Frameworks of Process Improvement for Mobile Applications

Haeng-Kon Kim

***Abstract*—Mobile application development belongs on the complicated, but quite regular software development environment that includes many different solutions possibilities in the development. The experiences from the mobile card game development lead to the development process that covers many distribution channels (different client platforms). Architecture and the readiness to react changes that were caused by technology requirement changes become one of the key points in successful development. Changes that came during the application development required quite clear architecture. By identifying, standardizing, and naming general process models, same models and code could be reused in several applications.**

In this paper we go through mobile development process and architectural structures and analysis of these with empirical mobile application development. We used different architectural views for analyzing mobile architectures and architecture role in development. The architecture and architectures role on the development has been studied in mobile application-and multi-platform service development.

***Index Terms*— Mobile Application, Mobile Architecture, Embedded Software, MDA (Model Driven Development), Model, Frameworks, Embedded Development Process**

I. INTRODUCTION

Embedded software applications are typically more difficult to design and build, both because of the problem domain and the constraints placed on them by the target environment. Embedded systems usually have limited memory and CPU processing power; these constraints are likely due to size, cost, power efficiency, and heat generation limitations. In addition, studies have shown that many embedded projects either fail altogether, or fail to meet the requirements set for them. Embedded system design needs to model together application and hardware architecture. For that a huge number of models are available, each one proposing its own abstraction level associated to its own software platform for simulation or synthesis. To produce a co-design framework, we are obviously obliged to support different models among all possible ones. Between these models we should produce automatic transformations. Each time a new model is included in the framework, we should develop a new transformation[1], [2]. To improve transformation engine development, Model Driven Architecture (MDA) techniques are useful. This approach permits to define the transformations at the meta model level. It guaranties to the framework the reuse of models and unifies the definition of the transformation rules. Modeling is frequently used in other engineering domains as a way to verifying design decisions before committing large resources to constructing or manufacturing a product. Models allow you to perform thought experiments on a design, and to try different approaches to solving the problem at a higher level of abstraction than the raw medium of the constructed product. In software this is not typically the case: although modeling tools have been around for decades, the desire, capability, and modeling tool maturity have not been sufficient. This is no longer the case, and using models -- not only to think about the problem but also to drive the design and construction of software -- is a reality. MDD tools can take models at a high level of abstraction and generate executable and efficient source code. Models (in particular visual ones) provide rich information on the structure and behavior of the system. Sharing and working with models on a software development team is now more efficient and less error-prone than working only in source code.

Embedded systems vary in size and complexity, and might include such divergent things as[3], [4]:

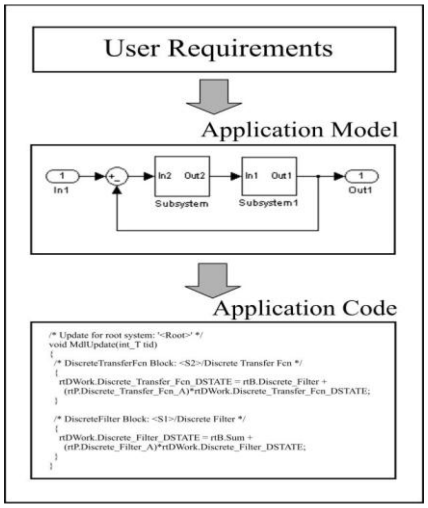
- Small applications embedded in a microcontroller running the interior lights in a car
- Large, multiprocessor telephone switches requiring millions of lines of code

Although the use of models is beneficial to both types, there are certain trade-offs required when using MDD on such systems. Small and resource-constrained systems may have such limited memory that using a Real-Time Operating System (RTOS) becomes prohibitive. In such cases, operating system services like threads, processes, or tasks are not available. Modeling tools that provide MDD capabilities need to take these diverse requirements into account. This article discusses the use of MDD techniques on various resource-constrained embedded systems, and looks at the trade-offs required in order to make MDD beneficial.

In this paper, we propose a framework of development process for the mobile embedded software, which is shows good

Heang-Kon Kim is with the Department of Computer Engineering, Catholic University of Daegu KyungSan, Daegu, 712-702, Korea (corresponding author to provide phone: 053-850-2743; fax: 053-850-2740;
e-mail: hangkon@ cu.ac.kr).

“This research was supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement)” (IITA-2008-(C1090-0801-0032))



```
/* Update for root system: '<Root>' */
void MdlUpdate(int_T tid)
{
  /* DiscreteTransferFcn Block: <S2>/Discrete Transfer Fcn */
  {
   rtDWork.Discrete_Transfer_Fcn_DSTATE = rtB.Discrete_Filter +
     (rtP.Discrete_Transfer_Fcn_A)*rtDWork.Discrete_Transfer_Fcn_DSTATE;
  }

  /* DiscreteFilter Block: <S1>/Discrete Filter */
  {
   rtDWork.Discrete_Filter_DSTATE = rtB.Sum +
     (rtP.Discrete_Filter_A)*rtDWork.Discrete_Filter_DSTATE;
  }
}
```

Fig. 1. Model Driven Approach for Mobile Applications

development utilization and filtering efficiency as well. The proposed process for embedded software uses multiple domain applications and achieves small number of the efficient real practices. In order to evaluate the productivity and quality of the proposed process development method, several experiments are conducted on the SPIC 2008 benchmark data[5]. The results of the experiments clearly show that we can decide a suitable process for the mobile devices and the proposed methods are also expected to be important technology for the mobile embedded development to be wide used recently.

II. RELATED WORKS

A. MODEL and REUSE-DRIVEN APPROACH

Their efficient development requires the Two approaches have emerged to tackle the use of techniques that can model entire families variability problem in the embedded world: the rather than just individual applications. *model-driven* and the *reuse-driven* approaches. With the former approach (see Figure 1), the application The non-functional challenge has long been requirements are expressed in a modelling recognized. It is addressed by several research environment that is capable of automatically generating the application code. The prototypical example of such an environment is the Matlab tool suite. The increase in efficiency arises from the fact that the software design and implementation phases are automated and that the control engineer can take direct control of the software development process without having to resort to the intermediary services of a software engineer.

In a reuse-driven approach instead (see Figure 2), the application code is built by configuring and composing a set of pre-defined software building blocks. The increase in efficiency now arises from the possibility of reusing existing software artifacts (modules, components, code fragments, etc.). Traditionally, the reuse-driven approach was implemented by developing libraries of reusable modules. More recently,

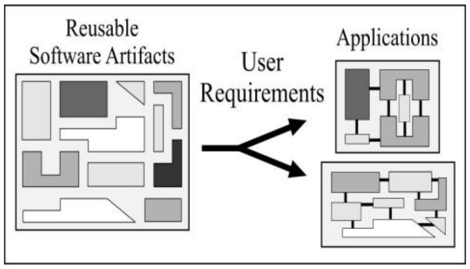


Fig. 2. Reuse Approach for Mobile Applications

software product families (Bosch, 2000) and software frameworks [6], [7] have emerged as more convenient reuse vehicles that allow reuse to take place at architectural as well as at the code level.

The problem for control engineers is that their applications tend to be multi-domain. A complete control application does not simply cover implementation of control laws. In fact, in most cases, the implementation of control laws is only a small fraction of the total control software. Most of the software normally is concerned with functionalities such as management of external sensors and actuators, management and generation of housekeeping data, management and processing of commands from some supervisory unit, implementation of failure detection and identification logic, implementation of failure recovery actions. A reuse approach may then be more appropriate. The cost of developing reusable building blocks is lower than the cost of developing model-driven tools and a reuse-driven approach is therefore affordable even for niche products – as most industrial control systems are. The reusable blocks can moreover be more easily targeted to the specific needs of their users and can therefore more easily match the often idiosyncratic needs of control applications.

B. Modeling Concurrency in UML

UML 1.x did not provide an easy way to represent a composite object (that is, one that contains other objects in a containment relationship). Real systems are typically made of smaller subsystems, and these subsystems are connected together in a unique way in order to satisfy the system requirements. UML 2.0 has addressed this by providing structured classes. UML 2.0 provides a powerful building block for system design through structured classes. In particular, the structured class concept can be specialized to model active objects -- with their own thread of control -- that communicate with other active objects, asynchronously, using message queues. These active objects address a key need of the embedded and real-time developer: the ability to model complex and concurrent objects (such as interfaces and devices) in the problem domain.

Contained parts of a structured class communicate via ports. Ports allow for two-way communication between objects that consist of *required* (incoming) and provided (outgoing) interfaces. Ports are wired together using a connector; once wired together, objects can communicate via the defined

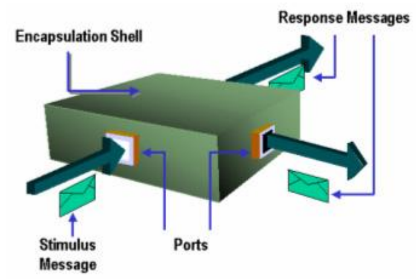


Fig.3. The active pattern for Mobile Application

interfaces. Ports also allow objects to be decoupled and plugged in anywhere a compatible set of connections is available. Active objects, furthermore, extend this capability by formalizing the communication. The message "format" that is used by ports on active objects is defined using protocols, which are the signals that are accepted (required) by and sent (provided) on a port.

The use of signals is analogous to events in the modeled system, and therefore useful for modeling the event-driven domain of embedded and real-time software.

Active objects have their own thread of control: they have an encapsulation boundary that protects the inner behavior and state and an external interface defined by ports (see Figure 3). This approach is similar to hardware design, where engineers build systems from components (for example, logic chips, ASICs, or micro-controllers) that have established interfaces (pin outs), and connect them together on a schematic diagram.

III. Mobile Embedded Software Development Process

MES(Mobile Embedded Software) can be considered as a particular case of embedded systems. MES design covers a lot of different viewpoints including as much the application modeling by the aggregation of functional components, as the assembly of existing physical components, as the verification and the simulation of the modeled system, as the synthesis of a complete end-product integrated into a single chip. As a rule a MES includes programmable processors, memory units (data/instructions), interconnection mechanisms and hardware functional units. These components can be generated for a particular application; they can also be obtained from IP (Intellectual Property) providers. The ability to re-use software or hardware components is without any doubt a major asset for a codesign system. We propose the MES development process as in figure 4.

(1) **Capture User Requirements**: The objective of this phase is to elicit, agree and document the customer requirements that the software system needs to fulfill. This includes establishing a common understanding with the customer on functional and non-functional requirements for MES domain. This phase includes the following activities: formalize the customer requirements in an MES Application Model and derive an initial Application PIM for MES and an initial functional requirements specification from the common infrastructure of reusable assets for MES.

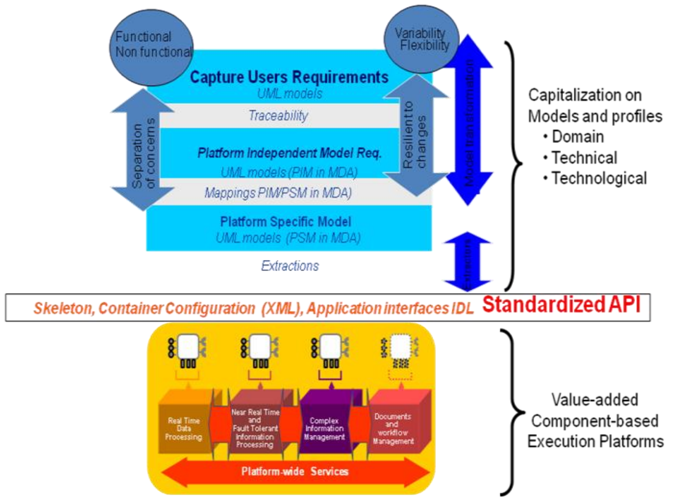

Functional
Non functional
Variability
Flexibility
Capture Users Requirements
UML models
Traceability
Platform Independent Model Req.
UML models (PIM in MDA)
Mappings PIM/PSM in MDA)
Platform Specific Model
UML models (PSM in MDA)
Extractions
Separation of concerns
Resilient to changes
Model transformation
Extractions
Capitalization on Models and profiles
• Domain
• Technical
• Technological
Skeleton, Container Configuration (XML), Application interfaces IDL
Standardized API
Real Time Data Processing
Near Real Time and Fault Tolerant Information Processing
Complex Information Management
Documents and workflow Management
Platform-wide Services
Value-added Component-based Execution Platforms

Fig.4. Mobile Embedded Development Process proposed

(2) **PIM Context Definition**: The objective of this phase is to clearly define the scope of the software system to be developed. The result is an unambiguous definition of the system, its objectives, and scope following a black-box approach. Main activities are:

- Establish the system goals and business principles.
- Describe the external actors that interact with the system.
- Identify the high-level services offered by the system and their key behavior.
- Define the business events, and exchanged business objects.

(3) **PIM Requirements Specification**: The objective of this phase is to build a model of customer requirements clear and complete and to have a unique requirements description that all subsequent models will use. In order to model the system functional and non-functional requirements, the main activities of this phase are:

- Refine the PIM Context
- Identify services, events and business objects produced and consumed by the system and the actors interacting with the system
- Specify capabilities (use cases), forces (non-functional requirements), and atomic requirements
- Identify and model the relationships between functional and nonfunctional requirements.

(4) **PIM Analysis**: The objective of this phase is to model the internal view of the system without any technological consideration and maintaining the separation of concerns between functional and non-functional aspects. The main activities of this phase are:

- Describe the system functionalities: the objects (with classes, attributes, packages, etc.), the functions (with operations), the system boundary (with interfaces), the behavior (with sequence diagrams), etc.
- Describe the system QoS aspects (refine the classes) and their application to the functional elements of the model.
- Maintain traceability with the Requirements PIM.

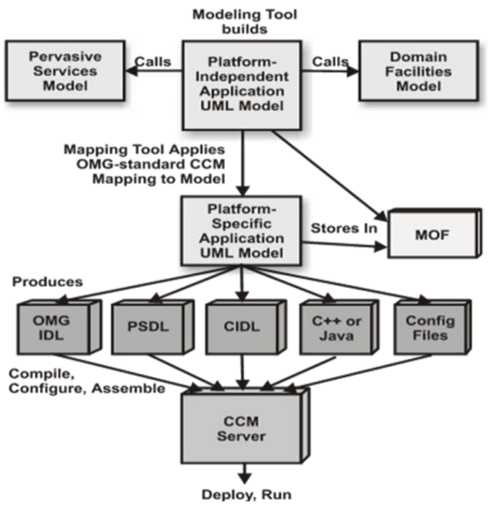

Modeling Tool builds
Pervasive Services Model
Calls
Platform-Independent Application UML Model
Calls
Domain Facilities Model
Mapping Tool Applies OMG-standard CCM Mapping to Model
Platform-Specific Application UML Model
Stores In
MOF
Produces
OMG IDL
PSDL
CIDL
C++ or Java
Config Files
Compile, Configure, Assemble
CCM Server
Deploy, Run

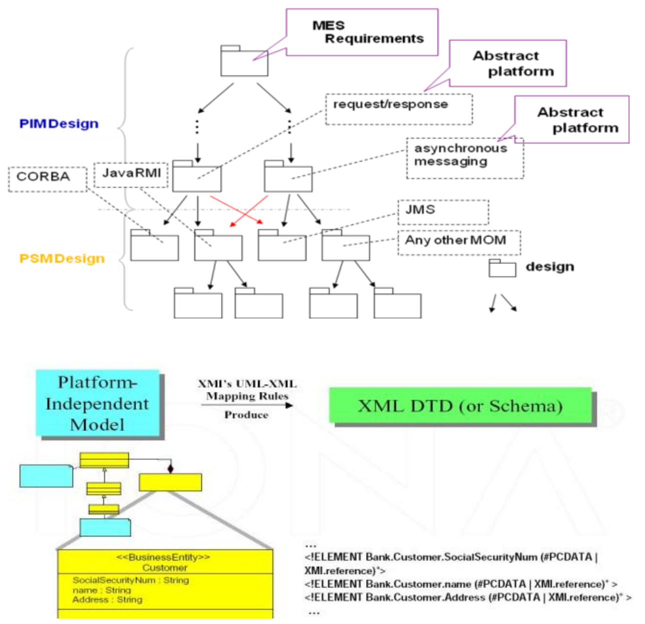
Fig.5. Upper Layer in Mobile Embedded Development Process

(5) **Design**: The objective of this phase is to model the detailed structure and behavior of the solution (software application) that fulfils the system functional and non-functional requirements. This implies making decisions on how the system will be implemented and which architectural style, patterns, standards and platforms will be used. Following an MDA approach, the design is performed in two steps:

- Specify and design a platform-independent solution (how) for all the requirements (what). The PIM will be defined with different elements depending on the architectural style selected for the solution, e.g., for a Components Design PIM the solution is expressed in terms of software components (component, interface, port, connector).
- Specify and design the platform-specific solution by refining the platform-independent solution. The PSM is intended to be automatically derived from the PIM through transformation engines. The PSM contains models specific of the platform (e.g., CCM, EJB, .NET) and is detail and complete enough to allow the codification and deployment of the solution as in the figure 6.

(6) **Coding & Integration**: The objective of this phase is to develop and verify the software code that implements the software design fulfilling the software requirements. This phase includes activities such as: develop the components and classes (according to the models used as inputs), define the organization of the code, execute unit tests, and integrate components and subsystems. Following a MDA approach, the code is intended to be automatically produced from the PSM through transformation engines.

(7) **Testing**: The objective of this phase is to demonstrate that the final software system satisfies its requirements. This phase includes activities such as: plan tests, prepare test model, test cases and test scripts, execute tests, correct defects and document testing results. Test models are traceable to PIM models (specially to PIM Requirements) and, following an MDA approach, test models will be refined from the PIM and test cases and test scripts will be automatically produced from the test model through transformation engines.



```
...
<!ELEMENT Bank.Customer.SocialSecurityNum (#PCDATA |
XMI.reference)*>
<!ELEMENT Bank.Customer.name (#PCDATA | XMI.reference)* >
<!ELEMENT Bank.Customer.Address (#PCDATA | XMI.reference)* >
...
```

Fig.6. Lower Layer in Mobile Embedded Development Process

(8) **Deployment**: The objective of this phase is to ensure a successful transition of the developed system to the final users (including resources, environment, schedule planning and execution). This phase includes activities such as: create a deployment plan (dates of installation, resources, etc.), create a deployment model (derived from the PSM Deployment model and adapted to the specific execution environment of the customer), create the product manuals, maintain records of the product that is being delivered to the client, and provide the installation of the product in the client premises.

IV. Evaluation

We propose the case study investigated possible configurations, with each one targeted for specific platform capabilities. The use of MDD for embedded systems is not an all-or-nothing proposition. Models are still applicable for resource-constrained devices if the use of models and subsequent code generation can provide productivity benefits.

The case study illustrates the trade-offs needed to support mobile application classes of embedded systems. The design activities match a successive refinement process between each domain according to various abstraction levels. The design flows from the functional domain, to the structural domain, finally to the physical domain, and so on while going down in the abstraction levels. This abstract model illustrates the interoperability requirements for different software applications owned by the same or different enterprises that are expected to interact with each other. It should be noted that the CIM model is abstract enough to encompass any system architecture, including both client-server and distributed approaches.

The use of MDA describes and represents interactions between entities, and not the way in which they will be implemented. The choice of architecture and components that will be utilized is up to the next model level.

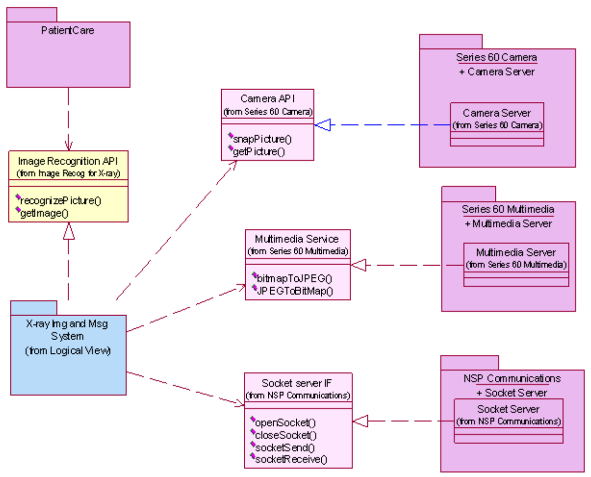


Fig.7. Lower Layer in Mobile Embedded Development Process

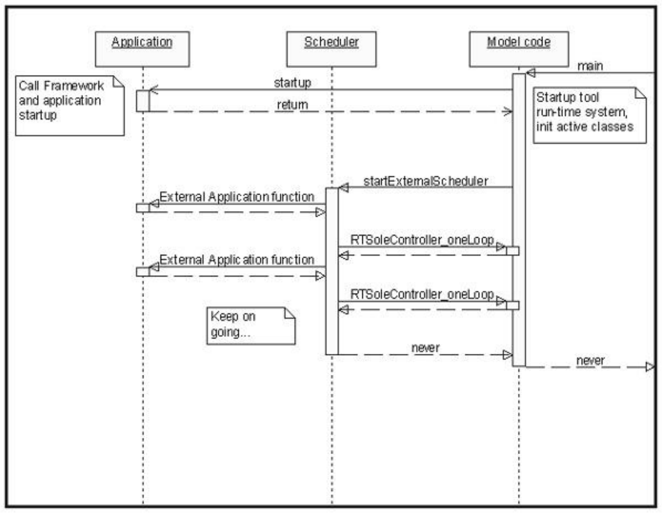

Application
Scheduler
Model code
main
Call Framework and application startup
startup
return
Startup tool run-time system, init active classes
startExternalScheduler
External Application function
RTSoleController_oneLoop
External Application function
RTSoleController_oneLoop
Keep on going...
never
never

Fig.8. Changing Thread of Control

(1) Capture the user requirements: Small Mobile complex embedded systems.

The best approach for designing large and complex systems (with several hundred active objects) is to use Rational Rose RealTime and C++ code generation, with a threaded runtime system. The usage of C++ gives you all the benefits of classes and inheritance (class, public, friend, and private), while the model code generator can make use of these C++ features to provide the most flexible modeling constructs (for instance, dynamic structure and multiple containment). You can therefore create abstract and derived objects, minimizing the design and implementation effort. In this configuration, the use of active objects, TargetRTS, and underlying RTOS allows the designer to take full advantage of the MDD tooling.

The multiplicity of the abstraction levels is appropriate to the modeling approach. The MES requirements with context model are used with a different viewpoint for each abstraction class in PIM level as figure 7.   This information is defined only once in a single model. The links or transformation rules between the abstraction levels permit the re-use of the concepts for a different purpose.

The external scheduler has control, and lets the Rational Rose RealTime TargetRTS process events one by one (at whatever time the scheduler decides). The model code contains all of the event-driven applications that are designed in Rational Rose RealTime. The scheduler shown above also calls external application functions that are designed outside the toolset (if applicable). An example of such an application could be a motor control loop, which is actually more frequency-driven than event-driven; the correct functioning depends on an exact scheduling frequency (for example, once per two ms) rather than a guaranteed maximum latency while reacting to an asynchronous event as in figure 8.

(2) System Decomposition

When decomposing a system with active classes on resource-constrained targets, you must find a balance regarding the percentage of active classes in the system. Active classes, although useful, do have memory and performance overhead associated with their use. A number of system services and basic applications at the bottom layers of a system model usually don't have to be active at all. Most engineers design application modules implementing the main functionality of a system, which should benefit from active objects due to the improved productivity from their use. Many of the details of concurrency, data access, and communication are accomplished using this paradigm. In Rational Rose RealTime, each state transition trigger for an active object must be triggered by an asynchronous event. Examples of such events, which are typically sent from the bottom layers of your system model, include:

- Action available
- End speed Mobile controller reached
- A simple external port became active

These events could arrive from other active classes, but also from passive classes using the external port described earlier (as shown in Figure 9). You should distinguish between active and

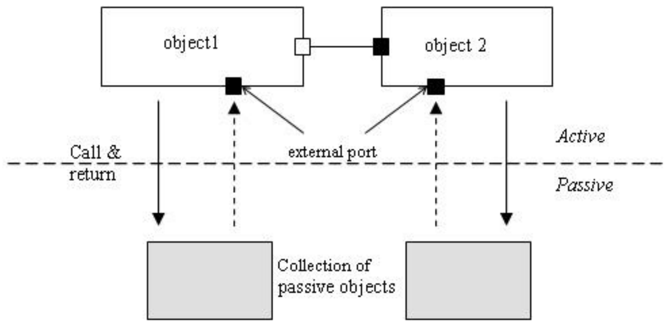


Fig.9. System decomposition example

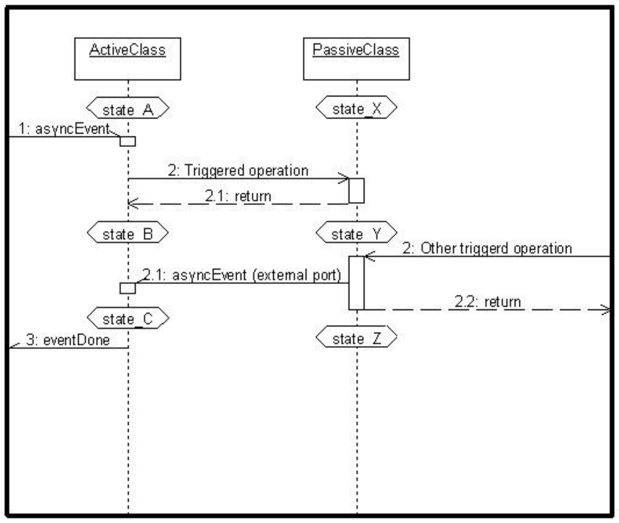


Fig.10. Interaction scenario between active and passive

passive classes early in the system design. The advantage of using passive classes is an improved memory footprint and better performance, and that they can also use state machines to describe their behavior. A common pitfall in using passive classes, on the other hand, is the fact that you may need to design some low-level services. For example a timer service is provided by the TargetRTS for active classes, but it is not available for passive classes. If a timing service for passive classes were required, you would have to design it.

Using the scheme above, active and passive objects can be combined in a single system. Figure 10, following, illustrates an example message sequence chart showing the interaction scenario.

V. Conclusions

Using software architecture gives quite good new concepts like classes, interfaces, components, modules, sub-systems, inheritance of classes, processes, messages, files, hardware components, communication models etc. Architecture model may concern either static structure of software like classes or dynamic structures like objects. It may also concern static relationships as well as dynamic behavioral models. Architecture can be described either as abstract model, which has not direct relation to software, or as concrete model. MDA prescribes certain model artefacts used along system development line, how those models may be prepared and their relationship. It is an approach to system development that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform. Main MDA artefacts are platform independent system models (PIMs), platform specific system models (PSMs) and system code. There is a clear distinction between PIM, PSM and system code although it depends on the context, the development process and the details of the system and target platform, where the border between PIM, PSM and system code is to be placed. Within these three abstraction levels, transformation techniques are used to translate model parts of one abstraction level into model parts on another abstraction level.

The idea of this paper is to suggest of development process to improve the Mobile applications system development. We used different architectural views for analyzing mobile architectures and architecture role in development. The architecture and architectures role on the development has been studied in mobile application-and multi-platform service development.

Acknowledgment

“ This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. R01-2008-000-20607-0) ”

References

[1] C. Reichmann, P. Graf, K.D. Müller-Glaser, “GeneralStore - A CASE-Tool Integration Platform Enabling Model Level Coupling of Heterogeneous Designs for Embedded Electronic Systems,” *11th IEEE Conference on the Engineering of Computer Based Systems 2004*, Brno, Czech Republic, May 2004.
[2] Telelogic, *Doors tool*, Available: http://www.telelogic.com/solutions/alm/rdt/index.cfm.
[3] T. Bienmüller, U. Brockmeyer, G. Sandmann, “Automatic Validation of Simulink/Stateflow Models, Formal Verification of Safety-Critical Requirements,” *OSC – Embedded Systems AG*, 2004.
[4] M. Stonebraker, J. Frew, K. Gardels, J. Meredith, “The SEQUOIA 2000 Storage Benchmark,” *Proc. of Int. Conf. on ACM SIGMOD*, 1993, pp. 2-11.
[5] Doo-Hwan Bae et al., SPICS midterm report, Aug. 2007.
[6] Carnegie Mellon Software Engineering Institute. *Software architecture documentation in practice – Documentating architectural layers*. 2002. Available: http://www.sei.cmu.edu/publications/documents/00.reports/00sr004.html
[7] Carnegie Mellon Software Engineering Institute. *A Framework for software product line practice – version 3.0*. 2002. Available: http://www.sei.cmu.edu/plp/framework.html, 2002.
[8] K.Y. Whang, R. Krishnamurthy, “The Multilevel Grid Files – a Dynamic Hierarchical Multidimensional File Structure”, *Proc. of Int. Conf. on Database Systems for Advanced Applications*, 1991, pp. 449-459.
[9] S. Shekhar, Y. Huang, J. Djugash, “Dictionary Design Algorithms for Vector Map Compression,” *Proc. of Data Compression Conf*, 2002, pp. 471.
[10] OMG. *Meta-Object Facility (MOF). version 1.4*. Available: http://www.omg.org/technology/documents/formal/ mof.htm
[11] OMG. Model-Driven Architecture. Available: http://www.omg.org/mda

Dr. Haeng-Kon Kim is currently a professor in the Department of Computer Engineering and was a dean of Engineering College at Catholic University of Daegu in Korea. He received his M.S and Ph.D degree in Computer Engineering from Chung Ang University in 1987 and 1991, respectively. He has been a research staff in Bell Lab. in 1988 and NASA center 1978-1979 in U.S.A. He also has been reserched at Central Michigan University in in 2000-2002 and 2007-2008 in U.S.A. He is a member of IEEE, KISS and KIPS. Dr. Kim is the Editorial board of the international Journal of Computer and Information published quarterly by ACIS. His research interests are Component Based Development, Component Architecture & Frameworks for Mobile Applications and Components embedded systems .