. In model-based development, models are used to describe all artifacts of the system, i.e., interfaces, interactions, and properties of all the components that comprise the system. This chapter discusses our related results: a modeling and model transformation system, how to support domain-specific modeling activities, what ways are suggested to define visual concrete syntax for domain-specific languages, the methods we apply for processing software models with model transformation, how to validate model processing, and how to support dynamic behavior of domain-specific languages.

3.2.1 A Modeling and Model Transformation System

Our modeling and model transformation framework is the Visual Modeling and Transformation System (VMTS) [Angyal et al, 2009] [VMTS]. VMTS is a metamodeling environment that supports editing models according to their metamodels. Models are

formalized as directed, labeled graphs. VMTS uses a simplified class diagram for its root metamodel ("visual vocabulary").

Also, VMTS is a model transformation system, which transforms models using both template-based and graph rewriting techniques. Moreover, the tool facilitates the verification of the constraints specified in the transformation step during the model transformation process. VMTS is developed within my group at BME AUT since 2003.

The VMTS approach uses graphical notation for control flow (the execution sequence of the transformation rules): stereotyped UML activity diagram [OMG UML]. The control flow language can express a transformation as an ordered sequence of the transformation steps. Classical graph grammars apply any production that is feasible. This technique is appropriate for generating and matching languages but model-to-model transformations usually need to follow an algorithm that requires a stricter control over the execution sequence of the steps, with the additional benefit of making the implementation more efficient. The control flow language supports the following constructs: sequencing transformation steps, branching, hierarchical steps, parallel execution of the steps, and iteration [Lengyel et al, 2006].

The branching construct is required, because often, the transformation that we would like to apply depends on a condition. In the control flow language, conditions assigned to the decision elements can choose between the paths of optional numbers, based on the properties of the actual host model and the success of the last transformation rule.

VMTS transformation rules have two specific attributes: *Exhaustive* and *MultipleMatch*. Remember that applying a model transformation rule means finding a match of LHS in the host model and replacing this subgraph with RHS. An *exhaustive* transformation rule is executed repeatedly, as long as LHS of the rule could be matched to the host model. The *MultipleMatch* attribute of a rule allows that the matching process finds not only one but all occurrences of LHS in the host model, and the replacement is executed on all the found places.

In VMTS, LHS and RHS of the transformation rules are built from metamodel elements. This means that an instantiation of LHS must be found in the host graph instead of the isomorphic subgraph of LHS. Rewriting rules can be made more relevant to software engineering models if the metamodel-based specification of the transformations allows assigning constraints to the individual transformation rules. This technique facilitates a natural representation for multiplicities, multi-objects and assignments of constraints to the rules with a syntax close to the UML notation.

Figure 3-1 introduces the principles of the VMTS model transformation. The input model corresponds to its metamodel. The transformation is defined by the transformation rules. The application order of the rules is determined by the control flow model. Both transformation rules and the control flow are also models, based on their metamodels, defined in the VMTS. Finally, the output model corresponds to its metamodel. In certain cases the input and output metamodels are the same metamodel.

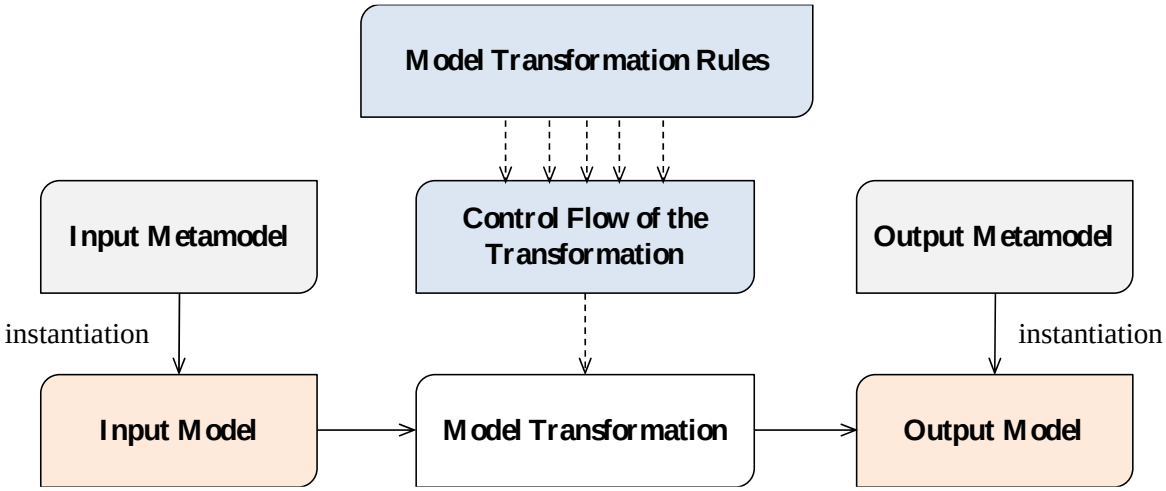

Figure 3-1 Principles of VMTS model transformations

Further details of the VMTS are discussed in Chapter 6.1.1.

3.2.2 Supporting Doman-Specific Modeling

Specifying systems on a higher abstraction level helps understand, develop and maintain software. A higher abstraction level can be achieved by software models and their model transformation. Model compilers provide a solution for automated source code generation from software models and mechanisms for software maintenance [Sztipanovits et al, 1997] [Sztipanovits et al, 2002].

Modeling software systems helps conceive and visualize the actual design, but real automation needs efficient model processing facilities, namely, model compilers.

The concept of metamodeling is founded at the beginning of the object-orientation. Metamodeling is an efficient construct for creating a large class of models [Atkinson et al, 2003]. For metamodeling tools, it simplifies the creation of generic environments modeling either UML or other domain-specific languages that are more specific on a given domain.

Metamodeling is a central technique in the design of language. Metamodels define the abstract syntax and static semantics of the domain. Metamodels specify the modeling process, how model objects are composed, what attributes they have, what connections can be created between them, what semantics are imposed on them. In the modeling phase, the modeling environment has to apply rules contained by the metamodel.

Once a metamodel is defined, instances of this metamodel, i.e. models which conform to this metamodel can be created. This type-instance relation can be generalized as done in the metamodeling approach where metamodels are instances of meta-metamodels, and so on.

The definition of the key concepts from *software models* to *metamodeling* are as follows:

- *Software model*: a description of atomic concepts and the relation between them.
- *Software modeling*: the process that creates *software models*.
- *Modeling language*: a set of rules defining what the instance models should fulfill.

- *Metamodel*: a special model that defines the creation rules of further models: element types, their attributes and the relation types between the elements. (I.e. the metamodel is a method to define modeling languages.)
- *Metamodeling*: the process that defines the *metamodel* and the instance models are created based on it.

Domain-Specific Modeling languages provide a viable solution for improving development productivity by raising the level of abstraction beyond coding [Sprinkle et al, 2004]. With domain-specific modeling, the models are composed of elements representing concepts that are part of the problem domain world, not the code world (unlike, for example, the core UML concepts). Domain-specific modeling languages follow domain abstractions and semantics, allowing developers to perceive themselves as working directly with domain concepts. In many cases, full final product artifacts can automatically be generated from these high-level specifications with domain-specific code generators. We apply domain-specific modeling because it improves and accelerates the software and system development processes.

3.2.3 Defining Visual Concrete Syntax for Domain-Specific Languages

Metamodeling is an efficient method to create and handle visual languages. A metamodel acts as a set of rules for its model: it specifies the modeled types, the attributes of the types and the possible relations between them. The relationship between a metamodel and its model is called instantiation. Considering the instantiation, there are two issues: instantiation of topological rules (e.g. the instantiation of the association in the UML class diagram as links in the UML object diagram), and the instantiation of attributes (e.g. the attributes in a UML class can have values in the object diagram).

Besides the instantiation issues, and the ability to create the topological definition of visual languages by using metamodeling, it is also important to be able to define their appearance. Usually, a metamodel does not describe the appearance, often called the concrete syntax of its models, thus, there is a need for an additional method to extend metamodel definitions to support this feature.

Without metamodeling, modeling frameworks have to implement a hard-wired set of modeling types with hard-wired visualization. Although metamodeling techniques usually do not support managing the concrete syntax of visual languages directly, there are several techniques to extend metamodeling in order to overcome this shortcoming. These techniques use metamodels as abstract input information and extend this information somehow in order to define the visualization. Although there does not exist a standard for applying this extension, different approaches can be categorized by the method they store the concrete syntax definition and how they apply the extension of the metamodel.

The simplest solution uses the same visualization scheme for all models. This domain-independent notation is a simplified visualization of the abstract representation of the models. The approach does not really extend metamodel definitions, but maps simple graphical objects (e.g. rectangles) to the model elements. The main advantage of the solution is that it does not require additional work when the list of domains increases. If there is a need for an

easy to understand representation of the models, but the customization of notation is not important, then this solution is appropriate. This is usually the case when creating the metamodel of a domain. It is important to distinguish the relations by their type (association, or inheritance), but the custom visualization of the nodes is not necessary.

Another way to define the concrete syntax is to extend metamodels with properties focusing on the presentation. This solution means that input metamodels contain properties both for the presentation and the data. Visualization properties are created when the metamodel is constructed. Since presentation logic is encoded in the metamodel items (as properties of the metamodel item), model items with a common metamodel can be treated uniformly. This solution has also its limitations: (i) The solution does not clearly separate the structural (topological) data and the presentation, which can cause problems of tangled code for the editing process. (ii) If the method is not combined with manual coding, then either the customization facilities are strongly limited, or the core framework is extremely complex. (iii) Different models with the same metamodel cannot be visualized differently.

The third solution is to define the concrete syntax not as part of the (meta)model, but as program code applicable in the modeling framework. The approach creates a mapping between the metamodel items and the classes of the code as well. On the one hand, the solution requires manual coding, but on the other hand it is much more flexible, than the first two solutions. In this case, the concrete syntax can be customized simply, but customization needs a great amount of additional work. The typical realization of the approach is the plugin-based architecture, where the presentation is coded in plugins assigned to a metamodel and each type defined in the metamodel, has one or more classes defining the visualization.

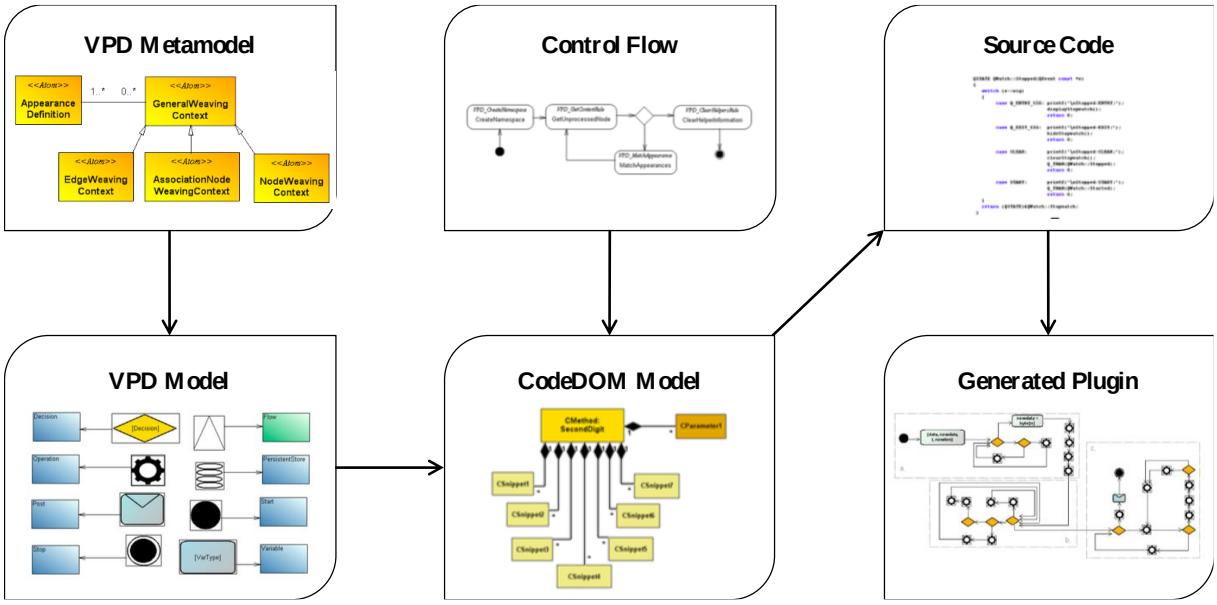

Figure 3-2 Overview of concrete syntax definition

The fourth solution models concrete syntax definitions applying a common Domain-Specific Language (referred to as Presentation DSL), and these definitions are attached to the metamodel. The original metamodel describes the structural (topological) definition of the target domain, while Presentation DSL (which is defined by a metamodel as well) describes

the presentation. Therefore, the different aspects of the domain (data and visualization) are clearly separated. The approach does not need manual coding, it can be standardized, and it allows multiple concrete syntax definitions for a single DSML. Additionally, the solution allows handling the concrete syntax in the same way as the standalone (meta)models, which simplifies editing, improves uniformity and flexibility. Another advantage of the solution is that it allows multiple concrete syntax definitions for a single DSML.

VMTS Presentation DSL (VPD) is a Presentation DSL realized within the VMTS framework. Figure 3-2 shows the main steps of the concrete syntax definition and processing. The VPD metamodel defines the metamodel for VPD, i.e. the structure of the concrete syntax definitions. By instantiating the VPD metamodel, concrete syntax definitions can be created. In order to facilitate the creation of the concrete syntax, a plug-in (the VPD Plugin) was implemented in the VMTS framework. To improve the effectiveness, there is an automatic solution for processing VPD models. This automatism applies model transformation techniques. The transformation converts VPD Models to *CodeDOM* models. *CodeDOM* is an abstract code representation. From the *CodeDOM* model, source code is generated with the .NET *CodeDOM* technology [Thai et al, 2003]. The source code implements a plug-in that can be used directly in VMTS Presentation Framework (VPF). This approach made it possible to avoid manual coding, and create plug-ins in a user-friendly, graphical way in the same environment as common DSL models.

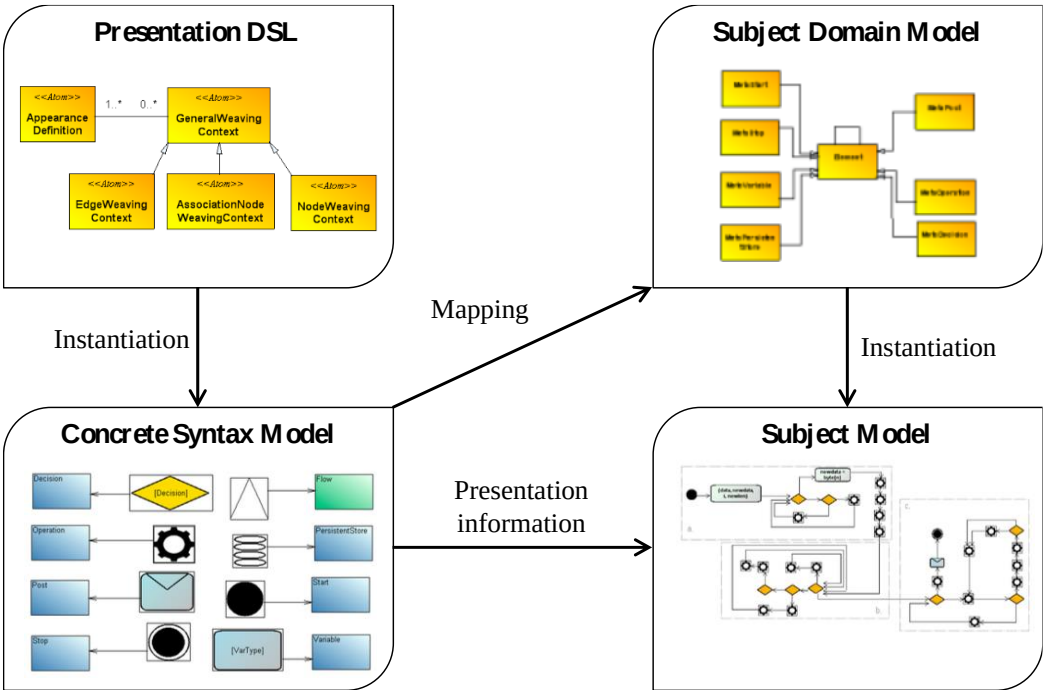

Figure 3-3 Overview of binding the abstract and concrete syntax definition

Figure 3-3 shows the metamodel – model, and the structural definition – concrete syntax relationships: (i) The Concrete Syntax Models are created by instantiating VMTS Presentation DSL. (ii) The structural definition of the domain, namely the *Subject Domain Model* is created, and the concrete syntax is mapped to the structural definition. From the *Concrete Syntax Model* a plugin is generated that is used to display the models of the target

domain. (iii) *Subject Model*s are created by instantiating the *Subject Domain Model*. (iv) The framework displays the model using the generated plugin [Mezei et al, 2006].

Now we discuss an example target domain, the Flowchart diagrams. The metamodel is depicted in Figure 3-4. The *Element* is an abstract base model item, it is used only as a common basis for the other node types, but it does not appear on the model level. The names of the model items refer to their functions. *InputOutput* statements are handled as a special type of generic *Statements*. Besides the inheritance relationships there are two relations available between the model items: one between *Element*s (between any kind of model items in Flowchart) and one between a *Condition* and an *Element*. The first relation expresses a single step between the elements, while the second relation can describe the output actions of the decision elements. The distinction between these two relations is not required, but in this way we can express which edge belongs to which decision result. The next step is to add attributes to metamodel items. *Condition*s have an attribute that describes the condition expression. *Statement*s have only one attribute that describes the statement code, while the relation between *Condition* and *Element* can describe a condition statement.

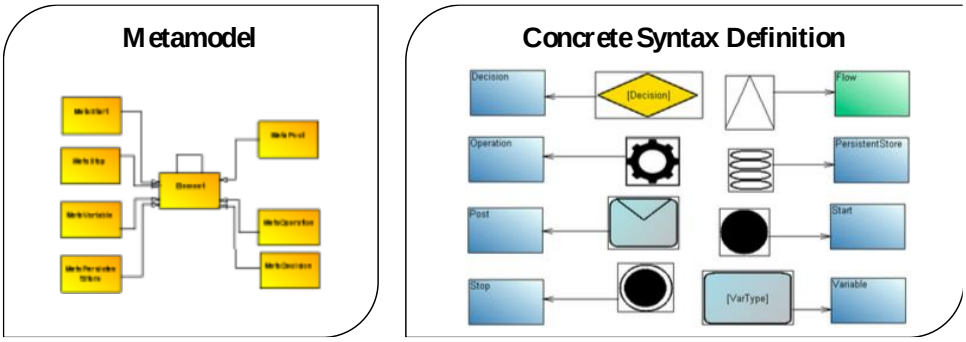

Figure 3-4 Example: the metamodel and the concrete syntax definition of the Protocol DSL language

As a next step, structural definition is mapped to a basic concrete syntax definition. This mapping, the creation of basic concrete syntax elements is partly automated. The user has to draw the customized notation (using VPD plugin) and customize the name of the model items, if required. As a result, the complete concrete syntax definition (Figure 3-4) can be created. Blue boxes on the figure are *NodeWeavingContext*, while green boxes are *EdgeWeavingContexts*. Other boxes are *AppearanceDefitions* associated with the corresponding weaving context.

The concrete syntax is processed by a model transformation (the VPD transformation) and a plugin source code is generated. The code generation is automated. Finally, the generated plugin is compiled and loaded into the modeling framework. Example Protocol DSL diagram with the defined concrete syntax is presented in Figure 3-5.

In a general case, the target domain does not require support for unique features, but the domain-specific visualization is required. In these cases the Presentation DSL is an efficient solution due to its advantages mentioned earlier. The approach is comfortable and quick in comparison with full manual coding. Research and development experiences have shown that creating visualization for a domain like UML Activity Diagram takes approximately two

weeks with manual coding, but only one day when using Presentation DSL. This ratio is even higher if we consider the cost of maintenance as well.

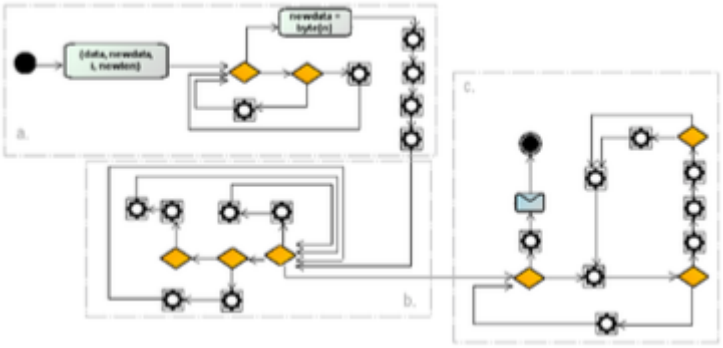

Figure 3-5 Example Protocol DSL diagram with the defined concrete syntax

3.2.4 Processing Software models with Model Transformation, Validated Model Processing

Model transformation means converting an input model that is available at the beginning of the transformation process to an output model, and it is a possible solution for model compiler realization. Model compilers can support properties to guarantee, preserve or validate them. [Lengyel and Charaf, 2015d] [Lengyel and Charaf, 2015e]

Model transformation lies at the heart of the model-driven approaches [Metzger, 2005] [Sztipanovits et al, 2002]. With model-driven software development, a modeling environment operates according to a modeling paradigm, which is a set of requirements that define how a system within a domain is modeled. The modeling paradigm is captured in the form of formal modeling language specifications referred to as metamodel. Once a metamodel is created for a particular domain, a modeling environment allows a modeler to create DSMs that can be synthesized into various artifacts.

Transformation approaches focus on models, support the flexible creation of modeling environments, and help to follow the changes of the models. At the same time they facilitate code generation and provide tool support for turning the created models into code artifacts. Metamodeling environments and model processors together form the tool support for model-driven solutions.

Models can be considered special graphs; simply contain nodes and edges between them. This mathematical background makes possible to treat models as labeled graphs and to apply graph transformation algorithms to models using graph rewriting. Often it is not enough to match graphs based on the topological information only, we want to extend the desired match by other properties, e.g. we want to match a subgraph with a node which has a special property or there is a unique relation between the properties of the matched nodes. For example we want to match a node with a special integer type property which value is between 1 and 4. The rewriting rules allow assigning constraints to the rules. Because these constraints are bound to the rules, they are able to express constraints local to the host graph area affected by the rules. This approach is inherently a local construct, because the elements not appearing in LHS or RHS cannot be directly included in the constraint statements. Although the specification has

this local nature, it does not mean that validating them does not involve checking other model elements in the input model: constraint propagation needs to be taken into account by both the algorithms and the user of the transformation on specifying constraints. The constraints which are enlisted in the LHS and RHS graphs affect the matched instances of the LHS and RHS graphs.

Design-by-contract. The purpose of contracts is to help to build better software by organizing the communication between software elements through specifying the mutual obligations [Meyer, 1988]. Contracts are used to guarantee that these communications occur on the basis of precise specifications of what these services are going to be. For the software to be able to guarantee any kind of correctness and robustness properties, they must know the precise constraints over such communications. In a client/supplier relationship, where the client needs a certain service and the supplier provides that service, the client has to fulfill certain obligations before calls the supplier. These preconditions are obligations for the client. In the other direction, we are going to express the conditions that the supplier routine must guarantee to the client on completion of the supplier's task. That is the postcondition of the contract, specifically, the postcondition of that particular routine. The postcondition is also an obligation for the supplier. Besides pre- and postconditions the third fundamental elements of contracts are invariants. A class invariant is a condition that applies to an entire class. It describes a consistency property that every instance of the class must satisfy.

To define precisely the transformation rules beyond the topology of the visual models, additional constraints must be specified which ensure the correctness of the attributes, or other properties to be enforced. Dealing with constraints provides a solution for these unsolved issues, because these problems can be addressed by constraint validation, but topological and attribute transformation methods cannot perform and express these kinds of model properties.

At the implementation level, system validation can be achieved by testing. Different tools, processes and methodologies have been developed and worked out to assist in testing the implementation of a system. Such approaches are for example, unit testing, mutation testing, and white/black box testing. In case of model transformation environments, beside the validation of the transformation engine, the transformation specification should also be validated.

There are several model transformation environments, which provide interfaces that allow implementing transformations using an optional object-oriented programming language. Related to the expected output there is nothing that can be guaranteed by these transformations.

There are few facilities provided for testing offline transformation specifications in an executable style. Also, there is a need for solutions that can validate model transformation specifications during its execution. This means that online validated model transformation approaches are required that guarantee if the transformation finishes successfully, the generated output is valid, and it fulfills the required conditions, i.e. the conditions expressed by the constraints.

A *precondition* assigned to a transformation rule is a *boolean* expression that must be true before the rule is executed. In a similar way, a *postcondition* is a *boolean* expression that must be true after the execution of the rule. If a *precondition* of a rule is not true then the transformation rule fails. If a *postcondition* of a rule is not true after the execution of the rule then the rule fails. A direct corollary of this is that a constraint in LHS is a precondition to the transformation rule, and a constraint in RHS is a postcondition to the rule. A transformation rule can be fired if all conditions enlisted in LHS are true. Furthermore, if a rule execution is finished successfully then all conditions enlisted in RHS must be true [Lengyel et al, 2008].

Constraints (pre- and postconditions) facilitates to specify precisely the execution of the rules contained by the whole transformation. Using constraints for each rule, the required conditions can be defined in details. Based on these considerations, executing a transformation rule follows the next steps:

1. Finding topological matches in the input model,
2. Checking the preconditions (constraints assigned to the LHS of the rewriting rule),
3. Performing the rewriting,
4. Checking the postconditions (constraints assigned to the RHS of the rewriting rule).

If a transformation contains rules specified properly with the help of constraints, and the transformation has been executed successfully for the input model, then the generated output model is in accordance with the expected result. This means that we should design the transformation rules, and fully specify them with constraints (pre- and postconditions). Next, if the execution of the transformation finishes successfully, it produces a valid result.

A sample transformation rule is presented in Figure 3-6. The rule represents the structure of both the left-hand side and right-hand side graphs, provides the metatype information (e.g. *<<Class>>*, *<<Table>>*), and assigns the constraints to the rule nodes (e.g. *Cons_C1*, *Cons_T1*)

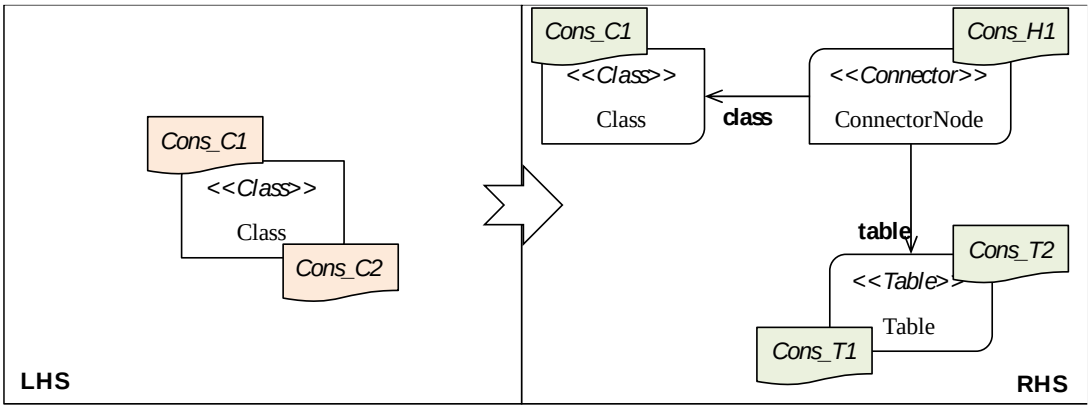

Figure 3-6 Example rewriting rule

3.2.5 Supporting Dynamic Behavior of Domain-Specific Languages

Domain-specific modeling is a powerful technique to describe complex systems in a precise but still understandable way. Rapid creation of graphical domain-specific languages (DSLs)

has been focused for many years. Research efforts have proven that metamodeling is a promising way of defining the abstract syntax of the language. It is also clear that DSLs can be developed to describe the concrete syntax. The visualization of models is supported by the VMTS Presentation Framework (VPF).

The VMTS Animation Framework (VAF) [Mészáros et al, 2009] is a flexible framework supporting the real-time animation of models both in their visualized and modeled properties. The architecture of VAF is illustrated in Figure 3-7.

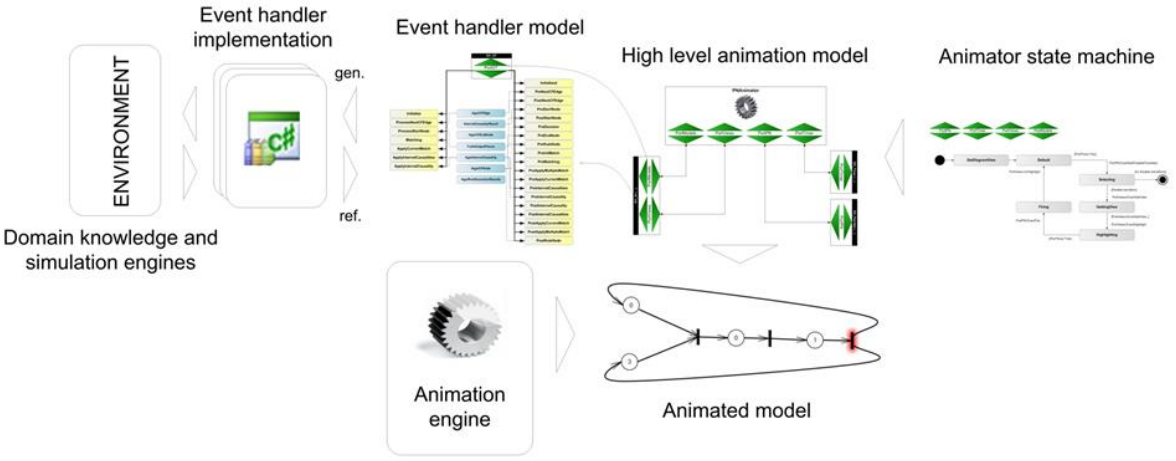

Figure 3-7 Architecture of the VMTS Animation Framework (VAF)

VAF separates the animation of the visualization from the dynamic behavior (simulation) of the model. For instance, the dynamic behavior of a graphically simulated statechart is different from that of a simulated continuous control system model. In the approach, the domain related knowledge can be considered a black-box, which integration is supported with different visual modeling techniques. With the help of this approach, we can integrate various simulation frameworks or custom components with event-driven communication. The animation framework provides three visual languages to describe the dynamic behavior of a model, and their processing via an event-driven concept. Event handling is the key point of the solution. Events are parametrizable messages that connect the different components of the environment. The services of the Presentation Framework, the domain-specific extensions, and possible external simulation engines (*ENVIRONMENT* block in Figure 3-7) are wrapped with event handlers, which provide an event-based interface. The communication is realized with events and event handlers. The definition of event handlers is supported with a visual language. The visual language defines the event handler, its parameters, the possible events, and the parameters of them - called entities (*Event handler model* in Figure 3-7). The default implementation of an event handler is generated based on the model, but the event handler methods which interact with the wrapped object have to be written manually (*Event handler implementation* block in Figure 3-7).

The animation logic is described using an event-driven state machine, called *Animator* (*Animator state machine* block in Figure 3-7). We have also designed a visual language to define these state machines. The state machine consumes and produces events. The transitions of the state machine are guarded by conditions (*Guard* property) testing the input events and

fire other events after performing the transition (*Action* property). States also define an *Action* property, which describes an operation that is executed when the state becomes active. The input (output) events of the state machine are created in (sent to) another state machine or an event handler. The events produced by the event handlers and the state machines are scheduled and processed by a DEVS [Zeigler et al, 2000] based simulator engine (*Animation Engine* in Figure 3-7).

The event handlers and the state machines can be connected in a high-level model (*High level animation model* in Figure 3-7). The communication between components is established through ports. Ports can be considered labeled buffers, which have a configurable but predefined size.

On executing an animation, both the high-level model and the low-level state machines are converted into source code, which is executed after an automated compilation phase.

Utilizing the capabilities of the Animation Framework we have solved several challenges, including the Model Transformation Debugger and the Matlab Simulink integration.

Model Transformation Debugger

The goal of building a debugger for visualizing model transformations was to facilitate tracing the transformation process, and to have the possibility to intervene at runtime. Therefore, we had enumerated following objectives: (i) the input and output models should be visualized and should always reflect the current state of the models; (ii) the control flow model should be animated to be able to exactly trace the execution of the transformation; (iii) the actually executed rewriting rule should be shown and in case of a successful match, the match should be visualized; (iv) the transformation should run step-by-step and continuously, the continuous running should be able to be interrupted by breakpoints, and the user should be allowed to perform jumps in the control flow, (v) the models, especially the host and the target models, could be edited at runtime.

The model of an event handler defines the events it can handle, the parameters of the events, and the interface of the event handler. The interface of a component is described with a set of ports: both event handlers and state machines provide their services through ports that can be connected to each other.

Figure 3-8 illustrates the event handler model of the model transformation engine. The event handler (*EH_GT*) defines a port (*PortGT*) to send and receive events. On the left-hand side the events received by the event handler are shown, on the right-hand side the events sent by the event handler are depicted. In the middle, the entities (the parameters used by the events) are enumerated.

The animation is described with the help of a visual language, which we use to model state machines. These state machines communicate via events: the state transitions trigger the existence of a specific event on a specific port, or a specific event combination on a set of ports, and fire events when performing the state transition. In the terminology the state machine is called *Animator*. Animators are modeled on two levels: (i) on the high-level representation several animators and event handlers can be connected, and their interaction

can be modeled, (ii) on the low level representation the individual states and transitions between the states of the state machine can be modeled. Figure 3-9 illustrates the composition of animators and event handlers implementing the model transformation debugger. Event handlers (*EH_UI*, *EH_GT*, *EH_Timer*) can be seen on the left and right sides of the figure. The high level representation of three animators (*SIM_GT*, *SIM_MatchHighlighter*, *SIM_Shortcut*) is depicted on the top and the bottom of the figure.

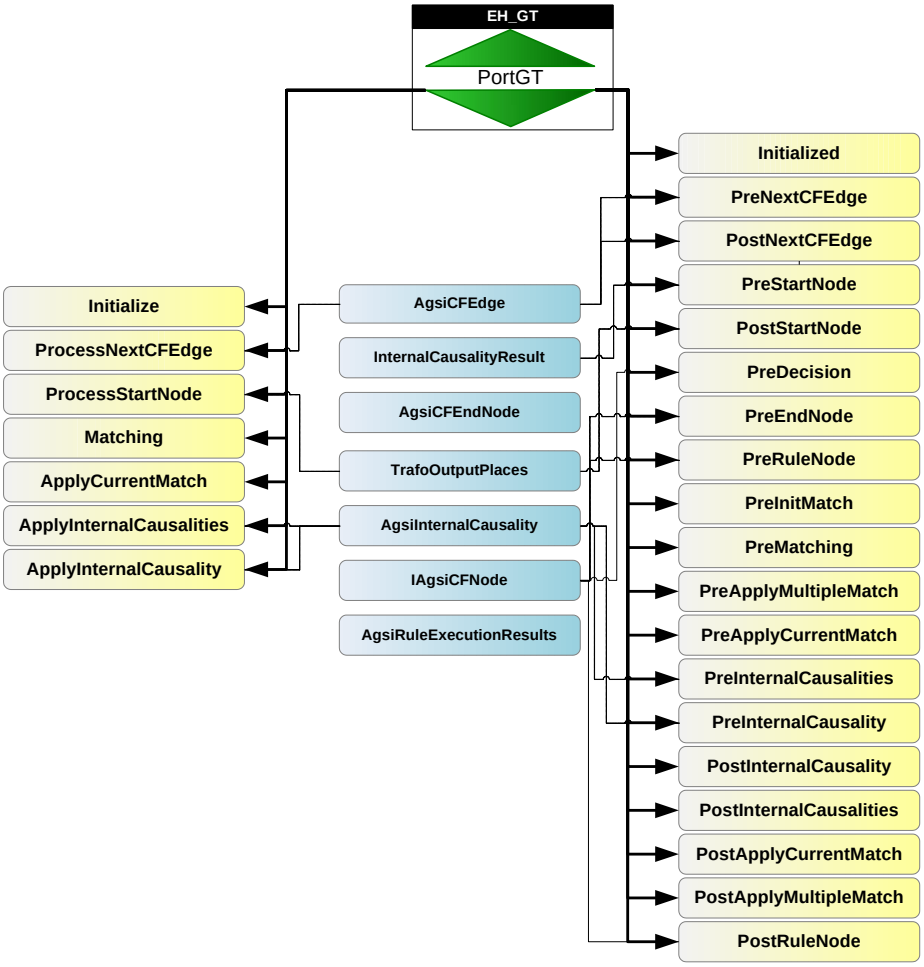

Figure 3-8 The event handler model of the transformation engine

In Figure 3-9, there are three animators: *SIM_GT*, *SIM_MatchHighlighter* and *SIM_Shortcut*. *SIM_GT* animates the control flow model, initiates the execution of the match and rewriting operations. *SIM_Shortcut* catches the key-presses, and instructs the *EH_Timer* to fire a *Tick* event if the key F11 was pressed. This feature is useful, if the timer is paused, and the user can execute the transformation step-by-step by pressing the key F11. *SIM_MatchHighlighter* catches the mouse events, and highlights matched and created elements in the host and the output model of the transformation, if the mouse hovers over an element in the rewriting rule. Thus, we can check which elements were matched by which item in the LHS of the rule, and which new elements were created after the application of the rule. Using several animators to provide a solution, we can clearly separate orthogonal aspects of the problem space.

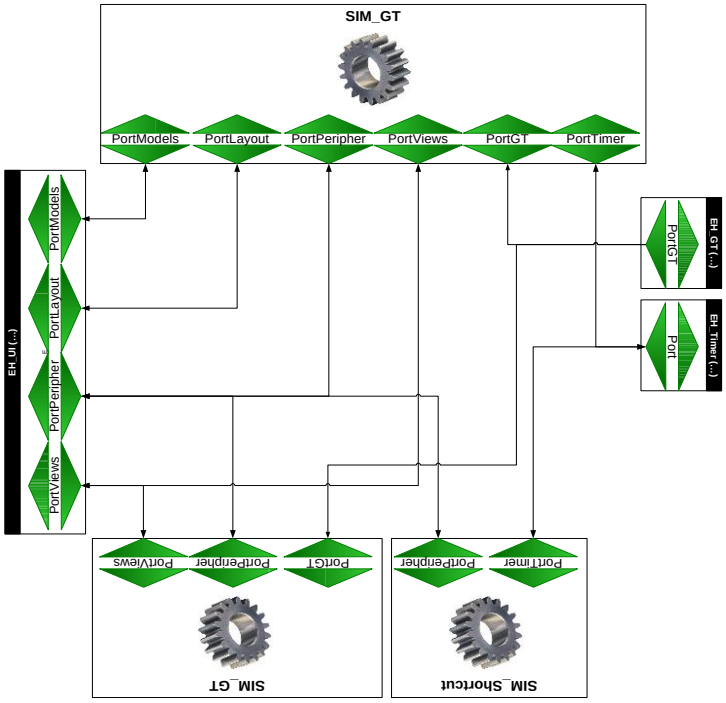

Figure 3-9 High level animation model of the debugger

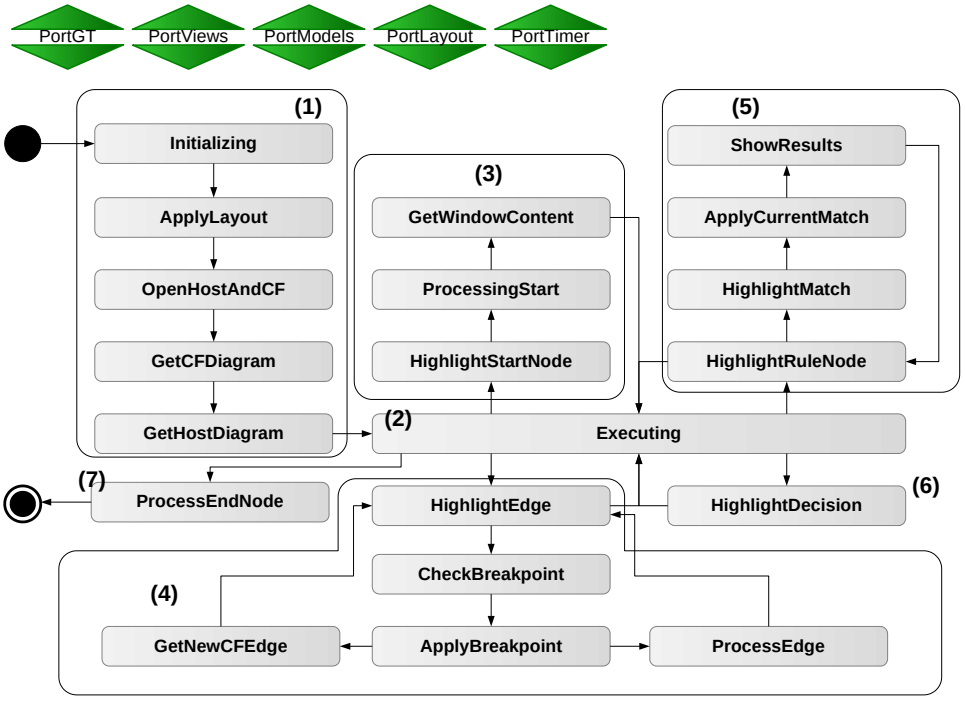

Figure 3-10 *SIM_GT* animator

Figure 3-10 presents the internal structure of the *SIM_GT* animator. This component is responsible for animating the control flow model, including detecting breakpoints and performing jumps, initiating the execution of rewriting rules, and visualizing the changes of the output model. States and transitions in block (1) are applied to initialize the transformation, to open the host and create the output model and to obtain a reference to the

opened diagrams. The *Executing* state can be considered a default state of the animation, the processing of the individual elements of the control flow model are initiated and finished in this state. Blocks (3), (4), (5), (6) and (7) are similar in the way that they are responsible for processing and highlighting the elements of the control flow, namely the start node, edges, rule nodes, decisions and the stop node.

Using the discussed solution the debugger is defined with the help of visual modeling techniques. Building on the VMTS Animation Framework, we could connect the animation of the user interface with the model transformation engine.

Within the introduced solution the problem area have been modeled on three different levels. (i) The event handler model is used to wrap the model transformation engine with an event based interface. (ii) The high-level animation model connects event handlers with animators defining orthogonal aspects of the problem. (iii) The state machine models integrate the messages of the user interface and the transformation framework. They decompose the events of the transformation engine to a set of UI events (e.g. opening several diagrams after processing the start node), and also integrate messages of the event handlers into one or several new events (e.g. sending *EventHighLight* events after receiving timer *PreApplyCurrentMatch* and *MouseOver* events). The skeleton of the event handler implementation is generated based on the event handler model; the animation model and the low-level state machines are used to generate the executable binaries implementing the debugger. Figure 3-11 depicts the VMTS Model Transformation Debugger in work.

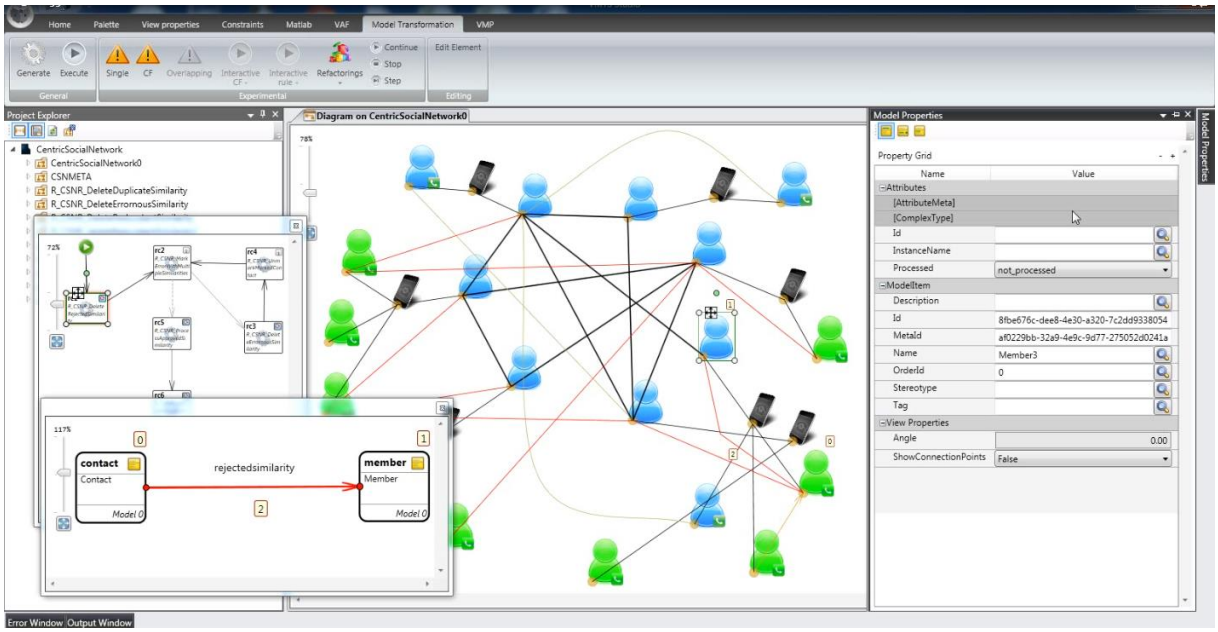

Figure 3-11 Model Transformation Debugger in VMTS

MATLAB Simulink Integration

A shortcoming of the previous solution was that the integration of external components (including simulation engines and third-party components) required too much work compared to using their API themselves. To integrate a component, we needed to create the event handler model including all the events for the components and we also needed to write the

gluing code between the generated event handler and the component by hand. This can be an error-prone task. If the interface of the adapted component changes for some reason, the changes had to be propagated to the event handler model by hand, and the gluing code had to be updated. To address this issue the approach has been extended to load the meta information of the actual component and to generate the event handler classes automatically, including both the event implementations and the gluing code. There are several types of .NET class-members we can assign events to. The most trivial one is the .NET event that is a language-supported solution to implement some kind of notification service. We subscribe delegates (roughly: function pointers) to the published events of an object, and they can be called by the publisher object using the Observer design pattern [Gamma et al, 1995].

Instead of modeling the necessary events to wrap a component, we provide tools to automate the wrapping. In addition to define events in the event handler model manually, one can extend the event handler by adding *fields* to it. The presented procedure is also supported by a graphical user interface which automatically provides available type members and ports to create *Member* attributes.

To illustrate the usage of the approach, we have integrated the MATLAB [MATLAB] API into VMTS, and we simulated a Simulink [Simulink] model and presented its results in VMTS. MATLAB does not have an official .NET-based interface, so we have used a free class library which wraps the most important MATLAB operations with a .NET class. The applied class library is the EngMATLib [EngMATLib]. Figure 3-12 overviews the main points of the Simulink-VMTS integration.

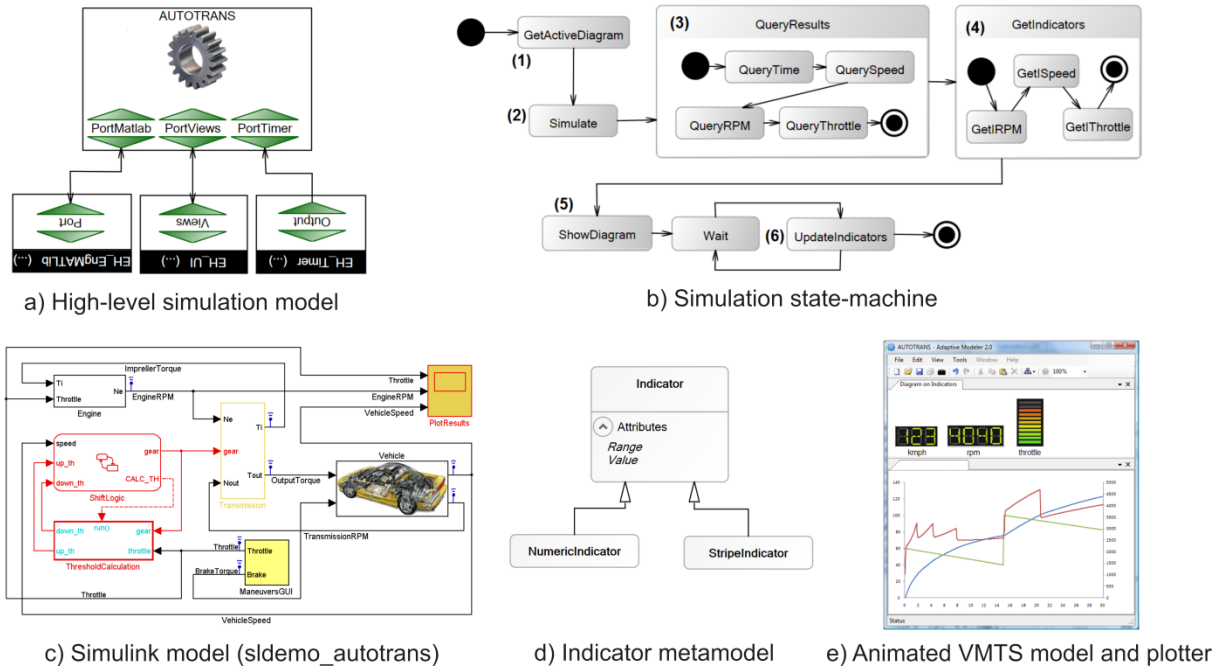

Figure 3-12 Integrating Simulink simulations into VMTS

The simulated Simulink model (called *sldemo autotrans*, Figure 3-12c is shipped with the MATLAB installation, and models an automatic transmission controller. If we execute the simulation within the Simulink environment, we can inspect the parameters of the car

(including speed, engine rotation-per-minute, and throttle state) at different conditions. In the following example, we present these parameters on diagrams and also we monitor them in VMTS. For this purpose simple visual language has been created, which can model indicators: numeric and stripe (Figure 3-12d). Each indicator has a *name* a *range* and a *value*, the modeled indicators are animated according to the results received from MATLAB.

Figure 3-12a and b present the high-and low-level (state machine) animation models. On the high-level animation model, the animator is connected to (i) the UI event handler (*EH UI*), (ii) to the generated *EngMATLib* event handler and (iii) to a *Timer* which schedules the animation by sending a *Tick* event every 0.1 secs. The state machine model consists of the following states: (1) query the active diagram, which contains the indicators to be animated (Figure 3-12e); (2) execute the simulation by sending an *Execute* event with a *"sim('sldemo autotrans')"* command to MATLAB; (3) query the time, speed, RPM and throttle state vectors using *GetMatrix* on the signal logging structure (*sldemo autotrans* output); (4) obtain a reference to the indicators using their names; (5) present the data-vectors on a diagram; (6) animate the indicators by iterating through the vectors, and updating the *Value* property of the indicators based on the data-vectors. Note, that the loop is controlled by the *Ticks* of the timer event handler.

Both of the examples, the Model Transformation Debugger and the MATLAB Simulink Integration, show the power of the VMTS framework in the field of model animation, i.e. in modeling and executing dynamic software models.

3.3 Related Work

Model-based application development necessitates the modeling of software requirements and the transformation of models between different stages of the design process. These transformations must be precisely specified. Many approaches have been introduced in the field of modeling, graph grammars and transformations to capture graph domains; for instance, Eclipse [Eclipse], GME [Lédeczi et al, 2001], MetaEdit+ [Tolvanen, 2006], GReAT [Karsai et al, 2003], FUJABA [Köhler et al, 2000], VIATRA [Varró et al, 2003], Attributed Graph Grammar (AGG) [Taentzer, 2004] and AToM^3 [de Lara et al, 2004]. These approaches are specific to the particular system, and each of them has some features that others do not offer.

Eclipse is a highly generic modeling environment. Eclipse Graphical Editing Framework (GEF) [GEF] is an open source infrastructure for creating and using graphical editors based on Eclipse. The underlying architecture of GEF is the Model-View-Controller pattern [Gamma et al, 1995]. The manageable data (model), the visualization (view) and the interaction features (controller) are separated into different classes. The model classes can be arbitrary Java classes, however, the controller classes should derive from the common *EditParts* class. The visualization can be performed by an arbitrary Java class by implementing a predefined interface (*IFigure*). GEF does not support automatic change-notification services, due to the underlying arbitrary model object. Notification mechanisms should be performed through manually written listener and adapter objects.

Graphical Modeling Framework (GMF) [GMF] is another Eclipse project, GMF utilizes GEF. Compared to GEF, GMF uses Eclipse Modeling Framework (EMF) [EMF] models as the underlying model object. EMF facilitates the serialization of models in various formats and the change notification mechanism is also a built-in feature. The editor environments in GMF are generated. The code generation is based on four models: (i) the EMF model (domain model) which serves as the model of the visual language; (ii) the Graphical Definition Model that defines the visualization of each model element; (iii) the Tooling Definition Model, which describes the additional visual elements required for the user interface (including editor palettes, menus) and (iv) the Mapping Definition Model, which realizes the mapping between the domain model and the visual models. The creation of models (ii)-(iv) and the generation of the editor from the input models is supported by the GMF Dashboard wizard.

Xtext [Xtext] is a framework for development of programming languages and domain specific languages. Xtext supports several aspects of a complete language infrastructure: parsing, linking, compiling, and interpretation.

Generic Modeling Environment (GME) is a general-purpose metamodeling and program synthesis environment. The modeling concepts of GME are represented by the MultiGraph Architecture (MGA) object network. MGA (and also GME) is built on the COM architecture. MGA objects provide change notification, transaction handling and support of multi-client collaboration on the same model. Model elements can be persisted through a unified storage interface to various storage types including relational database and XML formats. MGA provides general classes to represent model elements the properties of which can be reached through the same interface. However, it is also possible to edit model properties through a typed interface called Builder Object Network (BON), which is generated for a specific domain. BON also builds on core MGA objects and only wraps their features. Compared to GME, our framework does not use wrappers to support writing model processors. Our framework generates the typed interfaces and objects for each domain.

MetaEdit+ is an integrated modeling and metamodeling environment for DSLs. In MetaEdit+, developer can define a domain-specific modeling language with roles, constrains and specify the mapping these elements to code fragments in domain-specific generator. The model execution can be emulated, the generated code and the model elements are linked to each other, MetaEdit+ shows which model elements is under execute and also shows the corresponding source code.

A common drawback of all the tools above is that each of them uses the same class and object hierarchy to perform various operations on them (including visualization, model transformation, custom model processing) and do not support to use optimized solutions for different purposes. In contrast, VMTS can provide optimized data-representation for different kinds of applications.

The GReAT framework is a transformation system for domain-specific languages (DSL) built on metamodeling and graph rewriting concepts. The control structure of the GReAT allows specifying an initial context for the matching to reduce the complexity of the general matching case. The pattern matcher returns all the possible matches to avoid the inherent non-

determinism in the matching process. The attribute transformation is specified by a proprietary attribute mapping language, whose syntax is close to C. The LHS of the rules can contain OCL constraint to refine the pattern.

GReAT has a construct called test rule. A test rule is a special expression and it is used to change the control flow during execution. A test rule has only LHS. If a test rule is successful (matching was successful), the rule after the test node is executable.

In FUJABA, the combination of activity diagrams and collaboration diagrams (story-diagrams) are used to express control structures. Story-diagrams are a visual programming language that facilitates the specification of complex application-specific object structures. Moreover, FUJABA extended story-diagrams by statecharts to story-charts. Story-charts use statecharts and activity diagrams to define complex control flows and collaboration diagrams to specify the entry, exit, do, and transition actions that deal with complex object-structures.

VIATRA (Visual Automated Transformations) is a model transformation framework developed mainly for the formal dependability analysis of UML models. In VIATRA, metamodeling is conceived specially: the instantiation is based on mathematical formalisms and called Visual Precise Metamodeling. The attribute transformation is performed by abstract state machine statements, and there is built-in support for attributes of basic Java types. The model constraints can be expressed by graph patterns with arbitrary levels of negation. The rule constraints are also specified by graph patterns. VIATRA uses abstract state machines (ASM) to define the control flow of the system.

VIATRA is a model transformation framework developed mainly for the formal dependability analysis of UML models. In VIATRA, metamodeling is conceived specially: the instantiation is based on mathematical formalisms and called Visual Precise Metamodeling. The transformation language of VIATRA supports type checking, negative patterns, attribute conditions and traditional pattern matching issues. The rule specification language is a proprietary pattern language with type information. The attribute specification is graph-based. The attribute transformation is performed by abstract state machine statements, and there is built-in support for attributes of basic Java types. The model and rule constraints can be expressed by graph patterns with arbitrary levels of negation. VIATRA uses abstract state machines (ASM) to define the control flow of the system

AGG is a visual tool environment consisting of editors, interpreter, and debugger for attributed graph transformation; attribute computation by Java; supports a hybrid programming style based on graph transformation and Java. In AGG, termination criteria are implemented for Layered Graph Transformation Systems (LGTS). The layers fix the order how rules are applied. The interpretation process first has to apply all rules of layer 0 as long as possible, and then all rules of layer 1, etc. Rule layers allow specifying a simple control flow graph transformation. Once the highest layer has been finished, the transformation stops, unless the option “loop over layers” is turned on.

VIATRA and AGG applies negative application conditions (NACs). Using NACs it is possible to express structures and attribute value, but multiplicity conditions cannot be defined without additional constraints.

The transformation and simulation tool AToM^3 uses model transformation to simulation traces in order to simulate the operations. The rule constraints can contain generalized negative application conditions and can be pre- and postconditions to events. Constraints can be both semantic and graphical constraints. Similarly to AGG, the control flow consists of layers; the rules are sequenced by priority numbers within the layers. A rule is executed only once, but in case of non-overlapping matches, the rules are applied to all the matches.

The Model-Driven Architecture offers a standard interface to implement model transformation tools. The transformation related part of MDA is the Query, Views, Transformation for MOF 2.0 [OMG QVT]. Three types of operations are provided: *queries* on models, *views* on metamodels and *transformation* on models.

Compared to other approaches, VMTS meets the expectations in model-to-model and model-to-code transformation. VMTS has state of the art mechanisms for validated model transformation, constraint management and control flow definition. The environment has several standalone algorithms and other solutions that make them efficient.

VMTS has a unique constraint management and online transformation validation support. It provides a high-level control flow language with several constructs that optimize and make the transformations highly configurable.

3.4 Conclusions

Model-based development necessitates the efficient techniques for modeling and transformation of models between different stages of the design process. Modeling languages should be effectively designed and used by modelers and domain experts. The model processing transformations must be precisely specified.

In this chapter, a graph transformation-based technique for specifying such a model transformation has been presented: model transformation rules and the control flow of the transformation. We have discussed the metamodeling, software modeling concepts. Also we have introduced several model-driven considerations and model processing related results: supporting domain-specific modeling, defining visual concrete syntax for domain-specific languages, processing software models with model transformation, validated model transformation, and supporting dynamic behavior of domain-specific languages.

These results form the basis for the domain-specific modeling of different software aspects, e.g. data layer, user interface, static application structure, dynamic behavior, or communication protocols, and also the effective model processing and software artifact generation.

3.5 Acknowledgements

Related to the topic of this chapter, domain-specific modeling and model processing, there are 6 PhD theses submitted and successfully defended within the group under my supervision.

- Tihamér Levendovszky, *Applying metamodels in software model transformation methods*, 2006.
- László Lengyel, *Online validation of visual model transformatons*, 2006.

- Gergely Mezei, *Transformation-based support for visual languages*, 2008.
- Tamás Mészáros, *Supporting model animation methods with graph transformation*, 2011.
- Márk Asztalos, *Automated offline verification of graph rewriting-based model transformations*, 2013.
- Tamás Vajk, *Constraint validation-based performance optimizations in domain-specific modeling environments*, 2014.

4 Increasing the Efficiency of Mobile Platforms

A reasonable part of the research activities related to mobile platforms has been performed throughout the cooperation between Nokia and my research team. According to the agreement, the main results could be and have been utilized by Nokia.

4.1 Introduction

The mobile industry has developed immensely in the recent years. Mobile devices are currently fully featured elements of different networks, e.g. peer-to-peer networks, and have several advantages because of their mobility. Nevertheless, the most significant shortcoming of all portable electronics also applies to mobile devices: their operation time is limited by their battery capacity.

Two approaches are available to extend the operation time. The obvious one is to design hardware components that consume less energy and design batteries with higher energy density. The other solution, which is the focus of this research, is to develop software artifacts that use the system's resources more efficiently. The main advantage of the software-based solution is that the designed solutions can be applied on existing devices and do not require any special equipment or hardware resources. Investigating the different factors of energy consumption of mobile phones reveals that the most significant source of power consumption is the wireless radio [Creus et al, 2007]. In addition to transferring less data and switching off the wireless radio for longer periods, there are other techniques, for example Wireless Sensor Networks sleeping methods, that are examined and partly exploited by some of the solutions presented in the thesis. Measurements results [Nurminen, 2010] [Xiao et al., 2010] show that with wireless radio, including WLAN and 3G, the higher the bit rate, the more energy efficient the communication is. This suggests that in order to save energy, we should arrange the activity of content download in a way that the mobile devices operate at a bit rate as high as possible. Basically by reshaping the continuous but slow flow of data into short high-speed bursts, the energy consumption can be considerably decreased. [Charaf et al, 2014] [Kundra et al, 2015]

Peer-to-peer (P2P) systems operate over a network of nodes that provide services in a distributed manner by the peers themselves instead of using centralized servers. According to a recent study [Sandvine, 2012], P2P applications are still the second largest source of Internet traffic. With the development of mobile hardware and wireless Internet access, it became feasible to use mobile devices in P2P networks. This research focuses on mobile P2P networks and the energy aspects of content sharing with BitTorrent.

BitTorrent is the most widely used content-sharing solution with over 5 million simultaneous users. It enables the rapid transfer of data among a set of peers in a distributed way. The key idea is that the peers that have downloaded parts of the data start sharing it with the other peers. The popularity of the world's first mobile BitTorrent client, SymTorrent [SymTorrent] (the name is coming from the Symbian operating system as one of the most popular first generation smartphone operating systems), which was developed in the preparation phase of this research, clearly shows that today's mobile phones can effectively join and contribute to

distributed content sharing networks. Since using BitTorrent on a standard smartphone can completely deplete the phone's battery in about 5-6 hours [Nurminen et al, 2008], improving the energy efficiency of the solution is necessary to enable real mobile usage without the need of frequent recharging.

Further aspect of the research work is mobile social network solutions. Currently social networking solutions are one of the most popular sites on the Internet. These systems attract millions of users and their functionality is increasing rapidly. The basic idea behind such networks is that users can manage personal relationships online on the web user interface. Since their introduction, social network sites such as Facebook, Myspace, and LinkedIn have attracted millions of users, many of whom have integrated these sites into their daily practices and they even visit them multiple times a day. These sites have become an environment, created by the people who are using it.

With respect to developing technologies and social networks, mobile devices play an important role. The capabilities of mobile devices and mobile phones allow them to participate in rich Internet applications. Facebook statistics show that there are more than 844 million active users currently accessing Facebook through their mobile devices [Facebook Statistics, 2015]. People that use Facebook on their mobile devices are almost 50% more active on Facebook than non-mobile users. Mobile phone support in general social networks are mainly photo and video upload capabilities and access to the social network site using the application or nowadays mobile web browser.

Content sharing is a common feature of social networks. Members are able to share photos, photo albums or even videos with their friends. In case of mobile related social networks mobile phones are involved as individual entities. The challenge is to involve them into content sharing efficiently.

Another objective of the research is to investigate the role of mobile devices in content sharing. One of the most efficient content sharing solutions nowadays is the peer-to-peer (P2P) based BitTorrent technology. Involving mobile phones into such large networks is challenging if we consider the limitations of mobile phones. Previously we did not apply any constraints to the mobile phones when we considered involving them in the social network. In this investigation our goal is also to support as many mobile devices, even with different platforms and limited resources, as possible.

The result of this research can also be considered a proof of concept that mobile phones even with limited resources have an important role in social networks and they are also able to participate in large peer-to-peer networks like BiTorrent.

4.2 Contributions

This chapter discusses the investigation of the mobile peer-to-peer networks, the mobile platforms related energy efficient solutions and the network coding driven data distribution for mobile peer–to–peer networks.

4.2.1 Investigating Mobile Peer–to–Peer Networks

Peer-to-peer (P2P) system differs greatly from client-server architectures, where the service is provided by servers and clients only use those services. In case of peer-to-peer solution service providing is distributed between peers. In this way, in many cases peers behave as clients and servers at the same time. Figure 4-1 shows the typical architecture of peer-to-peer and client server systems, respectively.

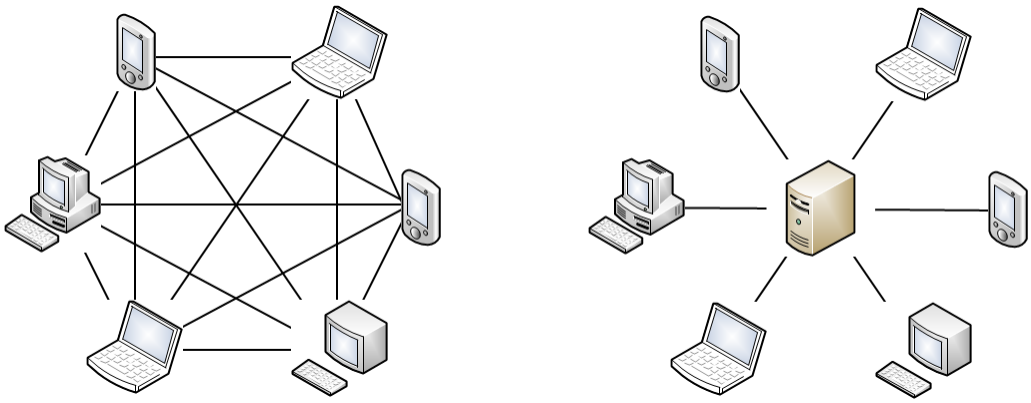

Figure 4-1 Typical architecture of peer-to-peer and client-server systems

Nowadays peer-to-peer systems are used in different situations: content sharing, audio- and video streaming, communication, cryptocurrencies, distributed search and distributed computing. Peer-to-peer solutions are very efficient from content distribution point of view (especially if the content is popular). In peer-to-peer systems there is no need for large central servers. The participants (usually PCs connected through the Internet) transfer the content between each another. In these systems the infrastructure and operational costs (internet access, energy consumption) are implicitly divided among the end-users. Content search is also conducted in a distributed manner.

Peer-to-peer systems are frequently mentioned as illegal services. The reason of this is that in many cases the technology is used to distribute copyrighted contents illegally. However, the technology has many advantages, therefore it was already applied in several solutions and in many cases the architecture is hidden from user perspective. The major advantages of the technology are scalability, fault tolerance and robustness.

From node types and roles point of view peer-to-peer systems are usually divided into two main categories:

- Pure peer-to-peer: Every node is equal with the same functionality. There is no central server or router. This decentralized topology follows the basis of the peer-to-peer architecture the most. Such systems are the original Gnutella [Gnutella], where network administration and search is also the responsibility of peers.
- Hybrid peer-to-peer: There are central elements in the networks that manage and overview the peers. There are also multi-level architectures where the group of the peers has special roles and they form an additional subnetwork. These entities are usually called as supernodes or superpeers.

BitTorrent [Cohen, 2003] is one of the most efficient content sharing solutions nowadays. BitTorrent is a swarming peer-to-peer protocol designed for the distribution of content between a large number of peers. The content is divided into small pieces and peers exchange these pieces in the swarm. The key idea is that when a piece has been downloaded to a peer, the peer starts sharing it with the others immediately. The protocol relies on the tit-for-tat strategy, which ensures the higher download rate is provided to good uploaders, while egoistic behavior is penalized. It has been estimated that peer-to-peer networks, primarily BitTorrent swarms, have accounted for roughly 20% of all download traffic and 50% of all upload traffic in the Internet as of 2012 [Sandvine, 2012].

Although the most popular uses of BitTorrent are entangled with legal issues, there are several legitimate uses as well. Facebook and Twitter uses BitTorrent for server deployment, Blizzard provides game updates via the Blizzard Downloader, which is hybrid BitTorrent/HTTP client, and several content providers shares downloadable content via BitTorrent [Braun et al, 2015].

Generally speaking, any file hosted in a centralized way can be made available via BitTorrent, which can decrease the server load and increase the download speed.

4.2.1.1 Examining the Semantic Information Retrieval Possibilities in Mobile Peer-to-Peer Networks

The problem of information retrieval has been one of the most serious challenges in the history of information technology. With the growing number of networked computers it becomes more and more difficult to find a specific document or other piece of information or resource.

We know that mobile devices with their current capabilities have joined the information retrieval networks. However, these devices require the use of novel protocols for efficient performance, because of their unique properties. Significant research efforts have aimed at designing an overlay network on the unstructured peer-to-peer protocols based on semantic data, in order to increase the efficiency of information retrieval.

We have examined the answering probability that can be achieved by semantic overlay in structured peer-to-peer networks. We built a model that helps compare the different approaches and reveals the parameters required to construct an optimal network layer.

We proposed algorithms and worked out a solution that facilitates to construct and maintain data structures, the semantic profiles that help approximate the parameters revealed by the model with the use of local decisions and low network traffic. We worked out an appropriate topology that can decrease the clustering observed in the query propagation path, which clustering reduces the efficiency of the protocol.

In order to illustrate the practical applications of the results in mobile environment, a specific protocol extension for the Gnutella protocol and a modular mobile client software package for the Symbian operating system have been developed.

4.2.1.2 Scalability and Performance Planning in Mobile Related Social Networks

In mobile related social networks, the number of identities is a key parameter from scalability and performance point of view. Therefore, we examined the distribution of identities raised by members, and created a model to estimate the total number of identities in the network. We have examined the accuracy of the proposed model, and provided a probability value for this accuracy.

When a member changes her or his profile, the number of synchronizations depends on the number of identity edges related to this member. Considering an online social network the operation that a profile is changed occurs quite often. In case of mobile-related social networks, not only the member profiles but the phonebook entries can be also changed. This way the number of identity links is a key parameter to the performance and scalability of the network.

Based on the research results, we have seen that the total number of identities increases approximately linearly with the number of members. Another explanation for this observation is that we have shown that the number of phonebooks follows exponential distribution thus the expected number of phonebook sizes based on that measurement is very accurate in this case.

For the measurements we used real data from the eight months operational period of our Phonebookmark application (a common project with Nokia). Based on measurements we have shown that the database provides common characteristics to other social networks and web graphs, which confirms the additional measurements related to similarities.

4.2.1.3 Efficient Content Sharing in Mobile Environments

Mobile phones have two major categories. Smartphones are considered basically as small computers with advanced hardware and software capabilities, while feature phones are simpler mobile devices but they also have several multimedia features. In this research we consider feature phones in order to support a wide range of devices.

When bringing new technologies, such as peer-to-peer technology to feature phones, it should be examined whether it is able to run on the limited resources (CPU, memory, network handling, file system, battery power) of the target devices well. We considered feature phones as mobile phones with limited capabilities. These devices usually have a maximum 1 or 2 MB large heap size. The target platform of the solution, the MobTorrent, was Nokia S40, however the application was written in Java ME, thus it runs on any device which support the proper Java version.

With a mobile BitTorrent solution mobile phones are able to download large amount of data in a relatively short time. For example, in the case of a mobile related social network where it is possible to share complete albums or even videos between friends and they can download these to their mobile devices. This way, when several users are downloading content, they can overload the server that decreases the performance and response time of the major functions and even similarity detecting and synchronizing.

The BitTorrent protocol uses HTTP connections for communicating with the tracker and TCP connections for the download and upload procedure from/to other peers. These communication protocols are supported in Java ME MIDP 2.0. While we were examining the network capabilities of the simple devices, it was found that there are some difficulties from peer-to-peer point of view.

We realized and checked with specialists that there are limitations about how many connections can be maintained at the same time. We also realized a socket handling problem in the Java ME implementation. A peer-to-peer application usually has to connect to several other peers. It is possible that some of the peers are off-line or not responding at all. The problem is that the timeout when the system realizes that the address is not responding is 244 seconds on S40 devices and 163 seconds on S60 devices.

Furthermore S40 platform (tested up to S40 5th Edition FP1) can handle only one connection request at the same time (S60 is able to handle 8 connection requests in parallel). These show that peer-to-peer applications have different platform requirements than other type of applications and that they bring up problems that are not experienced by other ones.

Processing Power Limitations. Implementing BitTorrent on resource-limited mobile devices sets special requirements to the algorithms. First, we should observe memory usage since its size is much more limited comparing to PCs. Therefore, a best practice is to store only the data that we are currently working on in the memory. Another important thing is to support some kind of thread handling, because BitTorrent uses several connections at the same time. With proper balancing of different connection threads, the application is able to download from several peers at the same time.

When we develop software for simple mobile devices we also have to consider that they have limited processing power. BitTorrent protocol uses the Secure Hash Algorithm version 1.0 (SHA1) for checking the consistency of the downloaded content. When the application has downloaded a piece it has to calculate the hash value of the content and compare it to the value in the original torrent file. This mechanism allows us to find and avoid the faulty or badly downloaded pieces. Experiences shows that one piece out of 100 can be bad, which means, without hashing, checking the probability of error is quite high even for medium-sized content.

File Handling Limitations. In BitTorrent it was required to store the downloaded content on the file system of the device. File handling in Java ME differs from the general Java file handling. The main difference is that we can access the file only via input- and output-streams.

The major difficulty is reading a file from a specific position, because there is no efficient file seeking mechanism implemented. The problem is that it is not possible to move to a specific position within the stream. For instance if we want to read the last 50 kB from a 40 MB file then we have to read the whole input stream until we reach the right position, and that requires time.

We managed to overcome this issue in case of content download, because we calculate the piece hash values iteratively as they are being downloaded, thus we do not need to read the piece from the file system when it has been downloaded completely. However this issue remains when we consider piece uploading, because piece requests can arrive from any position of the file.

We have examined the limitations of feature phones and shown that the feature phones with limited resources can participate in large peer-to-peer networks like BitTorrent. We have designed a BitTorrent solution that considers the limitation of feature phones. We have modeled the energy consumption of a mobile peer-to-peer solution. Furthermore, we have created an efficient content distribution solution for mobile related social networks. We have analyzed the solution and concluded that it does not overload the system.

4.2.1.4 Bringing Mobile BitTorrent into Content Sharing Systems

Content distribution over computer networks has always been a big challenge. First of all we have to decide whether to design a client-server solution with central units or a peer-to-peer solution where the content is distributed by using the resources of the network nodes.

The general architecture becomes even more complex if we plan to support mobile phones and other types of mobile devices as well. Another important objective is to create an efficient content sharing solution that does not overload the server(s) but at the same time is able to support several clients including mobile devices.

We were involved at Nokia Research Center (later Nokia Siemens Networks) in the Swarm project where the goal was to design a general hybrid peer-to-peer architecture that decreases the load of the central server, therefore it is ideal for service providers.

The final architecture is based on the efficient BitTorrent protocol. This protocol is known as a peer-to-peer protocol, but it has certain central elements (tracker, torrent file catalogs) that make it a good building block for a hybrid system.

In the Swarm hybrid peer-to-peer architecture a central element is responsible for the content management (which album belongs to the selected member, etc.) and it operates as a backend seed if no else shares the specific content, however when more members try to download the same content, the BitTorrent based peer-to-peer engine decreases the load of the server.

From the usability point of view, the main advantage of Swarm architecture is that it hides the technological backgrounds from the user. In order to begin a download we do not have to know anything about peer-to-peer technologies or BitTorrent, we just have to select the proper content via the interface of the service provider (e.g. a social network).

The proposed Swarm architecture is basically a general content sharing solution that can be embedded into different systems like social networks. When a service provider decides to implement a content sharing solution for users, the first idea is to create a client-server like solution. However creating and maintaining that type of system is rather expensive if we consider bandwidth requirement, CPU usage, and even energy consumption.

In case of the proposed hybrid architecture, significant part of the costs is distributed to user side; consequently it decreases the capital investments and operational costs for service provider. We analyzed the overhead of Swarm and we compared the operating cost of Swarm to a client-server solution. We have shown that the cost decreases when the same content is requested by more peers.

4.2.2 Energy Efficient Solutions for Mobile Platforms

A mobile device has several hardware components that drain the battery. The three highest energy consumers are the wireless interfaces (3G/Wi-Fi radio), the display and the application processor. These three are responsible for more than 70% of the device's energy consumption for most use cases. This thesis does not focus on user interface design or display technologies, thus the display factor can be neglected from this investigation. The effect of the processor highly depends on the use case; however, considering most peer-to-peer applications, its share in the overall energy consumption is still much smaller than the radio's [Nurminen et al, 2008]. Even under a relatively CPU-intense application, such as video conferencing, where the device needs to do the encoding and decoding of video frames, the CPU accounts for only around 17% of the total energy profile [Creus et al, 2007].

In a cellular device, two factors determine the energy consumption caused by network activity. First is the transmission energy that is proportional to the length of a transmission and the transmit power level. Second is the Radio Resource Control protocol that is responsible for channel allocation and scaling the energy consumed by the radio based on inactivity timers.

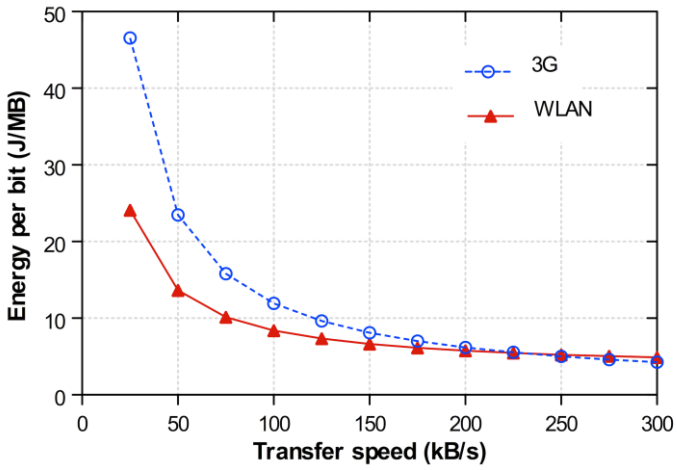

Figure 4-2 Measurement results: energy consumption per bit as a function of communication speed

An important observation in mobile phone energy consumption is that higher bitrate increases the energy-efficiency of data transfer. This is the result of the observation that, especially for 3G cellular, the power level of TCP data transfer is almost constant having only a weak dependency on the bitrate [Nurminen, 2010]. Only when bitrate drops to zero, meaning that there is no communication, the device is able to enter a sleep state with very low power level.

An extensive set of measurements for WiFi energy consumption [Xiao et al, 2010] with three different mobile phone models shows very similar results for 802.11g WLAN.

The measurement results illustrated in Figure 4-2 highlights how the energy consumption per bit varies as a function of communication speed. The shape of the curves shows that the higher the bit rate, the more energy-efficient the communication is. This suggests that in order to save energy we should arrange the content download activity in a way that the mobile device experiences as high bitrate as possible. One way to achieve this is to alternate between active transfer periods with high bitrates and idle periods with no traffic. So instead of receiving data at a constant low speed we prefer to communicate in high-speed bursts separated by idle periods. The higher speed we are able to reach during the active periods the more energy-efficient the data transfer will be. Note that the average download speed does not change, only the shape of the traffic.

To further investigate the energy characteristics of bursty traffic over wireless radio, we have performed measurements with varying transfer speeds and burst sizes. Figure 4-3 illustrates how burst size influences phone's energy consumption when communication is taking place over 3G and Figure 4-4 shows the WLAN results. In all cases 3 bursts of data was downloaded with 1 minute idle periods in between. Each measurement case was performed three times. The energy cost is determined by dividing the total energy consumption with the amount of data received.

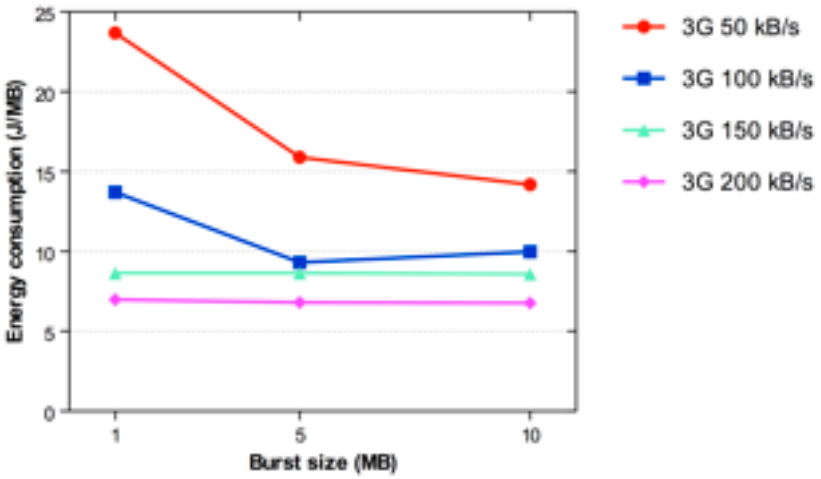

Figure 4-3 Energy cost of receiving data in bursts over 3G. Results of measurements performed with different burst size and transfer speed values

As expected, increasing the transfer speed decreased the energy cost in all cases. Bigger chunk size also had a positive effect on the energy in the majority of the cases. With WLAN, higher transfer speeds and larger chunk sizes reduce the energy consumption up to 25%. For 3G the differences are bigger with up to 70% decrease in energy-consumption. This can be mostly attributed to the radio deactivation timers. Normally it takes several seconds after sending the last data bit before the radio returns to the idle state. Therefore, when the data is sent in bigger chunks, the number of times the radio is activated and deactivated is reduced. As a result, the total time when the radio is powered-on but not communicating is reduced

resulting into energy savings. WLAN's much tighter idle timer works well in practice giving relatively good results even with the smallest burst size.

In summary, we can say that if the device is connected via WLAN, the optimal value of burst size is of less importance, but using larger bursts still has a measurable effect. On the other hand, with 3G cellular connections larger burst sizes and faster download speeds have a major impact on the energy consumption. Increasing the burst size from 1 MB to 5 MB had a more significant effect than going from 5 MB to 10 MB. With the current technologies, we cannot expect significant saving when increasing the burst size beyond the 5 MB - 10 MB range. Figure 4-3 also shows that we have an interesting trade-off between download speed and chunk size. For instance, when the download speed is reduced from 100 kB/s to 50 kB/s the phone consumes roughly the same amount of energy if at the same time the burst size is increased from 1 MB to 5 MB.

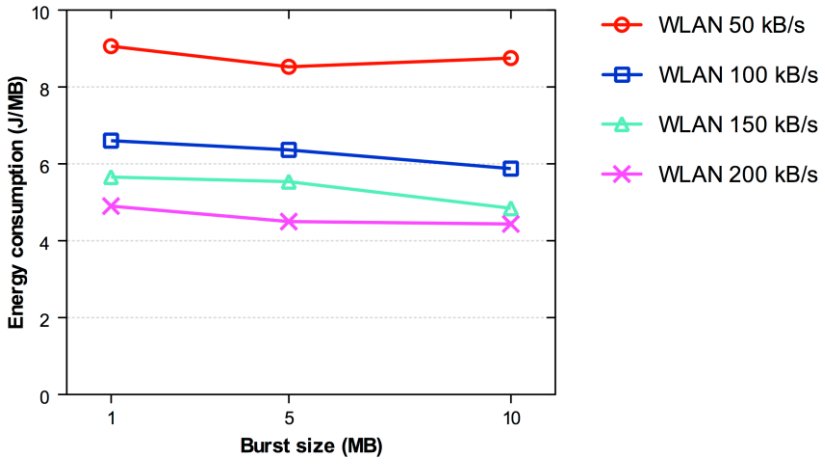

Figure 4-4 Energy cost of receiving data in bursts over WLAN. Results of measurements performed with different burst size and transfer speed values

A key characteristic of mobile phone communication is that sending data with a higher bit rate, in bursts, can save energy. Therefore, we arranged the content download activity so that the mobile device is able to experience a bit rate as high as possible. Furthermore, we have worked out resource-limited proxies for energy-efficient mobile BitTorrent.

4.2.2.1 CloudTorrent Protocol

We have designed and implemented the CloudTorrent system, which is a centralized BitTorrent proxy experiment. CloudTorrent consists of two main parts: a phone application communicating with the cloud and a server hosting a remote BitTorrent client. The phone application is a fork of the SymTorrent project. The client is able to control the remote BitTorrent server, check the download progress and transfer the downloaded files from the server to the mobile device. All communication with the server is carried out via HTTP requests. On the server side, uTorrent [uTorrent] is used, which is a popular free PC BitTorrent client with most of its functions available via an HTTP-based API. Since the uTorrent API does not support file downloading, we also run a separate web server that is used to transfer the downloaded files to the mobile devices.

Figure 4-5 shows the download speed and the energy consumption of the mobile clients. It is clearly visible, that CloudTorrent significantly outperformed SymTorrent: downloading the same amount of data required 672 J energy with SymTorrent and only 248 J with the CloudTorrent client.

As an alternative solution to hosting the proxies in the Internet, we have investigated the use of broadband routers as platforms for running BitTorrent proxies. Broadband routers that connect homes to the Internet are commonly available. It should be noted that even if the routers are located at home they do not limit the mobility of the users because they are accessible from everywhere via the Internet. If the mobile user happens to be at home, he could use the local WiFi connection but he can also connect to the proxy remotely through cellular data connection or through a public WiFi access point.

The router platform is attractive for a number of reasons. Since most homes are equipped with broadband routers, the installed base is large and the routers are frequently idle when their owners are on the move. Routers are typically powered up all the time and the energy consumption of the router is almost constant no matter how actively it is used. There is thus plenty of spare capacity that could be taken into use without additional costs for the users. From the technical point of view, the firmware of many routers can be changed, which allows extending the original functionality of the router.

Nevertheless, a number of new problems arise because broadband routers have limited resources. Although some models allow hardware extensions with USB devices, and some high-end router models even have built-in support for torrent download, typically, the memory size is limited and no mass storage is available. Future routers are likely to be more capable but we think that targeting minimalistic hardware is important both for utilization present router base and for the generality of the idea.

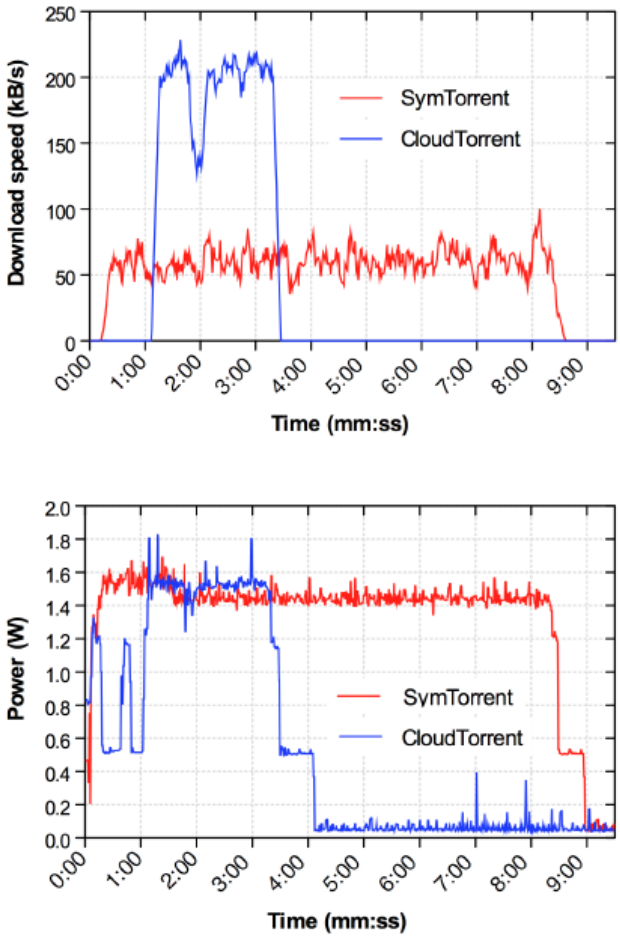

Figure 4-5 Energy consumption and download speed of a torrent download session using SymTorrent and CloudTorrent

4.2.2.2 BurstTorrent Protocol

We have developed a new BitTorrent-based protocol referred to as BurstTorrent, which enables improved energy efficiency for its participants. A fundamental difference between this and proxy-based BitTorrent solutions is that BurstTorrent does not require helper peers (proxies) or any kind of extra components. Nevertheless, all peers, including mobile devices that want to benefit from the lower energy consumption must implement a new protocol.

Legacy peers using standard BitTorrent can still participate in the swarm but has certain limitations. The essence of BurstTorrent is that instead of transferring data at a constant but suboptimal speed, peers negotiate time intervals with each other in which they transfer data at full speed. This can be thought as a high level traffic shaping solution that creates a bursty traffic schedule. Between the bursts, peers can enter idle (low energy) state, conserving energy.

Creating a bursty traffic schedule among a large group of peers is a difficult task that requires several peers to be synchronized and coordinated together either by a central component or in distributed way. The administration costs of this would most likely ruin the whole concept.

Therefore, BurstTorrent does not try to create a common transfer schedule for all peers but operates between a pair of peers instead. Creating a schedule between only two peers is a simpler task, which can be carried out with relatively small overhead.

In the context of BurstTorrent, all peers, regardless of them being fixed computers or mobile devices, run some kind of BitTorrent client. This client is either a standard, unmodified BitTorrent client or the extended BurstTorrent client. The peers are divided into three categories based on the type of BitTorrent client they host and their role in the transfer scheduling.

In BurstTorrent, bursty peers negotiate time intervals with regular peers when the regular peers would promise to use all the necessary resources to send content to the downloading peer with the agreed speed. Regular peers maintain an upload schedule in which they store points of time when data is required to be sent to bursty peers. Although regular peers could schedule multiple transfers, the protocol allows storing only one scheduled transfer at a time for each regular peer to minimize the size of the schedules both on side of downloading peers and on uploading peers. This means that the regular peers know only the next scheduled upload they are going to perform, and bursty peers need to wait until this upload is done to issue new schedule request to the regular peer. This way the excessive growth of the schedules can be prevented.

Regular peers serve other regular/legacy peers and bursty peers alternating in time. Each scheduled transfer period is followed by a period in which the regular peer serves only other regular or legacy peers. Regarding which peers are served, there is no difference between regular and legacy peers: both categories are treated equally. The length of the serving period is calculated based on the time spent serving the previous bursty peer. Each regular peer schedules only one transfer at a time and refuse further request until the transfer is carried out. This guarantees that the regular peers in the swarm always "get back" the bandwidth that was taken away from them during the scheduled transfer time.

4.2.3 Data Distribution for Mobile Peer–to–Peer Networks Using Network Coding

This research topic examines the possibility of data distribution over multiple mobile platforms forming wireless peer–to–peer networks. State-of-the-art mobile networks are centralized and base station or access point oriented. Current developments work with device to device communication. We have realized that the advantage of network coding in this context would be that the source devices only need a minimal amount of knowledge about the sinks received packets and therefore only a minimal amount of feedback is required to ensure reliable data delivery.

In recent years the number of Mobile Ad-hoc NETwork (MANET)-oriented applications has increased. The research and development activity is mainly driven by the increasing spread of mobile devices such as smart-phones, PDAs and tablets, but also by the increasing demand for ad hoc media and content sharing services e.g. for the class-room, conference and office [Vision, 2012]. For these types of one-to-many services the traditional point-to-point data distribution paradigms provide a poor utilization of the available network resources. In both wired and wireless networks multicast provides a favorable solution, where data is sent along

branches of a multicast tree and thus consuming a minimal amount of resources. Although IP multicast provides suitable delivery semantics for certain applications and services in small scale MANETs, it suffers from one major drawback namely the lack of reliable data delivery mechanisms, which is also required for multimedia content distribution among mobile devices [Pedersen et al, 2010].

Reliable multicast is required by an increasing amount of services not only the content sharing services mentioned, but also in e.g. health care networks, military networks, and including a wide range of one-to-many services used in the Internet [Mankin et al, 1998]. However, providing reliable data transport in MANETs brings up several challenging problems due to the high mobility, unstable nature and typical resource constrained mobile devices e.g. in terms of energy, computational power, and memory. This is further complicated as the requirements posed by different reliable multicast applications typically cannot be supported by a single underlying protocol or single set of transport mechanisms. Different multicast applications have different characteristics and requirements. For example in terms of, error handling, delay sensitivity, support for multiple sources, scalability, impact on other traffic types, and so on [Mankin et al, 1998]. This differs from reliable unicast protocols where the requirements remain reasonably general, allowing the delivery semantics of the Transmission Control Protocol (TCP) protocol to support a wide range of applications. This diversity in requirements has resulted in a variety of proposed reliable multicast protocols. Most proposals rely on two key approaches namely Automatic Repeat-reQuest (ARQ) in which a source retransmits lost packets and Forward Error Correction (FEC) in which the source transmits redundant parity bits allowing for error recovery at the sinks. Pure ARQ, suffers from the so-called feedback implosion and exposure-problem. Furthermore, we can mention the stochastic supressing mechanisms as well. Whereas, pure FEC schemes cannot provide full reliability. To mitigate these issues Hybrid ARQ schemes combining ARQ and FEC have been proposed and investigated, and their performance is in general well understood. Although traditional ARQ and FEC schemes have proved very useful in the construction of efficient reliable multicast schemes, recent results indicate that further improvements may be achieved for MANETs via the combination of two emerging technologies namely; Cooperative Wireless Networks (CWNs) and Network Coding [Zhang et al, 2007a].

Both cooperative communication in wireless networks and network coding are new research areas, that have generated a significant amount of interest in recent years [Fitzek et al, 2006]. Cooperative wireless networks abandon the traditional client/server network structure to create a number of advantages for cooperating devices. It has been proven that cooperative wireless networks in some setups can be used to increase throughput, reduce delay, and reduce energy consumption for the cooperating nodes [Fitzek et al, 2006]. Network coding offers properties that support cooperative data distribution schemes by allowing re-coding of data flows at intermediate cooperating nodes. Furthermore, network coding offers a higher flexibility and potential gains over traditional end-to-end codes. Network coding changes the current paradigm in switching computer networks often referred to as store-and-forward or routing. In case of network coding, information flows are no longer considered atomic entities but mixed at nodes in the network. This mixing has been shown in theory to exhibit several

advantageous properties related to throughput, robustness, security, and complexity in communication networks [Fitzek et al, 2014]. In both areas a vast amount of the promising results have been verified through simulation and/or analysis, however issues arising from real implementations are still largely unexplored.

The problem of data dissemination in MANETs is a particularly interesting research topic. Assume that we are about to share some multimedia content, such as pictures or video files, with other users in the vicinity. This content may originate from a single source or from multiple devices in the network. The size of the data set to be shared can vary greatly depending on the scenario. The content is always divided into numerous packets due to the limitations of the underlying communication systems. The goal is to ensure reliable delivery for the content as a whole, and not only for the individual packets.

However, supporting reliability in MANETs is really challenging because of the high mobility and the unstable nature of the wireless channel. Therefore, a significant number of packets can be lost during transmission. To improve reliability in packet erasure networks, ARQ has been employed in unicast communications. On the other hand, ARQ is not always useful in multicast due to the feedback implosion problem, which occurs when too many network nodes are sending feedback at the same time.

Since the same data set should be delivered to all receivers, the broadcast nature of the wireless channel allows for an efficient delivery of innovative information. The existing reliable broadcasting protocols for mobile ad hoc networks [Pagani et al, 1999] generally focus on the delivery of individual packets, which is an inherent limitation of these protocols that results in significant control overhead. It is more efficient to consider the entire data set as a unit, and apply a coding scheme on top of it. With a rate-less erasure code (EC), a single coded packet can convey useful information about the whole data set. Thus it suffices for a receiver to retrieve any subset of coded packets of size slightly larger than the set of source packets.

Research activities have shown that network coding [Fragouli et al, 2006] has its clear advantages for mobile communication systems that make network coding well-suited for the data distribution.

The core idea behind network coding is to take several original packets and combine them into one coded packet, which has the length of one original packet. If a sender is combining N packets, the receiver needs to receive at least N coded packets successfully to decode the original packets. Each coded packet is holding the encoding vector to give the receiver the knowledge which packets have been combined.

The operations performed in a network coding system are depicted in Figure 4-6. The encoder generates linear combinations of the original data packets in the current generation. Addition and multiplication are performed over a finite field. Therefore, a linear combination of several packets will result in a packet having the same size as one of the original packets. With Random Linear Network Coding, the coding coefficients are selected at random. Note that any number of encoded packets can be generated for a given generation. The middle layer represents the wireless *channel*, where packet erasures may occur depending on the channel

conditions. The received encoded packets are passed to the *decoder*, which will be able to reconstruct the original data packets after receiving at least *g* linearly independent packets.

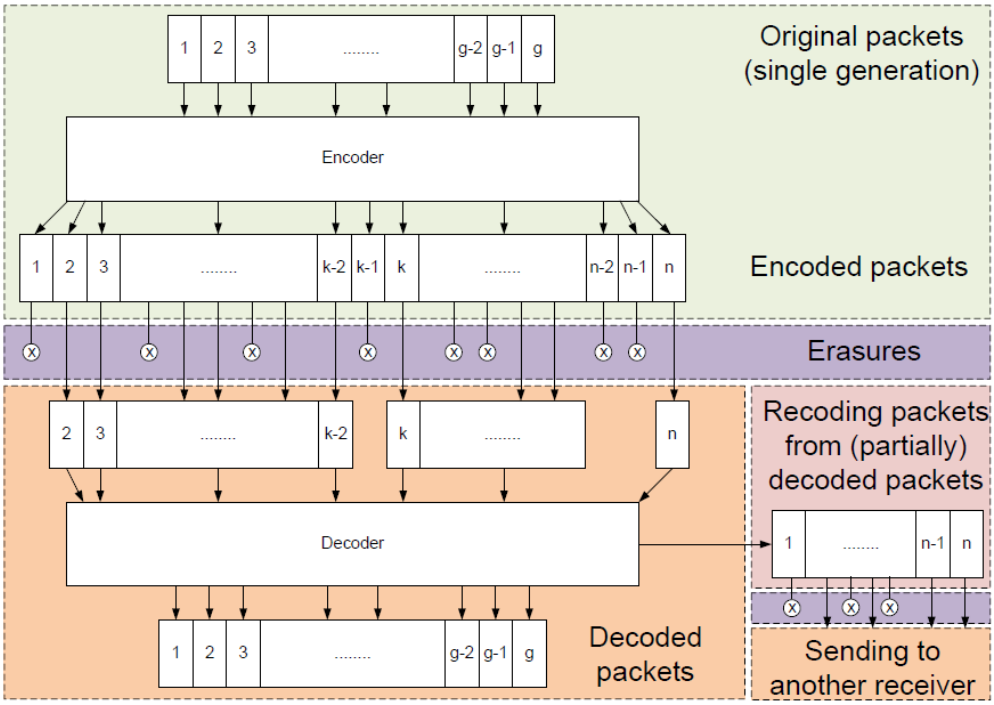

Figure 4-6 Overview of Network Coding [Fitzek et al, 2006]

Network coding can provide throughput improvements and a high degree of robustness in packet networks. Network coding allows any network node to recombine several input packets into one, instead of simply forwarding them. This research area was about to investigate and provide a method to distribute data in a wireless ad hoc network consisting of several mobile nodes [Vingelmann et al, 2011].

The approach was investigated from numerous aspects. Different types of networks in terms of mobility (static vs. dynamic) and topology (single-hop vs. multi-hop) are considered. There are several different scenarios considering the initial distribution and the flow of information that is being distributed in the network. In the simplest case, the entire data set is initially available on a single device that wants to distribute this content to its neighbors.

We investigated different schemes for data distribution in multi-hop ad-hoc networks via simulations. We developed a custom-built simulator that supports arbitrary network topologies and transmission schemes. With the help of this simulator environment we tested different scenarios and concluded measurement results. We have implemented the random linear network coding on various mobile platforms. We demonstrated that it is beneficial to use the binary finite field for random linear network coding calculations on mobile devices. We have evaluated the protocols designed within the simulator. The first scenario was a static, single-hop network where one mobile user has the multimedia content that is intended to be shared with others in the vicinity. The second scenario was where the information comes from an external source at a central entity (e.g. a base station in a cellular network), and the receivers try to cooperate in order to repair packet erasures (of the primary data flow) within

the cluster by exchanging missing packets with one another using a secondary network interface. We designed and implemented a protocol that minimizes the number of packets exchanged on the secondary network while maximizes the number of packet errors recovered on the primary network. Using network coding in a cooperative cluster proved to be an efficient way of realizing the packet exchange among the mobile devices that can simply broadcast recoded packets which most likely convey information that is useful to other receivers.

Our research efforts have also included the investigation of data distribution protocols for highly dynamic, multi-hop networks. A sufficient level of mobility was provided by a robot platform. We designed and built autonomous robots to carry several mobile devices in various environments. A key assumption was that all nodes are constantly moving, and they are connected in a multi-hop fashion where some devices can only be reached over multiple hops. The robots were meant to imitate human motion to some extent. We have worked out two scenarios and based on them measured and compared the different data distribution strategies. In the first scenario, initially one of the nodes has all the information that should have been transferred to all other devices. In the second scenario, initially each node had only a subset of the whole data set that should have been exchanged among all nodes in the network. The performance of several alternative strategies was compared using numerous metrics that showed the clear advance of network coding-based schemes.

We have developed an application for iOS platform that can disseminate video content in a single-hop network. We measured the performance of several alternative strategies for data distribution, and concluded that network coding can significantly reduce the completion time in the investigated scenarios. This also leads to relevant energy savings [Vingelmann et al, 2012].

4.3 Related Work

Since its inception, several studies have examined the performance of BitTorrent. [Xia et al, 2010] presents a survey of performance studies of BitTorrent from 2003 to 2008. Many of these studies also suggest modifications to BitTorrent's mechanisms to further improve its performance. Some research aim to improve the performance of BitTorrent by changing the way peers establish their connections to other peers while joining and thereafter while maintaining the connections [Zhang et al, 2007b] [Yamazaki et al, 2007]. Other works mostly focus on improving the piece exchange mechanism by promoting collaboration among peers using the underutilized download bandwidth, addressing the issue of free-riding [Wang et al, 2010] or suggesting improvements to the mechanisms [Ma et al, 2009].

The popularity of the BitTorrent appeared in many studies focusing on simulations. We tested several and finally used two different BitTorrent simulation frameworks [Bharambe et al, 2006]. The Peersim project [Montresor et al, 2009] is a popular general purpose P2P simulator created with Java, which covers the implementation of some popular protocols. BIT-SIM [Katsaros et al., 2009] is a modular and extensible simulator using OMNeT++ version 3. Ebit-Sim [Evangelista et al, 2011] is a BitTorrent simulation framework

implemented using OMNeT++ version 4 and INET, which uses an accurate simulation model based on the source code of mainstream BitTorrent clients.

Considering modeling BitTorrent, no prior research has investigated the performance of a network with storage-limited peers. A number of studies have modeled heterogeneous networks. However, none of them was directly suitable for our needs because the models only allow varying the bandwidth capacity of the peers. [Chow et al, 2009] claims to be the first paper modeling heterogeneous BitTorrent swarms with arbitrary number of peer classes. [Meulpolder et al, 2009] follows a fluid modeling approach based on differential equations to model a swarm where peers have arbitrary upload and download bandwidths. [Fan et al, 2009] uses multiple peer classes and analyzes different rate assignment strategies. [Liao et al, 2007] investigates how the performance of high bandwidth and low bandwidth peers compare and introduces a novel token-based scheme to analyze performance/fairness. The work presented by [Chang et al, 2011] quantifies the performance impact of network address translation (NAT) on BitTorrent by presenting a model with two peer classes (NAT peers and public peers).

Tribbler [Pouwelse et al, 2008] is a BitTorrent-based protocol, which uses helper nodes that can download content on request. This is somewhat similar to our proxy approach; however, the focus in Tribbler is on faster downloads, not on energy consumption as in our case.

4.3.1 Energy efficiency

So far the energy efficiency of BitTorrent, or peer-to-peer file downloading in general, has mainly been investigated from two different aspects: how to reduce the standby energy consumption of PC peers [Blackburn et al, 2009] and how to perform active content download energy-efficiently with a mobile peer [Nurminen, 2010] [Siekkinen et al, 2012]. In another track, the use of helper nodes to speed up BitTorrent downloads have shown promising results [Garbacki et al, 2006]. Our approach of using of memory-limited devices to help torrent downloads seems to be absent in prior work.

4.3.2 Mobile BitTorrent

We created the first BitTorrent client (SymTorrent) for mobile devices, which runs on Symbian OS. Soon after the release of SymTorrent, MobTorrent [Ekler et al, 2008a], a Java-based client was also released, which runs on feature phones with some limitations. Research related to the mobile usage of BitTorrent mostly concentrates on MANETs or wireless ad hoc networks [Sbai et al, 2008]. The focus of these works is different than ours, covering topics such as tuning of the peer discovery phase [Rajagopalan et al, 2006] or distributing content from one source to a potentially large number of destinations [Michiardi et al, 2007], but neglecting energy consumption.

4.3.3 Social peer-to-peer networks

The functioning of any peer-to-peer system relies on cooperation between the peers. In a social P2P system, communication occurs almost exclusively between nodes whose owners have a trustworthy social relationship. These systems are also known as darknets [Rogers et al, 2007] or friend-to-friend (F2F) systems [Li et al, 2006].

There are several P2P systems that use the concept of social networks in one way or another. In [Li et al, 2006b] an F2F system for scalable and durable storage was proposed. The Freenet project [Clarke et al, 2002] turns a social network into a privacy-preserving file sharing application with focus on efficient search [Sandberg, 2006]. Tribler [Pouwelse et al, 2008], another file-sharing application, allows the creation of social groups. Peers can recruit their friends from the group as download helpers to improve the performance. The helpers contribute their bandwidth twice, downloading from the source and then uploading to the friend that requested the help. In contrast, in our system there is no recruitment but we do take advantage of the fact that two friends are downloading the same content. None of the above work looks at the social network topology as the sole means of content distribution and does not clearly measure the performance benefits gained from using the social links.

The popular BitTorrent client, Vuze [Vuze], has added the friends feature. Users can allocate more upload bandwidth to their friends and easily share torrent links with them. The eMule client [Kulbak et al, 2010] has friend lists as well. Users can allocate upload slots to friends so they do not need to wait in the queue.

In [Galuba et al, 2010] the authors evaluate a socially-aware BitTorrent, in which peers connect both to the peers obtained from the trackers and to the peers belonging to the user's friends. The three core problems that they address are: (1) whether the social network topology alone can function as a scalable and efficient content distribution medium, (2) whether it is possible to take advantage of the social links for solving the freeriding problem without sacrificing performance and (3) what is the number of downloaders in the social network at which the benefit of using the social links becomes significant. They found that a hybrid solution in which peers download from both their friends and other peers obtained from the trackers has the highest robustness to freeriding, shortest download completion times and the most balanced upload bandwidth utilization.

4.4 Conclusions

MobTorrent is the first BitTorrent client for feature phones with limited resources. MobTorrent is a full-featured BitTorrent client, it supports both download and upload as well. During the research we discovered limitations related to feature phones which show that peer-to-peer solutions have different requirements than simple applications. Despite the limitations we were able to implement MobTorrent which is based on a general mobile BitTorrent engine that can adapt to the capabilities of the target device. We have demonstrated the portability of this engine by applying it on Android platform as well.

We have proposed a model to estimate energy consumption of MobTorrent. We have shown measurements related to MobTorrent performance, memory requirements and energy consumption. The measurements proved that the performance is convincing and MobTorrent is able to run on phones with limited resources and the resource requirement remains on a relatively constant level.

We have introduced a hybrid peer-to-peer based architecture for content distribution that supports mobile devices and decreases the load of the server. The solution can be used by

different type of service providers and it is also applicable in social networks. The main advantage of the architecture is that it also considers mobile phones and provides backend functionalities in case of low share ratio as well. For the content distribution a modified BitTorrent protocol have been applied.

The proposed architecture had practical results; it was implemented at Nokia Siemens Networks in the Swarm project. Swarm architecture requires less storage capacity, energy consumption and processing power on the service provider side comparing to a client-server solution, thus it can be provided even on lower prices.

The main advantage of this solution is the cost efficiency for service providers, because the produced traffic distributes in the network and significant part of the infrastructure and operation cost is handled by the Swarm client applications.

Regarding to the broadband routers, this work has both a practical side and a more theoretical side. On the practical side, it shows that using broadband routers as proxies for BitTorrent downloads is feasible and results in energy savings and user experience improvements via shorter download times. The concept can thus be deployed, and it would be rather easy to take into practical use because it is compatible with existing BitTorrent clients and does not harm the efficient operation of BitTorrent communities.

On the theoretical side, we have investigated how BitTorrent works on memory-limited devices. Only a relatively small number of pieces are required to achieve full upload utilization and reach good download speeds. Most of the available memory should be allocated to serving other peers. This memory allocation can be done statically using our analytical formulas or with an adaptive algorithm. The amount of data that is sent to the mobile device in one pass is another important parameter that influences the performance. In general, a larger chunk size can save energy but it may increase the download time.

Several aspects of the protocol have been investigated, and the results show that the worked out BurstTorrent approach has a lot of potential in heterogeneous content sharing networks which consist of both energy-limited and regular peers. BurstTorrent clearly outperforms standard BitTorrent in almost all cases in terms of the energy consumption of mobile peers, while the download time of regular peers is not affected significantly.

Finally, we have concluded that smartphones are capable of encoding and decoding data using random linear network coding even at high data rates. Furthermore, the benefits of network coding in terms of throughput and energy consumption are realistic in many scenarios.

4.5 Acknwedgements

The measurements provided in this chapter were made by Imre Kelényi and presented in his PhD thesis. Related to the topic of this chapter, efficient mobile platforms, there are 5 PhD theses worked out and successfully defended within the group under my supervision.

- Bertalan Forstner, *Semantic information retrieval in mobile Peer-to-Peer networks*, 2009.

- Péter Schönhofen, *Extracting document features to improve classification and clustering*, 2009.
- Péter Ekler, *Analysis and performance evaluation of mobile related social networks*, 2011.
- Imre Kelényi, *Energy efficient mobile peer-to-peer systems*, 2013.
- Péter Vingelmann, *Data Distribution with Network Coding in Device-to-Device Communication Scenarios*, 2013.

A part of this research activity was performed within an international collaboration with Aalborg, Massachusetts Institute of Technology, and Dresden Universities.

5 A Model-Driven Methodology for Supporting Multiple Mobile Platforms

This chapter utilizes the previously introduced results and provides a method for efficient application and service development. One of the key aspects of the method is the quality of the resulted software artifacts.

5.1 Introduction

The development of mobile applications has recently become an important field in the software industry. With the increasing number of different target devices, the need for reliable and efficient applications is also rapidly growing. Due to the diversity in mobile platforms, the identical applications must be independently developed for each platform. A further downside to the use of mobile devices is the limited power capabilities, namely battery life. Based on these factors, we identified the following issues to be addressed by a model-driven application development methodology:

- The rise in popularity of mobile devices calls for the designing and developing of efficient mobile applications for different mobile platforms. The goal is to design the same application once and generate the executable from the same common models for non-compatible target mobile platforms. This issue requires applying platform independent software modeling, in which all aspects of mobile applications (database, business rules, user interface, communication protocol, etc.) can be planned and manipulated. Next, we should apply different model processors, one for each target platform, to create executable applications.
- The generated mobile applications should apply energy efficient solutions to optimize battery power. These newly generated applications should be based on energy efficient software frameworks and these frameworks should be available for each target platform.
- In order to further support energy efficient solutions, it is necessary to utilize the capabilities of cloud-based solutions. Computation-intensive tasks can be outsourced into the cloud and the results made available to the mobile applications.

In order to achieve the above goals we build on our key research areas and results that we obtained during the last twenty years. The research areas that we apply are presented in the previous two sections, such as software modeling, model processing, and various methods to support mobile platforms. The motivation is to provide a model-driven methodology for the development of reliable, multi-platform, mobile applications. The realization of the goals is a modeling and model processing framework for different aspects of mobile applications in a platform-independent way, while integrating the use of cloud services and to provide automatisms that generate mobile application related artifacts for different platforms [Charaf, 2013a] [Charaf, 2013b].

5.2 Contributions

The architecture of the application generation process is depicted in Figure 2-1. The modeling of mobile applications and the processing of these models are performed in a framework. We use mobile, domain-specific languages to define the required application logic, data structure, communication and user interface. Platform-specific model processors are applied to generate the executable artifacts for different target mobile platforms. The generated code is based upon the previously assembled mobile, platform specific frameworks. These frameworks provide energy efficient solutions for mobile applications. Furthermore, some data processing or computationally intensive tasks are passed into the cloud to save the battery power of the mobile device.

DSLs address the connection points and commonalities of the most popular mobile platforms. These commonalities are the basis of further modeling and code generation methods. The main areas, covered by these domain-specific languages, are the user interface, static structure, business logic (dynamic behavior), database structure and communication protocol. Using these textual and visual languages, we are able to integrate the use of cloud services into the business logic.

For each target platform, a separate transformation should be realized since, at this step, we convert the platform independent models into platform specific executable code. The transformation expects the existence of the aforementioned frameworks and utilizes its methods. The generated source code is essentially a list of parameterized activities (commands) that certain functions of the mobile application should perform. This means that the core realization of the functions is not generated but utilized from mobile-platform specific frameworks, i.e., the generated code contains the correct function calls in an adequate order, and with appropriate parameters. The advantages of this solution are the followings:

- The modeler has an easier task. The model processors are simpler. The model processing is quicker. The generated source code is shorter and easier to read and understand.
- We use optimized mobile, platform-specific frameworks. These frameworks are prepared by senior engineers of the actual mobile platform and subject, e.g. energy efficiency. Furthermore, these frameworks are tested and can therefore be (ideally) reused in several applications.

We have been working on the modeling of different aspects of the mobile applications and processing these models to produce software artifacts more than ten years. There were two main focus points of this research activity. In the first wave, we concentrated on the Symbian [Harrison et al., 2004], Microsoft .NET Compact Framework (.NET CF) [Yao et al., 2004], Windows Phone 7 [WP] and Java 2 Micro Edition (J2ME) [Knudsen et al., 2005] platforms. The second wave of the research activities are focusing on the Android [Android], iOS [iOS] and Windows Phone 8 [Foley, 2012] [Rubino, 2012] platforms. Next sections discuss the solutions and results related to both of these waves. These results concentrate on two issues: (i) modeling the different aspects of the mobile applications and (ii) model processing, i.e. generating the software artifacts for different mobile platforms.

5.2.1 The First Wave for Supporting Multiple Mobile Platforms

Designing and developing applications for mobile devices require special attention, which is often a larger effort compared to the development for desktop computers and servers. This is because of the diversity of the increased number of mobile platforms and the different development style required by them.

Symbian [Harrison et al., 2004] used to be one of the most popular mobile platforms. There are several versions of Symbian OS. Symbian provides a robust architecture and API to support development. There are two important ways to develop applications for Symbian platform. The first choice is to use C++, which is the native language of the Symbian OS and the second option is to use Java, which runs on top of OS layer.

The **Microsoft .NET Compact Framework** (.NET CF) [Yao et al., 2004] is a subset of the full Microsoft .NET Framework [Thai et al, 2003]. The full .NET Framework was downscaled to fit resource-constrained devices without compromising user scenarios in such a way that the developers would experience enhanced performance with majority of the functionality at a reduced size. The most significant benefit is that the programming model for .NET Compact Framework devices is identical to that used by the developer using .NET to build applications for desktop PCs and servers.

Windows Phone is a series of mobile operating systems developed by Microsoft, and is the successor to the Windows Mobile platform. Windows Phone is primarily aimed at the consumer market rather than the enterprise market. Windows Phone was first launched in 2010. The latest release of Windows Phone is Windows Phone 8 and soon Windows 10, which has been available to consumers since 2012. With Windows Phone, Microsoft created a new user interface. Furthermore, the software is integrated with third party services and Microsoft services, and sets minimum requirements for the hardware on which it runs.

Windows Phone 7 was the first release of the Windows Phone mobile client operating system. Windows Phone 7 featured the new user interface, based upon Microsoft's Windows Phone design system, commonly referred to as Metro. The home screen, called the *Start screen*, is made up of *Live Tiles*, which have been the inspiration for the Windows 8 live tiles. Tiles are links to applications, features, functions and individual items. Tiles are dynamic and updated in real time, e.g., a tile could display a live update of the weather or the tile for an e-mail account could display the number of new e-mails.

The **Java 2 Micro Edition** (J2ME) [Knudsen et al., 2005] was also a popular platform for developing applications for mobile devices. J2ME is a smaller Version of Java 2 Standard Edition targeted towards consumer end embedded and small devices.

In the general purpose application development for mobile devices, there are several frequently appearing domains: user interface, static structure of the application (interfaces and classes), dynamic behavior (functions and function bodies), data structure, network communication and protocol definition languages.

We start the discussion with the modeling environments for the user interface development. There are different domain-specific languages (DSLs) worked out for this purpose, one for

each supported platform. We outline the model transformation for Symbian platform. Next, the user interface model porting possibilities are discussed. Then, we introduce a DSL describing protocols and network communication for mobile devices. Furthermore, we give a brief description of the code generator for .NET CF.

5.2.1.1 Modeling the different aspects of the mobile applications

Figure 5-1 and Figure 5-2 depict the metamodels for some mobile platforms. These metamodels illustrate the fundamental differences between the user control libraries of the target platforms. Although the difference between the naming conventions of the controls can be resolved by defining the correspondence between the appropriate metamodel elements, the conceptual differences cannot be handled in such a way. Note that several platform specific controls and features are modeled as attributes, thus they are not visible in the figures.

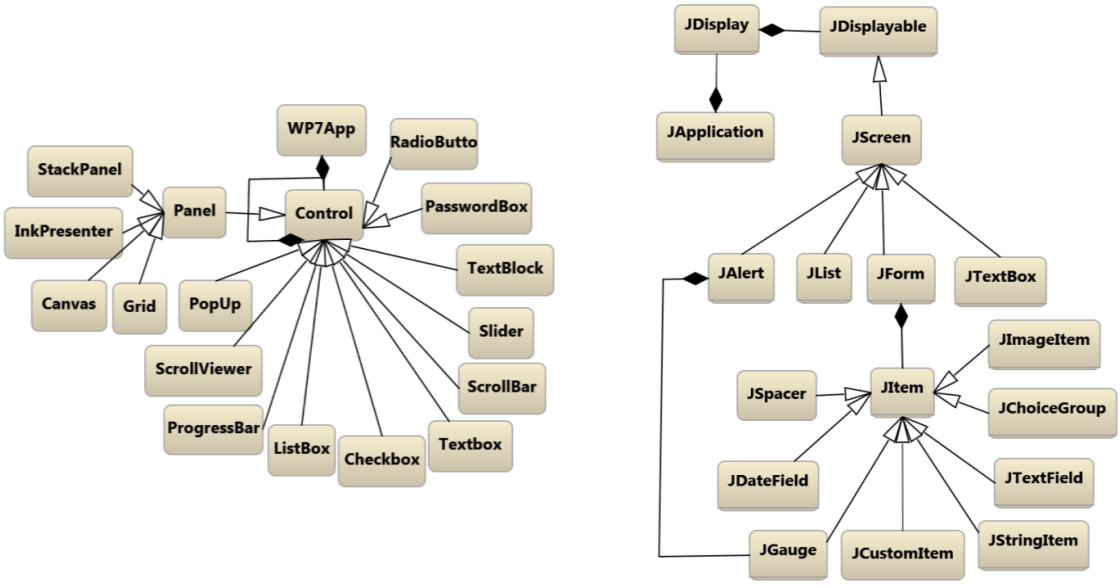

Figure 5-1 The user interface metamodels for Windows Phone 7 (WP7) and Java 2 Micro Edition platforms

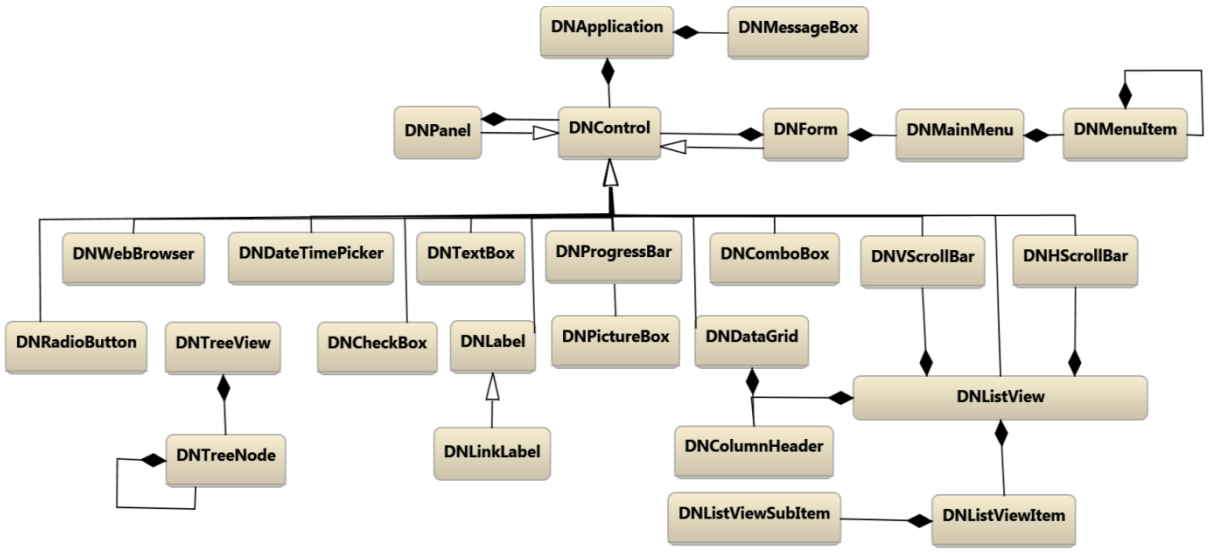

Figure 5-2 The user interface metamodels for .NET Compact Framework

Based on the metamodels, we can create user interface models. Example instance models for some of these DSLs are depicted in Figure 5-3. As we can observe, this is possible to define a concrete syntax for the appearance of a metamodel instance. The solution offers a plugin-based architecture, which means that each instance is attached to three components in a plugin, according to the Model/View/Controller pattern [Gamma et al, 1995]. These components specify the appearance and the behavior of the model item. The WYSIWG ("What You See Is What You Get") characteristic of the model presented in Figure 5-3 is facilitated by this architecture.

Figure 5-3 Example UI models for (a) Java, (b) Symbian, and (c) .NET CF platforms

5.2.1.2 Simplian Framework

Even tough Symbian was one of the most popular mobile platforms, Symbian-based software development is far more difficult and requires more specific skills than the development of desktop applications. This stems not only from the C++ characteristics, but also from the absence of easy-to-use integrated development tools.

We can mention, for example, the complicated memory management (for instance cleanup stack and two-phase object construction), the special exception handling (leave-mechanism), string- and array handling, or the unique aspects of resource-management. Also, developers must strictly take care of these uncommon circumstances, since ignoring them could lead to bugs that are hard to reveal.

A class library can help the programmer by covering all the recurring tasks deriving from the mentioned facts. The Simplian Framework, developed at BME AUT, provides an API for constructing the user interface of the application, and other simple means to bind data to the widgets, or send and receive them through different communication channels. Simplian also contains tools to generate the C++ code from a well-defined, platform-independent XML file, which can be constructed by the modeling tool [Aczél and Charaf, 2005].

We provide an example here, showing how the class library can help the user interface development. The GUI elements of Symbian are mostly designed to be constructed from resource files. Using resource files has admittedly many advantages, but it could be problematic if there is no flexible way to construct them dynamically from source code.

Considering an example of the class design in Symbian, the *CEikDialog* class can generate a multi-page form only from resource files. Counting the number of the pages contained by a form at runtime is possible only with several uncomfortable workarounds. After examining the class in depth, we have found that it has further properties that are not acceptable for model-based, generated software.

Designing the framework, we paid special attention to all the functions of the widgets to be accessible at runtime, because we decided to ignore the resource files. Therefore, we were not able to build on the existing *CEikDialog* class, thus, a new one was designed. The new element is a general form class with the following considerations: (i) It is easy and fast to construct a form and its widgets; (ii) The widgets have same look and feel as the characteristics specific to the executing operating system (for example the Nokia S60); (iii) Widgets are scalable for different screen sizes. (iv) The source code is easier to read and understand; (v) There is no need to maintain enumerations for the widgets, they are identified by variables; (vi) The widgets can be constructed and fully reconfigured at runtime, and (vii) the control is performed by a proprietary event handler system.

User Interface. The GUI widgets are usually divided into input and output elements. The classes for user input simplify the use of dialogs. It is possible to create multipage dialogs with input widgets. An elaborated mechanism checks the user input against the required circumstances. As S60 devices have a rather small screen, the different commands are assigned to menus, which can be activated with hardware softkeys. The class library for the input widgets implements a state machine that is triggered by event handlers. However, the transition structure can also be reconfigured at runtime. The source code is easier to read, therefore, code maintenance is also simpler.

Regarding the data presentation, we want to provide a wider variety of output widgets than S60 does. It is necessary to wrap the existing S60 lists and grids to be able to use them through the APIs of the class library. Furthermore, we elaborated new widgets that can show data tables, because the new devices with other user interfaces can support larger screens.

The APIs of the widgets are defined in a way that they can be simply connected to the form elements and the classes of the data access layer. It is guaranteed by the approach that, based on all these classes, the modeling tool can generate an event-driven, data flow-based source code that is easy to design and validate.

Data Binding. The goal of data binding is to connect the controlling widgets of the user interface to the data storage units, for instance, variables or database. This process can also be hand-coded by the programmer, but then this recurring task should be done to all widgets when loading and saving their value. This can be automated by the framework, hence it also eliminates some possibilities of failure that way. Designing the data binding architecture we focused on the following points:

- The programming interface should be as simple as possible (“set and forget”): once set, the binding synchronizes the elements automatically.

- Flexibility: the data binding module should handle different kinds of data sources. Binding to a database should be as easy as binding to a variable.
- Extensibility: the programmer can define custom types of data sources.
- Configurability: most settings can be overridden by the programmer.

The data sources bound to the controlling elements are wrapping classes to miscellaneous storage resources, for example data tables or data streams. These resources can be divided into two types. The first type can store only one single value or record, while the second type can handle sources with multiple records, for example a database table.

5.2.1.3 Challenges of Porting Mobile User Interface Models

Platform diversities make the porting process extremely difficult. For instance, the user interface controls of Windows Phone 7 (WP7) are flexible, since most of the controls are not restricted to contain only textual data, the content can be almost any other user interface control. The user interfaces of the other platforms (such as J2ME or .NET CF) are stricter. A given control can contain only a few predefined controls. In these cases, the generation process from WP7 to *strict* platforms is challenging. A further obvious difference is the number of the available controls. Numerous controls do not have any correspondent element in the other mobile platforms. For instance *TreeView* control exists in .NET Compact Framework, but not in Windows Phone 7. In these cases, when a corresponding model element is not available, we have to replace the control to be ported with other controls. For instance a *ListBox* is able to replace certain number of functions of the *TreeView*, however the exact same functions cannot be provided.

Another issue arises when although a control in the given platform has corresponding element in the other platform, but their behaviors are not the same. For instance, the *Menu* control of the .NET CF has the corresponding element in the WP7 metamodel, but their appearances are different. Figure 5-4 shows an example menu structure in .NET CF and its ported equivalent WP7 menu.

As depicted in Figure 5-4 the .NET CF is able to define two separated menus (*Menu* and *Exit*), while the WP7 defines only one, thus the second menu cannot be ported. Furthermore, the WP7 cannot define *SubMenu* controls.

Figure 5-4 Different menu concepts of the .NET CF and Windows Phone 7 platforms

In these cases the generator process has to replace the original user interface concept with a different one. For instance, the menu items may be ported as a *pivot* or *panorama* page in WP7 as it is illustrated in Figure 5-5.

Figure 5-5 The Windows Phone 7 pivot page is able to replace the .NET CF menu

5.2.1.4 Model Processing: Generating the Software Artifacts

The next component of the model-driven development solution is the model processor. Figure 5-6 outlines the code generation process for the Symbian platform. The model processor is a model transformation method defined on the basis of graph rewriting. The transformation takes the metamodel of the input and output models, the input model itself, a set of transformation rules and a control flow definition.

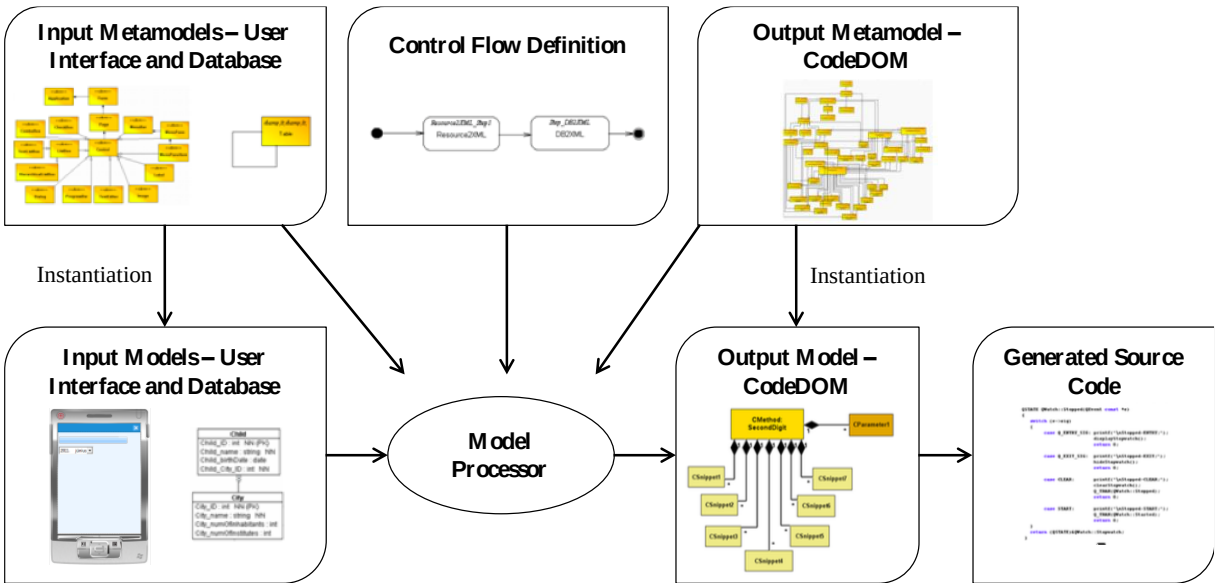

Figure 5-6 The transformation process for Symbian-based mobile devices

In the current case the metamodel of the input model is that depicted in Figure 5-1b. Furthermore, the Symbian platform related model processor supports data binding and database generation, thus, a database metamodel is also provided as an input. The model processing solution uses the Microsoft CodeDOM technology [Thai et al, 2003] for code generation. The CodeDOM consists of classes representing the syntactic elements of the .NET languages, such as C# and managed C++. The .NET framework has a code generator for these elements that generates syntactically correct code. Our modeling and model processing

solution has a CodeDOM metamodel, and there is a built in code generator for the instance models. Thus, the metamodel of the output model is the CodeDOM metamodel.

The input model of the transformation is a Symbian user interface model accompanied with a database model. The output model of the transformation is a CodeDOM model from which we automatically generate the appropriate C++ code.

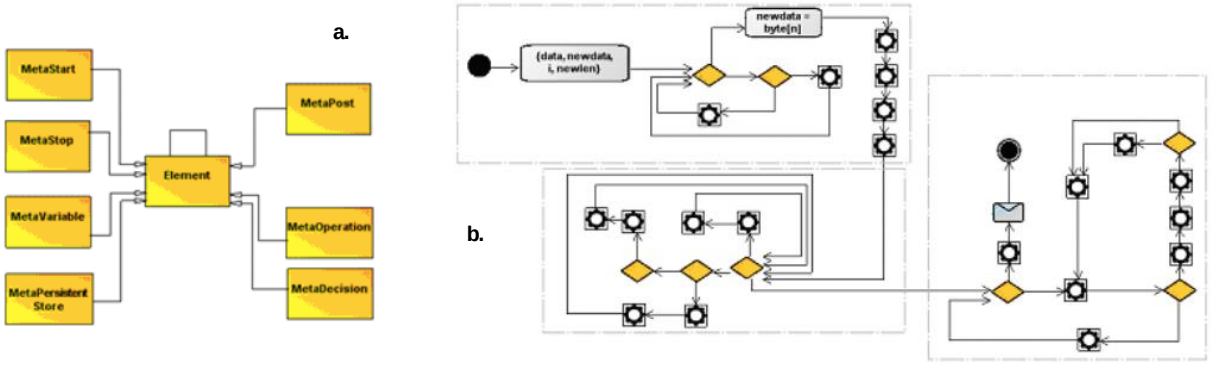

Figure 5-7 (a) *MobilCom* metamodel and (b) an example model describing protocol

The *MobilCom* metamodel that defines a language for network communication and an example for its instance model is depicted in Figure 5-7.

Figure 5-8 introduces the overview of the transformation process that generates source code from mobile protocol models (*MobilCom* models). Source code generation is again based on the CodeDOM technology. For each element, the transformation builds a separate class which inherits from the base class. Each class has a *Process* method with a *byte[]* and an *int* parameter and a *byte[]* return type. Data exchange between the different elements is based on byte arrays. This method is intended to define the logic of the protocol.

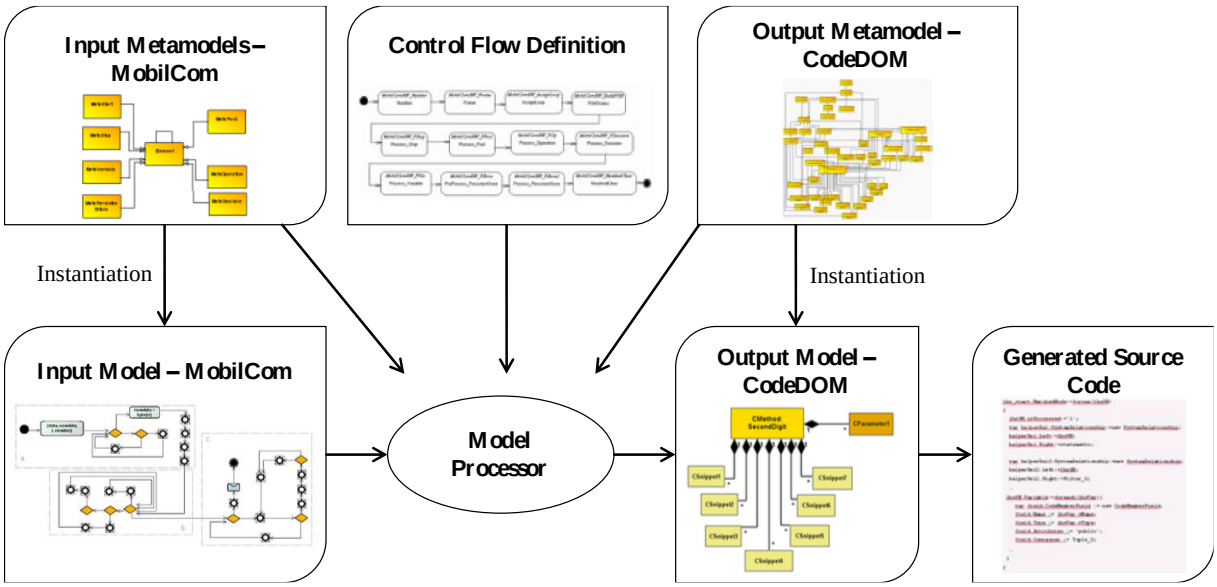

Figure 5-8 Code generation process for .NET CF-based mobile devices

We do not introduce the model processors related to all aspects of different mobile platforms, but we note that the main difference between the two transformations (Symbian and .NET CF

related transformations) is in the resource model (user interface model) processing. For the .NET CF platform certain properties of the user controls are generated based on the attributes of the resource model. The rewriting rules do not use these attribute values during the generation for Symbian platform, because the controls are placed strictly one below other.

The presented approach facilitates to use visual languages to define user interface, database and communication models to define the different aspects of mobile applications. The solution provides model processors to generate the platform-specific source code. The solution has been realized with domain-specific language engineering and graph rewriting-based model transformation.

5.2.2 The Second Wave for Supporting Multiple Mobile Platforms

The second wave efforts of the model-driven methodology for supporting multiple mobile platforms target the Android [Android] [Ekler et al, 2012], iOS [iOS] [Hillegass et al, 2012] and Windows Phone 8 [Foley, 2012] [Rubino, 2012] platforms.

Android is a Linux-based operating system that is designed primarily for touchscreen mobile devices, e.g. smartphones and tablet devices. Android was introduced in 2007, and the first Android-based phones reached the market in late 2008.

Android is open source, since Google releases its source code. This open source code and the licensing construct make it possible to freely modify the software and distribute by different participants of the market: developers, device manufacturers and/or mobile network operators. Currently, Android has the largest community of developers writing applications. These apps extend the functionality of devices. Based on different surveys, in 2012, there were more than 600,000 apps available for Android platform, and the estimated number of applications downloaded from Google Play, which is the primary Android app store, was more than 22 billion. At the end of 2012, Android had a worldwide smartphone market share up to 72% [Vision, 2012].

These factors resulted that Android is the most widely used smartphone platform, overtaking Symbian at the end of 2010. Android is an appropriate choice for those companies who require a low-cost, customizable, lightweight operating system for high tech devices.

iOS is a mobile operating system developed and distributed by Apple. First time it was released in 2007 for the iPhone and iPod touch platforms. The iOS has been extended to support further Apple devices such as the iPad and Apple TV. At the end of 2012, there were more than 700,000 iOS applications in Apple's App Store. The number of downloads was more than 30 billion. At the end of 2012, iOS had a 21% share of the smartphone mobile operating system units. Only Google's Android was in front of iOS [Vision, 2012].

The user interface of iOS is based on the concept of direct manipulation, using multi-touch gestures. Interface control elements consist of sliders, switches, and buttons. Interaction with the OS includes gestures such as swipe, tap, pinch, and reverse pinch, all of which have specific definitions within the context of the iOS operating system and its multi-touch interface. Internal accelerometers are used by some applications to respond to shaking the device or rotating it in three dimensions.

iOS is derived from OS X, i.e. iOS is Apple's mobile version of the OS X operating system used on Apple computers. In iOS, there are four abstraction layers: the Core OS layer, the Core Services layer, the Media layer, and the Cocoa Touch layer. iOS currently runs on iPhone, iPad, iPod touch and Apple TV.

Windows Phone 8 is the second generation of the Windows Phone mobile operating system. Windows Phone 8 replaces its CE-based (Windows Embedded Compact) architecture used on Windows Phone 7 devices with the Windows NT kernel found on many Windows 8 components. Therefore, Windows Phone 8 is the first mobile OS from Microsoft to use the Windows NT kernel, which is the same kernel that runs Windows 8. The operating system adds improved file system, drivers, network stack, security components, media and graphics support.

Windows 10 is an operating system for 32 and 64-bit PCs and also runs on the ARM platform for smaller tablets and smartphones. Furthermore, Windows 10 is going to run on phones, it is the new version of Windows Phone.

5.2.2.1 Modeling the Mobile Applications

As we have already discussed developing software for different mobile devices requires more and more time and work investment because of the diversity of the increased number of mobile platforms. The two most important development areas for mobile platforms are the web application and the native mobile application development. Our solution focuses on the native application development. Figure 2-1 shows that the generated native source code is based upon the platform-specific frameworks.

Instead of discussing all aspects of mobile platforms that the methodology support, and therefore be able to introduce only the beginning of each, we decided to select one area of the covered field and provide more details about it. The selected area is the *rapid prototyping of REST-based communication channels*. The methods supporting this area, i.e. the way in which we model some aspects of the mobile platforms and how the model processors generate the source code, are the same for several different aspects of mobile platforms as well.

Depending on the expectations of the customer, the mobile application developers need to develop either only the mobile clients for an existing server application or the server applications supporting the clients as well. The commonality is that there is one server-side application, but the mobile clients should be developed for multiple target platforms concurrently. Therefore, the communication layer should be developed for each platform. However, the concrete implementations of the layers on the different platforms show significant similarity, it is not possible to reuse the source code between the platforms because of the different languages and runtime environments. These facts are motivating issues to apply platform-independent modeling and model processing to support these types of challenges.

Mobile applications usually apply a REST-based (Representational State Transfer) [Richardson et al, 2007] communication channel when it is about data exchange between the different devices. In REST, the operations of the server application can be accessed with the

help of a properly formatted HTTP request. The parameters of the request may be encoded either into the request URL itself, or into the body of the HTTP request. In the latter case, the formatting of the parameters may be arbitrary, though, the two most often applied serialization mechanisms for the parameter objects are XML and JSON. The server application responds to such a request with a HTTP response that contains the response parameters in its body (again, usually as XML or JSON). Even though we know if the request and response body is formatted as XML or JSON, their concrete schema may be arbitrary and is not tied to such strict rules as in case of e.g. SOAP. Consequently, the serialization and deserialization procedures as well as the URL generators and interpreters should be developed on both the client and the server sides. Furthermore, in case of the client application, the same development must be performed in case of each targeted mobile platform as well.

As an initial step, we focus on REST APIs exclusively, and automate this error prone, monotonous coding for the client side. We have elaborated a modeling language that is able to describe server-side operations, the applied data types and the way of parameter serialization. We have also prepared the generators that automate the creation of the client-side communication layer based on the models.

There are various forms to represent a modeling language. For practical reasons and to speed up the initial development we have chosen to realize this language based on an already existing programming language, the C#. The idea is to define the communication API with the help of C# interfaces and to generate the concrete implementations based on these interface definitions for various platforms. Utilizing C# as a base language has the benefit that the syntactic and semantic verification of the language can be performed using existing C# compilers, and by compiling the interfaces into a .NET assembly, we can easily traverse and interpret the models by traversing the assembly using reflection.

With the help of interfaces, we can precisely define the methods that can be called on the server application, and we can also define the possible data types that we can pass to these methods or we can expect from these methods as a result value. The way how such a method call is performed, how the parameters are serialized when passing them to the methods and how the result values are deserialized on a successful call can be customized with the help of .NET attributes. The code generators processing these custom domain-specific models (these interfaces) will discover the attributes attached to the elements of an interface definition and modify the code generation accordingly.

Of course, we may not write arbitrary interfaces (allowed by C#) and may not use arbitrary attributes, just the ones interpreted by the code generator. These rules follow from the C#-based custom domain-specific language worked out for this purpose.

5.2.2.2 Model Processing: Generating the Software Artifacts

Having the features of a network service, including its methods and the applied data types, already defined, the next step is that based on the interfaces generate executable source code which is able to perform the communication with the server component. There are various solutions that can be utilized when it is about code generation, we have chosen the Microsoft T4 (Text Template Transformation Toolkit). T4 is a mixture of static texts and procedural

code: the static text is simply printed into the output while the procedural code is executed and it may result in additional texts to be printed into the output. Recall that, the interface definitions are compiled into a .NET assembly that can be loaded and traversed using reflection afterwards. The T4 templates we write also work on the reflected content of the assemblies.

In the first round we have targeting two mobile platforms: Windows Phone 8 (C#) and Android (Java). Therefore, we need to prepare two different T4 templates for the two platforms. In case of Windows Phone, we expect the data transfer objects be represented by C# classes, the fields of the DTO entities should be represented as .NET properties, and the generated classes should also support some kind of change notification about changes in the properties.

Appendix A provides a case study for supporting multiple mobile platforms.

5.2.2.3 Summary

In conclusion, we claim that our objective targets one of the most pressing problems in the area of mobile software development, originating from the diversity of mobile platforms. To address this problem, we have provided a modeling language family for mobile applications and developed optimal frameworks for all platforms which support the code generated from the models. The result is a complete methodology that allows the designing of mobile applications and generates working applications for each major mobile platform. The main points of the methodology is presented in Figure 2-1.

The actual wave is the Internet of Things (IoT). To realize the IoT vision of bringing technology to people anytime, anywhere, with any device, service, or application, not only must users be aware of their devices' capabilities but the "things" must also be aware of users' activities, preferences, and context. We have successfully transformed and applied the above introduced methods to support IoT based solution developments. The *SensorHUB* concept, introduced in the applications section of this thesis, provides a framework and tools to support application domain specific service development.

5.3 Related Work

This section introduces the most known cross-platform development frameworks and solutions. We summarize their capabilities and compare their capabilities and achievements with our methodology.

PhoneGap [PhoneGap] is a mobile development framework enabling developers to build applications for mobile devices using standards-based web technologies, such as JavaScript, HTML5 and CSS3, instead of device-specific languages such as Java, Objective-C or C#. The resulting applications are hybrid, meaning that they are neither truly native, because all layout rendering is done via web views instead of the platform's native UI framework, nor purely web-based, because they are not just web apps. Applications are packaged as apps for distribution and have access to native device APIs. It is possible to mix native and hybrid code snippets.

Earlier versions of PhoneGap required a developer making iOS apps to have an Apple computer, a developer making Windows Phone apps to have a computer running Windows, and so on. Currently, the *PhoneGap Build* service allows a programmer to upload his source code to a *cloud compiler* that generates apps for every supported platform.

Appcelerator Titanium [Appcelerator] is a platform that, similarly to PhoneGap, using web technologies supports developing mobile, tablet and desktop applications. The Appcelerator Titanium framework is available since 2008. Appcelerator Titanium Mobile framework allows web developers to apply existing skills to create native applications for iPhone and Android. However, in the case of Appcelerator Titanium Mobile, developers should not only be familiar with web technologies and JavaScript syntax, but they also have to learn the Titanium API, which is different from familiar web frameworks such as jQuery.

All application source code gets deployed to the mobile device where it is interpreted. Being interpreted means that some errors in the source code will not be caught before the program runs. Program loading takes longer than it does for programs developed with the native SDKs, as the interpreter and all required libraries must be loaded before interpreting the source code on the device can begin.

At the end of 2012, there were more than 30,000 applications shipped to the app stores built with Titanium. Appcelerator also offers cloud-based services for packaging, testing and distributing software applications developed on the Titanium platform.

Adobe Integrated Runtime, also referred as Adobe AIR [Adobe AIR], is a cross-platform run-time system provided by Adobe Systems. The goal of Adobe AIR is to support building Rich Internet applications using Adobe Flash, Apache Flex (formerly Adobe Flex), HTML, and Ajax, that can be run as desktop applications or on mobile devices. The runtime supports desktop applications on Windows, Mac OS and some mobile operating systems, e.g. BlackBerry Tablet OS, iOS and Android. Originally Linux was also a supported platform.

Appcelerator Titanium is often compared to Adobe AIR for developing desktop applications for Windows, Mac and also Linux. Titanium enables developers to use standard Web technologies such as HTML, CSS and JavaScript to develop applications that can be deployed to multiple platforms, including the desktop, the browser or the mobile device. Titanium, unlike traditional Web applications, which are limited to operating within the browser, enables developers to create applications that are able to read and write local data on the desktop and interact with the operating system.

Xamarin [Xamarin] is a company created by the engineers that created Mono [Mono] MonoTouch and Mono for Android, which are cross-platform implementations of the Common Language Infrastructure (CLI) and Common Language Specifications (Microsoft .NET).

Xamarin.Mobile is a library that exposes a single set of APIs for accessing common mobile device functionality across iOS, Android, and Windows platforms. The solution allows to use C# programing language and with the same code support all the mentioned platforms. Xamarin.Mobile currently abstracts the contacts, camera, and geo-location APIs across iOS,

Android and Windows platforms. In the future it will include notifications and accelerometer services.

Comparing our methodology to PhoneGap, Appcelerator, Xamarin.Mobile and other multi mobile platform solutions, we can say that the goal is similar but not exactly the same. Available solutions target to produce the final executable files, i.e. the applications that are ready to use, that can be downloaded and installed. This approach is quite comfortable from both the end users and the developers point of view. But, we can imagine that these types of applications are limited to certain functions. Automatically generated applications can contain only those functions that have the appropriate implementation or support in the mobile platform-specific libraries, in the supporting SDKs or APIs. In contrary, the goal of our solution is to accelerate the development and not to eliminate the native programming. We use software modeling to design different aspects of the mobile applications and generate some part of the source code for different mobile platforms. Our approach supports and often requires further development activities after the source code generation, e.g., to integrate the generated source code into the already existing source code, or to extend the functionalities with platform-specific native code. We still believe that each software application requires some human contribution on the programming level. The goal is to cut down the required time to complete the tasks, to effectively support development efforts, but not to fully eliminate programming.

Further difference is that the presented multi-mobile platform solutions are providing hybrid applications, i.e. the applications are partly web apps and partly they are based on platform-specific libraries. Our solution produces truly native applications, therefore usually they are providing better performance, and usually it is easier to perform their testing and management.

5.4 Conclusions

This chapter has provided a model-driven methodology for developing multiplatform mobile applications. We have integrated the four main research areas that establish the structure of such a solution. Namely, we have applied the research results from the field of software modeling and model processing, mobile platforms, distributed systems and cloud services, as well as data technologies. These areas have a significant history in our research group, and we are currently actively working on them.

We have introduced the two waves of our model-driven solution for developing energy efficient mobile applications for multiple mobile platforms. This approach increases both the efficiency of mobile application development and the quality of the resulting software artifacts. This is achieved by providing a mobile, platform-independent, high-abstraction level environment for mobile application design. We support it with innovative, mobile domain-specific languages and effective model processing technologies.

Appendix A, Appendix B, and Appendix C introduce three case studies, which jointly apply the results of the main research areas. The case studies highlight the relevance of the research groups that pertain to several software areas. We can conclude that the strength of both research and development groups are based on the multidisciplinary knowledge. These types

of groups are able to work out effective solutions for a variety of problems in the software area.

6 Application of the Results

Model-based approaches are widely recognized as a potential way of increasing productivity in software engineering. Model-based development is driven by model transformations that attempt to bridge the semantic gaps between high-level models and lower level languages. There are several industry related requirements for researching the ways in which model transformation can become efficient, highly-configurable and really appropriate part of software development processes.

The thesis has introduced a methodology that applying model-driven solutions facilitates the efficient multiplatform mobile application development. The resulted mobile applications are based on energy efficient mobile platform specific libraries. The key techniques applied by the approach have been introduced. These techniques are (i) domain-specific modeling and model processing, (ii) mobile platform related energy efficiency solutions, and (iii) the cloud computing capabilities. Appendix B, C, and D underpin the applicability of the introduced mobile application development approach.

We believe that the role of the mobile devices and the need for native platform specific mobile applications are continuously growing, therefore, the relevance of the solutions, like the presented one, is high.

This section introduces several applications and application areas, where the results of the thesis have been successfully utilized. Furthermore, three long term research and development projects are also introduced, where the related topics have been actively researched and applied.

6.1 Software Applications and Tools Developed within the Scope of the Research Activities

This chapter summarizes the key aspects of the Visual Modeling and Transformation System, next discusses a model-driven application development method to support the IBM smart city concept, furthermore, introduces various mobile and network coding based solutions, and finally, an Internet of Things (IoT) related direction.

6.1.1 Visual Modeling and Transformation System

We have already established that domain-specific visual languages play an essential role in software engineering. By illustrating the problem set in a graphical way, these languages permit to raise the level of abstraction and help in defining the steps of the software lifecycle. The increasing popularity of domain-specific languages requires applications that are capable of visualizing these languages and offer a user-friendly way to edit the models interactively. At first glance it seems simple to create a modeling framework for this purpose; however, there are many additional requirements to support.

The *diversity* between the needs of different domains is huge. Not only the list of available modeling items can be different, the appearance of the modeling toolbar must be customizable, but it is also a common need to support custom user actions (e.g. mouse hover selects to control, or it means switching to edit mode). The framework must be flexible and it must allow customizing almost each and every framework service.

The framework must be capable of supporting various *storage* types (e.g. database, file-based/memory-based model repositories). There are several well-known model definition formats such as XMI, or GXL. For a generic purpose modeling framework, it is clearly not enough to support only one of these. Moreover, storages are important due to the performance as well.

This is not enough to create a framework that can only edit and visualize models, unless the models are used for documentation purposes only. There is a need to *process* the models: create new models, generate source code from them, or evaluate/execute the model definitions. Without supporting dynamic modeling features, models would be static, uncomfortable definitions only. Therefore, model traversing features or other model processing methods must be part of the modeling framework, or at least modeling tools must offer an easy way to apply these services on models via an external component.

The *performance* of the modeling framework is also essential. It is usual to have models with a few dozens of model items; however, there are domains, where thousands or even millions of model items have to be handled. Visualizing this amount of items is rarely required, because these models are too complex to display them, but it is common that the model processor must compute algorithms even for such large models in a reasonable time.

Although, several modeling frameworks exist, we have found that none of these tools fulfill all the mentioned requirements. Usually, there are two main problems with these tools: (i) they focus on an aspect of modeling and model processing and not on the whole process, or (ii) they prefer either customizability or performance (usually the two oppositional feature), but they rarely allow the user to choose.

Our research team has analyzed existing modeling frameworks and summarized our experiences including the experience gained from implementing the previous version of our framework, Visual Modeling and Transformation System (VMTS) [VMTS]. We have found that it is possible to create a solution, which is highly customizable and still fast. Modular structure of the framework allows us to support various domains and to store the models in different storage types. Moreover, the user can decide at *run-time* how important customizability and performance are. The presented solution can visualize optional domain-specific language models and it can handle millions of model items, if required. This section introduces the structure of the VMTS solution. It is important to emphasize that the presented approach is not specific to our system, but it uses solutions adoptable in any other modeling framework.

Visual Modeling and Transformation System addresses the above requirements, namely diversity, various storage types, model processors and performance. Moreover, VMTS allows fine tuning performance and customizability. In order to achieve these aims, we applied a

component-based architecture, where the well-defined interfaces allow replacing any of the components. The framework provides built-in services to support generation of these components.

Recall that Visual Modeling and Transformation System is a graph-based metamodeling system. Because of the metamodeling support, we can create models not only for predefined modeling languages, but we can define new modeling languages as well.

The architecture of the VMTS framework is depicted in Figure 6-1. The bottom most level is the Data Repository that represents the different data persisting stores that can be connected to the system. Above repositories, the VMTS Exchange Framework (VXF) and VMTS Modeling Framework (VMF) can be found. VXF and VMF form together the VMTS Data Interface (VDI). These two layers (VXF and VMF) are used to store the data in the repositories and to store the model data in memory during editing. At the top level of the architecture, different components (VMTS Applications) can be found; these components use the VDI to manipulate the data.

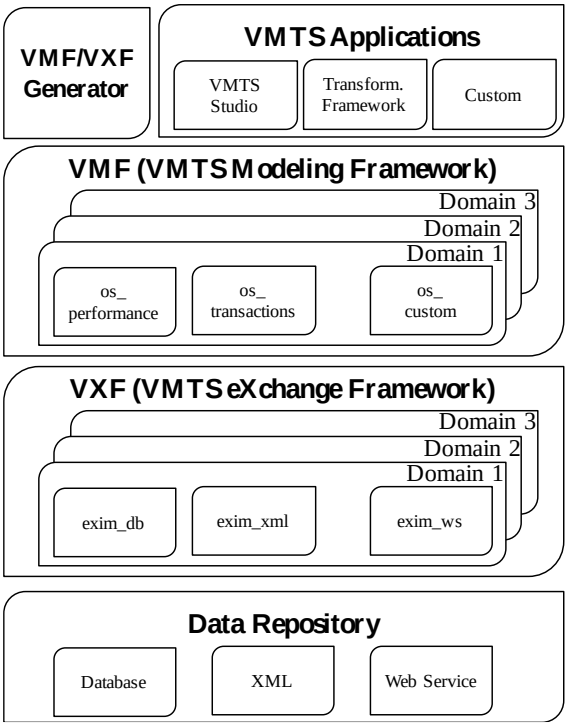

Figure 6-1 Architecture of VMTS

One of the most important user requests was to support multiple views of models and to provide extensibility so that different models can be visualized in a customizable way. Different views are required to be able to focus simultaneously on different aspects of models, while customization is needed to support domain-specific languages. The Document-View architecture is an appropriate choice to satisfy this requirement. For each model element, VMTS associates exactly one Document and an arbitrary number of Views. To facilitate the customization of the appearance, we have designed a flexible plugin system. Each plugin is attached to a metamodel with the help of the unique identifier of the metamodel. However,

metamodels can have more than one plugins attached, the user can select the plugin to use at run-time.

Our research team has been working on building the tool since 2003. We have already developed three different versions of VMTS framework. Each of the versions provided more facilitates, covered higher application area, and made the usability and user experience better. During the design, we have taken into consideration the requests from our industrial partners, the knowledge of other modeling frameworks and the conclusions of workshops, where VMTS has been presented e.g. [Graph Transformation Contest, 2007] [Grabats Tool Contest, 2008].

We have successfully applied the modeling and model processing capabilities of the VMTS framework in several research and development software projects. Some of these projects are introduced in Appendix B, Appendix C and Appendix D. In the case study presented in Appendix D, we show the efficiency of model-driven development that is supported by modeling and model processing environments like VMTS. Next section also introduces the capabilities of model-based solutions and the role of the modeling and model processing frameworks in current application development processes.

6.1.2 Model-Driven Application Development to Support the Smart City Concept – The IBM Smart City Project

The goal of the project was to provide a solution, which facilitates that entitled city inhabitants can report failures (traffic problems, electricity problems, etc.), disasters and other issues related to the city infrastructure. Supported channels / clients are (i) mobile client (SMS channel), (ii) web client (web service channel) and (iii) desktop client (web service channel).

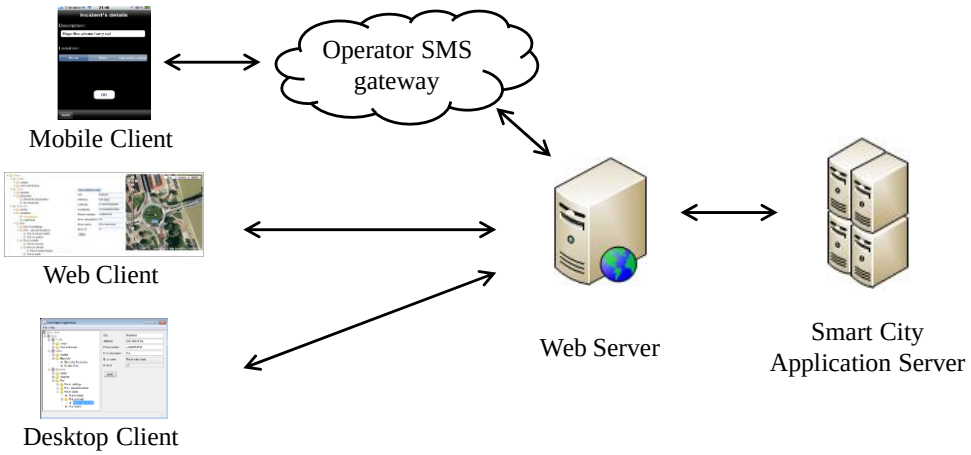

Figure 6-2 Architecture of the Smart City solution

The reported incidents are collected by a central server that generates the appropriate tasks for the infrastructure maintainers. The solution provides three clients (mobile, web, and desktop), and a visual modeling tool that facilitates the editing of the incident menu structure. The output of the menu structure definition is an XML file that is imported and used by the clients.

This makes possible that the same XML file drives all of the clients. The architecture of the solution is presented in the Figure 6-2.

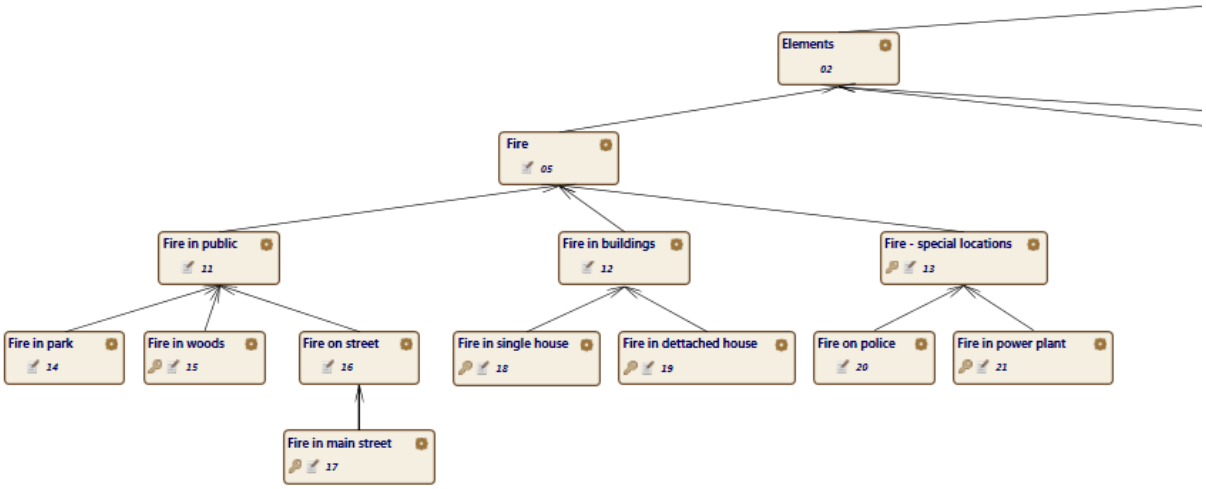

Figure 6-3 Example part of an incident tree within the VMTS

The domain-specific visual language that facilitates to define the menu structure of the incident tree is created within the VMTS environment. An example part of an incident tree model is depicted in Figure 6-3.

The client applications make possible to submit the unique failure code, user ID, phone number, PIN code of the user (if required), geographical position, user provided address, timestamp of the report, and pictures (optionally).

Screenshots of the three mobile clients (Java ME, iOS and Android mobile clients) are shown in Figure 6-4. The desktop and web clients are presented in Figure 6-5.

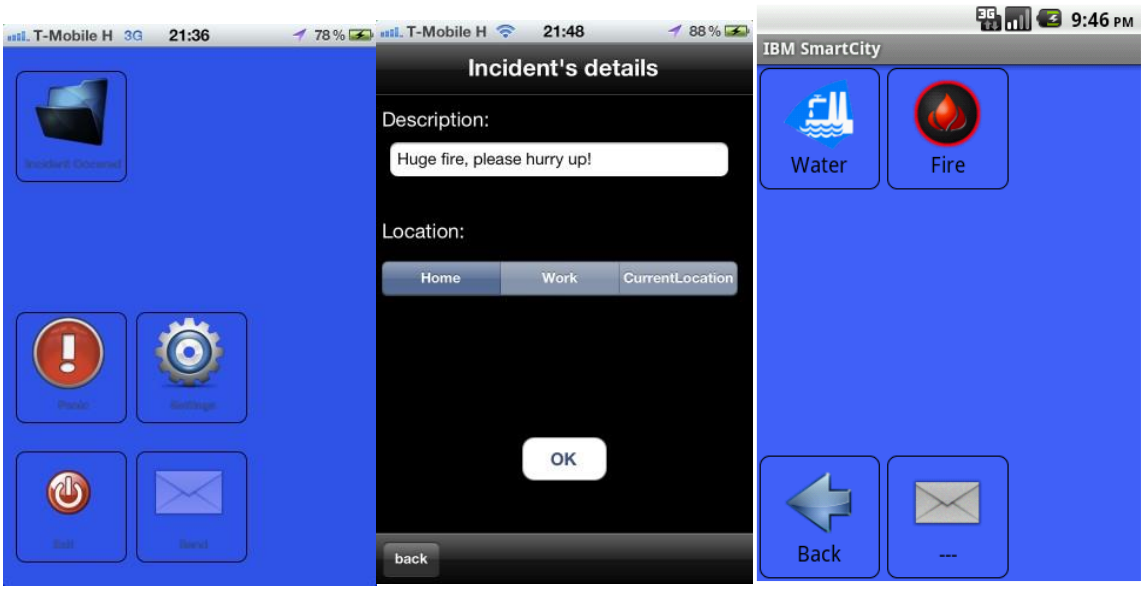

Figure 6-4 Smart City mobile clients

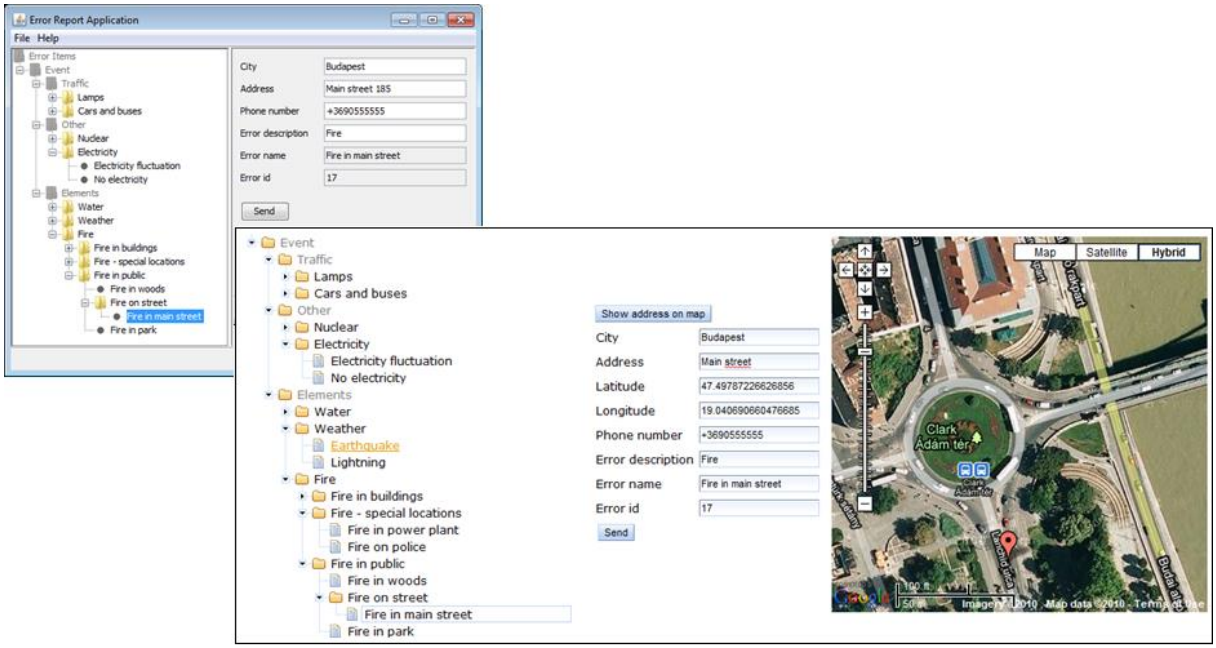

Figure 6-5 Smart City desktop and web clients

6.1.3 Mobile Solutions

Mobile peer-to-peer technology is a natural demand fueled by the appearance of smartphones on the market. Our research group did pioneering work in this area. *Symella* [Molnár et al, 2007], the first *Gnutella* client for Symbian OS, has been downloaded by more than 400,000 users since its first public release in the summer of 2005. *SymTorrrent* [Kelényi et al, 2011] is the first *BitTorrent* client for mobile phones. The first free public version was available in 2006, as of writing the software has been downloaded by about 500,000 clients.

At the beginning, the low computing performance and limited storage capacity of the mobile devices prevented to run fully functional P2P clients on them till the last couple of years. With the spread of smartphones, this situation has changed. Our primary goal was a Gnutella client for handheld devices that adapts to the characteristics of the mobile environment, still capable of connecting to clients running on any other platform. Moreover, we designed it modular to be extensible in order to implement and test the results of the latest researches and protocol extensions such as *SemPeer*. We selected the Symbian OS as the mobile platform to develop for, as it is the most mature and most popular mobile operating system available of its kind. The application was named *Symella*, after the connection of the words *Symbian* and *Gnutella*.

A mobile related social network was implemented with Nokia Siemens Networks in the *Phonebookmark* project. We have participated in a peer-to-peer research project at Nokia Research Center, where we analyzed the behavior of *BitTorrent* technology in low end mobile environment. In order to involve mainstream phones into P2P networks, we have developed a *BitTorrent* client named *MobTorrent* for Java ME platform [Ekler et al, 2008a] [Ekler et al, 2011]. The original goal was to examine whether mainstream phones are able to run such complex applications. The experiment has met the expectations, and *MobTorrent* became a suitable for communicate with the *BitTorrent* network. We made *MobTorrent* publicly

available as an open source project at Budapest University of Technology and Economics. The more than 10,000 downloads make it also possible to conduct measurements to analyze the behavior of mobile phones as members in a large peer-to-peer network. We emphasize that based on feedbacks *MobTorrent* operates well in real environment.

We examined a content sharing architecture that is applicable in social networks and decreases the load of the server. We took part in the implementation of this architecture at Nokia Research Center and Nokia Siemens Networks. Since mobile phones are key elements in mobile-related social networks, in our research we focused on involving them in the content distribution as well. This *BitTorrent* based hybrid solutions can be applied in several situations in social networks, e.g. distributing videos or whole photo albums between friends.

The research results provide mechanisms and protocols for real-life problems and actual use-cases. Since the majority of the research was carried out in cooperation with Nokia Research Center and Nokia Siemens Networks, it was a prerequisite that the results could be utilized in real-life use-cases. *BurstTorrent* have fully functional protocol implementation, which were tested in real networks.

SymTorrent is the world's first mobile *BitTorrent* client. *SymTorrent* was released as an open source application. Nowadays it is still being used by several thousands of users with Symbian based mobile devices.

The *BurstTorrent*, protocol was implemented as a simulation, but the phenomenon on which it is based on was observed, validated and measured using *SymTorrent* in real-life networks. Furthermore, in cooperation with Nokia, we have published an international patent on the mechanism [Nurminen et al, 2010].

A sloppy Distributed Hash Tables (DHT) mechanism was implemented in a mobile client, resulting in the first mobile Distributed Hash Tables client. This was also released as an open source project which was reused by other research groups, including the University of Stirling, Oulu University and Aalborg University. The sloppy DHT mechanism was also published as an international patent in [Nurminen et al, 2009].

6.1.4 Network Coding-Driven Solutions

In Section 2.4, we have seen that network coding is a promising paradigm providing several benefits both in networks and applications. The technology and the results of the field have been successfully applied for different areas. Such examples are for example (i) the tight relation of security and network because the data that is transmitted is coded at the sender, and thus appears to be "random" to an adversary, (ii) user cooperation in ad hoc networks and user formed groups also offers a long list of potential benefits, (iii) distributed storage system solutions, and (iv) synchronized multimedia streaming [Pahlevani et al, 2014].

We have successfully worked on the latter two areas and now summarize the relevant results here.

Network Coded Distributed Storage

One of the primary challenges of present day distributed storage technologies is the reliable management of stored data. In order to increase data availability and reliability, files are

distributed across several storage devices and an erasure code is applied. Since devices can loose network connectivity with a given probability, there is a dynamic recovery mechanism in place. In the case of a node failure a new recovery node gets connected to a given subset of the operating nodes and receives a part of the stored data. Random linear network coding (RLNC) can be suitable for such scenarios given its ability to recode with partial data and is the main subject of analysis in this work.

We have investigated data survival as a function of topology and communication overhead, defined by the number of connections and the number of transmitted packets to the recovery node, respectively. A sufficient condition for infinite longevity of the stored data is derived with respect to the number of nodes used for storage and the number of packets to be exchanged in the case of node failures. We have provided a comparison using experimental results that shows that RLNC can be up to 50% more effective than traditional erasure codes. These results support constructing effective low cost distributed storage architectures.

Reliable distributed storage has been one of the driving forces behind most online services in the last decade. It has also played a key role in the creation of entire new fields such as cloud computing and big data. Many traditional distributed storage systems that are employed in controlled, observable and predictable scenarios, use replication. For example, the widely used Apache Hadoop File System (HDFS) uses 3-way replication by default [Apache Hadoop, 2015]. Windows Azure Storage was one of the first large services to make use of erasure codes [Huang et al, 2012b]. Facebook employed an extended version of HDFS-RAID [Weiyan, 2014] that introduces Locally Repairable Codes for storing rarely accessed cold data. An evolution of this called HDFS-Xorbas [Sathiamoorthy et al, 2013] was also considered. Google has stated that Colossus, the successor to the Google File System [Ghemawat et al, 2003] will also make use of a Reed-Solomon code.

On the other hand, in distributed storage systems that lack a central entity to direct the repair process and for which the exact system state is hard to observe and predict, these traditional codes have proved less effective [Fitzek et al, 2014]. [Sipos et al, 2015] has shown using simulation data, that it can outperform replication-based storage and Reed-Solomon codes even in traditional centrally controlled systems if the amount of storage and repair traffic is limited. These include general P2P storage, such as mobile and vehicular storage clouds and sensor networks. These systems behave in a more dynamic way, nodes leave and join the system regularly, and therefore it is crucial to limit the transmission cost associated with maintaining data integrity. In this paper we advocate the use of random linear network coding, which is better suited for this dynamic scenario.

We have proposed a model and shown that if the values of certain parameters that define the storage system are chosen correctly, data integrity can be guaranteed. Next, we have defined the constraints to choosing the appropriate values. Finally, we have shown the most cost-effective sets of values and compared RLNC with other erasure codes using results from simulations. In summary, the main contribution of the area is that the analytical results can be used almost directly to create cost-effective, reliable storage systems.

Synchronized Multimedia Streaming

The idea was to stream multimedia content from a single source to multiple receivers with direct or multi-hop connections to the source. First we analyzed existing solutions for video streaming on the iPhone that use point-to-point architectures. After acknowledging their limitations, we have proposed a solution based on network coding to efficiently and reliably deliver the multimedia content to many devices in a synchronized manner. Then we have introduced an application that implements this technique on the iPhone. We also worked out a testbed, which consists of 16 iPod Touch devices to showcase the capabilities of our application [Vingelmann et al, 2011].

Multimedia content distribution has received a lot of attention lately in the mobile world. New ways to convey multimedia content to mobile devices are discussed after the recent failure of DVB-H and DVBM. Besides the bare technology there is also the question how mobile users are looking at multimedia content. So far the main architecture was designed such that the overlay network with its highly centralized architecture is providing the content, and the mobile users are consuming it. But more and more users are starting to generate and collect their own content that they would like to distribute among each other in local area networks.

We investigated the possibility of sending multimedia content from one device to many devices in closer proximity. Earlier the Mobile Device Group of Aalborg University has shown that it is possible to share photos and audio files among mobile devices even across different platforms. Therefore, the next step was address mobile video as the next logical step.

Synchronized video playback can be used among friends to show their latest videos to each other. Exchanging music videos is especially interesting at social events if everybody can play them at the same time. Another fascinating application is for home entertainment: we can deploy a simple server that broadcasts a live video stream (e.g. a sporting event) that is accessible on every mobile device in the household.

We have applied network coding in order to address the channel characteristics of wireless networks and the limited energy of mobile devices. In this area we have introduced a way to disseminate multimedia content in a synchronized manner. We have proposed a method based on network coding to efficiently deliver data from a single source to many receivers. An application running on Apple iPhone/iPod Touch devices has been presented to show the feasibility of this approach. The first commercial implementation of this technology can be found at [Steinwurf, 2015].

6.1.5 Internet of Things Based Solutions

A pillar of the Future Internet, the Internet of Things, will comprise many billions of Internet-connected physical objects or "things" that can sense, communicate, compute, and potentially actuate, as well as have intelligence, multimodal interfaces, physical/virtual identities and attributes. The IoT is an enabler and often driver to many application domains including production lines, supply chain management, transportation and logistics, aerospace, and automotive.

A world is saturated with "things" that form diverse and heterogeneous networks with overlapping capabilities in massively distributed IoT-based systems, therefore it is important to efficiently utilize resources, including power efficiency and sensor data based capabilities.

6.1.5.1 The *SensorHUB* Framework

We have worked out the *SensorHUB* [SensorHUB] [Lengyel et al, 2015b] framework that utilizes the state of the art open source technologies and provides a unified tool chain for IoT-related application and service development. *SensorHUB* is both a method and an environment to support IoT-related application and service development, furthermore, it emphasizes and supports the data monetization approach, i.e. provides a method to define data views on top of different data sources and analyzed data. In this way, the framework is a novel approach compared to the available frameworks, development environments and methods. The framework is available in a Platform as a Service (PaaS) model and has been applied for the vehicle, health, production lines and smart city domains. We discuss some implementations on top of the *SensorHUB* framework, which verified the architecture of the *SensorHUB*, for example the *VehicleICT* platform that supports connected car domain related application and service development.

A further advantage of the *SensorHUB* is that it makes it possible to develop and reutilize domain-specific software blocks, e.g. components for the health or the vehicle domain that are developed once and built into different applications. The framework makes these available by default and provides various features to support developers working in this field.

To realize the IoT vision of bringing technology to people anytime, anywhere, with any device, service, or application, not only must users be aware of their devices' capabilities but the "things" must also be aware of users' activities, preferences, and context. The *SensorHUB* concept provides a framework and tools to support application domain specific service development.

The architecture of the *SensorHUB* concept is depicted in Figure 6-6. The whole system contains the following areas:

1. sensors, data collection, local processing, client side visualization and data transmission (bottom left)
2. cloud based backend with big data analysis and management (bottom right)
3. domain specific software components (middle)
4. applications, services, business intelligence reports, dashboards (top)

Sensors cover different domains: health, smart city, vehicle, production line, weather and others areas. Local processing and data transmission makes up a local platform, which performs core services, i.e. data collection, data aggregation, visualization, secure communication, and data transmission. This component also provides information as a local service interface for different applications.

The cloud component provides the historical data storage, big data management, domain specific data analysis, extract-transform-load (ETL) mechanisms and data query interface. Its architecture was designed specifically for cloud deployments, although it can also be deployed on conventional server instances. In the core, there is a Microservice Repository, which holds the implementation of the different services. The most notable domain-agnostic services are the data ingestion service and the general querying service. Among the domain-specific services are the push notification service, which is applicable in all domains that have smartphones on the client side, and the proximity alert service, which can be used to determine if the user is located inside a noteworthy area and is extremely useful in the vehicular domain.

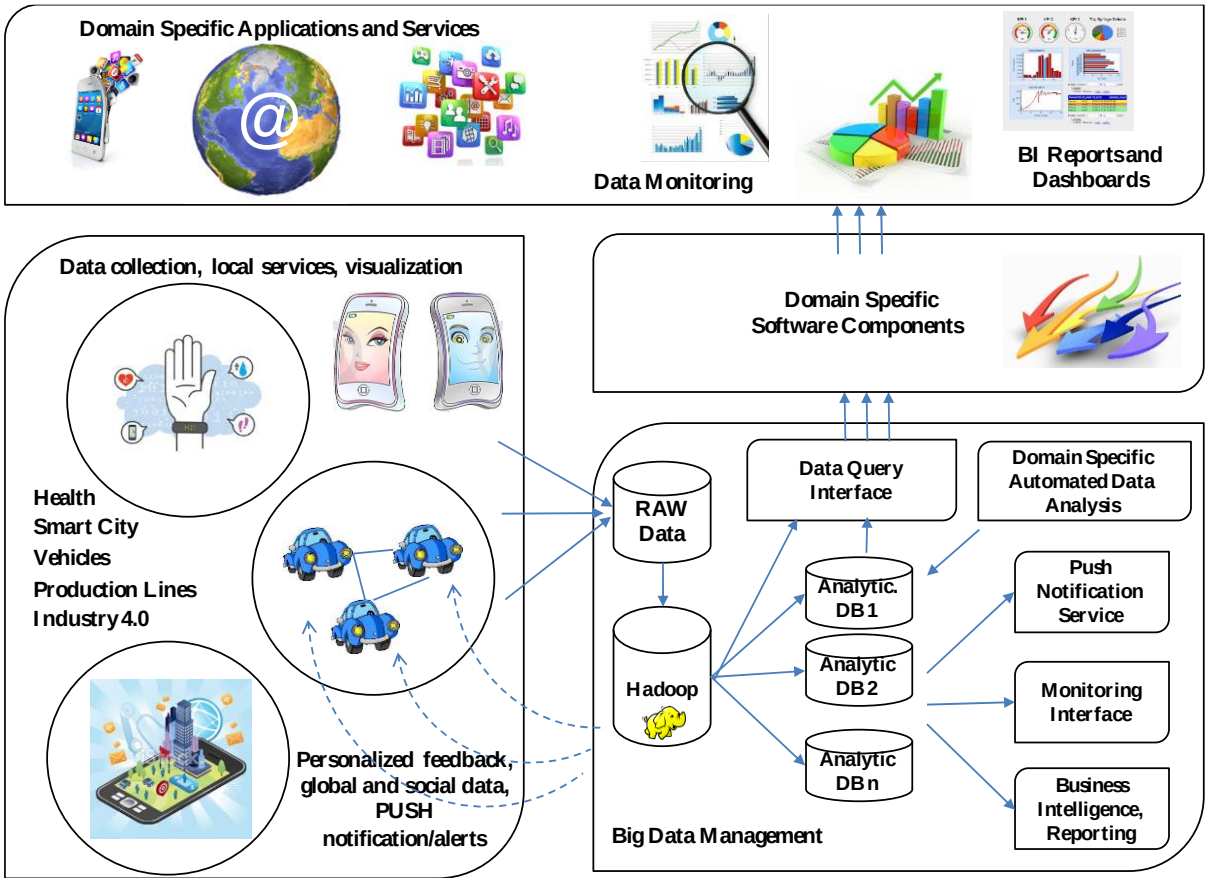

Figure 6-6 The *SensorHUB* architecture

The fourth layer comprises applications that implement the specific user-facing functionalities. From this point of view, it might seem that those are the components designed in the last phase, but in fact, the applications are what drive the design of the *SensorHUB* framework's architecture. These data-driven applications, independently of their purpose, eventually face the same problems repeatedly. Without the *SensorHUB* framework, applications would have to find a way to collect their data (client-side code on the sensors), to store them reliably in large amounts and in a scalable way (ingesting the data into a database with all the related difficulties), to transform them into a format that makes it possible to access them either for analytic purposes or present them on a dashboard (the contemporary problem of utilizing Big Data) and also to act on them in time, when the information is still

relevant (building a stream processing pipeline). Solving these problems is not at all trivial, and can account for the majority of the development effort if done one-by-one for every different application. The main purpose of the *SensorHUB* framework is to function as a platform for these applications, providing the implementation of the previously described tasks, so the application developers can focus on the domain-specific problems they intend to solve. The implementation includes client-side software components, which make it possible to collect sensor data with ease; a data processing pipeline that aides tasks from the data ingestion to the visualization of the data in an efficient, scalable way; and also several domain-specific components, which are not as commonly needed as the data processing pipeline itself, but are also reusable in different applications.

6.1.5.2 The *VehicleICT* Platform

The *VehicleICT* platform [VehicleICT] [Ekler et al, 2015] [Lengyel et al, 2015a] is an implementation on top of the *SensorHUB* framework targeting the vehicle domain. The implementation of the *VehicleICT* platform helped to distill the architecture of the *SensorHUB*. The *VehicleICT* platform utilizes the capabilities of the *SensorHUB* and provides a vehicle domain related layer with several reusable components and features. This means that the *VehicleICT* platform itself can be considered a test environment that verifies the different aspects of the *SensorHUB* framework.

The idea behind the *VehicleICT* platform was to identify a reasonably rich set of functionalities that typical connected car applications need, and then to implement and test these functionalities, and finally offer them as building blocks in a centralized manner. The approach enables application developers to focus on their domain-related logic. By using the building blocks, application development becomes more efficient and leads to more stable software artefacts.

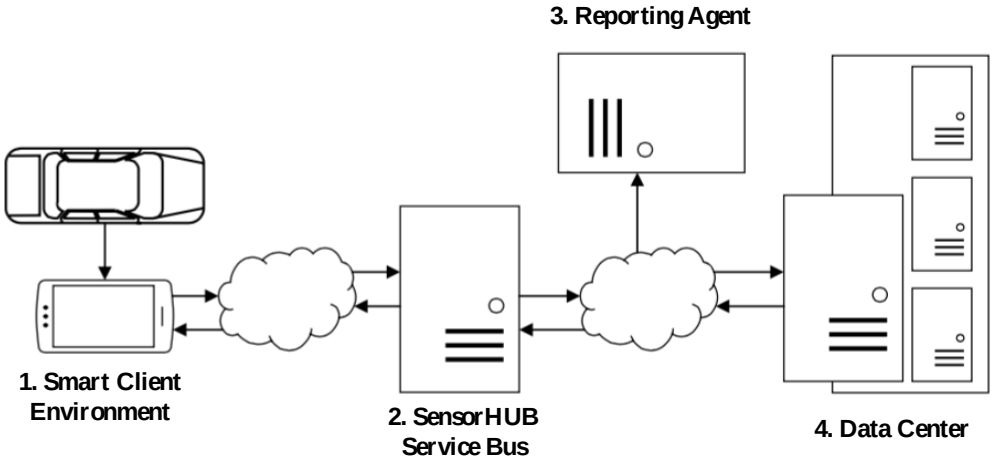

Figure 6-7 The *VehicleICT* architecture

Applications and services in the connected car domain can be divided into three separate parts: (i) the sensors, (ii) the local processing and visualization, and (iii) the background processing and analytics. The *VehicleICT* platform meets developers' needs in all three of aforementioned areas [VehicleICT]. Figure 6-7 introduces the architecture of the platform.

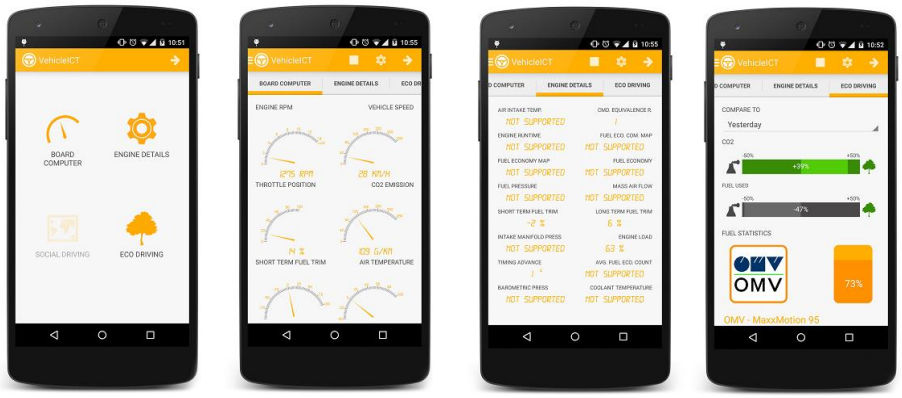

Figure 6-8 *VehicleICT* proof of concept smartphone application screenshots

Figure 6-8 depicts screenshots from the proof of concept mobile application of the *VehicleICT* platform. The first screen is a dashboard for further navigation. On the second screen, the Board Computer can be found, which contains the most frequently used indicators like vehicle speed, engine RPM and ambient temperature. This screen displays the current values on an interface designed to be accessible during driving. The third screen contains the Engine Details view, which presents all available vehicle information (except those on the previous screen), in a simple form, more suitable for diagnostics. The last screen is the Eco Driving view, which displays fuel consumption and CO_2 emission related information, utilizing the services provided by the infrastructure.

The *SensorHUB* concept continuously evolves due to its usage in both R&D and industrial projects. The concept has been defined after the first few similar IoT projects driven by the close requirements and solutions of the different projects. This is a natural process in the software industry that having solved the same or similar task for more than twice, we are about to work out a solution that can be utilized in different projects only by configuring the general components. The incremental and iterative development of the framework is driven by both the introduction of the new IoT domains and also by the evolving end user and corporate requirements targeting the data processing methods, reporting and data monetization ways.

VehicleICT was the first project, where both the client and server parts of the *SensorHUB* framework have been utilized. In certain cases, based on the actual conditions and requirements, we apply only a part of the whole framework, i.e. either the data collector sensor area (client side) or the backend component with extensive reporting.

Within the frame of two EIT Climate-KIC [EIT Climate-KIC] projects we utilize primarily the sensors related client part and the data upload components of the framework. These Climate-KIC projects are referred as URBMOBI and SOLSUN. In addition, in the SOLSUN project we are developing several web based and reporting components on top of the *SensorHUB* platform. The experience shows that the framework increases both the ease of development of the certain features and also the quality of the resulted software components. Furthermore, the extension of the services and the maintenance of the source code and the components of the solutions are also supported by the framework methods. In summary, the

framework with integrated solutions (data management, analysis, reporting, push notification), moves the development activities to a higher abstraction level and provides an up-to-date professional environment for the developer teams.

6.2 Research Projects Utilizing the Results

The following sections introduce three research and development (R&D) projects. These projects utilized several results introduced in this thesis. In each of the following project I have significant role in project management and development directions.

6.2.1 Mobile Innovation Centre

The Mobile Innovation Centre (MIC) was a project within the NKTH Asbóth Oszkár Program. The overall goal of the project was to improve the domestic research, development and innovation, furthermore, to support the local and regional industrial and economic development. The Mobile Innovation Centre performed its activity between 2005 and 2009. The goals of the Mobile Innovation Centre were as follows:

- Support of research and development of high-speed mobile and wireless communication technologies.
- Subserving initiation of modern mobile and wireless communication technologies and network services.
- Supporting the development and practical application of the newest mobile and wireless communication technologies and services.
- Promoting close research and development co-operations of universities and industrial parties.

The BME Mobile Innovation Centre basically performed its 4 year professional activity according to the originally appointed aims, meanwhile it was also continuously adapted to the dynamic progression of related specialties. Accordingly, for the sake of efficiency the 16 starting projects continued their activities in 8 so called integrated projects after the first 2 years [MIC Final Report, 2009].

An important result of the MIC's professional work was that apart from the original consortium several domestic small and medium enterprises were involved in professional co-operations and ensured them efficient R&D support.

Beyond this, the MIC has started an inward project in order to attain professional competences that can give the base for the further work of the Centre. The professional activity of the MIC is organized in three R&D programs: (1) Mobile Radio Technologies, (2) Integration and management of heterogeneous mobile networks, and (3) Mobile Development of Services and Applications.

I was the Service Development Director of the whole project and the Project Director of R&D Program 3. As a Service Development Director, I was working on the management and organization of service-development activity of the Centre. During the 4 years, under my direction the following important results were achieved within this project.

6.2.1.1 Integrated Project 3.1 Service and Application Development for Mobile Systems

The objectives of the project were the followings:

- Development of an intelligent client and gateway based remote monitoring system to enable cost effective remote monitoring of industrial objects based on mobile technologies.
- Research, design and development of a context-aware framework, research on Web Services and enhancement of the web accessibility of the online SZTAKI dictionary.
- Investigation of different location methods, building up algorithms and applications, based on the most important radio networks (Bluetooth, WLAN, 3G and RFID).

The objectives of the R&D results could be summarized in a tool chain that supports the multi-platform development of SCADA systems:

- Making the tool Visual Modeling and Transformation System suitable for modeling mobile environment ("VMTS Mobile Toolkit").
- Modeling the most popular mobile platforms in order to support applications using graphical user interfaces and network communications.
- Creating source code generators.
- Creating a general simulation environment.

The results can be divided into two categories. Firstly, the theoretical results related to the development of novel technologies, secondly, the set of the practical results. In summary, the project team has worked on location-based, content management, remote controlling, multiplatform applications and application development frameworks.

We were continuously collaborated with several representatives of Magyar Telekom (Hungarian T-Com), Pannon Inc., GKI Inc., Montana and Oracle Hungary Ltd. on content management areas. We were actively participating in two working groups of W3C Mobile Web Initiative and the Mobile and Multimedia Cluster.

There was a continuous collaboration between the Department of Automation and Applied Informatics the Department of Telecommunications (both from BME) and the Computer and Automation Research Institute (from the Hungarian Academy of Sciences). We have also been cooperating with Infoware Zrt. in the field of remote monitoring systems.

During the research and development we had cooperation with important industrial partners, thus we could use their support and experiences to develop the framework further. The most important partners include the T-Group, Nokia, Nokia Siemens Networks and we had common tenders with companies like ITWare.

6.2.1.2 Integrated Project 3.2 Investigation of User Behavior

During the interacts with intelligent environment and other devices the cell phone is handled as a special interface that can connect to given communication points according to the situation and problem. In aspect of this development the testing possibilities of the cellular

phone as an interaction interface and possible various context-dependent solutions seem to be very important. Within this examination line the prototype of cell phone-based presentation controller was elaborated. The weblog analyzer software package can also be highlighted in this project. This system enables retrieval and presentation of important data from large sized weblogs.

6.2.1.3 Summary

We can state that both of the integrated projects drew the attention of industrial partners. This is proven by the fact that our project results generated novel R&D projects with industrial partners. Of course, the research related publication activity should be also emphasized, because several conference and journal articles were published based on the practical results.

6.2.2 BME Innovation and Knowledge Centre of Information Technology – BME(IT)^2

The strategic objective of the BME Innovation and Knowledge Centre of Information Technology was to continuously keep both the competences of Budapest University of Technology and Economics (BME) and its industrial partners, furthermore to increase the competitiveness of the region [BME(IT)^2]. The IT^2 performed its main activities between 2007 and 2009.

The results of the four R&D programs are utilized in the exploitation of four major fields and in the related development activities. Within the Research & Development Programs I was the head of the program *Development methodology and framework*, and within the application-development directions I was the project director of the e-Document project.

We have worked out a development methodology based on multi-layer and multi-dimensional metamodels and a development framework based on this methodology. The framework has a modular structure and can be extended with domain-specific modules. This allows application development for a specific domain efficient.

Developer frameworks can be supplemented using application specific plug-in modules. Document management tasks can be implemented with a plug-in module that over and above storing the documents and archiving them directly or as meta data, manages access rights, automatically forwards documents and reply documents to the appropriate recipients, allows for prescribing and tracing the information-flow that documents realize, as well as by means of elaborating the option of mobile document management. The goal was to develop software components that provide services for these products, and moreover to formulate additional editing possibilities for generated electronic documents. We worked out an application interface that allows for the articulation of requirements at the user level, and coordinates the system of corporate processes and documents accompanying production in a flexible manner which adapts to any possible change.

The results of the Knowledge Centre had a direct impact on the university teachers and students as well. By taking part in the work of the Knowledge Centre, BSc, MSc and PhD

students gained first-hand experience and insight into up-to-date and relevant industrial practices.

6.2.3 Research University – Research and development, technology and knowledge transfer at the BME

Budapest University of Technology and Economics (BME) as a research university targets the training of engineers capable of solving creative and innovative R&D challenges, furthermore, elaborating and implementing new products. This can be done if we actively participate in high level research programs regularly, and we earn even international recognition in some fields [BME Research University Strategy, 2010].

The BME Research University project has performed its main activities between 2010 and 2012. For the first step in the comprehensive development program, BME has identified five research areas:

- Sustainable energy,
- Vehicle technology, transport and logistics,
- Biotechnology, health and environment protection,
- Nanophysics, nanotechnology and materials science,
- Intelligent environment and e-technologies,

I was the Program Director of the *Intelligent environment and e-technologies* research area.

A key role is assigned to information and communications technologies (ICT) in the implementation of the strategic plans of European Union Member States and in the enhancement of the EU world market competitiveness. Because of its importance, recent years have seen a host of analyses in this area at both European and national levels. Research and development achievements in the ICT area have fundamentally determined the global technological development underpinning the development of world economy; within that the competitiveness of the domestic industry and its development options.

Our daily activities are increasingly supported and monitored by intelligent environments and e-technologies in which software and intelligent signal processing, data management and planning systems in the form of software, play dominant roles. The research activity of intelligent environments and e-technologies is directed towards areas where large distribution systems, consisting of a host of intelligent services, millions of computers and data collection points, will form the basic infrastructure of the knowledge based society, economy and service systems in the foreseeable future [BME Research University Milestones, 2012].

The main ICT strategies were the basic technologies and application oriented research directions. Within the scope of the *Intelligent environment and e-technologies* research area the Department of Automation and Applied Informatics has realized two projects: (i) Location-based mobile services, and (ii) Modeling and model processing. Our R&D results support the location-based solutions both for mobile and other applications areas, domain-specific modeling, effective model processing techniques and handling large models.

During the project we have realized several long term R&D results, which have strengthened the research groups, provided the continuity of the active research work and established industrial cooperation projects.

7 Summary

This chapter summarizes the main scientific results.

7.1 Thesis I: Domain-Specific Modeling and Model Processing

Related key publications: [Asztalos et al, 2007] [Lengyel et al, 2005] [Lengyel et al, 2006] [Lengyel et al, 2008] [Lengyel et al, 2015c] [Lengyel and Charaf, 2015d] [Lengyel and Charaf, 2015e] [Levendovszky et al, 2005a] [Levendovszky and Charaf, 2005b] [Mészáros et al, 2009] [Mezei et al, 2006] [Mezei et al, 2007].

The results of Thesis I are the followings:

Supporting Domain-Specific Modeling

- *A methodology that supports to define the concrete syntax (appearance) of visual domain-specific languages. A domain-specific language and its application to define the concrete syntax (appearance) of visual domain-specific languages.*
- *The methodology applies model processing to combine the abstract syntax (defined by the metamodel) and the concrete syntax. The result is a visual domain-specific language that can be used for modeling.*

Supporting Dynamic Behavior of Domain-Specific Languages

- *A methodology that supports the dynamic behavior (animation) definition of domain-specific languages. The methodology contains the event-driven architecture that supports the definition of the dynamic behavior.*
- *A methodology that applies model transformations to execute the models describing the dynamic behavior with.*

Model Transformation-Driven Software Model Processing

- *A methodology that supports the graph rewriting-based model processing. Models are represented as graphs and the model processing is defined as a sequence of graph rewiring rules. It has been proven that the method provides efficient algorithms to find the left-hand side graphs in the processed models, and to replace these matches with the right-hand side graphs of the rewriting rules.*
- *The algorithms of the methodology facilitate the followings: (i) based on the metamodel multiplicities it is decidable whether valid instance models can be constructed or not, (ii) in the context of the inheritance relation, the identification of the topology relations between the metamodel and instance models. Conditions have been worked out (i) to decide the applicability of the rewriting rules and (ii) to analyze the serial and parallel independence of the rewriting rules and decide whether the rules can be executed in parallel and/or in different order. The algorithms and the conditions provides offline (before the transformation execution) analyzing possibilities of the rewriting rules.*

Validating Model Transformations

- *A methodology that supports the online validation of graph rewriting based model transformation. The validation is performed by the pre- and postconditions assigned to the rewriting rules. Based on the methodology, the execution steps of the rewriting*

rules are as follows: (i) finding a topological match according to the left-hand side graph of the rewriting rule, (ii) validating the preconditions on the matched sub-graph, (ii) performing the rewriting (model processing), and (iv) validating the postconditions on the result of the rewriting.

- *Algorithms with proven correctness, which facilitate the validation of both the rewriting rules and whole model transformations.*
- *A methodology to analyze the behavior of graph rewriting-based model transformations. The analysis is supported by a control flow language, which facilitates to define the exact execution order of the rewriting rules, and by the algorithms that make possible to contract transformation rules.*

7.2 Thesis II: Increasing the Efficiency of Mobile Platforms

Related key publications: [Aczél and Charaf, 2005] [Braun et al, 2015] [Ekler et al, 2008a] [Ekler and Charaf, 2008b] [Ekler et al, 2010] [Ekler et al, 2015] [Fitzek et al, 2014] [Forstner et al, 2005] [Kelényi et al, 2007] [Kundra et al, 2015] [Pahlevani et al, 2014] [Pándi and Charaf, 2013] [Pándi and Charaf, 2015] [Vingelmann et al, 2011].

The results of Thesis II are the followings:

Analyzing Mobile Peer-to-Peer Networks, Retrieving Semantic Information

- *Analyzing unstructured peer-to-peer networks. A methodology, which supports that based on semantic information predict the probability whether the required information would be provided (the query could be answered).*
- *The method to design data structures, which support the methodology. Algorithms that based on semantic information build and utilize the data model.*

Examining Mobile Platforms-related Social Networks

- *A methodology that supports to model and discover the relations between social networks and mobile device phonebooks.*
- *Analyzing the scalability of mobile-based social networks. A methodology that supports the scalability of mobile-based social networks.*
- *A model that supports the estimation of energy consumption of mobile devices with limited resources.*

Energy Efficient Mobile Peer-to-Peer Solutions

- *A methodology facilitating that instead of partitioned data transfer to mobile devices the data is transferred in bursts (larger packages). The solution provides significant energy saving for mobile devices.*
- *Algorithms and a protocol that makes data transfer to mobile devices more energy efficient.*

7.3 Thesis III: A Model-Driven Methodology for Supporting Multiple Mobile Platforms and IoT Devices

Related key publications: [Charaf, 2013a] [Charaf, 2013b] [Charaf et al, 2014] [Ekler et al, 2015] [Lengyel et al, 2015a] [Lengyel et al, 2015b] [Lengyel et al, 2015c] [Lengyel and Charaf, 2015d].

The results of Thesis III are the followings:

Modeling Common Aspects of Different Mobile Platforms

- *Investigated and selected the technology that based on the research goals could be applied to work out the mobile applications related textual and/or visual domain-specific languages.*
- *Domain-specific languages facilitating to define the different aspects of mobile applications: user interface, data layer, static structure, dynamic behavior and communication protocols.*
- *A methodology that supports the common application of the different domain-specific languages.*

Supporting the Development for Different Mobile Platforms and IoT Devices with Model Processing

- *A model processing methodology that supports different mobile platforms at the same time.*
- *The correctness of the model processing algorithm has been proven. The produced artifacts are quality outputs and fulfill the requirements. Measurements verify the efficiency of the methodology.*

Methods for Supporting Application in Environments with Limited Resources

- *Investigation of those software and hardware factors that have effect on energy consumption in current mobile devices. The results have been analyzed.*
- *Based on the results of the analysis, development methodologies (best practices) and design patterns have been worked out. These results target the environments with limited resources, for example mobile devices.*

Applying Cloud-based Technologies

- *A method and related algorithms that applies cloud-based technologies for supporting domain-specific modeling.*
- *A method that applies cloud-based technologies for supporting model processing.*

To summarize, the objectives have targeted one of the most pressing problems of mobile software development, which derives mainly from the diversity of mobile platforms. To solve this problem, the mobile platforms have been analyzed from different perspectives. A modeling language family has been designed for mobile applications and an optimal framework has been worked out for all platforms supporting the code generated from the models. The result is a complete methodology and a system that allows designing mobile

applications and with model processing effectively supports the application development for each major mobile platform.

This thesis shows that my research results, the continuous work in the field of information and communication systems, and the results of my group (Applied Informatics Group at the Department of Automation and Applied Informatics) from the last 15 years have a great impact in several fields and promoted several results: among others, the motivation of the PhD students, active research work of my colleagues, close relationships with several industrial partners, successful R&D projects.

A. Appendix – Sample for Supporting Multiple Mobile Platforms

In this appendix we introduce examples related to the multiple mobile platform support. First we show an example where the server has a simple method that is used to create a new user in the target system. The method expects one parameter (the name of the user) and returns nothing. The corresponding C# interface is the following:

```
public interface MyService
{
    void insertUser(string userName);
}
```

To indicate that this interface defines the API of a server application we denote it with the *[RESTAPI]* attribute above the interface:

```
[RESTAPI]
public interface MyService
{
    void insertUser(string userName);
}
```

By finding this attribute the code generator will recognize that this interface should be treated as a REST API interface, and it should generate the client-side proxy class for that. But how would the code generator know what kind of Url should be invoked on calling this method? This is specified using the *[RESTUrl]* attribute above the method:

```
[RESTAPI]
public interface MyService
{
    [RESTUrl(Url = "insertuser.php")]
    void insertUser(string userName);
}
```

This way, the generated proxy will navigate to the *insertuser.php* page, and pass the username as url parameter (like *insertuser.php?userName=XXXX*).

If we would like to use different parameter names instead of the names of the parameters in the C# interface, we may customize that using the *[RESTParam]* attribute.

```
[RESTAPI]
public interface MyService
{
    [RESTUrl(Url = "insertuser.php")]
    void insertUser([RESTParam(Name="usr")]string userName);
}
```

In the example above, though, the name of the parameter is username, it is mapped to the "usr" http GET parameter (*insertuser.php?usr=XXXX*). A straightforward question is what happens if we need to use different HTTP methods, e.g. POST instead of GET to call the server-side service. We can specify it as the *CommandType* a parameter of the *[RESTUrl]* attribute:

```
[RESTAPI]
public interface MyService
{
    [RESTUrl(Url = "insertuser.php", CommandType = RESTCommandType.POST)]
    void insertUser([RESTParam(Name="usr")]string userName);
}
```

Changing the command type to POST (possible values are GET, POST, PUT, DELETE) we just instruct the code generator to generate a proxy code that uses the HTTP POST command to send the request. The passed parameters are again encoded into the request url. If we would like to pass the parameter inside the body of the http request as HTTP form parameter instead of the request url itself, then we can set it up using the *Mapping* parameter of the *RESTParam* attribute as follows:

```
[RESTAPI]
public interface MyService
{
    [RESTUrl(Url = "insertuser.php", CommandType = RESTCommandType.POST)]
    void insertUser([RESTParam(Name="usr", Mapping =
          RESTUrlMappingType.Body)]string userName);
}
```

The default value for the *Mapping* parameter is *RESTUrlMappingType.Url*. Although, we can already pass single parameters both in the request URL and in the HTTP body as form parameters, often the argument should be handled as not a parameter of the target resource but a locator for the target resource. For example the user we create is assigned to a specific client. The client is not passed as a parameter to the *insertuser.php* page, but the *insertuser.php* page is located inside the appropriate client folder like *http://..../client1/insertuser.php...* . To map a specific parameter into the resource Url at a specific position we should set the *Mapping* argument for that parameter to *RESTUrlMappingType.Custom*, indicate the position of this parameter with the *$* character.

```
[RESTAPI]
public interface MyService
{
    [RESTUrl(Url = "$client/insertuser.php", CommandType = RESTCommandType.POST)]
    void insertUser([RESTParam(Mapping = RESTUrlMappingType.Custom)]string client,
      [RESTParam(Name="usr", Mapping = RESTUrlMappingType.Body)]string userName);
}
```

Since server methods usually have a return value as well, it must be handled by the generated proxy code as well. Assume that the *insertUser* method returns the unique id (e.g. a *Guid*) of the newly created user inside the HTTP response as plain text. This return value can simply be returned by the generated proxy method by setting the return type of the *insertUser* method to string:

```
[RESTAPI]
public interface MyService
{
    [RESTUrl(Url = "$client/insertuser.php", CommandType = RESTCommandType.POST)]
    string insertUser([RESTParam(Mapping = RESTUrlMappingType.Custom)]
       string client,
```

```
        [RESTParam(Name="usr", Mapping = RESTUrlMappingType.Body)]string userName);
}
```

Declaration of the custom data types. In most practical cases, the parameters expected by the methods or the return values of them are not only primitive types like string, integer or a floating point number, but some complex type consisting on multiple subfields.

For this purpose, we have invented another interface-level attribute called *RESTDTO* that is the abbreviation of REST Data Transfer Object. Interfaces marked by this attribute will be handled by the code generator as simple classes used for representing and transmitting data. Of course, in addition to the standard data storing feature of such an object, the code generator may extend it with various additional features such as change notification and equality comparison.

Assume that when creating a new user, we would like to pass also the full name and the age of the user to be created. And we do not want to use separate method arguments for them but handle them as one unit. Then we may wrap these parameters into a new DTO interface:

```
[RESTDTO]
public interface User
{
    string UserName { get; set; }
    string FullName { get; set; }
    int Age { get; set; }
}
```

A complex type cannot be simply encoded into the request Url, thus, we must set it up to be serialized inside the body of the HTTP POST request:

```
[RESTAPI]
public interface MyService
{
    [RESTUrl(Url = "$client/insertuser.php", CommandType = RESTCommandType.POST)]
    string insertUser([RESTParam(Mapping = RESTUrlMappingType.Custom)]
       string client,
       [RESTParam(Mapping = RESTUrlMappingType.Body)]User user);
}
```

Using the default settings, the user parameter would be serialized as converting its fields into HTTP form parameters. But typically, in the REST communication rather XML or JSON serialization is applied. The way how complex parameters should be serialized can be specified with the *Format* argument of the *RESTParam* attribute:

```
[RESTAPI]
public interface MyService
{
    [RESTUrl(Url = "$client/insertuser.php", CommandType = RESTCommandType.POST)]
    string insertUser([RESTParam(Mapping = RESTUrlMappingType.Custom)]
      string client,
      [RESTParam(Mapping = RESTUrlMappingType.Body, Format =
          RESTFormatType.XML)]User user);
}
```

In a similar way, we may also expect complex return values that should be parsed as a custom DTO object. The way how the HTTP response should be deserialized can be specified with the help of the *RESTResponseType* attribute. For example, the following declaration indicates that the *insertUser* method returns some kind of *UserInfo* data, and it should be deserialized from the HTTP response as XML.

```
[ResponseType(Format = RESTFormatType.XML)]

[RESTUrl(Url = "$client/insertuser.php", CommandType = RESTCommandType.POST)]
UserInfo insertUser([RESTParam(Mapping = RESTUrlMappingType.Custom)]string client,
          [RESTParam(Mapping = RESTUrlMappingType.Body, Format =
              RESTFormatType.XML)]User user);
```

Of course, the same data may by serialized as XML in several different ways. By default, we assume each member field to be serialized as an XML tag identified by the name of the tag, while the value of the field is serialized as the content of the XML tag. If this value is a primitive type, then simply its printed value, if the value is of a complex type, then the same method is applied recursively. For example, the presented *User* DTO would be serialized as XML as:

```
<User>
    <UserName>joe</UserName>
    <FullName>John Doe</FullName>
    <Age>30</Age>
</User>
```

If we would like to change the way how the XML document is generated and parsed, we can perform it using the standard .NET XML formatter attributes by attaching them to the DTO definition.

```
[RESTDTO]
public interface User
{
    [XmlAttribute(AttributeName = "usr")]
    string UserName { get; set; }
    [XmlAttribute(AttributeName = "full")]
    string FullName { get; set; }
    [XmlAttribute(AttributeName = "age")]
    int Age { get; set; }

}
```

The above DTO definition results an XML document like the following one:

```
<User usr=”joe” full=”John Doe” age=”30”/>
```

Model Processing: Generating the Software Artifacts

Having the features of a network service, including its methods and the applied data types, already defined, the next step is that based on the interfaces generate executable source code which is able to perform the communication with the server component. There are various solutions that can be utilized when it is about code generation, we have chosen the Microsoft T4 (Text Template Transformation Toolkit). T4 is a mixture of static texts and procedural

code: the static text is simply printed into the output while the procedural code is executed and it may result in additional texts to be printed into the output. Recall that, the interface definitions are compiled into a .NET assembly that can be loaded and traversed using reflection afterwards. The T4 templates we write also work on the reflected content of the assemblies.

In the first round we are targeting two mobile platforms: Windows Phone 8 (C#) and Android (Java). Therefore, we need to prepare two different T4 templates for the two platforms. In case of Windows Phone, we expect the data transfer objects be represented by C# classes, the fields of the DTO entities should be represented as .NET properties, and the generated classes should also support some kind of change notification about changes in the properties. A possible implementation of the template is the following:

```
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Runtime.Serialization",
"3.0.0.0")]
[System.Runtime.Serialization.DataContractAttribute(Name="<#=type.Name#>")]
public partial class <#=type.Name#> : System.ComponentModel.INotifyPropertyChanged
{
<# foreach (var pi in type.GetProperties()) { #>

    private <#= pi.PropertyType.FullName #> <#=pi.Name#>Field;

    public <#= pi.PropertyType.FullName #> <#=pi.Name#>
    {
        get
        {
            return this.<#=pi.Name#>Field;
        }
        set
        {
<# if (!pi.PropertyType.IsValueType) { #>
            if (!object.ReferenceEquals(this.<#=pi.Name#>Field, value)) <# } else {
#>
            if (!this.<#=pi.Name#>Field.Equals(value))<# } #>

            {
                this.<#=pi.Name#>Field = value;
                this.RaisePropertyChanged("<#=pi.Name#>");
            }
        }
    }
<#}#>

    public event System.ComponentModel.PropertyChangedEventHandler PropertyChanged;
    protected void RaisePropertyChanged(string propertyName)
    {
        System.ComponentModel.PropertyChangedEventHandler propertyChanged =
           this.PropertyChanged;
        if ((PropertyChanged != null))
        {
            PropertyChanged(this, new
               System.ComponentModel.PropertyChangedEventArgs(propertyName));
        }
    }
}
```

The input of the template (*type*) is the reflected DTO type. The resulting code will describe a partial class the name of which corresponds to the name of the interface. Then, the template

iterates through all the fields declared in the interface, generates a private variable and a wrapper property for the variable. When setting up the value of the property, it checks if the new value is really different from the previous one, and changes the value of the underlying variable if it is really different. Then it also calls the *RaisePropertyChanged* method (passing the name of the changed property to it as parameter) that fires the *PropertyChanged* event. The concrete implementation of the DTO class generated for the interface *User* will look like the following one:

```
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Runtime.Serialization",
"3.0.0.0")]
[System.Runtime.Serialization.DataContractAttribute(Name="User")]
public partial class User : System.ComponentModel.INotifyPropertyChanged
{
    private System.String UserNameField;
    public System.String UserName
    {
        get
        {
            return this.UserNameField;
        }
        set
        {
            if (!object.ReferenceEquals(this.UserNameField, value))
            {
                this.UserNameField = value;
                this.RaisePropertyChanged("UserName");
            }
        }
    }

    private System.String FullNameField;
    public System.String FullName
    {
        get
        {
            return this.FullNameField;
        }
        set
        {
            if (!object.ReferenceEquals(this.FullNameField, value))
            {
                this.FullNameField = value;
                this.RaisePropertyChanged("FullName");
            }
        }
    }

    private System.Int32 AgeField;
    public System.Int32 Age
    {
        get
        {
            return this.AgeField;
        }
        set
        {
            if (!this.AgeField.Equals(value))
            {
                this.AgeField = value;
                this.RaisePropertyChanged("Age");
            }
        }
```

```
    }

    public event System.ComponentModel.PropertyChangedEventHandler PropertyChanged;
    protected void RaisePropertyChanged(string propertyName)
    {
        System.ComponentModel.PropertyChangedEventHandler propertyChanged =
           this.PropertyChanged;
        if ((propertyChanged != null))
        {
            propertyChanged(this, new
               System.ComponentModel.PropertyChangedEventArgs(propertyName));
        }
    }
}
```

For the Android (JAVA) implementation we do not need the DTOs to support any special features thus, here we just generate plain old JAVA (POJO) classes collecting private fields with getter and setter methods. The corresponding T4 template is much simpler as well:

```
<# Type type = this.DTOType; #>
public class <#=type.Name#> {
<# foreach (var pi in type.GetProperties()) { #>

    private <#= TTHelper.MapType(pi.PropertyType) #> <#=pi.Name#>Field;

    public <#= TTHelper.MapType(pi.PropertyType) #> get<#=pi.Name#>() {
        return this.<#=pi.Name#>Field;
    }

    public void set<#=pi.Name#>(<#= TTHelper.MapType(pi.PropertyType) #> value) {
        this.<#=pi.Name#>Field = value;
    }
<#}#>
}
```

There is a big difference compared to generating target code for C#, though. Since the interface was also written in C# and uses the primitive types of the .NET Base Class Library, and we are traversing the .NET assembly, we must translate the .NET types to JAVA types. In case of the C# generator template, we could simple jump over this step, since we could use the same type names as in the interface definition itself (see *PropertyType.FullName* in the template). In case of the code template for JAVA, we perform this translation using the *TTHelper.MapType* method. This is a custom implementation handling only some primitive types only, but can be arbitrarily extended with further types as well. The resulting JAVA code originating from the same *User* DTO interface is the following:

```
public class User {
    private String UserNameField;
    public String getUserName() {
        return this.UserNameField;
    }
    public void setUserName(String value) {
        this.UserNameField = value;
    }

    private String FullNameField;
    public String getFullName() {
        return this.FullNameField;
    }
    public void setFullName(String value) {
```

```
        this.FullNameField = value;
    }

    private int AgeField;
    public int getAge() {
        return this.AgeField;
    }
    public void setAge(int value) {
        this.AgeField = value;
    }
}
```

In general, it is advised to keep the generators as simple as possible, and outsource all the common implementations into helper classes or base classes. We followed this principle during realizing the code templates that generate the service proxy classes.

In case of the Android implementation, all the communication-specific parts of the implementation are outsourced into the *RESTTask* class that implements an Android Activity. In case of an Android Activity, the network communication is performed on an asynchronous thread, and the caller of the thread is notified about the result via a *BroadcastReceiver*.

B. Appendix – Case Study 1

The case study provides an example for a mobile application that is quite useful on all relevant mobile platforms. These types of mobile applications are within the focus of the presented methodology.

We know that we can easily get into difficult situations in foreign countries, if we do not speak the national language. For example in a restaurant, it is usually important to understand exactly what we will eat. A few decades ago, printed dictionaries were the only solution, but nowadays they are often replaced by a dictionary application running on the mobile devices. Unfortunately, these dictionaries rarely support translation of a whole sentence, or complex expressions, they offer a word-by-word translation instead. This is not always enough. Another problem is to handle non-Latin alphabets, e.g. to translate a Cyril, or an Arabic text, since it is hard to find out the word to translate. This case study proposes a solution for this issue.

The mobile application offers a smart mobile dictionary. The application takes photos using the mobile device, recognizes the text on the photo and translates it into the preferred language. Since text recognition and translation require serious computational power, these functions are not executed on the mobile device directly, but they use services running in the cloud environment. The translator application is not free, users have to register in order to use the application and they have to pay a small fee (via PayPal) for each translation. The application logic is represented in Figure B-1.

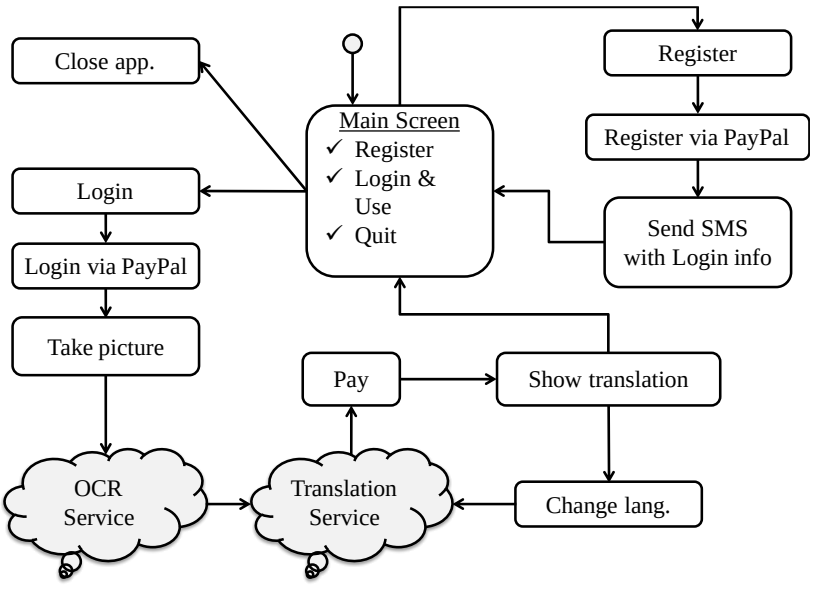

Figure B-1 Overview of the translation service case study

The figure provides a domain-specific model. This language can be used to define a set of similar applications. Shortly, we will see that by using the language, we can use typical elements of mobile application development on a high abstraction level. These elements can be simple or complex tasks, and they can be based on cloud services. For example, translation is a complex task. Now we investigate the model to understand its mechanisms.

The application starts with the *Main Screen*, which displays the main menu, with three options: *Registration, Login & Use* and *Quit*.

Quit menu item closes the application. It is used to define an exit point only.

Registration starts with a welcome screen (*Register*) containing a standard registration how-to and a few marketing images. Next, we have to register via PayPal. The model does not describe the steps of this registration, since applications within this domain always register in the same way, using the same steps. For example, we have to log in to our PayPal account and confirm access to the application. The procedure does not depend on whether we want to pay for a translation application, a web warehouse, or anything else. Common procedures, like this are implemented in an underlying domain-specific library. When generating code from the model above, we only have to generate a function call from this step and the framework will handle the rest. Thus, common tasks of the domain are not detailed in the model, the focus is on the unique, domain-specific issues and their solutions.

If the PayPal-based registration is successful, the user receives a text message with login information. SMS sending is not part of the PayPal registration, domain applications may use it, but it is not necessary to do so. However, the technical details of sending an SMS are not elaborated in the model, it is implemented in the library.

The *Login* also starts with a welcome screen (*Login*). Next, we force the PayPal login and validate whether the PayPal account is empty. The PayPal-based login is also part of the underlying library. Therefore, the framework provides a common way to handle it. The next step however, is completely application specific. We take a photo by using the camera of the mobile device. This photo contains the text to translate. We do not process the photo on the mobile, but call a cloud-based *OCR* service (Optical Character Recognition service). Although modern smart devices could apply OCR algorithms locally, it is not optimal. By applying the text recognition in the cloud, we can save energy. This is beneficial for the environment as well. The OCR service finds the text on the photo and forwards the text to another cloud service, the *Translator* service. Here again, it is possible that we could translate the text locally, but forwarding this task to the cloud is more energy efficient. In case of cloud service calls, the domain-specific framework allows us to specify the exact address of the services easily. This is the same principle as the one mentioned earlier: we hide technical details and focus on domain related questions. When the translation is ready, we pay. After paying, we can read the translated text. This *pay-first* approach makes cheating with the application more difficult. After reading the translation, the user can choose another target language (useful in a multi-language group) and the cloud service translates the source text again.

As it has been illustrated, the domain language and the domain-specific framework made possible to specify PayPal registration and login, or SMS sending very simple. Moreover it has simplified the usage of cloud services from mobile. We have successfully identified the common elements of our domain and built a language on these elements. By using these bricks and others such as welcome screens, we can build our application model easily and rapidly. For example the application logic of a package delivery system could also be

specified in a similar way. Moreover, since the language is rather specific to a domain, we can create efficient solutions for this language. Each domain model will use these solutions; therefore improving the efficiency affects all domain applications immediately.

Finally, note that the model is domain-specific, but not platform specific. We can process and generate applications for multiple mobile platforms (e.g. Android, iOS and Windows Phone) based on the same application logic specified by the model. We create domain development environments (code generation templates, domain framework) for each platform and use these environments in model processing. We do not implement upgrades separately from the platforms. Changes in the application logic can be propagated into the applications by code generation.

C. Appendix – Case Study 2

The second case study provides an example of a software service that follows the three-layered software architecture. The service is useful for all relevant mobile platforms. The solution applies the modeling, model processing, mobile platforms-related technologies, data handling and cloud computing-related research results. These types of software services are the target of the methodology provided by the thesis.

Organizing the invoices, tracking spending and balancing our account are essential tasks when it is time for our family and/or company to manage the budget. Applying intelligent software solutions can save time and provide more accurate results. Moreover, electronic data storage provides greater security for the data. The methods used to manage our spending data can greatly affect the process of organizing your money.

The main idea of the application bundle is to provide centralized, cloud-based data storage and services, which allow the manipulation of data using different clients on a variety of platforms. The native mobile clients provide the user interface to login to the system, manage spending and insert new data. The role of mobile devices is crucial in this scenario, since they are always with us and we can use them to immediately insert new spending. The camera feature of the device can be utilized to take a photo of an invoice, a receipt, or a bill. This photo can be automatically processed by software and provide the data regarding spending. When using a credit card, it is also common that we receive a text message to our phone with the details of our purchases. These text messages can also be automatically processed by the application. Furthermore, mobile device-based payment solutions are increasingly being used and have a notable role in the market. These solutions also allow users to use their phones instead of their bank cards.

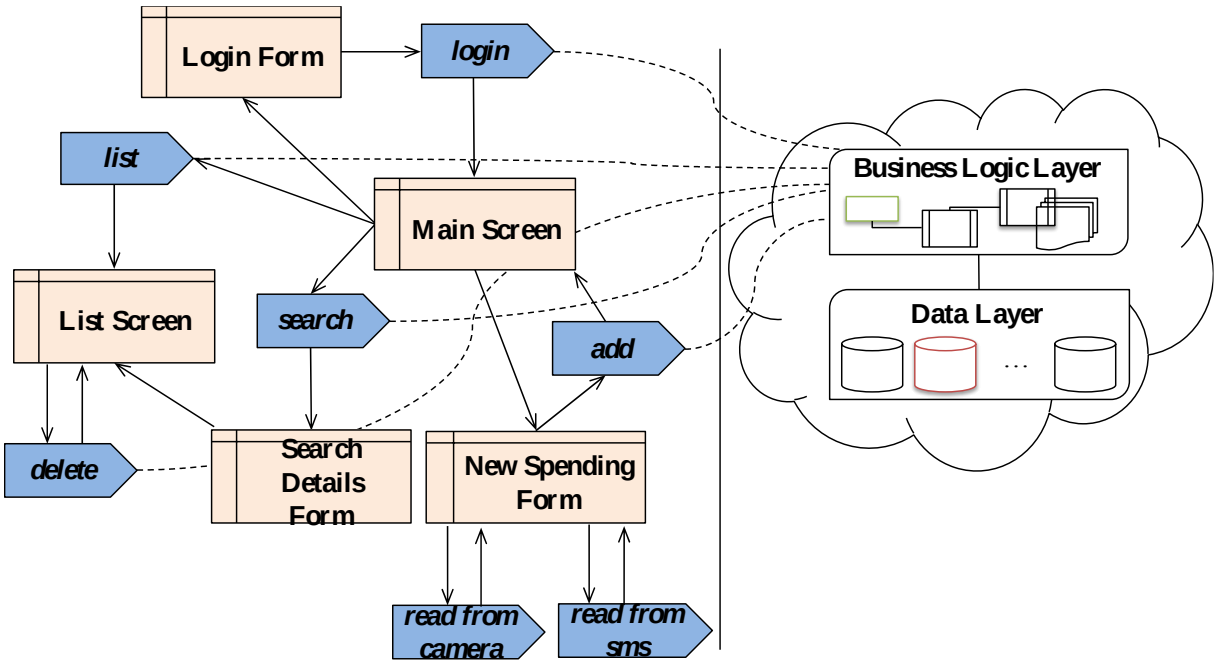

Figure C-1 Overview of the spending tracking case study

The flow of the application functions that run on the mobile devices is presented in Figure C-1. Rectangles on the left represent the main screens of the user interface, while the arrow-like rectangles are functions implemented in the application. The storage of the data and the application programming interface (API), which is used as a service to manipulate this data, is provided by the server and hosted in the cloud. From the main screen, we can navigate to the login screen. To perform the login, the login function needs to reach the cloud service to authenticate the user. This is denoted by the dashed line connecting the login function to the server side. Several functions are available from the main screen. We can query our spending list, search for specific data, or add new data. From the list screen, we are able to delete some of the data. From the *New Spending Form*, we can reach two functions: *read from camera* and *read from sms*. These elements represent that the camera or the text messages, stored on the mobile device, can be used to initialize the data on the form regarding new spending data. Note that these two functions do not need to reach the server.

We have seen that, aside from the data storage, a server component is required to manage access to the data and provide a platform independent application programming interface. Using this API, native clients can be implemented for the different platforms. The different artifacts, requiring development, are as follows:

- Based on the schema of the data, a data store needs to be initialized. For example, if the data is stored in a relational database, the appropriate tables should be created.
- A server component needs to be implemented, which can access the data store and provides a platform independent API for the actual functions used to manipulate the data. Besides these functions, the server should provide other services, e.g. notifications regarding data changes or built-in user management.
- Native clients must be implemented for the different mobile platforms.
- Finally, for administration purposes, a web client is required.

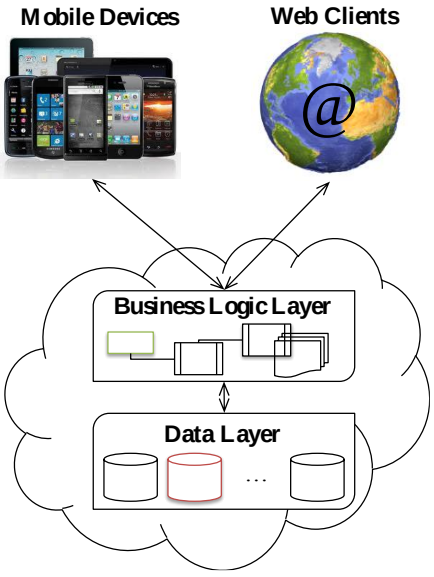

Figure C-2 The architecture of the case study application

Through the generalization of the architecture in this concrete application, the need to specify a data model becomes apparent. In order to complete this, several data manipulation functions and clients who provide user interfaces for these functions are required. The flow of the functions needs to be defined in order to provide the client's logic. Moreover, the structure of the client's forms can be derived from the data model. Note that the implementation of several functions can also be derived from the data model, e.g., listing, inserting and deleting. Moreover, several services, such as user management, are not specific to this application.

This is clear that, in providing two domains, we can make better use of the concept of domain-specific modeling. Within the first domain, we can model data schemata and the needed functions (data manipulation model). In the second domain, we can specify platform independent and high-level descriptions of the flow regarding the client's functions (function flow model). Having completed the function flow model, the three layers of the architecture above can be generated automatically.

- The schema of the data storage can be generated from the data manipulation model.
- The code of the server component can be generated from the data manipulation model and the function flow model. The source code for several, simple data manipulation tasks can be automatically generated, e.g., deleting data, inserting new data, listing of data, etc. Naturally, if a more complex application-specific login needs to be implemented, it must be manually provided by the developer.
- The code of the mobile clients is also based on both models. The function descriptions provide the connections between the different pages. Meanwhile, the structure of the forms, used to display data, can be generated from the data manipulation model.

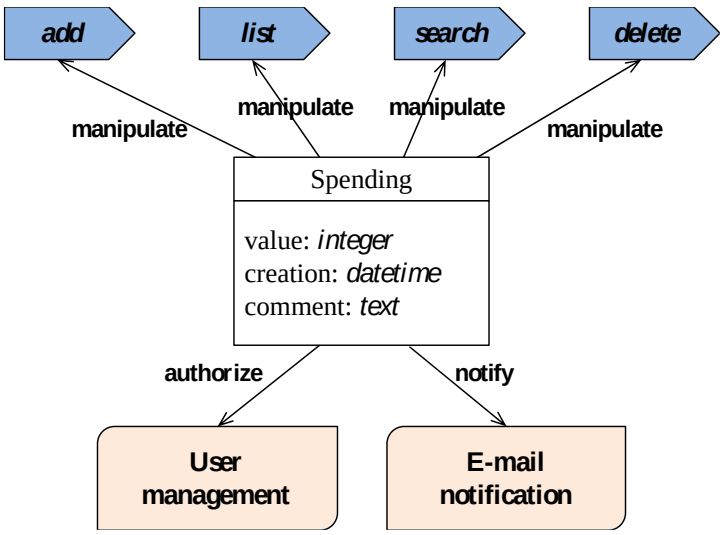

Figure C-3 Data manipulation in the case study

Assume that the function flow of our original budget tracking application is modeled as it is presented in Figure C-1. A sample model of the data manipulation is provided in Figure C-3. The rectangle in the middle provides the description of a simple data item we will store. For

each transaction we provide its numerical value, its creation date, and a comment. This is also specified that four functions are needed (add, list, search and delete) for data manipulation. Moreover, we attach two further components for this data, which are provided by the framework: User management and Email notification.

As the language of the two modeling domains becomes more powerful, the whole concept becomes more efficient and less code needs to be added manually. We highlight that this solution allows the code generation for different mobile platforms and also supports the service-side realization of the whole solution. However, custom validations and/or further custom behavior, which requires to manually extend the implementation, are also worthy of further examination.

The case study highlights the capabilities of our research areas when they are applied jointly. This also underpins that in the case of the software industry, only research groups and implementation groups can have a relevant role. This is because individuals cannot collect and maintain a deep knowledge throughout several areas, e.g. in the field of modeling, model processing, mobile platforms, energy efficient development, data technologies and cloud computing.

D. Appendix – Case Study 3

The third case study is based on [Levendovszky et al, 2009b]. The proliferation of mobile peer-to-peer systems made a next generation mobile BitTorrent client an appropriate target to compare two different development approaches: the traditional manual coding and domain-specific modeling languages accompanied by generators. The case study presents two domain-specific modeling languages for mobile communication modeling, and one for user interface development. We compare the approaches by development time and maintenance, using our modeling and transformation tool Visual Modeling and Transformation System.

MobTorrent is a *BitTorrent* client for Java ME platform. Having developed the manually coded version, we started with creating domain-specific languages that can be used to describe P2P systems for mobile applications. We identified two main functionality groups where the domain-specific modeling languages are useful: processing the protocol messages and designing the user interface. As a domain-specific modeling language platform, we used the metamodeling and model transformation tool VMTS. In VMTS, we created the domain-specific modeling languages, and we defined the model processors that translate the models into Java ME code.

We were about to address the following issues:

- How can Mobile P2P application benefit from domain-specific modeling languages?
- Does the domain-specific modeling language technology pay off at all in mobile P2P development in time?
- Do the domain-specific models require less maintenance effort?
- Could the domain-specific modeling language approach accelerate the development of the future versions?

We start with answering the first question by giving an insight of the used domain-specific languages, moreover, we show how we got the facts that underpin the answers.

In VMTS, we developed an integrated environment to visually model different aspects of Java ME mobile applications, and code generators to turn the models into executable Java code. The *Java Resource Editor* domain-specific language is appropriate for the rapid development of the static components of mobile applications, while the *Java Network Protocol Designer* can be used to model the static components and the dynamic behavior of simple, message-based network protocols.

Java ME Network Communication Support

In order to download content via *BitTorrent*, we need a torrent file. This small file contains some meta-data describing the content and the address of at least one central peer called *Tracker*, which manages the traffic. After we have the torrent file, the *BitTorrent* client connects to the *Tracker*, which sends a set of addresses of other peers back to the client. Then the client connects to these addresses and concurrently downloads the content from them via a

BitTorrent-specific protocol. In BitTorrent, we can download the content simultaneously from different peers.

We developed two domain-specific modeling languages for modeling the static and dynamic aspects of message-based network protocols, an integrated configuration environment and code generators to support the rapid modeling and implementation of communication through the network. It is capable of describing the peer-wire protocol and its processing logic. This solution exploits the fact that numerous well-known and widely used network protocols take a message-based approach. This means that the entities communicating with each other use a well-defined language, which consists of exactly identifiable elements with a predefined structure. The *MessageStructure* language models the messages (the static components) of such a protocol. Furthermore, the *MessageProcessor* language is provided to describe the logic of a protocol. We use hierarchical state machines to define this logic: we can declare the possible incoming messages in a state and the messages to be sent when leaving a state.

With the help of model processors, we generate a standalone network library, which can be adapted to the user interface or to business logic components. The generated network library provides its services through a unified callback interface. Using this interface this is possible to subscribe to numerous events fired by the library during communication.

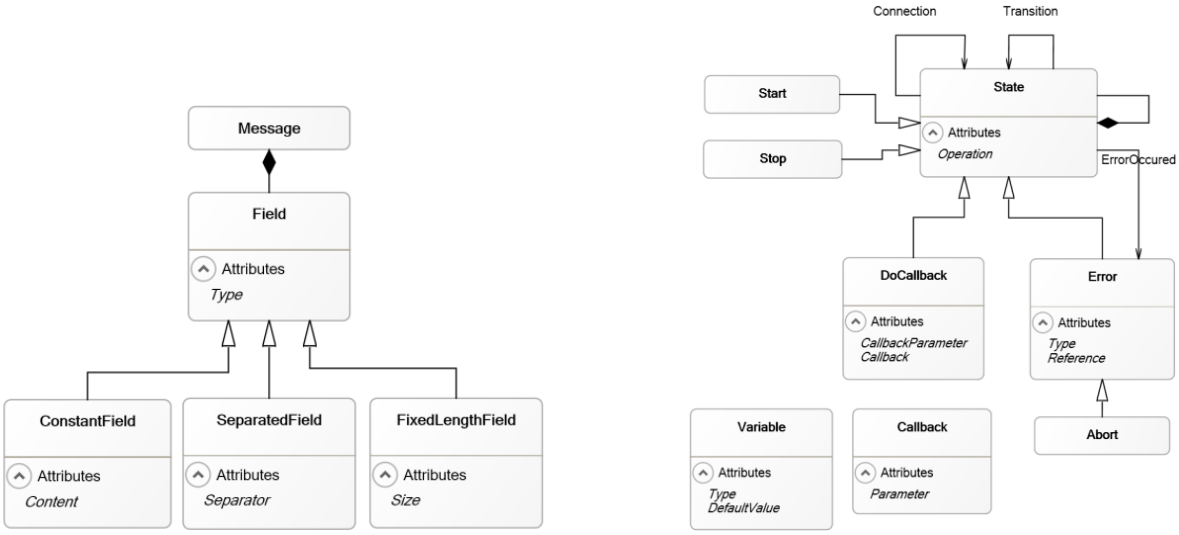

Figure D-1 Metamodels for modeling the static and dynamic properties of message-driven state machines

Modeling messages. Figure D-1 presents the metamodel of the *MessageStructure* domain-specific modeling language. The *Message* is the unit of the communication of the approach. Each message consists of several *Field*s. *Fields* have a *Type* attribute which corresponds to a simple Java type. The supported types are *int, byte* and *String*. We distinguish three different types of fields: *ConstantField, FixedLengthField,* and *SeparatedField.* A *ConstantField* has an additional *Content* attribute which is used to define the exact content of such a field at modeling time. In a protocol where a user ID (e.g.: 123) is sent in the format of *#userid#123*, the *#userid#* part of the message is a *ConstantField*. When reading a message from the network stream, the content of a *ConstantField* must be found at the position defined by the field in the model. Otherwise, the message processing fails.

FixedLengthFields have a predefined size (*Size* attribute). This means that the field represents a buffer for *Size* pieces of elements of type *Type. The Size* attribute does not have to be a constant value, instead, it can be contained by a field of the same message or global variable, or even an aggregated value of those. This means that if we recognize a *FixedLengthField* in a message it is possible that the size of this *FixedLengthField* depends on the already read content of a previous field in this message. *SeparatedField*s do not have a predefined value or size. Their start and end are marked by a character sequence specified in their *Separator* attribute. Reading such a field is finished with reading the value of the *Separator attribute* from the stream.

During code generation, Java classes are created based on the message elements. The contained fields of the messages will correspond to the fields of the Java class. Based on the model and the order of the fields of the messages, the member methods to read or write the message from or to the network stream are also generated. With the help of modeling messages and generating their wrapper classes, our solution completely hides byte-wise network stream operations, and provides an interface based on Java objects to the upper layers of the application.

BitTorrent messages. In order to discover and filter the incoming messages described with the *MessageStructure* domain-specific modeling language, we have implemented a message discovery algorithm. After a message is parsed, a callback method is being called which carries the different type of *MessageFields as* parameters. This callback method is used by the *MobTorrent* framework to execute BitTorrent-specific functions such as save the incoming data in a file.

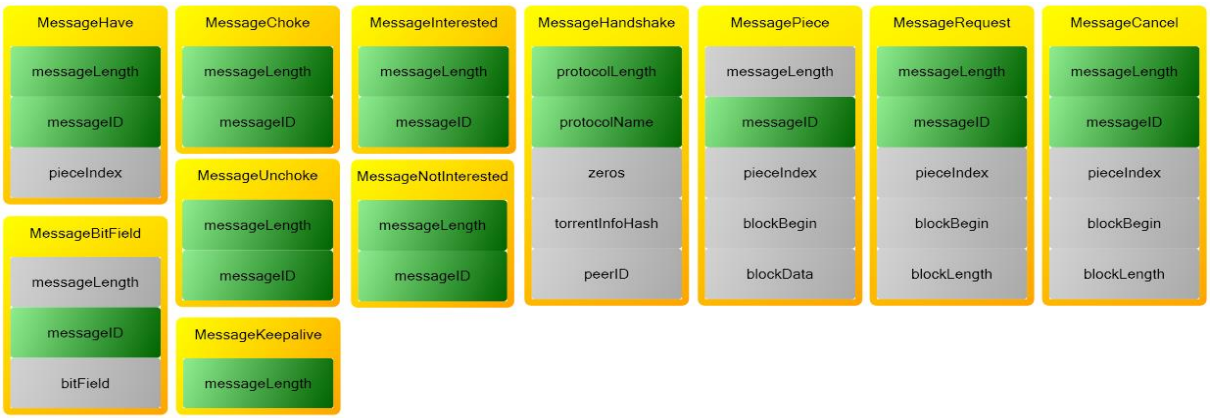

Figure D-2 Message objects used by the BitTorrent protocol

Figure D-2 presents the model of the BitTorrent protocol messages. The green fields are the *ConstantFields* and the grey ones are the *FixedLengthFields*. BitTorrent protocol does not use *SeparatedFields*. Usually in every message-based protocol, the messages have a common structure. In the case of BitTorrent we can separate the messages into two parts. The first part contains the *MessageHandshake* (Figure D-2) only, which is used during the *peer-wire protocol* to determine whether two peers are compatible with each other. *MessageHandshake* starts with two *ConstantFields* followed by three *FixedLengthFields*.

Figure D-2 shows that the *messageLength* field is green in all the messages, except for *MessagePiece* and *MessageBitfield*. The length of these two messages depends on the amount of data they carry. Following the *messageLength*, each message contains a *messageID* field which makes it easy to filter the messages. Only the *MessageKeepAlive* does not have this *messageID* field, because it contains only a *messageLength* constant field.

Modeling dynamic behavior. The core concept of the approach is that communication layer performs status changes as a consequence of receiving specific messages from the network stream. In addition, we may also instantiate and send network messages during a status change. The communication layer can run standalone, and it informs the connecting components of the application through a callback interface about the important events of the communication. The business logic can influence the behavior of the communication layer through the parameters of the layer and by sending messages directly through the network stream. The behavior of the network layer can be modeled with the help of a message-driven state machine.

Figure D-1 presents the metamodel of the *MessageProcessor* DSL. There are two special types of states: the *Start* and the *Stop* states. The *Start* state indicates the entry point of the state machine, while the *Stop* state indicates the exit point. States can be nested, therefor the start and end states may also be used as the entry/exit point of a sub-state machine. States can be connected with the help of *Transition* edges. A *Transition* edge may trigger the reception of a specific message from the stream: the type of the expected message is defined by the *MessageTypeIn* attribute of the edge, which references an already modeled message. If several outgoing transition edges are connected to the same state, then the transition whose triggered message first arrives will be chosen. If a transition is chosen, the state pointed by its right end will be the next active state. When activating a state, the instruction described in its *Operation* attribute is executed.

The state machine may could be customized with *Variables*. Each variable has a name, a type and a default value. Variables can be considered global parameters, which can be accessed by all states and edges.

A callback method is generated on the callback interface for each error node. Furthermore, a method is generated for each stop state as well. The generated methods can be parameterized with the help of the *Parameter* attribute. The callback methods of the interface can be invoked in two ways: either through a *DoCallback* state, or with the transition edges, as a method invoke can be assigned to each transition.

Figure D-3 illustrates the model that has been created for the BitTorrent protocol. The yellow boxes represent the global variables of the state machine: (*i) peerAddress* – the address of the peer we are connected to, (ii) *torrentInfoHash* – the hash of the downloaded torrent, (iii) *peerId* – the unique identifier of the connected peer and (iv) *ownPeerId* – our own identifier. The blue boxes with a small yellow lightning denote the callback methods created on the callback interface.

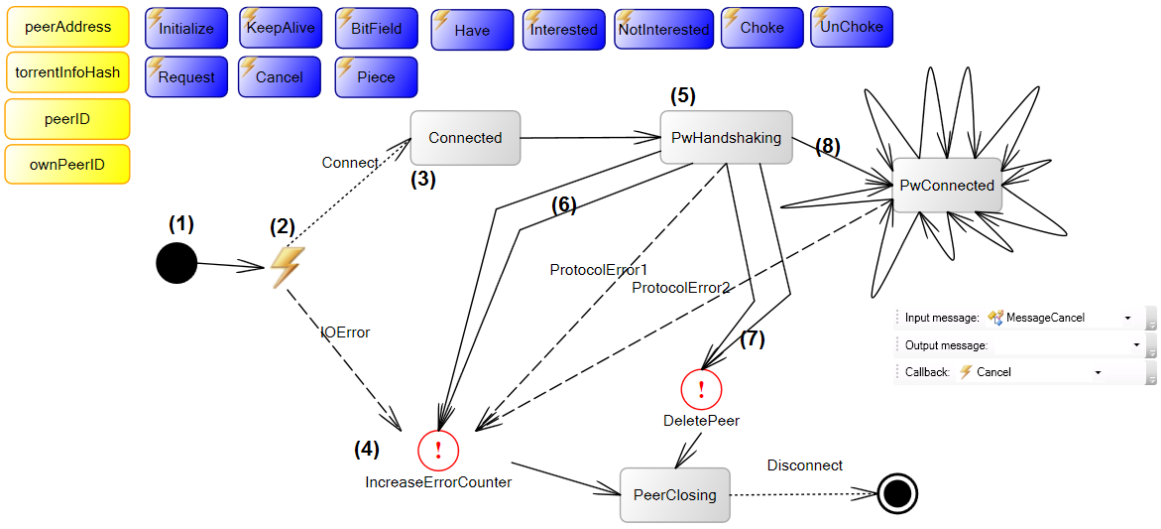

Figure D-3 BitTorrent client protocol model

The protocol works as follows. After the start state (1), we call the *Initialize* callback (2) to instruct the framework to perform initialization steps. Then the protocol tries to connect to the target host (the *Connect* edge is parameterized with the *peerAddress* variable). If the connection succeeds, we get to the *Connected* state (3), otherwise an I/O error occurs (4). On error, we perform an *IncreaseErrorCounter* callback, and disconnect the stream. Moving from (3) to (5) a *MessageHandshake* (Figure D-2) message is sent to the remote peer. Edges (6) and (7) trigger the answer-*MessageHandshake* message, and check if the parameters of the answer are valid. On an invalid handshake answer, either the *IncreaseErrorCounter* or the *DeletePeer* state will be active, and the communication is closed with the current peer. Edge (8) is a fallback edge meaning that edge (8) is chosen if neither of error transitions (6-7) can be selected. The state *PWConnected* can be considered the default state of the protocol: almost any type of messages can be received at this state (that is why there are so many loop edges around it), and each message arrival performs the appropriate callback invocation. Figure D-3 shows that the edge parameters (and also other model parameters) can be changed with the help of smart tags. They appear when the mouse is hovered over an item. State *PwConnected* can be left only if a protocol error occurs, or the business logic over the network layer changes the current state.

Mobile DSL for User Interface Development

Having generated code from the network model and integrated it with the *MobTorrent framework*, we continue the work with the user interface of our new mobile BitTorrent client. With User Interface (UI) domain-specific modeling languages, we can model the static structure of user interfaces, and generate the platform-specific source code according to the models. The UI domain-specific modeling language also has a metamodel, We support all the *Screens*, *Commands* and *Controls* available in Java ME both on the modeling and the generator level. In P2P applications we usually download multiple contents at the same time and these downloads are displayed in a list where the icon of the list item represents the status (downloading, finished, error, etc.) of the download. With the help of an *ImageList* we can access image resources and use them in other components, for example in a List.

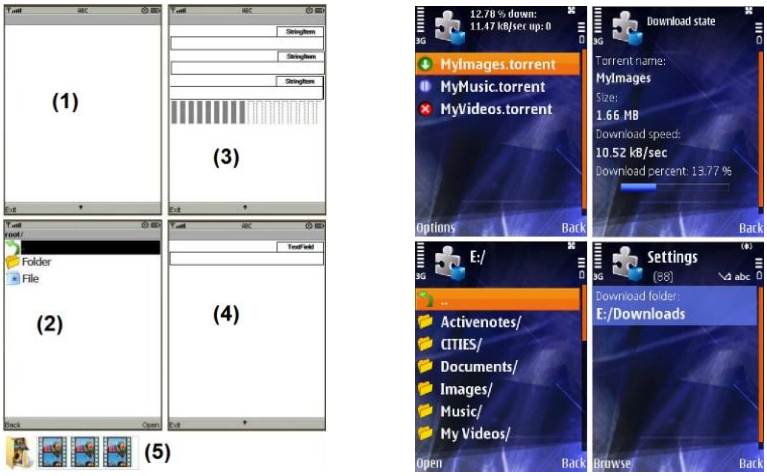

Figure D-4 User Interface model of the mobile BitTorent client in VMTS

In Figure D-4, the four screens of the application can be seen both at modeling time (a), and when executing the application on a real hardware (b). Screen (1) is used to present the torrents being processed (*TorrentList*). It is modeled with a simple *JList* item which is replaced with a class derived from Java ME *List* during code generation. Screen (2) is the *FileSelectDialog* itself, with which one can browse for a torrent file to be processed. Screen (3) is used to show the download state of the selected torrent. Screen (3) is built from a *JForm* item, which contains three *StringItems* for presenting the name of the torrent file, the size of the downloaded data, and the actual transfer rate. A *JGauge* element represents a progress bar which shows the progress of the download. Finally, screen (4) is used to modify the application settings such as the download path. It is also based on a *JForm* element, which contains a *JTextField* item. (*JTextField* corresponds to the *TextField* Java ME class).

We can also set the commands (menus) for the screens. The *TorrenList* contains commands for torrent handling such as add torrent file, start download, pause download, and commands responsible for navigating to another screen like *Settings* or *Download state*. Thus, we can also describe the high-level UI logic. The model also contains an *ImageList* (5) with three icons. This list represents the icon set used by the *Screen* (1). Finally, after modeling and generating the network layer and the user interface, one task remains: integrating the generated components with the *MobTorrent* framework. The integration is not supported with model processing techniques, i.e. the gluing code has been written manually.

In order to integrate the UI with the *MobTorrent framework* we have applied the *Observer design pattern*. The framework provides an interface which we have to implement in the UI code. This interface contains functions that are called from the framework when the status of the download changes, such as download speed changed, download progress increased. When we initialize the framework we have to set which object implements the observer interface in the UI. By using this observer the framework can notify the UI if something changes and the relevant information can be displayed on the screen of the mobile phone easily.

Conclusions

So far we have shown how mobile P2P development can benefit from domain-specific modeling technology. We found well-separated functionality groups, and supported them by domain-specific modeling languages and code generators. Table 1 depicts the development times with manual coding and with domain-specific modeling languages taking one developer into account who had previous experience of this sort of application.

Functionality	**Time with manual coding**	**Time with DSL**
User interface	5 days	2 day
Peer network connection	8 days	1 day
Peer-wire protocol	6 days	1 day
Message handling	10 days	2 days

Table D-1 Development time with and without DSMLs

Additionally, there were functions, which we did not support with domain-specific modeling languages. These required the following amount of time:

- File and database handling: 8 days
- BitTorrent specific functions: 13 days
- Tracker communication: 5 days
- Download for other clients: 8 days

The domain-specific modeling language infrastructure, i.e. the languages and the generators, is developed by an engineer with extensive domain-specific modeling language and tool experience. The time spent per person is the following:

- *MessageStructure* and *MessageProcessor* DSMLs: 4 days
- *MessageStructure* and *MessageProcessor* generators: 5 days
- UI DSML: 9 days
- UI generator: 10 days

So the development effort for the functions supported by DSMLs is as follows:

- With DSMLs: 6 days
- Without DSMLs: 29 days

The development time without the time for the DSML development:

- With DSMLs: 40 days
- Without DSMLs: 63 days

Including the DSML infrastructure development:

- With DSMLs: 68 days
- Without DSMLs: 63 days

These numbers highlight the fact that domain-specific modeling language technology is a generative technique: a generator is much harder to develop than the generated code once. Therefore, the more times we run the generator, the more the domain-specific modeling language approach pays off. From the second time on, the domain-specific modeling language and generator development does not appear as an additional cost.

As long as only the models need to be modified, domain-specific modeling languages increase the maintainability. If the generator must also be modified, the necessary effort can arbitrarily increase. These domain-specific modeling languages can be reused for any Java ME mobile development where UI or network support is required, but the approach is not limited to the Java ME platform, since it can be extended to other platforms by modifying the code generators. The proposed case study can be used as well in other solutions where BitTorrent technology is used for content distribution.

Bibliography

[Aczél and Charaf, 2005] Aczél, K. and Charaf, H., Automatic user interface code generation in symbian, MicroCAD 2005: International Scientific Conference, Hungary, pp. 1-5, 2005.

[Adobe AIR] Adobe AIR homepage, URL: http://www.adobe.com/hu/products/air.html, 2015.

[Ahlswede et al, 2000] Ahlswede, R., Cai, N., Li, S.Y.R. and Yeung, R.W., Network information flow. IEEE Transactions on Information Theory, Vol. 46, Issue 4, pp. 1204–1216, 2000.

[Alajrami et al, 2014] Alajrami, S., Romanovsky, A., Watson, P. and Roth. A., Towards Cloud-Based Software Process Modelling and Enactment, 2nd Workshop on MDE for and in the Cloud (CloudMDE 2014), Spain, pp. 6-15, 2014.

[Android] Android webpage, URL: http://www.android.com/, 2015.

[Angyal et al, 2009] Angyal, L., Asztalos, M., Lengyel, L., Levendovszky, T., Madari, I., Mezei, G., Mészáros, T., Siroki, L. and Vajk, T., Towards a fast, efficient and customizable domain-specific modeling framework, In Proceedings of the IASTED International Conference, pp. 11–16, Innsbruck, 2009.

[ANTLR] ANother Tool for Language Recognition website, URL: http://www.antlr.org, 2015.

[Apache Hadoop, 2015] The Apache Software Foundation, Apache Hadoop, URL: http://hadoop.apache.org/, 2015.

[Appcelerator] Appcelerator platform homepage, URL: http://www.appcelerator.com/, 2013.

[Assmann, 1996] Assmann, U., How to uniformly specify program analysis and transformation with graph rewrite systems, Lecture Notes in Computer Science, pp. 1060, 1996.

[Asztalos et al, 2007] Asztalos, M., Lengyel, L., Levendovszky, T. and Charaf, H., Graph Transformation Contest - UML to CSP Transformation, In: Applications of Graph Transformation 2007 (AGTIVE) - Graph Transformation Tool Contest, Kassel, Germany, Paper 9, 2007.

[Atkinson et al, 2003] Atkinson, C. and Kühne, T., Model-driven development: A metamodeling foundation, IEEE Software, Vol. 20, Issue 5, pp. 36–41, 2003.

[BBC Research Report, 2015] BBC Research Report, Global Markets and Technologies for Sensors, URL: http://www.bccresearch.com/market-research/instrumentation-and-sensors/sensors-ias006f.html, 2015.

[Beuche et al, 2006] Beuche, D. and Dalgarno, M., Software product line engineering with feature models, Methods and Tools, 2006.

[Bharambe et al, 2006] Bharambe, A. R., Herley, C. and Padmanabhan, V. N., Analyzing and improving a bittorrent networks performance mechanisms, 25th IEEE International Conference on Computer Communications, pp. 1 –12, 2006.

[Blackburn et al, 2009] Blackburn, J. and Christensen, K., A simulation study of a new green bittorrent, IEEE International Conference on Communications Workshops, pp. 1–6, 2009.

[BME(IT)^2] BME Innovation and Knowledge Centre of Information Technology – BME(IT)^2, URL: http://www.it2.bme.hu/en, 2009.

[BME Research University Milestones, 2012] Research University Milestones 2012, Research and development, technology and knowledge transfer at the BME, Budapest University of Technology and Economics, 2012.

[BME Research University Strategy, 2010] BME Research University Strategy Summary, Budapest University of Technology and Economics, 2010.

[Braun et al, 2015] Braun, P.J., Sipos, M., Ekler, P. and Charaf, H., Increasing data distribution in BitTorrent networks by using network coding techniques, 21th European Wireless Conference, Budapest, Hungary, pp. 196-202, 2015.

[Brooks et al, 2008] Brooks, C., Lee, E.A., Liu, X., Neuendorffer, S., Zhao, Y. and et al, Heterogeneous concurrent modeling and design in java, Technical report, 2008.

[Bruneliere et al, 2010] Bruneliere, H., Cabot, J. and Jouault, F., Combining model-driven engineering and cloud computing, 6th European Conference on Modeling Foundations and Applications, Paris, France, pp. 1-2, 2010.

[Budinsky et al, 2003] Budinsky, F., Steinberg, D., Merks, E., Ellersick, R. and Grose, T.J, Eclipse modeling framework, Addison-Wesley Professional, 2003.

[Chang et al, 2011] Chang, L., Liu, Y., Wei, Z. and Pan, J., Optimizing BitTorrent-like peer-to-peer systems in the presence of network address translation devices, Peer-to-Peer Networking and Applications, Vol. 4, Issue 3, pp. 274–288, 2011.

[Charaf, 2013a] Charaf, H., A methodology for model-driven multiplatform mobile application development, International Journal of Computer Engineering and Technology, Vol. 4, Issue 1, pp. 61-70, 2013.

[Charaf, 2013b] Charaf, H., Technology for multi-platform mobile application development, Wulfenia Journal, Vol. 20, Issue 2, pp. 427-438, 2013.

[Charaf et al, 2014] Charaf, H., Ekler, P., Mészáros, T., Kelényi, I., Kővári, B., Albert, I., Forstner, B. and Lengyel, L., Mobile Platforms and Multi-Mobile Platform Development, Acta Cybernetica, Vol. 21, Issue 4, pp. 529-552, 2014.

[Chiu et al, 2006] Chiu, D.M., Yeung, R.W, Huang, J. and Fan, B., Can network coding help in P2P networks, 2nd Workshop of Network Coding, 2006.

[Chou et al, 2003] Chou, P.A., Wu, Y. and Jain, K., Practical network coding, Annual Allerton Conference on Communication Control and Computing, Vol. 4, pp. 40–49, 2003.

[Chow et al, 2009] Chow, A. L. H., Golubchik, L. and Misra, V., BitTorrent: An extensible heterogeneous model, INFOCOM 2009, pp. 585–593, 2009.

[Clarke et al, 2002] Clarke, I., Miller, S. G., Hong, T. W., Sandberg, O. and Wiley, B., Protecting free expression online with freenet, IEEE Internet Computing, Vol. 6, pp. 40–49, 2002.

[Clasen et al, 2012] Clasen, C., Fabro, M.C.C. and Tisi, M., Transforming Very Large Models in the Cloud: a Research Roadmap, 1st Workshop on MDE for and in the Cloud (CloudMDE 2012), Denmark, 2012.

[Cohen, 2003] Cohen, B., Incentives build robustness in bittorrent, 1st Workshop on Economics of Peer-to-Peer Systems, 2003.

[Creus et al, 2007] Creus, G. B. and Kuulusa, M., Optimizing mobile software with built-in power profiling, Springer, 2007.

[Czarnecki et al., 2000] Czarnecki, K. and Eisenecker, U.W., Generative programming: Methods, tools, and applications, Addison-Wesley, 2000.

[Dimakis et al, 2011] Dimakis, A.G., Ramchandran, K., Wu, Y. and Suh, C., A survey on network codes for distributed storage, IEEE, Vol. 99, Issue 3, pp. 476 –489, 2011.

[Dong et al, 2011] Dong, M. and Zhong, L., Self-constructive high-rate system energy modeling for battery-powered mobile systems, 9th International Conference on Mobile Systems, Applications, and Services (MobiSys '11), ACM, New York, NY, USA, 335-348, 2011.

[Donohoe, 2012] Donohoe, P. (ed.), Software Product Lines: Experience and Research Directions, Springer Science & Business Media, 532 pages, 2012.

[EBNF, 1996] Information technology - Syntactic metalanguage - Extended BNF, 1996.

[Eclipse] Eclipse homepage, URL: http://www.eclipse.org, 2015.

[Ehrig et al, 1999] Ehrig, H., Engels, G., Kreowski, H-J. and Rozemberg, G. (ed.), Handbook on graph grammars and computing by graph transformation: Application, languages and tools, Vol. 2, World Scientific, Singapore, 1999.

[Ehrig et al, 2006] Ehrig, H., Ehrig, K., Prange, U. and Taentzer, G., Fundamentals of algebraic graph transformation, Monographs in Theoretical Computer Science, Springer, 2006.

[EIT Climate-KIC] EIT Climate-KIC, Knowledge & Innovation Community, URL: http://www.climate-kic.org, 2015.

[Ekler et al, 2008a] Ekler, P., Kelényi, I. and Charaf, H., BitTorrent at mobile phones, 5th IEEE Consumer Communications & Networking Conference, Las Vegas, 2008.

[Ekler and Charaf, 2008b] Ekler, P. and Charaf, H., Analyzing the Concept of Involving Low End Devices in a Cooperative Network, IEEE International Conference on Communications Workshops. Peking, China, pp. 102-106, 2008.

[Ekler et al, 2010] Ekler, P., Lukovszkiy, T. and Charaf, H., Evaluating Dynamically Evolving Mobile-Based Social Networks, Acta Cybernetica, Vol. 19, Issue 4, pp. 735-748, 2010.

[Ekler, 2011] Ekler, P., MobTorrent P2P mobile application 1.1, Budapest University of Technology and Economics, URL: http://amorg.aut.bme.hu/projects/mobtorrent, 2011.

[Ekler et al, 2012] Ekler, P., Fehér, M., Forstner, B. and Kelényi I., Android-alapú szoftverfejlesztés - Az Android rendszer programozásának bemutatása, SZAK Kiadó, 2012.

[Ekler et al, 2015] Ekler, P., Balogh, T., Ujj, T., Charaf, H. and Lengyel, L., Social Driving in Connected Car Environment, 21th European Wireless Conference, Budapest, Hungary, pp. 190-195, 2015.

[EMF] Eclipse Modeling Framework homepage, URL: http://www.eclipse.org/emf, 2015.

[EngMATLib] EngMATLib homepage, URL: http://www.thecodeproject.com, 2015.

[Evangelista et al, 2011] Evangelista, P., Amaral, M., Miers, C., Goya, W., Simplicio, M., Carvalho, T. and Souza, V., Ebitsim: An enhanced bittorrent simulation using omnet++ 4, In Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 437–440, 2011.

[Facebook Statistics, 2015] Facebook Statistics, http://newsroom.fb.com/company-info, 2015.

[Fan et al, 2009] Fan, B., Lui, J. C. S. and Chiu, D.-M., The design trade-offs of bittorrent-like file sharing protocols, IEEE/ACM Trans. Netw., Vol. 17, Issue 2, pp. 365–376, 2009.

[Farooq et al, 2015] Farooq, M.U., Waseem, M., Khair, A.i and Sadia, M., A Critical Analysis on the Security Concerns of Internet of Things (IoT), International Journal of Computer Applications, Vol. 11, 2015.

[Fayyad et al, 2006] Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P., From data mining to knowledge discovery in databases, AI Magazine, 37-54, 1996.

[Fitzek et al, 2006] Fitzek F.H.P. and Katz, M. (ed.), Cooperation in wireless networks: Principles and applications – real egoistic behavior is to cooperate!, ISBN 1-4020-4710-X, Springer, 2006.

[Fitzek et al, 2014] Fitzek, F., Toth, T., Szabados, A., Pedersen, M., Roetter, D., Sipos, M., Charaf, H. and Medard, M., Implementation and performance evaluation of distributed cloud storage solutions using random linear network coding, IEEE International Conference on Communications, pp. 249-254, 2014.

[Foley, 2012] Foley, M.J., Microsoft's Windows Phone 8 finally gets a 'real' Windows core, URL: http://www.zdnet.com/blog/microsoft/microsofts-windows-phone-8-finally-gets-a-real-windows-core/12975, 2012.

[Forstner et al, 2005] Forstner, B., Lengyel, L., Levendovszky, T., Kelényi, I. and Charaf, H. Supporting Rapid Application Development on Symbian Platform, Proc. of IEEE Eurocon 2005 The International Conference on "Computer as a tool". Beograd, Serbia, pp. 72-75, 2005.

[Fowler, 2010] Fowler, M., Domain-specific languages, Addison-Wesley Professional, 2010.

[Fragouli et al, 2006] Fragouli, C., Boudec, J.L. and Widmer, J., Network coding: An instant primer, SIGCOMM Comput. Commun. Rev., Vol. 36, Issue 1, pp. 63–68, 2006.

[Galuba et al, 2010] Galuba, W., Aberer, K., Despotovic, Z. and Kellerer, Leveraging social networks for increased bittorrent robustness, 7th IEEE Consumer Communications and Networking Conference (CCNC 2010), Las Vegas, USA, 2010.

[Gamma et al, 1995] Gamma, E., Helm, R., Johnson, R. and Vlissides, J., Design patterns: Elements of reusable object-oriented software, Addison-Wesley Professional Computing Series, 1995.

[Garbacki et al, 2006] Garbacki, P., Iosup, A., Epema, D. and Van Steen, M., 2fast: Collaborative downloads in p2p networks, 6th IEEE International Conference on Peer-to-Peer Computing, pp. 23–30, 2006.

[Gartner, 2010] Gartner survey 2010, URL: http://www.gartner.com/it/page.jsp?id=1529214, 2010.

[Gkantsidis et al, 2006] Gkantsidis, C. and Rodriguez, P., Cooperative security for network coding file distribution, IEEE Infocom, 2006.

[Gnutella] Gnutella Protocol, URL: http://www.gnu.org/philosophy/gnutella.en.html, 2014.

[Grabats Tool Contest, 2008] Grabats Tool Contest, URL: http://www.fots.ua.ac.be/events/grabats2008, 2008.

[Graph Transformation Contest, 2007] Graph Transformation Contest - UML to CSP Transformation, Applications of Graph Transformation 2007 (AGTIVE) - Graph Transformation Tool Contest, 2007.

[GEF] Graphical Editing Framework, URL: http://www.eclipse.org/gef/, 2015.

[Ghemawat et al, 2003] Ghemawat, S., Gobioff, H. and Leung, S.-T., The google file system. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, ACM, New York, NY, USA, pp. 29-43, 2003.

[GMF] Graphical Modeling Framework homepage, URL: http://www.eclipse.org/gmf, 2015.

[GReAT] GReAT: Graph Rewriting and Transformation, URL: http://www.isis.vanderbilt.edu/tools/GReAT, 2011.

[Gregory et al, 2000] Gregory, A.R., Foundations of multithreaded, parallel, and distributed programming, Addison–Wesley, ISBN 0-201-35752-6, 2000.

[Gray et al, 2004] Gray, J., Zhang, J., Roychoudhury, S. and Baxter, I., C-SAW and genAWeave: a two-level aspect weaving toolsuite, 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, ACM, New York, 2004.

[Harrison et al., 2004] Harrison, R., Symbian OS C++ for mobile phones: Programming with extended functionality and advanced features, John Wiley & Sons, ISBN 0470856114, 2004.

[Heide et al, 2011] Heide, J., Pedersen, M.V, Fitzek, F.H.P. and Médard, M., On code parameters and coding vector representation for practical rlnc, IEEE International Conference on Communications (ICC) - Communication Theory Symposium, Kyoto, Japan, 2011.

[Hillegass et al, 2012] Hillegass, A. and Conway, J., iOS programming: the big nerd ranch guide (3rd ed.). Pearson. p. 590. ISBN 978-0-321-82152-2, 2012.

[Huang et al, 2012a] Huang, J., Qian, F., Gerber, A., Mao, Z.M, Sen, S. and Spatscheck, O., A close examination of performance and power characteristics of 4G LTE networks, 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12), ACM, New York, NY, USA, pp. 225-238, 2012.

[Huang et al, 2012b] Huang, C., Simitci, H., Xu, Y., Ogus, A., Calder, B., Gopalan, P., Li, J. and Yekhanin, S., Erasure coding in windows azure storage. In: Proceedings of the 2012 USENIX Conference on Annual Technical Conference, USENIX Association, Berkeley, CA, USA, 2012.

[Imre et al, 2012] Imre, G. and Mezei, G., Parallel graph transformations on multicore systems, MSEPT 2012, pp. 86-89, 2012.

[iOS] iOS, URL: http://www.apple.com/ios/, 2013.

[ITU, 2015] Internet of Things Global Standards Initiative website, URL: http://www.itu.int/en/ITU-T/gsi/iot/, 2015.

[Kalnins et al, 2005] Kalnins, A., Barzdins, J. and Celms, E., Model transformation language MOLA, 2003 European Conference on Model Driven Architecture: Foundations and Applications (MDAFA'03), Springer-Verlag, Berlin, Heidelberg, pp. 62–76, 2005.

[Karsai et al, 2003] Karsai, G., Agrawal, A., Shi, F. and Sprinkle, J., On the use of graph transformation in the formal specification of model interpreters, Journal of Universal Computer Science, Special Issue on Formal Specification of CBS, 2003.

[Kelényi et al, 2007] Kelényi, I., Csúcs, K., Forstner, B. and Charaf, H., Peer-to-Peer file sharing for mobile devices, Mobile Phone Programming: Application to Wireless Networks, ISBN 978-1-4020-5968-1, Springer, 2007.

[Kelényi et al, 2011] Kelényi, I., SymTorrent P2P mobile application 1.50, Budapest University of Technology and Economics, URL: http://amorg.aut.bme.hu/projects/symtorrent, 2011.

[Kelly et al, 2008] Kelly, S. and Tolvanen, J.P., Domain Specific Modeling, Wiley, 2008.

[Klein et al, 1999] Klein, T., Nickel, U. A., Niere, J. and Zündorf, A., From UML to Java and back again. Technical report, University of Paderborn, 1999.

[Köhler et al, 2000] Köhler, H.J., Nickel, U.A., Niere, J. and Zündorf, A., Integrating UML diagrams for production control systems, 22nd International Conf. on Software Engineering (ICSE), ACM Press, Limerick, Ireland, pp. 241-251, 2000.

[Krohn, 2004] Krohn, M.N., On-the-fly verification of rateless erasure codes for efficient content distribution, IEEE Symposium on Security and Privacy, pp. 226–240, 2004.

[Knudsen et al., 2005] Li,S., Knudsen, J., Beginning J2ME: From Novice to Professional, Apress, ISBN 1-59059-479-7, pp. 480, 2005.

[Kulbak et al, 2010] Kulbak, Y. and Bickson, D., The emule protocol specification, 2010.

[Kundra et al, 2015] Kundra, L., Ekler, P. and Charaf, H., Orientation estimation in modern wearables with visual feature tracking, Journal on Multimodal User Interfaces, Vol. 9, 2015.

[de Lara et al, 2002] de Lara, J. and Vangheluwe, H., AToM3: A tool for multi-formalism and meta-modelling, FASE, pp. 174–188, 2002.

[de Lara et al, 2004] de Lara, H., Vangheluwe , H. and Alfonseca, M., Meta-modelling and graph grammars for multi-paradigm modelling in AToM, Software and Systems Modeling (SoSyM), Vol. 3, Issue 3, pp. 194-209, 2004.

[Lengyel et al, 2005] Lengyel, L., Levendovszky, T., Mezei, G. and Charaf, H., Control flow support in metamodel-based model transformation frameworks, Eurocon 2005 The International Conference on Computer as a Tool, Serbia-Montenegro, pp. 595-598, 2005.

[Lengyel et al, 2006] Lengyel, L., Levendovszky, T. and Charaf, H., Constraint validation in model compilers, Journal of Object Technology, pp. 107-127, 2006.

[Lengyel et al, 2008] Lengyel, L., Levendovszky, T. and Charaf, H., Validated model transformation-driven software development, International Journal of Computer Applications in Technology, pp. 106-119, 2008.

[Lengyel et al, 2015a] Lengyel, L., Ekler, P., Ujj, T., Balogh, T., Charaf, H., Szalay, Zs. and Jereb, L., ICT IN ROAD VEHICLES – The VehicleICT Platform, 4th International Conference on Models and Technologies for Intelligent Transportation Systems, Budapest, Hungary, pp. 457-462, 2015.

[Lengyel et al, 2015b] Lengyel, L., Ekler, P., Ujj, T., Balogh, T. and Charaf, H., SensorHUB – An IoT Driver Framework for Supporting Sensor Networks and Data Analysis, International Journal of Distributed Sensor Networks, Vol. 2015, Article ID 454379, 12 pages, 2015.

[Lengyel et al, 2015c] Lengyel, L., Meszaros, T., Asztalos, M., Boros; P., Mate, A., Madacs, G., Hudak, P., Kovacs, K., Tresch, A. and Charaf, H., Quality Assured Model-Driven Requirements Engineering and Software Development, The Computer Journal, Vol. 58, Issue 8, 2015.

[Lengyel and Charaf, 2015d] Lengyel, L. and Charaf, H., Open Issues in Model Transformations for Multimodal Applications, Journal on Multimodal User Interfaces, Vol. 9, 2015.

[Lengyel and Charaf, 2015e] Lengyel, L. and Charaf, H., Test-driven verification/validation of model transformations, Frontiers of Information Technology & Electronic Engineering, Vol. 16, Issue 2, pp. 85-97, 2015.

[Lédeczi et al, 2001] Lédeczi, Á., Bakay, Á., Maróti, M., Völgyesi, P., Nordstrom, G., Sprinkle, J. and Karsai, G., Composing domain-specific design environments, Computer, Vol. 34, Issue 11, pp. 44–51, 2001.

[Levendovszky et al, 2005a] Levendovszky, T., Lengyel, L., Mezei, G. and Charaf, H., A Systematic Approach to Metamodeling Environments and Model Transformation Systems in VMTS, Electronic Notes in Theoretical Computer Science, Vol. 127, pp. 65-75, 2005.

[Levendovszky and Charaf, 2005b] Levendovszky, T. and Charaf, H., Pattern Matching in Metamodel-Based Model Transformation Systems, Periodica Polytechnica Electrical Engineering, Vol. 49, Issue 1-2, pp. 87-107, 2005.

[Levendovszky et al, 2009a] Levendovszky, T., Lengyel, L. and Mészáros, T., Supporting domain-specific model patterns with metamodeling, Software and Systems Modeling, Vol. 8, Issue 4, pp. 501-520, 2009.

[Levendovszky et al, 2009b] Levendovszky, T., Mészáros, T., Ekler, P. and Asztalos, M., DSML-aided developement for mobile p2p systems, 9th OOPSLA Workshop on Domain-Specific Modeling, Orlando, USA, pp. 51-56, 2009.

[Levine, 2009] Levine, J.R., Flex & Bison: Unix text processing tools, O'Reilly, 2009.

[Li et al, 2006a] Li, S.Y.R. and Yeung, R.W., On convolutional network coding, IEEE International Symposium on Information Theory, pp. 1743–1747, 2006.

[Li et al, 2006b] Li, J. and Dabek, F., F2F: Reliable storage in open networks, 5th International Workshop on Peer-to-Peer Systems (IPTPS 2006), 2006.

[Li et al, 2010] Li, Y., Soljanin, E. and Spasojevic, P., Collecting coded coupons over overlapping generations, IEEE International Symposium on Network Coding (NetCod), pp. 1–6, 2010.

[Li et al, 2011] Li, X., Mow, W.H. and Tsang, F.L., Singularity probability analysis for sparse random linear network coding, IEEE International Conference on Communications (ICC), 2011.

[Liao et al, 2007] Liao, W., Papadopoulos, F. and Psounis, K., Performance analysis of bittorrent-like systems with heterogeneous users. Performance Evaluation, Vol. 64, Issue 9-12, pp. 876–891, 2007.

[Ma et al, 2009] Ma, Z. and Qiu, D., A novel optimistic unchoking algorithm for bittorrent, 6th IEEE Consumer Communications and Networking Conference (CCNC 2009), pp. 1–4, 2009.

[Mankin et al, 1998] Mankin, A., Romanow, A., Bradner, S. and Paxson, V., IETF Criteria for evaluating reliable multicast transport and application protocols, RFC 2357, 1998.

[Mannadiar, 2012] Mannadiar, R., A multi-paradigm modelling approach to the foundations of domain-specific modelling, PhD thesis, McGill University, 2012.

[MATLAB] MATLAB homepage, URL: http://www.mathworks.com, 2015.

[Maurice et al, 2008] Maurice, H.P. and Nir S.N., The art of multiprocessor programming, Morgan Kaufmann, ISBN 0-12-370591-6, 2008.

[Mens et al, 2006] Mens. T. and Van Gorp, P., A taxonomy of model transformation, Electron. Notes Theoretical Computer Science, Vol. 152, pp. 125-142, 2006.

[Metzger, 2005] Metzger, A., A systematic look at model transformations, Model-Driven Software Development, Vol. 2 of Research and Practice in Software Engineering, Springer, 2005.

[Meulpolder et al, 2009] Meulpolder, M., Pouwelse, J., Epema, D. and Sips, H., Modeling and analysis of bandwidth-inhomogeneous swarms in bittorrent, IEEE Ninth International Conference on Peer-to-Peer Computing (P2P'09), pp. 232–241, 2009.

[Meyer, 1988] Meyer, B., Object-oriented software construction, Prentice Hall, New York, 1988.

[Mészáros et al, 2009] Mészáros, T., Mezei, G. and Charaf, H., Engineering the dynamic behavior of metamodeled languages, Simulation-Transactions of the Society for Computer Simulation International, pp. 793-810, 2009.

[Mészáros et al, 2010] Mészáros, T., Mezei, G., Levendovszky, T. and Asztalos, M., Manual and automated performance optimization of model transformation systems, International Journal on Software Tools for Technology Transfer, Vol. 12, Issue 3-4, pp. 231-243, 2010.

[Mezei et al, 2006] Mezei, G., Lengyel, L., Levendovszky, T. and Charaf, H., A model transformation for automated concrete syntax definitions of metamodeled visual languages, Electronic Communications of the EASST, pp. 1-12, 2006.

[Mezei et al, 2007] Mezei, G., Levendovszky. T. and Charaf, H., Formalizing the Evaluation of OCL Constraints, Acta Polytechnica Hungarica, Vol. 4, Issue 1, pp. 89-110, 2007.

[Michiardi et al, 2007] Michiardi, P. and Urvoy-Keller, G., Performance analysis of cooperative content distribution in wireless ad hoc networks, 4th Annual Conference on Wireless on Demand Network Systems and Services (WONS'07), pp. 22–29, 2007.

[MIC Final Report, 2009] Mobile Innovation Centre (MIC) Final Report, NKTH Asbóth Oszkár Program, Budapest University of Technology and Economics, 2009.

[Microsoft T4] Microsoft T4, Code Generation and T4 Text Templates, URL: http://msdn.microsoft.com/en-us/library/bb126445.aspx, 2013.

[Molnár et al, 2007] Molnár, B., Forstner, B. and Kelényi, I., Symella P2P mobile application (1.40), Budapest University of Technology and Economics, URL: http://amorg.aut.bme.hu/projects/symella, 2007.

[Mono] Mono Project homepage, URL: http://www.mono-project.com, 2015.

[Montresor et al, 2009] Montresor, A. and Jelasity, M., Peersim: A scalable p2p simulator, IEEE 9th International Conference on Peer-to-Peer Computing (P2P'09), pp. 99–100, 2009.

[Nurminen et al, 2008] Nurminen, J. and Noyranen, J., Energyconsumption in mobile peer-to-peer - quantitative results from file sharing, 5th IEEE Consumer Communications and Networking Conference (CCNC 2008), pp. 729 –733, 2008.

[Nurminen et al, 2009] Nurminen, J. and Kelényi, I., A method and device for network messaging, Patent Application US 20090252071, 2009.

[Nurminen, 2010] Nurminen, J. K., Parallel connections and their effect on the battery consumption of a mobile phone, IEEE Consumer Communications and Networking Conference (CCNC), 2010.

[Nurminen et al, 2010] Nurminen, J. and Kelényi, I., Method and apparatus for controlling energy consumption during resource sharing. Patent Application US 20100191994, 2010.

[OMG] OMG Homepage, URL: http://www.omg.org/, 2015.

[OMG OCL] OMG Object Constraint Language Specification, Object Management Group URL: http://www.omg.org/spec/OCL/2.2/PDF, 2010.

[OMG QVT] OMG Query/View/Transformation (QVT) specification, Meta Object Facility 2.0 Query/Views/Transformation Specification, OMG doc. ptc/07-07-07, URL: http://www.omg.org/, 2007.

[OMG MDA] OMG Model-Driven Architecture (MDA) specication, OMG document ormsc/01-07-01, URL: http://www.omg.org/, 2001.

[OMG UML] OMG UML specification, version 2.5, OMG document formal/15-03-01, URL: http://www.uml.org/, 2015.

[Pahlevani et al, 2014] Pahlevani, P., Hundebøll, M., Pedersen, M.V., Lucani, E.D, Charaf, H., Fitzek, F.H.P, Bagheri, H. and Katz, M., Novel Concepts for Device to Device Communication using Network Coding, IEEE Communications Magazine Vol. 52, Issue 4, pp. 32-39., 2014.

[Pagani et al, 1999] Pagani, E. and Rossi, G.P., Providing reliable and fault tolerant broadcast delivery in mobile ad-hoc networks, Mobile Networks and Applications, Vol. 4, Issue 3, pp. 175–192, 1999.

[Pedersen et al, 2009] Pedersen, M.V, Heide, J., Fitzek, F.H.P. and Larsen, T., Pictureviewer - a mobile application using network coding, 15th European Wireless Conference (EW), Aalborg, Denmark, 17-20 2009.

[Pedersen et al, 2010] Pedersen, M.V., Heide, J., Vingelmann, P., Blázovics, L. and Fitzek, F.H.P., Multimedia cross-platform content distribution for mobile peer-to-peer networks using network coding, International Conference on Multimedia (MM'10), Firenze, Italy, pp. 1091-1094, 2010.

[PhoneGap] PhoneGap homepage, URL: http://phonegap.com/, 2013.

[Pouwelse et al, 2008] Pouwelse, J., Garbacki, P., Wang, J., Bakker, A., Yang, J., Iosup, A., Epema, D., Reinders, M., van Steen, M. and Sips, H., Tribler: a socialbased peer-to-peer system, Concurrency and Computation, Vol. 20, pp. 127, 2008.

[Pándi and Charaf, 2013] Pándi, K. and Charaf, H., Performance Metrics Based Mobile, Acta Cybernetica, Vol. 21, Issue 1, pp. 165-176, 2013.

[Pándi and Charaf, 2015] Pándi, K. and Charaf, H., Mobile Resource Management Load Balancing Strategy, Acta Cybernetica, Vol. 22, Issue 1, pp. 171-181, 2015.

[Rajagopalan et al, 2006] Rajagopalan, S. and Shen, C., A cross-layer decentralized bittorrent for mobile ad hoc networks, IEEE 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services, pp. 1–10, 2006.

[Richardson et al, 2007] Richardson, L., Sam, R., RESTful web service, O'Reilly Media, 2007.

[Rogers et al, 2007] Rogers, M. and Bhatti, S., How to disappear completely: A survey of private peer-to-peer networks, In SPACE, 2007.

[Rozenberg, 1997] Rozenberg, G. (ed.), Handbook on graph grammars and computing by graph transfor-mation: Foundations, Vol. 1, World Scientific, Singapore, 1997.

[Rubino, 2012] Rubino, D., Overview and review of Windows Phone 8, URL: http://www.wpcentral.com/overview-and-review-windows-phone-8, 2012.

[Rud, 2009] Rud, O., Business intelligence success factors: Tools for aligning your business in the global economy, Hoboken, N.J, Wiley & Sons, ISBN 978-0-470-39240-9, 2009.

[Sandberg, 2006] Sandberg, O., Distributed routing in small-world networks, 8th Workshop on Algorithm Engineering and Experiments, 2006.

[Sandvine, 2012] Sandvine's global internet phenomenon report: 1h 2012, URL: http://www.sandvine.com/news/global_broadband_trends.asp, 2012.

[Sathiamoorthy et al, 2013] Sathiamoorthy, M., Asteris, M., Papailiopoulos, D., Dimakis, A.G., Vadali, R., Chen, S. and Borthakur, D., Xoring elephants: novel erasure codes for big data, Proceedings of the 39th International Conference on Very Large Data Bases, PVLDB'13, VLDB Endowment, pp. 325–336, 2013.

[Schürr and Selic, 2009] Schürr, A. and Selic, B., As opposed to the original designers of UML, Model Driven Engineering Languages and Systems, Lecture Notes in Computer Science, 2009.

[Sbai et al, 2008] Sbai, M., Barakat, C., Choi, J., Hamra, A. and Turletti, T., Adapting bittorrent to wireless ad hoc network,. Ad-hoc, Mobile and Wireless Networks, pp. 189–203, 2008.

[Sendall et al, 2003] Sendall, S. and Kozaczynski, W., Model transformation: The heart and soul of model-driven software development, IEEE Software, Vol. 20, Issue 5, pp. 42-45, 2003.

[SensorHUB] The SensorHUB project website, URL: https://www.aut.bme.hu/SensorHUB/, 2015.

[Steinwurf, 2015] Steinwurf ApS, http://www.steinwurf.com, 2015.

[Shixing et al, 2011] Shixing, L., Hong, W., Tao, X. and Guiping, Z., Application Study on Internet of Things in Environment Protection Field, Lecture Notes in Electrical Engineering, Vol. 133: 99–106, 2011.

[Siekkinen et al, 2012] Siekkinen, M., Hiienkari, M., Nurminen, J. K. and Nieminen, J., How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4, In Wireless Communications and Networking Conference Workshops, IEEE, pp. 232-237, 2012.

[Silva et al, 2009] Silva, D., Zeng, W. and Kschischang, F.R., Sparse network coding with overlapping classes, Workshop on Network Coding, Theory, and Applications (NetCod), pp. 74 –79, 2009.

[Simulink] Simulink homepage, URL: http://www.mathworks.com/simulink/, 2015.

[Sprinkle et al, 2004] Sprinkle, J. and Karsai, G., A domain-specific visual language for domain model evolution, Journal of Visual Languages and Computing, Vol. 15, Issue 2, 2004.

[Sztipanovits et al. 1997] Sztipanovits, J., Karsai, G., Model-Integrated Computing, IEEE Computer, pp. 110-112, 1997.

[Sztipanovits, 1998] Sztipanovits, J., Integrated Engineering of Computer-Based Systems. IEEE Computer, Vol. 31, Issue 12, pp. 68-69, 1998.

[Sztipanovits et al, 2002] Sztipanovits, J. and Karsai, G., Generative programming for embedded systems, Generative Programming and Component Engineering (GPCE), Vol. 2487 of LNCS, Springer-Verlag, Pittsburgh, Pennsylvania, pp. 32-49, 2002.

[SymTorrent] SymTorrent homepage, URL: http://symtorrent.aut.bme.hu, 2012.

[Taentzer, 2004] Taentzer, G., AGG: A graph transformation environment for modeling and validation of software, Application of Graph Transformations with Industrial Relevance (AGTIVE 2004) , Springer, Vol. 3062 of LNCS, pp. 446-453, 2004.

[Thai et al, 2003] Thai, T. and Lam, H., .NET framework essentials, O'Reilly, 2003.

[Tolvanen, 2006] Tolvanen, J.P., MetaEdit+: integrated modeling and metamodeling environment for domain-specific languages, Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications (OOPSLA '06), ACM, New York, NY, USA, pp. 690–691, 2006.

[Ujhelyi et al, 2015] Ujhelyi, Z., Bergmann, G., Hegedüs, Á., Horváth, Á., Izsó, B., Ráth, I., Szatmári, Z. and Varró, D., EMF-IncQuery: An integrated development environment for live model queries, Science of Computer Programming, Vol. 98, pp. 80-99, 2015.

[uTorrent] uTorrent BitTorrent client, URL: http://www.utorrent.com, 2015.

[Vajk et al, 2009] Vajk, T., Kereskényi, R., Levendovszky,T. and Lédeczi, Á., Raising the abstraction of domain-specific model translator development, 16th IEEE Int. Conf. and Workshop on the Engineering of Computer Based Systems, San Francisco, USA, pp. 31-37, 2009.

[Vajk et al, 2012] Vajk, T., Deák, L., Mezei, G. and Levendovszky, T., Performance evaluation of model-based data access layers in NoSQL databases, 1st Workshop on MDE for and in the Cloud (CloudMDE 2012), Denmark, 2012.

[Varró et al, 2003] Varró, D. and Pataricza, A., VPM: A visual, precise and multilevel metamodeling framework for describing mathematical domains and UML, Journal of Software and Systems Modeling, 2003.

[VehicleICT] The VehicleICT project website, URL: https://www.aut.bme.hu/VehicleICT/, 2015.

[Verdú et al, 2012] Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., de Castro, J. P. and Queirós, R., A distributed system for learning programming on-line. Computers & Education, Vol. 58, Issue 1, pp. 1-10, 2012.

[Vingelmann et al, 2011] Vingelmann, P., Fitzek, F.H.P., Pedersen, M.V., Heide, J. and Charaf, H., Synchronized multimedia streaming on the iphone platform with network coding, IEEE Communications Magazine, pp. 126-132, 2011.

[Vingelmann et al, 2012] Vingelmann, P., Pedersen, M.V., Heide, J., Zhang, Q. and Fitzek, F.H.P., Data dissemination in the wild: A testbed for high-mobility MANETs, IEEE ICC 2012 - Ad-hoc and Sensor Networking Symposium, Ottawa, Canada, 2012.

[Vision, 2012] Vision: Developer Economics 2012, URL: http://www.visionmobile.com/devecon, 2012.

[VMTS] Visual Modeling and Transformation System, URL: http://www.aut.bme.hu/vmts, 2015.

[Visual Studio] Visual Studio homepage, URL: https://www.visualstudio.com, 2015.

[Vuze] Vuze homepage, URL: http://www.vuze.com/, 2013.

[Wang et al, 2010] Wang, J., Shen, R., Ullrich, C., Luo, H. and Niu, C., Resisting free-riding behavior in bittorrent, Future Generation Computer Systems, Vol. 26, Issue 8, pp. 1285–1299, 2010.

[Weiyan, 2014] Weiyan Wang, H.K., Saving Capacity with HDFS RAID, https://code.facebook.com/posts/536638663113101/saving-capacity-with-hdfs-raid, 2014.

[WP] Windows Phone, URL: http://www.windowsphone.com, 2012.

[Xamarin] Xamarin homepage, URL: http://xamarin.com/, 2013.

[Xia et al, 2003] Xia, Y. and Glinz, M., Rigorous EBNF-based definition for a graphic modeling language, 10th Asia-Pacific Software Engineering Conference Software Engineering Conference (APSEC ’03), pp. 186–196, Washington, DC, USA, 2003.

[Xia et al, 2010] Xia, R. and Muppala, J., A survey of bittorrent performance, IEEE Communications Surveys & Tutorials, Vol. 12, Issue 2, pp. 140–158, 2010.

[Xiao et al, 2010] Xiao, Y., Bhaumik, R., Yang, Z., Siekkinen, M., Savolainen, P. and Yla-Jaaski, A., A system-level model for runtime power estimation on mobile devices, IEEE/ACM International Conference on Green Computing and Communications, 2010.

[Xtext] Xtext home page, URL: http://www.eclipse.org/Xtext/, 2013.

[Yamazaki et al, 2007] Yamazaki, S., Tode, H. and Murakami, K., Cat: A cost-aware bittorrent, 32nd IEEE Conference on Local Computer Networks (LCN 2007), pp. 226–227, 2007.

[Yao et al., 2004] Yao, P. and Durant, D., .NET compact framework programming with C#, Addison-Wesley Professional, 2004.

[Zakerinasab et al, 2012] Zakerinasab, M.R. and Wang, M., An update model for network coding in cloud storage systems, 50th Annual Allerton Conf. on Communication, Control, and Computing, 2012.

[Zanella et al, 2014] Zanella, A., Bui, N., Castellani, A., Vangelista, L. and Zorzi, M. Internet of Things for Smart Cities, IEEE Internet of Things Journal, Vol. 1(1), 2014.

[Zeigler et al, 2000] Zeigler, B.P., Praehofer, H. and Kim, T.G., Theory of modeling and simulation, Second Edition, Academic Press, 2000.

[Zhang et al, 2007a] Zhang, Q. and Fitzek, F.H.P., Cognitive wireless networks – Cooperative retransmission for reliable wireless multicast services, Springer, ISBN 978-1-4020-5978-0 25, pp. 485–498, 2007.

[Zhang et al, 2007b] Zhang, L., Muppala, J. and Tu, W., Exploiting proximity in cooperative download of large files in peer-to-peer networks, Second International Conference on Internet and Web Applications and Services (ICIW’07), pp. 1–1, 2007.