Developing Service-Oriented Applications: a method engineering based approach

Alfredo Garro, Wilma Russo, and Andrea Tundis
Department of Electronics, Computer and System Sciences (DEIS), University of Calabria, via P. Bucci 41C, 87036 Rende (CS), Italy
{alfredo.garro, wrusso, andrea.tundis}@unical.it

Abstract - *The Service-Oriented paradigm, which conceives software resources as discoverable services available on a network, is proving an effective approach for providing business solutions in distributed and heterogeneous computing environments. However, due to the different and numerous issues to face, it is witnessing a growing interest in the use of methodologies suitable for supporting the development of service-oriented applications. The paper proposes an approach, centered on the Method Engineering paradigm, which enables the definition of new methodologies tailored to address specific issues arising in developing of service-oriented applications through the exploitation of fragments of methodologies existing and experimented. In particular, it is shown how to obtain, through composition of method fragments, a complete process which covers from requirements specification to testing of service-oriented applications. The complete definition of a method fragment (MF-Web Services Builder) and a related CASE tool are also presented along with a case study showing their exploitation for building a real service.*

Keywords: Service-Oriented Applications; Web Services; Method Engineering; Service Component Architecture; Component-Based Development

1 Introduction

The service-oriented approach promotes the reuse of existing assets in the development of new services and represents an effective interoperability solution in distributed and heterogeneous environments [6, 20]. In particular, the availability of wide adopted Web Services standards (WSDL, SOAP, UDDI, etc.) [20] for the description, communication, and discovery of services enables the jointly exploitation of services independently from the specific implementation technologies. Despite these advantages, the development of a service-oriented application faces several challenges concerning: the requirement specification, the application definition, the discovery, deployment, composition, and integration of services, and, finally, the application testing. To address these topics, several software engineering methodologies and related tools have been proposed in the service-oriented domain [18], some of these cover the whole application lifecycle (from requirements to testing) [16, 2], whereas others address specific aspects [9, 11, 3, 13]. However, existing methodologies often cannot be used "as they are" because of the specific characteristics of the application to develop. In these cases, significant efforts are required which are focused or on customization of existing methodologies or on definition of new ones without any fruitful reuse of those existing [18].

A solution, which makes it possible defining methodologies that fit specific necessities without losing the advantages coming from the exploitation of existing and experimented ones, can be represented by the adoption of the Method Engineering paradigm [4, 5] which has already proved its effectiveness in both the object-oriented and agent-oriented software engineering communities [12]. According to this paradigm, a methodology is obtained by assembling pieces of methodologies (method fragments), *ex-novo* defined or obtained from those existing and available in a repository (Method Base) [7].

In this context, this paper aims to bring the benefits arising from the Method Engineering paradigm in the service-oriented domain. In particular, it is shown how a complete development process, which covers from requirements specification to application testing, can be obtained by composing method fragments addressing some specific and recurrent aspects in the service-oriented domain. The complete definition of a method fragment (MF-Web Services Builder) which addresses a specific aspect concerning the development of Web Services is also provided according to the IEEE FIPA Specifications [8]. Moreover, a CASE tool related to MF-Web Services Builder is presented along with a case study showing its exploitation for building a real service.

The paper is organized as follows: Section 2 introduces fundamentals of the Method Engineering paradigm and exemplifies its exploitation in the development of service-oriented applications; a method fragment (MF-Web Services Builder) is completely defined in Section 3, and the related CASE tool along with a case study showing its exploitation is shown in Section 4; finally, conclusions are drawn and future works delineated.

2 Method Engineering and development of Service-Oriented applications

In this Section, an approach centered on the Method Engineering paradigm which enables, as well as in others software engineering domains, the exploitation of fragments of existing and experimented methodologies in defining new ones, is presented. In particular, Section 2.1 introduces fundamentals of the Method Engineering (ME) paradigm whose exploitation in the service-oriented domain is exemplified in Section 2.2 through a process which is obtained by composing method fragments and covers from requirements specification to testing of a service-oriented application.

2.1 The Method Engineering paradigm

ME allows for obtaining Software Engineering Processes (SEPs) by defining and combining method fragments able to support specific phases of a development process and/or to address specific issues or application aspects. Method fragments which can be either defined ex-novo or obtained by fragmentizing existing methodologies, are auto-consistent and reusable methodological chunks stored in a repository, called Method Base, from which they can be retrieved and assembled during the construction of a SEP.

According to the IEEE FIPA Specifications [8], a method fragment defines a process which receives a set of input work-products and produces a set of output work-products by possibly managing intermediate work-products. A fragment is further characterized by application guidelines that illustrate how to use the fragment and the related best practices, a glossary of the exploited terms, composition guidelines which describe the issues addressed by the fragment, and dependency relationships which give information about others possible related fragments. These meta-data constitute the fragment description which can be codified and stored in the Method Base for facilitating fragment retrieval, adaptation and composition [8].
The definition of a SEP according to the Method Engineering paradigm requires the following main steps (see Figure 1):

1. definition of the characteristics of the SEP and the activities to be carried out (SEP Lifecycle Definition);
2. selection of the methodological fragments from the Method Base (if available) on the basis of the activities defined in step 1 (Method Fragments Selection);
3. definition of new fragments to cover process activities that are not supported by fragments available in the Method Base (Method Fragments Definition);
4. adaptation and composition of the (selected and/or ex-novo created) method fragments to obtain the SEP (Method Fragments Adaptation and Composition).

With reference to step 4, the fragments can be easily integrated through a work-product driven approach (input work-products of a fragment should be derived from output work-products of other fragments, possibly adapted) [7, 10, 17].

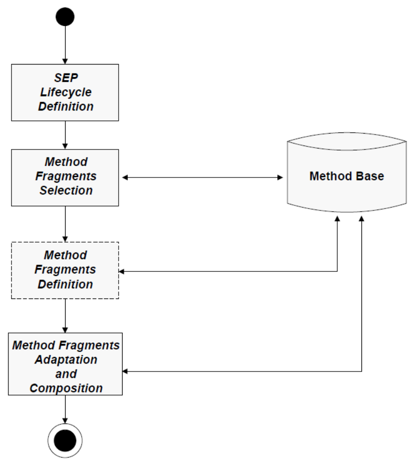

SEP Lifecycle Definition
Method Fragments Selection
Method Base
Method Fragments Definition
Method Fragments Adaptation and Composition

Figure 1. Definition of a SEP according to the ME paradigm

2.2 Exploiting Method Engineering in the service-oriented domain

According to the ME paradigm the development of a specific service-oriented application can be addressed by the composition of method fragments either retrieved from a Method Base and possibly adapted or defined ex-novo. In particular, the following method fragments can be identified:

1. Requirements Specification, which formalizes the application requirements as output work-product.
2. Application Definition, which, starting from application requirements, makes available a choreography of an application.
3. Service Discovery, which discovers, on a given research domain, the set of services able to cover the roles of a given choreography of an application.
4. Service Development, which builds and makes available a service adhering to a specific service contract.
5. Service Composition, which specifies the interactions of a set of services selected to cover the roles of an application choreography.
6. Service Integration, which implements the communication infrastructure among the services constituting an application.
7. Application Testing, which aims at validating and evaluating an application.

An example of composition of method fragments for obtaining a complete process for the development of service-oriented applications, which covers from requirements specification to application testing, is showed in Figure 2.

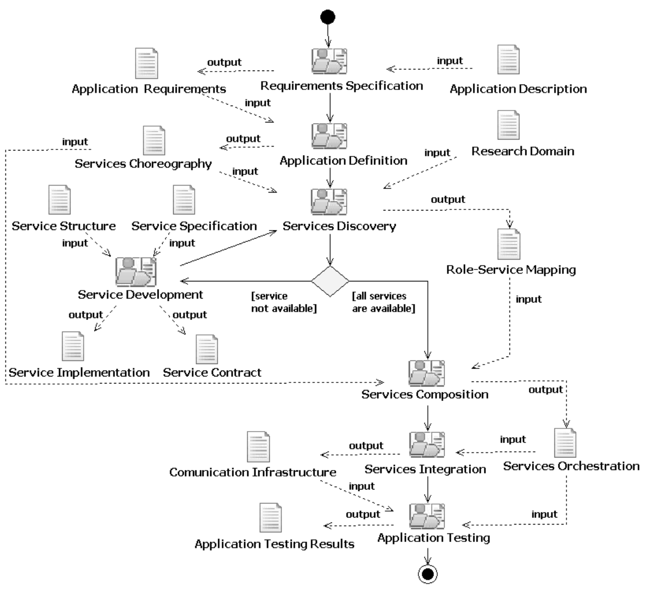


Figure 2. An example of method fragments composition

The process adheres to a typical iterative-incremental lifecycle and the involved fragments are integrated through a work-product driven approach (input and output work-products are reported in Table 1)

TABLE I. Method fragments and related work-products

Method Fragment	**Input Work-Products**	**Output Work-Products**
Requirements Specification	- Application Description	- Application Requirements
Application Definition	- Application Requirements	- Services Choreography
Services Discovery	- Services Choreography - Research Domain	- Role-Service Mapping
Service Development	- Service Structure - Service Specification	- Service Contract - Service Implementation
Services Composition	- Services Choreography - Role-Service Mapping	- Services Orchestration
Services Integration	- Services Orchestration	- Communication Infrastructure
Application Testing	- Services Orchestration - Communication Infrastructure	- Application Testing Results

It is worth noting that: (i) starting from a process definition, the availability of method fragments for the service-oriented domain would allow covering the different process phases and/or addressing the different development issues in different and specific ways depending on the chosen fragments; as an example, the availability of different fragments for Service Discovery would allow to address this aspect with different approaches and techniques [11]; (ii) specific SEPs can be defined by composing the method fragments selected on the basis of the desired software development lifecycle; as an example, if the services that constitute the application are all available and their interactions have been already defined, a light process which covers only Service Integration and Testing could be defined and related fragments selected and composed.

3 Component-Based Development of Web Services

In the development of a service-oriented application, a central issue as that concerning the implementation of new services can be addressed through a specific method fragment (see Section 2). In the following, the complete definition of a method fragment for (Web)Services Development (MF-Web Services Builder) is provided. This fragment, once available in a Method Base, could be exploited, as is or after suitable adaptation, in the composition of specific processes that require development of services, processes that can be both complete as that exemplified in Section 2.2 or cover only some phases of the application lifecycle. According to the IEEE FIPA Specifications [8], the description of the process defined by the fragment as well as that of other meta-data concerning the features and the use of the fragment are described in Sections 3.1 and 3.2 respectively.

3.1 Process definition

MF-Web Services Builder conceives and structures a Web Service as a set of interconnected and jointly working components built using the same or different technologies and executed on the same machine or across a network [14].

The concrete model exploited by MF-Web Services Builder for defining Web Services following this component-based approach adheres to SCA Specifications [14], and, in particular to the SCA Assembly Model [14] which models a service-oriented application as a SCA domain consisting of a set of Services called composite which are in turn structured in components. Each component offers a set of business functions (or services) to other components and can have settable properties which influence the execution of business functions; moreover, dependencies of each component on services provided by other components are called references. The configuration of a component requires both to set values for its properties and to wire its references to services provided by other components. In a composite a component is responsible to make available and exploitable the services provided by the composite (the focus component). A composite has settable properties which are related to those of its components and references to other composites. A service contract, which can be codified by a WSDL file, gathers the services provided by a composite.

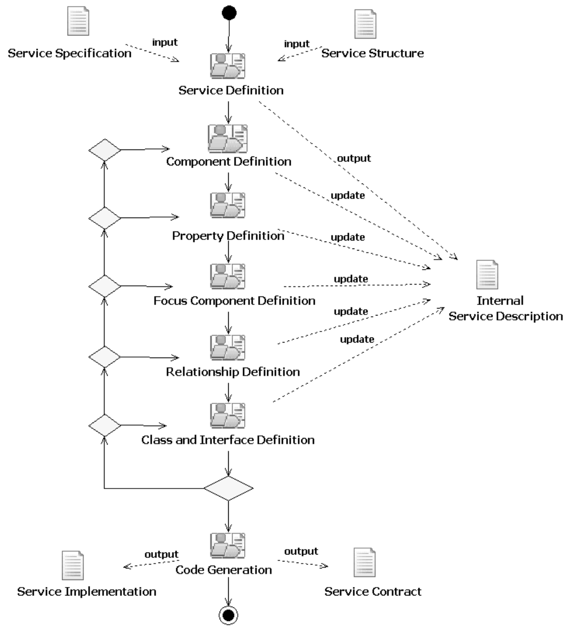


Figure 3. The process provided by MF-Web Services Builder

The adoption of the above described SCA Assembly Model allows defining a complete process for the development of a Web Service conceived as a SCA composite. In Figure 3 the process, which constitutes the core element of the MF-Web Services Builder method fragment, is showed. In particular, the process starts with the Service Definition activity which deals with the analysis of the input work-products (Service Specification and Service Structure) described in Section 3.2. The result of this activity is a clear definition of the Service requirements which is also captured in a refined version of the input work-products. Then, a component-based structure of the composite implementing the Service is obtained and the capabilities and role of each component are specified (Component Definition activity). For each component its properties which capture the component state are introduced in the Property Definition activity. One of the components is selected as an interface with the environment (Focus Component Definition activity). After selecting the Focus Component, the URI and the contract of the Service are defined. At this point of the process, the references (and wires) among the Service components (Component Relationship Definition activity) and the interfaces and classes which constitute each component are defined. The concrete implementation of Service components if not available should be also provided. During these activities, an Internal Service Description which contains a description of the components, their relationships and properties, is constantly updated (see the support work-products in Section 3.2).

The described process can be iterated until the service requirements are met. Finally, the Service Contract is generated along with the Service Implementation (Code Generation activity) which can be refined by adding functions and methods on the basis of both the Service business logic and the chosen implementation (see the output work-products in Section 3.2).

3.2 Meta-data

The IEEE FIPA meta-data related to MF-Web Services Builder are the following:

- Fragment Prerequisites: the specifications of the Service to be developed must be available and formalized in the input work-products (Service Specification and Service Structure).
- Input work-products: a text document (Service Specification) with an informal description of the service to be developed and an UML class diagram (Service Structure) representing a high-level view of the service structure should be provided.
- Output work-products: the fragment produces an implementation of the Service (Service Implementation) and a WSDL file (Service Contract) that describes the offered services and how to structure messages and data for their exploitation.
- Support work-products: a SCDL (Service Component Description Language) file (Internal Service Description), which describes the Service components, their relationships and properties, is constantly updates during the development process.
- Application Guidelines: a set of best practices, examples and case studies which show when and how to effectively exploit MF-Web Services Builder, are provided with the fragment documentation.
- Glossary of terms: to avoid misunderstanding and an uncorrected use of the fragment, a glossary which reports the definition of the main terms used for the fragment definition is delivered. In particular, a definition of the terms related to the SCA Assembly Model [14] is provided.
- Composition Guidelines: useful information for guiding the work-product based composition of MF-Web Services Builder with other fragments is provided (see Section 2).
- Dependency Relationships: the input work-products for MF-Web Services Builder should be provided by those method fragments addressing the identification of Web Services (e.g. the Application Definition method described in Section 2.2); the output work-products of MF-Web Services Builder can represent input for others method fragments as the Services Composition fragment (see Section 2.2).

4 MF-Web Services Builder CASE Tool

To allow a concrete exploitation of the methodological approach to Web Services development provided by MF-Web

Services Builder, a CASE tool (MF-Web Services Builder CASE Tool) has been implemented; its architecture and provided functionalities are presented in Section 4.1, whereas a case study showing its effectiveness for building a real service is reported in Section 4.2. The fragment documentation along with the source code of the CASE Tool is available on the OpenKnowTech Project web site [15].

4.1 An overview of the CASE tool

MF-Web Services Builder CASE Tool, which supports the execution of the process defined by MF-Web Services Builder (see Figure 3), has been designed and implemented as a plugin for the Eclipse platform and according to the SCA specifications by exploiting, in particular, the SCA Composite Designer which is part of the Eclipse STP/SCA project [19] and allows a Model-Driven and component-based service development. The CASE Tool is able to:

- support the execution of the process activities through a wizard-based visual interface;
- create and constantly update the SCDL XML file containing the description of the service components, their relationships and properties;
- automatically generate the Service Implementation (currently in Java) and the Service Contract as a WSDL file;
- deploy the developed service in a SCA runtime environment for making it available at a specified URI.

An overview of the MF-Web Services Builder CASE Tool architecture is reported in Figure 4.

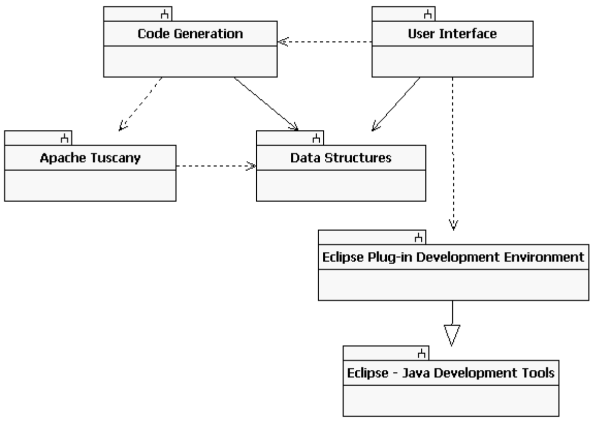
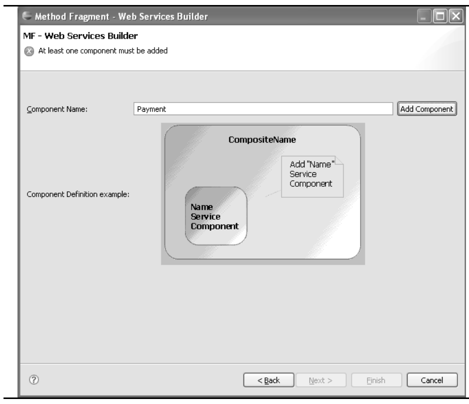

Code Generation
User Interface
Apache Tuscany
Data Structures
Eclipse Plug-in Development Environment
Eclipse - Java Development Tools

Figure 4. The architecture of the MF-Web Services Builder CASE Tool

In particular: (i) internal data structures (Data Structures) map the basic concepts of the SCA Assembly Model; (ii) the visual user interface has been obtained by defining a set of wizards (User Interface) through the Eclipse PDE (Plug-in Development Environment); (iii) the generation of the WSDL service contract and the service deployment are obtained by using the Tuscany Apache framework [1] (Code Generation) and the Tuscany SCA runtime environment (Apache Tuscany) respectively, choosing Tuscany among the implementations of the SCA Specifications [14] as it is complete, well-documented and widely adopted.

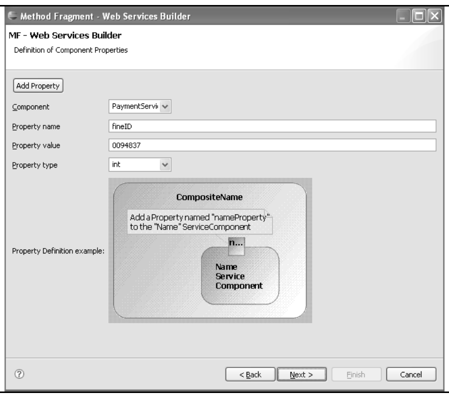

Method Fragment - Web Services Builder
MF - Web Services Builder
Add Component
CompositeName
Add "Name" Service Component
Name Service Component
< Back
Next >
Finish
Cancel

At least one component must be added

Component Name: Payment

Component Definition example:

5.a


Method Fragment - Web Services Builder
MF - Web Services Builder
Add Property
CompositeName
Add a Property named "nameProperty" to the "Name" ServiceComponent
n...
Name Service Component
< Back
Next >
Finish
Cancel

Definition of Component Properties

Component: PaymentServi

Property name: fineID

Property value: 0094837

Property type: int

Property Definition example:

5.b

Figure 5. Component (5.a) and Property Definition (5.b) wizards

4.2 Exploiting the CASE Tool

The feasibility of MF-Web Services Builder and its supporting CASE tool in the development of Web Services, and, in particular, the significant reduction of programming and implementation efforts are demonstrated through a simple but real case study concerning the development of the FinePayment Service, a Web Service for the online payment of fines by the Local Police.

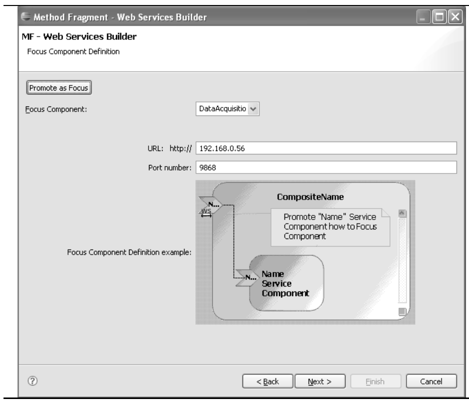
According to the process provided by MF-Web Services Builder (see Figure 3), on the basis on the analysis and definition of the service requirements (Service Definition activity) the Component Definition activity has identified the following three components:

- Payment, which manages data to perform fine payment;
- Data Access, which allows retrieving fine data and storing payments;
- Data Acquisition, which temporarily stores the fine data and exploits the functionalities provided by the other service components.

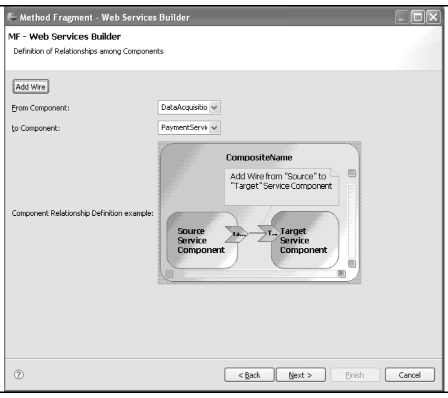
In the Property Definition activity the following component properties were defined: (i) amount of fine, interest on late payment, and total amount (Payment properties); (ii) data base connection settings (Data Access properties); (iii) fine ID, vehicle plate number, driver name, and bank account for the payment (Data Acquisition).

In Figure 5 the definition of the Payment component and its properties through the wizards provided by MF-Web Services Builder CASE Tool are reported.

In the Focus Component Definition activity, the Data Acquisition component was promoted as Focus and the URL and the port number to identify the Service on the network was specified (see Figure 6.a). In the Component Relationship Definition activity the references among components were introduced (see Figure 6.b).


Method Fragment - Web Services Builder
MF - Web Services Builder
Focus Component Definition
Focus Component: DataAcquisitio
URL: http:// 192.168.0.56
Port number: 9868
Focus Component Definition example:
Promote "Name" Service Component how to Focus

6.a

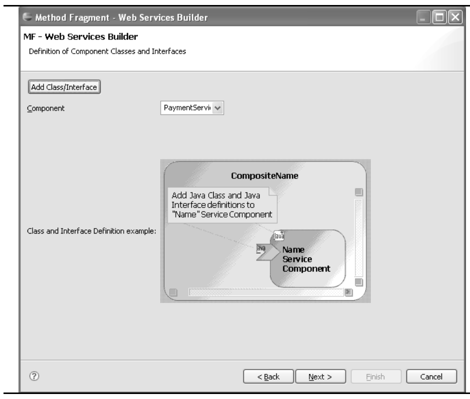

Method Fragment - Web Services Builder
MF - Web Services Builder
Definition of Relationships among Components
From Component: DataAcquisitio
to Component: PaymentServi
Component Relationship Definition example:
Add Wire from "Source" to "Target" Service Component

6.b

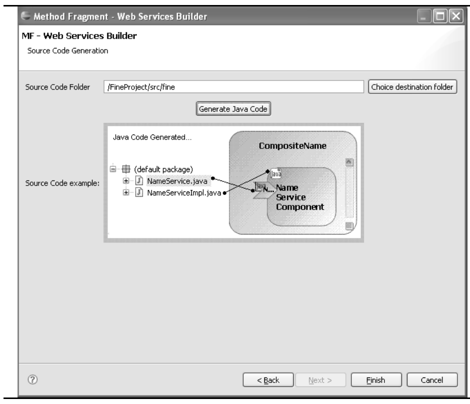
Figure 6. Focus Component Definition (6.a) and Relationship Defintion (6.b) wizards

In the Class and Interface Definition activity classes and interfaces (available or defined *ex-novo*) were associated to each component for implementing its functionalities, and, finally, the Service Implementation and its Contract were automatically generated (see Figures 7.a and 7.b).

Figure 8 shows the high level architecture of the real system for the management of fines in which the *FinePayment* service has been deployed. This system is currently distributed by a European software vendor and used by some public administrations.


Method Fragment - Web Services Builder
MF - Web Services Builder
Definition of Component Classes and Interfaces
Component: PaymentServi
Class and Interface Definition example:
Add Java Class and Java Interface definitions to "Name" Service Component

7.a


Method Fragment - Web Services Builder
MF - Web Services Builder
Source Code Generation
Source Code Folder: /FineProject/src/fine
Source Code example:
Java Code Generated...
(default package)
NameService.java
NameServiceImpl.java

7.b

Figure 7. Class and Interface Definition (7.a) and Code Generation (7.b) wizards

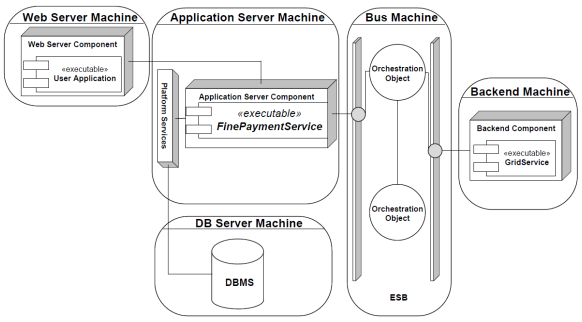


Figure 8. Execution environment of the FinePayment service

5 Conclusions and future work

The variety of contexts in which to develop service-oriented applications actually makes it rather difficult to use a unique methodology and flexible enough to effectively support the development of any application regardless of the specific context in which the application relates. Therefore, due to the heavy and time-consuming efforts required to adapt an existing methodology, when it is necessary to address specific issues arising in developing a specific service-oriented application, often make it more profitable to define a new methodology without any fruitful reuse of the existing ones. This paper has proposed the exploitation of the Method Engineering (ME) paradigm in the service-oriented domain which allows defining methodologies which fit specific needs through the composition of method fragments, extracted from existing methodologies or defined ex-novo. In particular, a set of method fragments able to address recurrent challenges in the development of service-oriented applications has been individuated and composed through a work-product driven composition so obtaining a complete development process. An example of the complete definition of a method fragment (MF-Web Services Builder) concerning the central issue of service development along with the implementation and experimentation of a related CASE tool have been provided. MF-Web Services Builder adopts as the reference logical and architectural service model that defined by the SCA Specifications; moreover, its definition conforms to the IEEE FIPA Specifications so enabling fragment storage in any FIPA compliant Method Base and composition with FIPA compliant method fragments in a specific service-oriented development process.

The ongoing experimentation aims at evaluating the benefits that could arise from the exploitation of the ME paradigm in the service-oriented domain and, in particular, from the availability of method fragments and related CASE tools to exploit or possibly adapt in the definition of development processes of service-oriented applications. Currently, main efforts are geared to: (i) define method fragments covering central issues in the development of service-oriented applications; currently a fragment for supporting the composition of Web Services is under definition; (ii) obtain new and more complete development processes through the composition of the defined method fragments; currently the integration of MF-Web Services Builder with a method fragment for testing the quality of Web Services is under consideration; (iii) define design patterns able to easily drive the execution of the component-based developed process provided by MF-Web Services Builder.

6 References

[1] Apache Tuscany Framework, documentations and software, http://tuscany.apache.org

[2] Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., Holley, K.: SOMA: A method for developing service-oriented solutions. In: IBM Systems Journal, vol. 47, n. 3, pp. 377--296, IBM Corp. Riverton, NJ, USA (2008).

[3] Baldoni, M., Baroglio, C., Martelli, A., Patti, V.: Reasoning about interaction protocols for customizing web service selection and composition. In: The Journal of Logic and Algebraic Programming, vol. 70, n. 1, pp. 53--73, Elsevier B.V., Amsterdam, The Netherlands (2007).

[4] Brinkkemper, S.: Method Engineering: engineering of information systems development methods and tools. Information and Software Technology, vol. 38, n. 4, pp. 275--280, Elsevier B. V., Amsterdam, The Netherlands (1996).

[5] Brinkkemper, S., Lyytinen, K., Welke, R.: Method engineering: principles of method construction and tool support. Springer-Verlag, Berlin Heidelberg, Germany (1996).

[6] Bruni, R., Lluch Lafuente, A., Montanari, U., Tuosto, E.: Service Oriented Architectural Design. In: Proceedings of the 3rd International Symposium on Trustworthy Global Computing (TGC'07). LNCS, vol. 4912, pp. 186--203, Springer, Heidelberg (2008).

[7] Cossentino, M., Fortino, G., Garro, A., Mascillaro, S., Russo, W.: PASSIM: a simulation-based process for the development of multi-agent systems. Int. J. of Agent-Oriented Software Engineering, vol. 2, n.2, pp. 132--170, Inderscience Enterprises Ltd., United Kingdom (2008).

[8] Cossentino, M., Gaglio, S., Garro, A., Seidita, V.: Method fragments for agent design methodologies: from standardisation to research. Int. J. of Agent-Oriented Software Engineering, vol. 1, n. 1, pp. 91--121, Inderscience Enterprises Ltd (2007).

[9] Di Penta, M., Canfora, G., Esposito, G., Mazza, V., Bruno, M.: Search-based testing of service level agreements. In: Proceedings of GECCO 2007, the Genetic and Evolutionary Computation Conference, London, England, UK (2007).

[10] Fortino, G., Garro, A., Russo, W.: An integrated approach for the development and validation of multi-agent systems. In: Int. J. of Computer Systems Science & Engineering, vol. 20, n. 4, pp. 259--271, CRL Publishing Ltd, Leicester, United Kingdom (2005).

[11] Giallonardo, E., Damiano, G., Zimeo, E.: onQoS-QL: A Query Language for QoS-Based Service Selection and Ranking. LNCS, vol. 4907, pp. 115--127, Springer, Heidelberg (2009).

[12] Henderson-Sellers, B.: Method engineering for OO systems development. Communications of the ACM, vol. 46, n. 10, pp. 73--78, ACM press (2003).

[13] Koehler, J., Srivastava, B.: Web service composition: current solutions and open problems, In: Proceeding of ICAPS 2003 Workshop on Planning for Web Services, pp. 28--35, 2003.

[14] Marino, J., Rowley, M.: Understanding SCA (Service Component Architecture). Addison-Wesley Professional (2009).

[15] OpenKnowTech Project, documentations and software available at http://www.openknowtech.it

[16] Papazoglou, M.P., Van Den Heuvel, W.J.: Service-oriented design and development methodology. In: International Journal of Web Engineering and Technology, vol. 2, n. 4, pp. 412--442, Inderscience Publishers, Geneva, Switzerland (2006).

[17] Ralyté, J., Rolland, C.: An assembly process model for method engineering. In: Proceedings of CAISE01, the 13th Conference on Advanced Information Systems Engineering, Interlaken, Switzerland (2001).

[18] Ramollari, E., Dranidis, D., Simons, A.J.H.: A survey of service-oriented development methodologies. In: Proceedings of the 2nd Young Researchers' Workshop on Service Oriented Computing, Leicester, UK, pp. 75--80 (2007).

[19] STP/SCA Tools project, components and tools, http://wiki.eclipse.org/SCA

[20] Sweeney, R.: Achieving Service-Oriented Architecture: Applying an Enterprise Architecture Approach. Wiley & Sons, Inc., New Jersey, USA (2010).