DEVELOPMENT OF SOFTWARE PROCESS MODEL FOR CONTROL SYSTEMS USING SPEM

Dept. of Industrial Automation, University of Chemical Technology and Metallurgy, Sofia, Bulgaria

Abstract: The main aim in achieving of agile software process model is to support the development of diverse and complex software tools and program components in order to deliver faster functionality with higher quality and reliability. In this paper the Eclipse Process Framework (EPF) based on Software & Systems Process Engineering Meta-model specification (SPEM) as a UML2.0 profile and the agile unified software development process OpenUP are used to develop software process model for the domain of process control. In order to support the development of Model Driven Development (MDD) of control systems the basic process OpenUP is extended with process models represented the MDD - OpenUP/MDD. Finally some conclusions are made.

Keywords: SOFTWARE PROCESS MODEL, CONTROL, SPEM, OPENUP, ECLIPSE, UML.

1. Introduction

Control systems are closely related to computer science and software engineering, since all modern control algorithms are implemented as software. Algorithms and software in the domain of control differ significantly from the traditional computer software. The most significant features are the dynamics of the system and the real time behaviour. An important consequence of wide penetration and use of control systems at different operative levels is the substantial increase in complexity and size of the software used. All this requires the use of advanced methods and approaches for software development in this domain with a view to deliver faster functionality with higher quality and reliability.

In general, software engineering has made significant progress over the past two decades. Structural programming, descriptive (declarative) specifications, object-oriented programming, formal methods and visual programming have emerged to assist software development to a level that allows the construction of correct and efficient software systems. For now, however, these hopes are justified, and while the processor performance is increasing annually by 48 percent, the capacity of communication networks with 78 percent, the productivity in software development is increasing only by 4.6% and the power of programming languages and tools with 11% per year [1].

One of the directions to cope with the above mentioned problems and to increase the efficiency and productivity in the field of software engineering is to use the approaches of Model Driven Engineering (MDE) based on UML as a general purpose modelling language [2]. At the core of MDE are the so called PIM (platform independent model) languages, allowing modelling the software application or system through specifying its functions, structure and behaviour. One of the best known and discussed initiatives of MDE is SPEM [3], MDE and MDA based standards recommend different technologies to support the concepts and principles for development of control systems, but they do not suggest a development methodology. This gap may be filled through combining the MDE techniques with the concept of Software Process Models (SPM) [4]. There are many different SPM and this high diversity is determined by too many modifications to the well known major brands namely: sequential, incremental, prototyping, evolutionary, and agile models. The development and application of agile Software Process Models (SPM) allows shortening the time to the customers of the developed software, characterized by high flexibility, adaptability and reconfigurability.

The main aim of the paper is to implement the Software & Systems Process Engineering Meta-model - SPEM for development of software process model for the domain of control. The open source platform Eclipse is chosen as one of the most successful, universal and robust platform providing an open integrated and extendable development environment for tools and desktop applications [5].

The next part of the paper proposes a short overview of the software process model development based on SPEM [6] under Eclipse and using Eclipse Process Framework (EPF) [7] and OpenUP [8]. After that the suggested software process model for the domain of control is presented. In the last part of the paper the suggested software process model is illustrated with a simple case study. Finally some conclusions and perspectives for future work are made.

2. Overview of SPEM 2.0

Software process models represent networked sequences of activities, objects, transformations, and events, embodying strategies to accomplish of software evolution. In difference to the software life cycle models they are more precise, detailed and formalized using sufficiently rich notation, syntax, or semantics, suitable for computational processing. The Software and Systems Process Engineering Meta-model (SPEM2.0) is one of the mostly used meta-models for developing various software process models for diverse domains and is specified as a MOF2.0 based meta-model as well as a UML2.0 Superstructure based profile.

SPEM 2.0 Meta-model. The SPEM2.0 Meta-model, based on MOF2.0 is a formal meta-modelling language for representing development methodologies and its final version is released in March 2007. The adopted meta-model is very general and may be used to describe any software development process such as Waterfall, Iterative, Incremental, Evolutionary or agile in any domain. Through a rich set of customization attributes different temporal guidance’s for the process elements may be specified, allowing their mapping to the project plans which are generated based on the underlying lifecycle model of the process.

The basic idea of SPEM is that the software development process may be seen as collaboration between abstract active entities called process roles that perform operations called activities on concrete, tangible entities, called work products. The SPEM model elements can be presented by Packages which contains one or more related extensions of standard UML semantics. SPEM defines the elements used in describing a process as well the elements used for structuring and managing the needed information. The basic packages used to structure and manage information are shown in the class diagram in fig.1. They are: “MethodLibrary”, “MethodPlugin”, “MethodContentPackage”, “ProcessPackage” and “MethodConfiguration”. “MethodLibrary” is a collection of “MethodPlugin” and “Method Configurations”. In order to simplify the information management, the “MethodPlugin” package is further sub-divided into “MethodContentPackage” and “ProcessPackage” packages and defines extensibility and variability mechanisms for these sub-packages. “MethodContentPackage” as shown in fig.2 defines highly re-useable content, such as definition of roles, tasks, work products and associated relationships. It includes guidance and categories but not timing information. “ProcessPackage” re-uses the content to create end-to-end processes such as phases, iterations, activities and milestones that define the
development lifecycle. It also defines when tasks are performed via Activity Diagrams and/or Work Breakdown Structures. The “MethodConfiguration” package defines a logical sub-set of a “MethodLibrary” package that will be published or exported and may be considered as a ‘filter’ applied to the library.

Two types of UML diagrams for software process modelling are used - Use Case Diagram and Activity Diagram. Use case diagram describes the assignment of “ProcessRoles” to “Phases” or “Activities” and represents the Work Breakdown Structure of the process. Activity diagram completes the construction of structural aspect including the sequence of a simulation model. Activity diagram can illustrate the behaviour of an activity with the “ActionStates” which represent “Steps” of an “Activity”. In general, the state chart diagram illustrates the behaviour of an activity.

SPEM 2.0 Profile defines stereotypes for UML 2.0 Superstructure model elements for every meta-model concept and uses tree type of diagrams – class, activity and state diagrams. Similar to the meta-model that separates the “MethodContentPackage” from the “ProcessPackage”, the UML2.0 profile of SPEM2.0 allows to define reusable behaviour with UML Activities which are being invoked by Actions presented in Activity diagrams. The profile specifies a few stereotypes, but it must exclude or constrain many basic concepts of UML. Reusable process components in SPEM 2.0 Profile are achieved through encapsulation with input and output ports, which allow the user to consider the actual definition of the work that produces the outputs as a “black box”. The SPEM 2.0 profile describes all structures and attributes needed to represent methods and processes as actually reuses elements from the UML 2.0 meta-model. Main disadvantage of this approach is inherited from the UML and cover the simple relationships, which need to be enforced with complicated OCL constraints, constraining the user much more than when modelling with UML [9].

All elements in SPEM packages together with their attributes and associations are directly represented by equivalent UML classes, attributes and associations. The attributes in the SPEM packages are represented by tagged values and associations are represented in a variety of ways. The SPEM meta-model is defined using four layer OMG’s architecture and supports the description of the basic concepts as “Tasks”, “Techniques”, “Roles”, “Products” and “Phases” in the following ways: “Tasks” are modelled by “Activity”; “Techniques” are modelled by “Guidance” or “Activity”; “Roles” are modelled by “ProcessRole”; “Products” are modelled by “WorkProduct” and “Phases” are modelled by “Phase”. The latest SPEM version improves the content and process re-usability, by introducing new capabilities as a clear separation between structural and dynamic methodology concerns.

Two types of UML diagrams for software process modelling are used - Use Case Diagram and Activity Diagram. Use case diagram describes the assignment of “ProcessRoles” to “Phases” or “Activities” and represents the Work Breakdown Structure of the process. Activity diagram completes the construction of structural aspect including the sequence of a simulation model. Activity diagram can illustrate the behaviour of an activity with the “ActionStates” which represent “Steps” of an “Activity”. In general, the state chart diagram illustrates the behaviour of an activity.

SPEM specification defines that “Step” should be used only in an activity diagram [10, 11].

3. Development of SPEM based software process model

The suggested approach for development of SPEM based software process model for the domain of process control is based on the Eclipse Process Framework (EPF) that is a new Eclipse technology aiming to provide an extensible framework and exemplary tools based on SPEM 2.0 concepts for defining and managing software development processes [7]. Within this framework, extensible process content for a range of software development and management processes supporting iterative, agile and incremental development is proposed, and is applicable to a broad set of development platforms and applications. EPF Composer is built upon the Eclipse platform to support many of the Eclipse plug-ins using different views and perspectives for presenting specific information and supporting a workflow. The standard perspectives are: authoring - for editing method content and browsing - for previewing published elements.

OpenUP is one of the main pillars of an iterative incremental software development process that is minimal, complete, and extensible. It includes only fundamental content which can be manifested as an entire process to build a system and can be used as a foundation on which process content can be added or tailored as needed. Its lifecycle is divided into 4 phases, each one pursuing its own purposes and milestone criteria for exit from the phase (fig.3):

♦ Inception - with the primary purpose to understand the scope of the problem and feasibility of a solution.
♦ Elaboration - with the primary purpose to validate the solution architecture (feasibility and trade-offs).
♦ Construction – with the primary purpose to develop and verify the solution incrementally.
♦ Transition – with the primary purpose to deploy the solution to the operational environment and validate it.

Supplementary in OpenUP the main objectives and activities for each phase are defined. For example the main objectives of Elaboration phase are to get a more detailed understanding of the requirements; to design, implement, validate, and baseline the architecture and to mitigate essential risks, and produce accurate schedule and cost estimates.

OpenUP consists of a base process (OpenUP/Basic) and extensions to this base process, such as agile database content or OpenUP/MDD. The last one is an extension to OpenUP/Basic that models OMG’s Model-Driven Architecture (MDA) approach [8, 12]. The OpenUP/MDD method content and process models represent the MDA approach with six key Role Definitions: Platform Expert, Business Analyst, Domain Expert, Language Engineer, Test Designer, and the Transformation Specifier. These roles are responsible for altogether 57 different Work Products, and perform 90 standard Tasks.
OpenUP/MDD is a model-driven development process which is a variation of the Open Unified Process for MDD. The OpenUP/MDD is a kind of generic model-driven development process defined to be customizable for each specific context. The structure and behaviour processes according OMG are modelled as meta-models at all levels – M0, M1 and M2. The elements of software development project are defined at M0 level, at the next M1 level all elements of process model are designed and finally on level M2 - the elements of detailed process meta-model are defined [12].

In order to achieve MDA based development of control applications the plug-in OpenUP/MDD is chosen to be included in the model architecture. According MDA there are three types of models: the Computational Independent Model (CIM), the Platform Independent Model (PIM) and the Platform Specific Model (PSM) [13]. The CIM is the model that describes the systems requirements without any references to a particular technology or technique and it does not show how the system is implemented. The CIM shows the systems in its environment and helps presenting what the system should do. The CIM is not widely used because of the vague definition of what it actually should be. The model is also not directly used to create a PIM. However, there must be a clear relation between CIM and PIM. The PIM is an important part of MDA. It describes the system without referring to the platform and thus gives a platform independent representation of a system. This model is the first step to create a system. The PIM can be transformed into a PSM using model transformations. The PSM models provide information about how the system is implemented and its relation to the platform. PSM contains the information from the PIM with specific details about the target platform.

A set of related skills, competencies and responsibilities of an individual or a set of individuals are defined by “Role”. Individuals should play their “Roles” performing “Tasks” that can be associated to input and output “WorkProducts”. A “Task” may comprise many “Steps” to describe a meaningful and consistent part of the overall work. The “Discipline” represents a collection of “Tasks” which are related to a major “area of concern” within the overall project. “WorkProducts” are in most cases tangible artifacts consumed, produced, or modified by “Tasks”.

In this approach four types of artifacts for “WorkProducts” are specialized – “UMLModel”, “TransformationRule”, “ExtraModel” and “Profile”. The “UMLModel” can be created in two ways - automatically generated by a transformation during the process execution or produced by a process role. The “TransformationRule” contains the rules for model transformation and code generation during the process execution. Basic or supplementary notations are used only for documentation in “ExtraModel”. Each modelling phases used to represent an UML profile is named “Profile”. In order to transform the MDA based UML models well defined transformation rules are needed.

Three types of UML diagrams for presenting of MDD meta-model are used – class diagram, use case diagram and activity diagram. The defined transformation rules for presenting the base UML elements with SPEM stereotypes are summarized as follow: in a use case diagram “Tasks” are modelled as use cases, while in the activity diagram they are modelled as “actions”. The use case diagrams are used to provide a specific view associating “Roles” to perform “Tasks” and also the used/produced “WorkProducts”. The activity diagrams are used to model the process workflow, i.e. the behaviour associated to the process execution in terms of “Phase”/“Iterations” and the selected “Steps” (TaskUse).

4. Case Study

In this section, an application for water level feedback control system using PID controller as case study is presented. The physical system is shown in fig.4 and consists of a cylindrical tank, filled with water until specified level that is registered with a level sensor and is controlled by changing the input valve position. The core of control system is a feedback controller, which controls the level in the tank according the PID principle. The discussed example is divided in two parts – physical system, named “Physical SubSystem” and control system called “ControlSubSystem”.

There are two basic parts for presenting of the software life cycle – development phase and maintenance phase. The development phase of the software process is organized in 4 lifecycle phases: Inception phase, Elaboration phase, Construction phase and Transition phase as is shown in fig.5. The software architecture is decomposed into components represented from a variety of viewpoints, all of which can be combined to create a holistic view of the system (fig.5-1). Each architectural view addresses some specific set of concerns, specific to stakeholders in the development process: users, designers, managers, system engineers, maintainers, and so on.

In the Inception phase one or more templates of use cases, scenarios or set of requirements as well functional requirements assigned to the control system are defined. The used templates requirements explain how to elicit, analyze, specify, validate, and manage the requirements for the system to be developed and functional requirements to the control system are defined as follow: the level set-point, the sampling period and the output flow rate is not controlled.

EPF provide 3 types of process diagrams. The main phases of the life cycle of water level control system are presented through two types UML diagram – activity diagram and activity detail diagram. The activity diagram (fig.5-2) gives the overall sequence in Elaboration phase.

The activity detail diagram (fig.5-3 and fig.5-4) shows the tasks within the same activity, arranged by the responsible role that
performs them and the mandatory input and output work products for each task. In this case the physical subsystem and control subsystem are presented as activities in Elaboration phase. PID_Calc, Integral and Derivative are the three sub-activities nested in “Control SubSystem” activity. Tank, Valve_in and Valve_out are sub-activities used for modelling of physical subsystem. The activity detail diagram from fig.5-4 presents the current level control in the Tank. After initialization and sensing of the current level in the Tank, the control subsystem takes a decision to send a message to the actuator (Valve_in) – Move_Pos1 or Move_Pos2, which depend on the tank level.

5. Conclusions

This paper aims to create an executable functional software process model including the early stages of the development process based on requirements specifications, possibility for formal specification, verification and validation of the system. The Eclipse Process Framework, based on SPEM2.0 meta-model and the plug-in OpenUP/MDD are used in order to describe the structure and behavior of the software process model.

The proposed approach for development of agile SPM allows to modelling the development processes and is supported by different libraries with improved reusability and extensibility capabilities. SPEM2.0 provides a range of mechanisms for extending and combining chunks of process and method descriptions.

As a future work, this approach will be applied to a variety of methods for quantitatively analyzing a descriptive process model in order to increase the efficiency, correctness and interoperability of the project components.

6. References