A Component-based Product Line Architecture for Workflow Management Systems

Itana M. S. Gimenes, Edson A. O. Junior
Universidade Estadual de Maringá, Departamento de Informática
Maringá, Brasil, 87020-900
itana@din.uem.br, edson.junior@pop.com.br

and

Fabrício R. Lazilha
Centro Universitário de Maringá, Departamento de Informática
Maringá, Brasil, 87050-390
fabricio@cesumar.br

and

Leonor Barroca
The Open University, Department of Computing
Milton Keynes, MK7 6AA, England
l.barroca@open.ac.uk

Abstract
This paper presents a component-based product line architecture for workflow management system. The process followed to design the proposed architecture was based on the Catalysis approach. Extensions were made to represent variability across the process. The domain of workflow management system has shown to be appropriate to the application of the product line approach as there are a standard architecture and models established by a regulatory board, the Workflow Management Coalition. In addition, there is a demand for similar workflow management systems but with some different features. The proposed architecture was evaluated by the Rapide simulation tools.

Keywords: Workflow Management Systems, Product line, Software Architecture, Reuse.
1 Introduction

According to Bass [1], a software product line approach is a collection of systems that share a manageable set of features amongst its main artefacts. These artefacts include a base architecture and a set of common components that populate the architecture. The design of an architecture for a product family must consider similarities and variabilities amongst its products.

The product line approach is appropriate for domains where there is a demand for specific products that can be modelled from a set of common features and well-defined variability points. The Workflow Management System (WFMS) domain is appropriate to the application of this approach due to the efforts of the Workflow Management Coalition (WFMC) [2]. This regulatory organisation established a generic architecture and a reference model for WFMS that can be used to customise products to market needs. Companies need workflow products that have similar features but with some different aspects. Examples are workflow products with either traditional and web user interface, or with different task scheduling algorithms.

This paper presents a component-based product line architecture for WFMS and its design process. It also describes the extensions made to represent variability throughout the development process. The proposed architecture was evaluated by the Rapide simulation tools [3]. Section 2 describes the proposed architecture and the notation used to represent variability; section 3 presents the architecture evaluation, and section 4 presents the conclusions.

2 The Product Line Architecture Design Process

Product line is still a recent approach, which demands new architecture and components design techniques. Some of the existing techniques are Synthesis (Software Productivity Consortium) [4], Family-Oriented Abstraction, Specification and Translation (FAST) [5], Product Line Software Engineering (PuLSE) [6] and Feature-Oriented Domain Analysis (FODA) (Software Engineering Institute - SEI) [7]. These techniques are based on domain engineering.

They provide mechanisms to capture and represent domain features and variabilities. However, they are less efficient to represent architectures and components. Thus, Component-Based Development (CBD) methods can be used in the design of product lines to bridge the gap between domain analysis and the architecture and component design and implementation. General purpose CBD methods are easier to understand and use. In addition, there are more commercial tools that may be used to support them.

The product line architecture design process proposed in this paper considers:

- domain analysis based on the generic architecture and reference models for WFMS of the WFMC [2];
- design of the architecture and components based on the Catalysis approach [8]; and

Catalysis was used, as it is a general purpose CBD approach based on UML [10] and encompasses important concepts such as the central role of software architecture, frameworks and patterns.

Product line approaches that also take CBD into account are Kobra [11] and GenVoca [12], but they are, as yet, less disseminated than Catalysis.

The architecture design process is composed of the following phases: requirement analysis; system specification, architectural design and component internal design. The following sections describe the application of this process to design a component-based product line architecture for WFMS.

2.1 Requirement Analysis

The design of a product line architecture for WFMS requires the conception of the domain model representing the objects and domain actions. In this phase, it is already possible to identify similar aspects and variation points amongst product line members. According to the product line terminology we are proposing an architecture for a WFMS family of products. Each specific product that can be generated from this product line is a member of the family.

The reference model and generic architecture for WFMS of the WFMC were used to extract the main set of features of the WFMS family. These models define the main components and interfaces that a WFMS should have in order to allow interoperability of subproducts from different suppliers. In addition, the WFMC defines important characteristics to allow specific WFMS to be built according to organisation needs, such as workflow for software production or workflow for financial administration. Figure 1 presents the generic architecture for WFMS of the WFMC [2]. The WFMC models also indicated the potential components of the product line architecture and their
interfaces. The process manager pattern [13] was also used to exploit the WfMS domain. This is an architectural pattern for definition of PSEE process managers developed from studies of existing environments and from experiences obtained in the development of the ExPSEE (Experimental Process-centred Software Engineering Environment) project (funded by CNPq - a Brazilian research funding agency).

![Diagram of Generic architecture for WfMS](image)

**Figure 1:** Generic architecture for WfMS [2].

A domain model in Catalysis is represented by objects and actions at a high level of abstraction and is independent of the software solution for the problem. The domain model is represented by UML use cases. There are three main actors in the WfMS domain: the workflow user, the workflow manager and the workflow architecture manager. Figure 2 shows only the architecture manager viewpoint, the others are omitted for lack of space.

One of the main aspects of a product line development is to capture and represent variabilities associated with the architecture and its components. The notation followed is the use case variability of Jacobson et al. [14] that suggest the stereotype «extend» to represent variation points in use cases. The extended use case to represent variations is annotated with a mark (blob). Figure 2 shows an example of this representation. The use case Define Architecture is marked as a variation point to represent the optional feature of allowing dynamic architecture updating. Another example appears in the use case DefineToolType that specifies two extensions: DefineInternalToolType or DefineExternalToolType.
2.2 System Specification

This stage represents the software solution identified from the domain model. The analysis of the system actions leads to the identification of the types and related actions. Types, in the Catalysis view, are specifications of object behaviour that document and show only the external view of an object. The main artefact of this phase is the static type diagram.

The representation of variation points also appears in this phase. Figure 3 shows the static type diagram for WiMS. Morisio [15] extended UML with a variability stereotype, indicated as a <<V>>. This stereotype is related with the concepts of specialization and aggregation. In Figure 3, <<V>> is used to represent types that can vary according to the product features. For instance, the stereotype <<V>> is used to represent a variable resource type, TypeResource, which can be specialised to material, actor or tool type of resources. The tool type is also extended to represent internal or external types of tool.
2.3 Architectural Design

From the type modelling several refinements need to be made to reach the level of components. Catalysis uses packages as a high-level decomposition unit. A package is an independent unit whose relationships to the rest of the system can be well established. These packages can be derived from the domain model and their relationships can be represented as import dependencies. The package partitioning follows the vertical and horizontal slices approach of Catalysis. The vertical slices represent the business level partitioning of the generic architecture according to the actions undertaken by the main actors that interact with the WFMS. These actors are the workflow architecture manager, who defines reusable workflow architectures; the workflow manager (supervisor), who controls the instantiation, resource allocation and task assignment for the workflow; and the workflow user, who executes workflow tasks. These actors are represented through the packages Defining Workflow Architecture, Managing Workflow and Executing Workflow. Gimenes and Barroca [16] propose an enterprise framework for workflow management systems based on the WfMC generic architecture that can be used as a basis to generate components of WFMS.

Figure 4 shows the high level vertical slice diagram. The horizontal slices represent the partitioning of the architecture separating higher level business model packages down to the infrastructure packages (e.g. middleware or system software). This partitioning identifies service packages that are shared with higher level packages. It also aims at reducing the package importation across the layers. The elements of the WfMC generic architecture are represent by the packages: WorkflowArchitectureMgr, WorkflowMgr, WorkflowExecutionMgr, ResourceAllocationMgr, TaskScheduler, Interpreter, ExternalApplicationMgr and ObjectMgr.

From the high level vertical slice diagram it is possible to get to the vertical layer diagram that represents the final partitioning of the system. This model joins the package of the previous diagram with the types associated to each package. Figure 5 presents the TaskScheduler package and its types. The whole diagram is not shown for lack of space.
The Task Scheduler package is responsible for the control and management of tasks and actions to be undertaken in the workflow, such as modifying an artefact using an external tool. Thus, some types that appear in the static type diagram (Figure 3) are associated with this package as shown in Figure 5. Variation points are associated with types Resource and Tool.

2.3.1 Component Architecture

The vertical layers diagram is the result of identification and specification of components. The components are represented by the generic packages encompassing its types and relationships. The component architecture designed from this diagram is shown in Figure 6.

Most of the packages were mapped to components. This is a result of our experience with the domain. Several iterations of this system specification were carried out. So, the specification architecture is closer to the component architecture.
A short description of the components and the variabilities associated with this architecture is as follows.

- **GraphicalInterfaceMgr:** this component is responsible for user interface management. One variation point is the interface being via web browser or conventional.

- **WorkflowArchitectureMgr:** this component supports the definition and maintenance of workflow architectures. This component makes the workflow definition more flexible as the workflow types are not static. Variation points include the resource type, the tool type, and the process language supported. Resource can be specialised into actor, tool and material types. Tool type can be either internal or external. Different process programming languages can be supported depending on the interpreter.

- **WorkflowMgr:** this component is responsible for the instantiation and management of projects that are associated with a workflow. The project includes the instantiation and execution of workflow architectures. For each workflow element in the architecture there is an object in the workflow. After that, tasks can be managed and executed allocating resources and other decision-making. No variation points were defined for this component.

- **WorkflowExecutionMgr:** responsible for the control and management of the process tasks supporting the interaction between workflow users and the environment, in particular supporting the selection of tasks to be executed. No variation points were defined for this component.

- **TaskScheduler:** this component is responsible for task and action control and management. It allows the interaction of the users to the WfMS. The users can choose their tasks in the system. Variation points include resources to be used: types of resources, tools to be used (external or internal), task priority definition and algorithms used for task scheduling.

- **AllocationResourceMgr:** this component is responsible for resource allocation (actors, tools or material). In addition to the resource type and tool type, variation points include allocation policies.

- **ExternalApplicationMgr:** this component is responsible for the management of external application during the workflow definition and task execution. Variation points include different mechanism to adapt external application to the workflow.

- **ObjectMgr:** this component is responsible for the mechanism for management of objects. It manipulates data such as workflow control data, relevant data that appears throughout the process and even workflow.

- **Interpreter:** this component is responsible for the execution of a workflow script written in a process programming language.
2.3.2 Specification of the Architecture for CORBA

According to the Catalysis process the modelling of the logical architecture as shown in Figure 6, is followed by the decision making regarding the implementation mechanisms to be used, for instance the middleware. The result is a technical architecture as shown in Figure 7. In this case a CORBA ORB (Object Request Broker) [17] was considered as the middleware.

![Diagram of the WfMS product line architecture in CORBA.](image)

The components of the architecture have well defined interfaces in the CORBA IDL (Interface Definition Language). These IDL interfaces can be generated from tools such as Rational Rose [18]. The representation of the component interface in IDL allows for the individual implementation of the components in any language, operating system or network. The communication between components is made through ORB. The requested component replies or not according to communication constraints defined in the component architecture.

Up to this phase the components of the architecture are specified and their interfaces defined. The next phase should be the component internal design. However, the implementation of the components is not in the context of the work presented in this paper. Previous work has already developed a Task Scheduler component [19] but did not follow the product line approach. Current work is developing the Workflow Execution Manager and Graphical Interface Manager components.

3. Architecture Evaluation

Bosch [20] identified four ways of evaluating product line architectures: scenarios, simulation, mathematical models and evaluation based on past experiences. Section 2 described the process of defining the proposed architecture. This process is based on the Catalysis approach that uses UML, however, Catalysis does not support simulation. Thus, in order to evaluate the proposed architecture without delving into implementation details, we opted for an Architecture Definition Language (ADL). This language is focused in the representation of high level structures abstracting away from implementation details. The ADL Rapide [9] was chosen to specify the proposed architecture as it is a general purpose ADL that allow the modelling of component interfaces and their externally visible behaviour. In addition, there is a support environment that allows the definition of the architecture and its simulation. The Catalysis specification of the architecture, represented in a UML component diagram (Figure 6), was manually translated to the ADL Rapide.

The simulation was carried out based on the selection of relevant scenarios for WfMS. Sequence diagrams were drawn to represent the interaction of each specific scenario using components in the place of objects. The
architecture was executed according to the scenarios to simulate the behaviour of the system. Each scenario
represented the view of each WfMS users: workflow architecture manager; the workflow manager (supervisor and
the workflow user. Once an erroneous message was observed, the sequence diagram was adjusted to correct the
components communication and the overall simulation was repeated.

The analysis of the dynamic behaviour of the components of the proposed architecture allowed the evaluation of
both the communication between components and the general features of the system avoiding implementation
issues. However, it was not sufficient to prove that the architecture is correct. The complete evaluation needs more
precise techniques for architecture evaluation. Criteria need to be established to allow derivation of statistical data
from the simulation.

4. Conclusions

This paper presented the process followed to define a component-based product line for WfMS. Extensions made to
represent variability across the process were presented. These extensions were based on Jacobson [14] and Morisio
[15].

We argue that CBD can be used in the design of product lines to bridge the gap between domain analysis and the
architecture and component design and implementation. General purpose CBD methods are easier to understand and
use. In addition, there are more commercial tools that may be used to support them. In this work we have used the
Catalysis process [8] to design the product line component architecture. The domain analysis was carried out based
on the generic architecture and reference models for Workflow Management System (WfMS) of the Workflow
Management Coalition (WfMC) [2]. The evaluation of the architecture was carried by its specification in the Rapide
ADL and simulation in the Rapide environment. This allowed the evaluation of the architecture based on selected
scenarios avoiding implementation details.

Members of the WfMS family can be generated from the proposed product line by providing their specific
requirements. These requirements are used to select and instantiate the variabilities of the component architecture.
Currently we are populating the architecture by either developing novel components or adapting previously
developed components [13, 19]. We are also working on formalising the product generation process.

References

System Architecture Using Rapide. IEEE Transactions on Software Engineering. Special Issue on Software
02.00.03, (November 1993).
methodology to develop software product lines. Symposium on Software Reusability (SSR99), (May 1999).
1998.


