Efficiency Requirements in PIM to PSM Transformations

Dariusz Gall

Computer Science and Management Faculty, Wroclaw University of Technology
Wybrzeże Wyspiańska 27, 50-370 Wroclaw, Poland

Abstract. In the paper, we provide extensions for MDA (Model Driven Architecture), taking into consideration efficiency requirements. The scope of the proposition is limited to the PIM (Platform-Independent Model) and the PSM (Platform Specific Model) extension. Efficiency requirements are maximal rates of users requests under which the system has to perform. We extend the PIM to the model containing efficiency requirements. We introduce a PSMSpec (PSM Specification) containing a system architecture, platform, efficiency requirements, and a PSMInst (PSM Instance) defining a deployment of the system. The PIM’s functional specification and efficiency requirements are transformed to the PSMSpec’s system architecture and efficiency requirements. Finally, efficiency characteristics are estimated and verified against the efficiency requirements, the PSMInst is suggested.

1 Introduction

Efficiency requirements shall be handled during development process from the earliest stages [1]. In order to make it possible, we have to specify the requirements with constraints in quantitative way, provide a transformation of the requirements from specification level to design/deployment level, provide verification of the requirements, and introduce changes to design/deployment based on the requirements verification [1], [2].

In the paper, we introduce handling of efficiency requirements in the development process, by extending the MDA (Model Driven Architecture). The MDA is software development approach, which is based on models and the concept of separation of specification from implementation. The concept is applied by transforming platform-independent models (PIMs) into platform specific models (PSMs), which in result are transformed to a system implementation [3].

In the related papers [2], [4], [5] idea of supporting efficiency requirements using MDD (Model Driven Development), in particular MDA, is introduced. Efficiency requirements are not supported by the MDA [2], but are implemented in the MDA by adding new efficiency models at platform independent, platform specific level and by providing transformations of the efficiency models [2], [4], [5]. The efficiency models are resolved and results are used for architecture/design refactoring, if required [2], [4]. In [5], a necessity of differentiating platforms' performance influence to the overall system performance is discussed, e.g. software and hardware platform
separation.

The paper is organized in the following way: in Chapter 2, requirements transformation, extension of PIM and PSMs are discussed, in Chapter 3 we introduce behavior metric and discuss a system’s efficiency estimation, an example of Payment System is presented in Chapter 4, and the article is summarized in Chapter 5.

2 Extending MDA

In order to introduce efficiency requirements into the MDA, we have to include into MDA models’ efficiency requirements, efficiency constraints, and transformations.

The PIM is platform independent model, which defines a software system’s functional requirements [3]. We add to the PIM efficiency requirements. The functional specification contains actors, use cases definition, and use cases’ scenarios. The use cases’ specification are defined by UML sequence diagrams in terms of «boundary» and «logic» classes or «entity» classes [6].

The PIM’s efficiency requirements and efficiency constraints are defined for pairs of an actor and an use case. We specify an efficiency requirement by defining a maximal rate of an use case’s scenario executions by an actor. The constraints on the requirements indicate most often executed scenarios of the use case. For each possible scenario we define a weight, indicating execution rate. The efficiency requirements and constraints are expressed using MARTE [7].

The PSM is a platform specific model defining a system architecture [3]. We suggest to introduce two PSMs: PSMSpec (PSM Specification) and PSMInst (PSM Instance). A PSMSpec consists of the software system architecture, deployment specification, platform within the system is developed, efficiency requirements, and efficiency constraints, required for a total computation effort calculation introduced in Chapter 3. We define platform by UML structural elements [6] and efficiency constraints, specifying computational complexity of platform’s operations. The deployment specification defines system deployment on specification level [6] and efficiency constraints [7], i.e. defines possible system deployments, e.g. number of nodes’ instances, nodes’ efficiency constraints.

The PSMSpec’s efficiency requirements and efficiency constraints are mapped from efficiency requirements and the constraints defined in PIM, expressed in terms of the software system architecture. An efficiency requirement is defined in context of the actor and the use case from PIM’s use cases.

A PSMInst is a platform-specific model, which contains an instance of the PSMSpec’s deployment specification [6] with efficiency characteristics of nodes [7], i.e. computation power of nodes. The PSMInst represents a particular deployment of the system.

The PIM is transformed into a PSMSpec. Two kind of transformation are performed. The first type is a transformation of the PIM’s functional specification into PSMSpec’s software system architecture and deployment specification, within a platform. The second type is a transformation of the PIM’s efficiency requirements with constraints into PSMSpec’s efficiency requirements with constraints.

The PSMSpec is transformed into PSMInst. A particular deployment of the system is stored in PSMInst. During the transformation, instances are created and
configurations of the instances are set. The transformation is made with respect to results of the efficiency assessment introduced in Chapter 3.

3 Efficiency Assessment

Fulfilling the efficiency requirements is necessary to find such a system’s configuration, i.e. the PSMInst, where computation power provided by node types is more than the computation power required by node types for the all system use cases. Thereby, we have to: assess a computation effort for each use case’s scenario for each node’s type, assess the required computation power for all system’s use cases for each node type, and check if it is possible to find the system’s configuration which provided computation power that is sufficient.

In first step, we assess the use case’s computation effort. We define the behavior metric, which value is used for assessing computation effort of a use case implemented within a particular PSMSpec. The metric is defined in terms of average number of virtual instructions for each node type required to execute a use case. We assume that each virtual instruction has similar time of execution on a given node type. The metric’s argument are: the UML sequence diagram representing scenarios of the use case and average lengths of scenarios’ input and context arguments. Each occurrence specification in a UML sequence diagram, e.g. operation invocation, combined fragments, etc. is mapped to a set of virtual instructions. A value of the metric for a given sequence diagram is a collection, in which for each node type, a number of virtual instructions corresponding to elements of the sequence diagram executed on the node type is given.

The mapping may differ for different platforms, we use herein Java platform, thereby a virtual instruction corresponds to a Java bytecode instruction [8] [9]. Because the virtual instruction is a bytecode, we are able to compute the metric’s values for Java platforms’ libraries operations and use the values in use case’s metric calculation [9]. We assume herein that for all necessary Java’s library operations the metric is computed.

We calculate a value of the metric for a given use case’s scenarios and attribute values taking into consideration each node type on which the scenarios are executed. We calculate the metric according to the following steps:

1. If a given operation or an use case has already calculated complexity, return the value, taking into consideration node type on which operation is executed. We assume that the value is already given for operations of platform’s libraries.
2. If a given operation or use case is defined by an UML sequence diagram than for each occurrence specification, taking into consideration node type:
   a. Calculate the metric for all called operations, i.e. repeat these steps from the beginning for each operation,
   b. If the occurrence specification is not inside any combined fragment then add the occurrence specification metric’s value to the given operation or use case total complexity,
   c. If the occurrence specification is inside alternatives, option combined fragment then add the occurrence specification metric’s value to the given
operation or use case total complexity multiplying it by weight attached to appropriate combined fragment’s operand,

d. If the occurrence specification is inside loop combined fragment then add the occurrence specification complexity to the given operation or use case total complexity multiplying it by number of loop iterations attached to the loop.

In next step, we have to calculate a required computation power per node type for all the system’s use cases taking into consideration the efficiency requirements. We calculate it by summing for all pairs of an actor and a use case, multiplication of each element of a collection of the use case metric’s value’s by a use case execution rate.

Finally, we have to find a proper system configuration. For each node type in the PSMSpec’s deployment specification, efficiency constraints on a computational power are set. We check for each node type whether it is possible to create a proper number of instance where computational power provided is more that computational power required. If yes, then the condition is fulfilled, otherwise, we have to modify the system architecture, or change the deployment specification constraints.

4 Payment System Example

We present the Payment System as an example of efficiency handling, which provides a money transfer functionality. We present in the paper one use case with efficiency requirements and constraints, which is the money transfer, see Fig. 1. The efficiency requirement, defined by «gaWorkloadEvent» and pattern tag, is to provide the use case execution’s maximal throughput. Weights of each possible scenario are defined by «paStep» and a weight tag [7], see Fig. 1.

We transfer the PIM’s functional requirements to the PSMSpec, with respect to client-server architecture pattern, adding the load balancer to provide system scalability [10]. Because of the space limit, we do not present the platform’s model in this paper. We present only transformation results in Fig. 2 and Fig. 3: i.e. the sequence diagram of the implementation of use case, and deployment diagram with types of...
nodes etc. The Fig. 2 and Fig. 3 already contain efficiency requirements and constraints, however, none of them are added during the PIM's functional requirements transformation. The deployment specification diagram is annotated by efficiency characteristics of nodes, using «gaExecHost» throughput tag, see DeploymentSpecification in Fig. 3. Next PIM's efficiency requirements and constraints are mapped from the PIM to the PSMSpec with respect to the system’s architecture, see Fig. 2.

**Fig. 2.** PSMSpec - The transfer money use case realization.

We assess a computation effort of the money transfer use case. Each lifeLine’s instance on Fig. 2 is linked with comment pointing out a node type. The sequence diagram on Fig. 2, contains the behavior metric’s values for operations, specified by $E(methodName)$. A computation effort per a node type for the use case is specified by $EP(nodeName)$. These values were calculated with respect to the steps introduced in Chapter 3.

**Fig. 3.** PSMSpec deployment specification and PSMInst deployment instance.

We assess the required computation power, taking into consideration the efficiency requirement and the results are: $RCP(\text{LoadBalancer})=2000\,[\text{1/s}]$, $RCP(\text{WebFrontEnd})=30000\,[\text{n/1/s}]$ and $RCP(\text{TransactionService})=12800\,[\text{1/s}]$. In next step, we have to find such number of nodes instances to fulfill the efficiency requirements. Taking into consideration the PSMSpec’s deployment specification
efficiency characteristics, we generate a PSMInst, see deployment instance on Fig. 3. Sum of nodes’ provided computation power for a given type is less than the required computation power for the node type, thereby the efficiency requirements is fulfilled.

5 Conclusions

We provide the MDA models and transformation extensions for handling efficiency requirements in the MDA. The PIM is characterized by efficiency requirements and constraints, the PSMSpec contains a software system architecture, a deployment specification and the efficiency requirements, and the PSMInst represent an instance of the deployment specification. In the paper, we provide the efficiency estimation, using the behavior metric and illustrate the method with an example.

We only consider the efficiency requirements for the system throughput, however, time constraints are another type of the efficiency requirements to be investigated. We also consider only an “average case”, though we should also be able to consider worst case, etc. In the future, it would be advisable to take into consideration several platforms types, e.g. Java, .Net, etc. Moreover, the method provided in the paper does not take into consideration the network communication overhead.

References