Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CADSSO: A DEVELOPMENT APPROACH FOR CONTEXT-AWARE SERVICE ORIENTED SYSTEMS

[1]M. LETHRECH, [2]A. KENZI, [3]I. ELMAGROUNI, [4]M. NASSAR, [5]A. KRIOUILE
[1,3,4,5]SIME Laboratory, ENSIAS, Mohammed V University, Rabat - Morocco
[2]Sidi Mohamed Ben Abdellah University, Fes - Morocco
E-mail:[1, 3]{mohammed.lethrech, issam.elmagrouni}@um5s.net.ma, [2]adil.kenzi@gmail.com
[4, 5]{nassar, kriouile}@ensias.ma

ABSTRACT

In this paper, we present our development approach named CADSSO (Context-Aware, Domain Specific and Service Oriented) for Context aware Service oriented Systems. CADSSO is organized in two stages: the modeling stage and the code generation stage. It is essentially based on the Domain Specific Modeling (DSM) approach. Indeed, the creation of the five models of the modeling stage is done using Domain Specific Languages (DSLs). The first model is Domain Specific Context Model, it represents the service execution context. The second is Domain Specific Service Model, it allows the services modeling. We also have the Service Variability Model which is used in the service variability modeling. The bond between Service Variability Model and Domain Specific Service Model is provided by Adaptation rules model. Finally, specific domain business modeling is done via domain specific business model. At code generation stage, a specific code generator transform all these models to final source code using a framework specifically designed for a given target platform.

Keywords: *Domain Specific Modeling (DSM), Domain Specific Language (DSL), Service Oriented Computing (SOC), Context-Aware Computing (CAC), Adaptation*

1. INTRODUCTION

Since the recognition of software development as an engineering discipline at the 1968 NATO Conference [1], the history of software engineering is essentially summarized in levels of abstraction rising, said Grady Booch in his 2002 talk "The Limits of Software". Raising the level of abstraction is a necessary condition for improving productivity. For example, the passage from assembler to third generation languages allowed increasing productivity by more than 400% according to [2]. The Domain Specific Modeling (DSM) approach is the main actor of another revolution on raising the level of abstraction [3][4]. DSM is characterized by, 1- the use of a Domain Specific Languages (DSLs) for modeling the solution of a narrow field, 2- full automatic generation of final solution from the high level specifications. A DSL is by nature semantically rich, since it uses directly the concepts and business rules of a specific domain. In addition to production increase, estimated between 300% and 1000% [5], adoption of DSM approach in software engineering provides a lot of advantages essentially a better quality of generated code and a better reactivity to business rules and technological changes [3].

On the other hand, the issue of "context" now becomes a hot topic in human computer interaction research and development especially with the pervasive use of mobile devices and the need for ubiquitous computing [6]. In these settings, adaptability becomes a key feature of services as it provides a way for an application to continuously change itself in order to satisfy new contextual requirements [7]. Responsiveness to business rules changes is also treated by the Context-Aware Computing (CAC) paradigm [8]. This last, mainly deals with the development of adaptive systems, that is to say, which can discover and take advantage of contextual information [9].

Other important quest in software engineering is to find a solution that simplifies development and implementation, supports effective reuse of software assets and facilitates the communication and integration of distributed systems. Service Oriented Computing (SOC) is the latest fad in this elusive quest [10]. SOC has eliminated huge amounts of redundant software automated manual process and increased productivity [11]. SOC uses

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Service Oriented Architecture (SOA) which is an architectural style of building software applications that promotes loose coupling of independent services, allowing linking resources on demand [12].

The adoption of SOC and CAC for the development of adaptable service oriented systems encounters several problems, essentially the lack of modeling approaches, processes, techniques and tools to facilitate construction of these systems [13] along with time and cost optimization.

In this paper we present our development approach for adaptable service oriented systems. We can organize our approach in two stages: modeling stage and code generation stage. Modeling stage requires the creation of five models. The first model is the Domain Specific Context model, which is responsible for modeling the system context of use. Context model is conforms to our proposed context meta-model. The second model takes care of service modeling is the domain specific services model. This last is conforms to a service meta-model which must be an extension to our generic service meta-model. The third model is service variability model. It formulates services forms of adaptations. This model is conforms to our service variability meta-model. The fourth model is adaptation rules model. It makes the join between service variability model and domain specific context model. The last one is domain specific business model, which is in charge of the domain specific business modeling. The domain developer must produce the domain specific business meta-model which is the abstract syntax of the domain specific business language. In the code generation stage the domain developer must produce modeling tools and code generator. This last converts the models produced during the modeling stage to final source code.

To sum up, the domain developer must produce the service meta-model (abstract syntax) as an extension of our generic service meta-model and the domain specific business meta-model. He must also produce the concrete syntax of modeling languages and related tools as well as the code generator.

The remainder of the paper is organized as follows: the second section describes the case study; the third introduces our CADSSO approach. The modeling stage and the code generation stage are respectively illustrated in the fourth and fifth sections. Our approach is evaluated in the sixth section. The related work is presented in the seventh section. Finally, the article ends with a conclusion and outlook.

2. A CADSSO SCENARIO

Our approach can be illustrated by a domain specific case study which presents a great adaptation potential. Our choice was fixed on tax domain. It should be noted that a tax information system is supposed to be updated for each finance law publication. Therefore, tax information system must be flexible, scalable and customizable as much as possible. This business change (functional dynamic) represents, according to Kelly [3], one of the essential points justifying the adoption of the DSM approach. We focused only on the calculation and restitution of corporation tax. The main services of our validation scenario are “TaxCalculation” and “TaxRestitution”. The former is a generic service which will be specialized by the “CTCalculationService” (corporation tax calculation service), specific for calculating the value of corporation tax. The latter is a business process composed of several services. It allows a restitution of the corporation tax. Based on our Tax meta-model (see section 6) we can generate models for each tax (corporation tax, income tax…).

The business logic of these services depends on several parameters, for example: ratepayer exemption, ratepayer category, ratepayer regime…. These parameters change from year to year. For example “CTCalculationService” service has four logical variabilities (see figure 7). In addition to this business variability it is possible to imagine other technical variabilities, for example the device used by the ratepayer, the ratepayer's position….These motivating scenario provides a good illustration of context awareness.

3. CADSSO APPROACH SYNOPTIC

Our CADSSO approach adopts the concepts and principles of Model Driven Software Development (MDSD). It aims to facilitate modeling and development of adaptable service oriented systems, using DSM approach. In order to have adaptable services, service variability should be modeled and designed at the first stages of a modeling approach. Also, separation of concerns is the cornerstone of system flexibility. In fact, our modeling approach ensures a separation between service variability, context and service adaptation rules.

Our CADSSO approach is divided into two stages: Modeling stage and code generation stage.

The modeling stage is based on five models:

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- (a) Domain specific services model: is a representation of the domain specific services. We have defined a generic abstract syntax which can be extended by a specific abstract syntax to produce a domain specific abstract syntax [14];
- (b) Service variability model: represents service variabilities. Each service is adaptable based on its variation points;
- (c) Domain specific context model: models the context elements, which influence the services adaptation;
- (d) Adaptation rules model: provides answer to the following question: when each service variation point must be used related to the context variation? In other words it represents the relation between service variability model and context model;
- (e) Domain specific business model: used to model the domain specific business. The domain specific business model is modeled by Domain specific language (see section 6.1.2.5).

In the code generation stage the language developer must produce the domain specific business meta-model, the specific service meta-model (abstract language) as an extension of our generic service meta-model, the language (concrete syntax), the domain framework and the code generator.

The domain specific code generator uses the domain specific framework to transform models into full source code. The domain framework essentially removes duplication from the generated code, hides the target environment and facilitates integration with existing code [3]. The figure 1 illustrates our CADSSO approach.

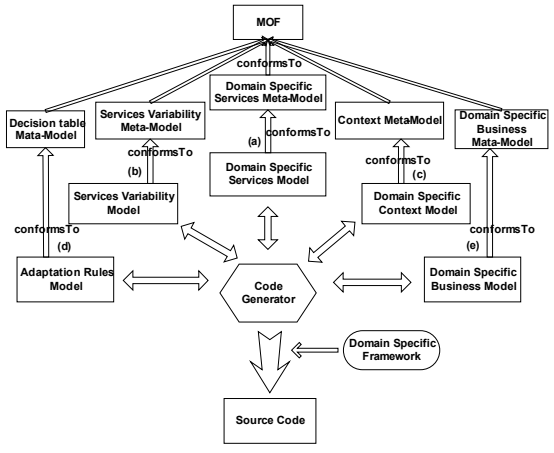


Figure 1: CADSSO approach Synoptic

4. CADSSO MODELING STAGE

In this section, we have been focusing on the modeling side of our approach [15]: context modeling, service modeling, service variability modeling, adaptation rules modeling and domain specific business modeling.

4.1 Context Modeling

A lot of context definitions are found in literature [16][17]. A comparison of these definitions is out of the scope of this paper. The most appropriate and the most referred [18][19] context definition is given by Dey [16].

"Any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves."

There is a growing body of research on context meta-models [20][21][22][23][24][18][25]. Our proposed context meta-model is directly joined to the service variabilities. Each service is adaptable based on its service variation points.

For us a Context is composed of ContextElements and a ContextElement is composed of ContextParameters. We have also defined the ServiceContextElement entity which is a ContextElement specific to one Service. The ServiceContextElement can contain ContextParameters from different ContextElements, it can also contain its own ContextParameters which is specific to the service. Each ServiceVariationPoint (from service variability meta-model) is joined to one instance of ServiceContextElement.

The context can be divided into a lot of sub-contexts. This division can be based on the source of sub-context elements, such as: environment (temperature, position, battery level…), user (user profile), computational, sensed …etc [26][21][20]. We have left the context meta-model open to support any type of sub-context. Our context meta-model is illustrated in the figure 2.

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

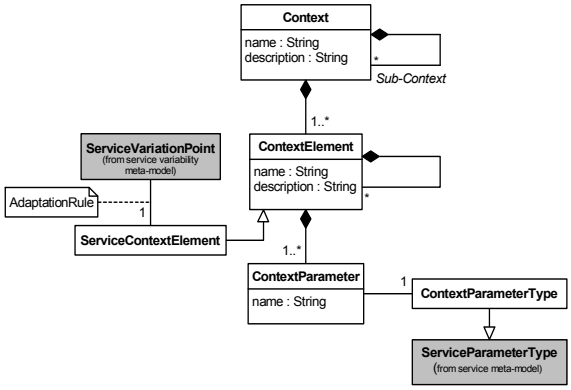


Figure 2: Context meta-model

4.2 Service Modeling: Generic Service Meta-Model

The service is the central element of a service-oriented information system. The urbanization of the latter passes, inevitably, by a service modeling stage.

We have categorized the service entity into three categories: Business service, Utility service and Scheduling service. The latter simply uses the already implemented BusinessServices through ShedulingStrategies (see CTRestitutionProcess in the section 6.1.2.1). The BusinessServices (see CTCalculationService in the section 6.1.2.1) implement business activities and the UtilityServices constitute non functional activities. The ShedulingStrategy identifies the services variabilities and their orchestration strategy used by a SchedulingService.

Each Service has at least one ServiceVariationPoint which represents a variability of the adaptable service. A service has one or more ServiceOperations and a ServiceOperation has ServiceParameters; a Service also has one or more Service Level Agreements (SLA). According to service attribute "type", the implementation will change (big web service, restful web service…etc). Each SLA has conditions which are a boolean expressions. Each condition has Reactions (e-mail, sms …etc) and a Reaction is an UtilityService. The requester -which represents a service user-, uses a service through a SLA. A ServicePackage groups one or more services. The ServiceInterface make it possible to explicitly model the operations provided by a service using this interface. A service has just one BaseServiceInterface which groups operations provided for all service users. A ServiceOperation has a set of ServiceParameters. A Process is a "ShedulingService". We use the "process-as-a-service" approach in all the life-cycle of a process. Finally, BusinessServices can use UtilityServices. The relation "uses" (from service A to service B) means that the service A can call the service B. Our generic service meta-model is represented in the figure 3.

The language developer adds his domain elements to our generic service meta-model [14] to have his domain specific meta-model. Our extension mechanism is based on specialization relation (see section 6.1.1.1). All our meta-models are in conformity with the MOF (Meta-Object Facility) model [27].

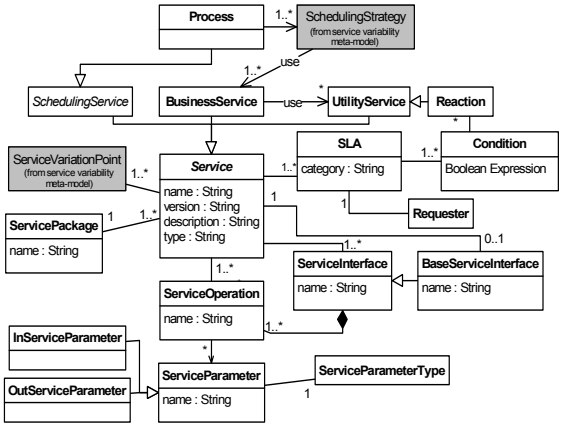


Figure 3: Generic service meta-model

4.3 Service Variability Modeling

Adaptation in software engineering received a lot of research attention these last years. Several categorizations of adaptation exist in the literature. Raman and al. [7] proposed a goal-oriented categorization. They distinguish, in his adaptation taxonomy, between corrective, adaptive, perfective, extending and preventive adaptation. In addition, they divided adaptive adaptation into context-aware, customization/personnalization and mediation adaptation. Bucchiarone [28] and al. differ between at run-time adaptation and designed adaptation. The first one for on-the-fly adaptation, the second, requires analyzing all the possible adaptation case at design time. Khouloud [20] listed three types of adaptation: reflexive adaptation, adaptation controlled by policies and adaptation by weaving aspects. The first is the ability of a system to observe and act on itself during its execution.

Using these categorizations we can classify the adaptation treated in our approach as designed and adaptive adaptation. To model adaptation we opted for modeling service variabilities through a dedicated model.

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Service Variability is a characteristic (logic, graphical presentation, persistence … etc) which may vary within a service, according to service context of use. The service variability is represented with variation points. A variation point is a place in service where the difference occurs.

Each service has a lot of variation points [29][30][31]. We have categorized these variation points into two categories: FunctionalVariation and TechnicalVariation. The former is divided into three classes: Logic, for the business logic variation; Interface, for methods signature and methods offered -by the service- variation and SchedulingStrategy, for the variation of the sequence of invocation of services used by a process. TechnicalVariation is divided into three classes: Locality, for the machine which hosts the service; GraphicalPresentation, which means that the graphical interface is adaptable according to the device used by the service requester and Persistence, for variation of Scheme or physical representation of persistent data. The persistence attribute "type" specifies the way in which the data will be persisted (relational, XML …etc). Also, SchedulingStrategy is specialized into two sub-strategies: SequentialSchedulingStrategy and ParallelSchedulingStrategy.

A service has a lot of ServiceVariationPoints and each ServiceVariationPoint is matched to a ServiceContextElement instance. Our service variability meta-model is illustrated in the figure 4.

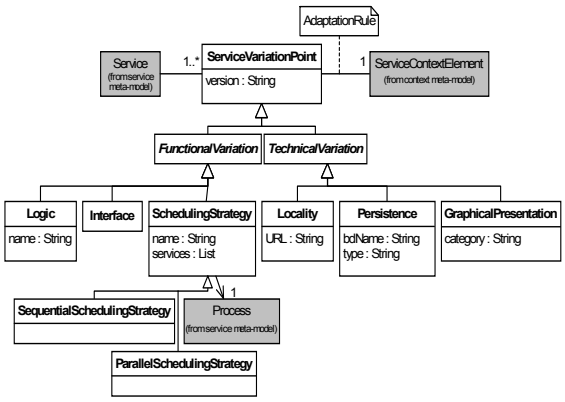


Figure 4: Service variability meta-model

To conclude, we have proposed a MOF [27] meta-model [14] (generic service meta-model and service variability meta-model) for modeling adaptable services using DSM approach. To support service adaptation modeling, we have integrated the service variability notion to our generic service meta-model. It allows modeling the various variants of a service. It should be noted that addition of new context elements or new service variability must be done in a transparent manner and without jeopardizing the service running and availability. In addition, all common concepts of studied service meta-models [20][22][23][32][24][25] are used in our generic service meta-model in order to facilitate the integration of service variability notion to extents service oriented systems.

4.4 Adaptation Rules Modeling

In this section we treat the relation between service variability model and the context model. This connection is ensured by the Adaptation Rules Model. This latter is modeled using complete decision tables with mixed-entries [33].

The concept of decision table is derived from the classical approach based on the data and treatment separation paradigm. Decision tables have been used extensively and successfully to produce specifications for complex systems, they offer the following benefits [33]:

- Easier to understand and review than code, even with many conditions, actions and rules;
- An accurate representation of the internal logic of the function;
- Thanks to the completeness mechanism, it is possible to ensure the completeness of rules for all received input data;

A decision table exhaustively expresses the relationship between the internal logic of a function, input data and output data:

- The provided input data represent the conditions of the decision table: in our case they are in the form of ServiceContextElement parameters; which includes ContextParameters from different ContextElements;
- The output data emanate from the actions triggered by the decision table: in our case the actions are in the form of ServiceVariationPoints ;
- The couples (ServiceContextElement parameters, service variation points) represent the rules of our decision table. This adaptation logic will be industrialized using a rule engine.

We have one adaptation decision table by service. The same is for processes adaptation rules, each process (service) has an adaptation decision table; the conditions represent ServiceContextElement parameters and the actions are in the form of process SchedulingStrategies.

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Decision tables excel at unitary treatment of rules, on the other hand they do not guarantee the exhaustiveness of the treatments of a function. This is where Domain Specific Business Model comes in. This latter allows modeling the internal behavior of methods. For domain specific business modeling, our fifth modeling step, the domain developer must produce a DSL (see section 6.1.2.5).

5. CODE GENERATION STAGE

In the code generation stage the language developer must produce the domain specific business meta-model, the specific service meta-model (abstract syntax) as an extension of our generic service meta-model, the language (concrete syntax), the domain framework and the code generator.

The domain specific code generator uses the domain specific framework to transform models into full source code. The domain framework essentially removes duplication from the generated code, hides the target environment and facilitates integration with existing code [3].

6. APPROACH EVALUATION: TAX CALCULATION AND RESTITUTION

In this paper, we have chosen the tax domain to illustrate our approach. The remainder of this section is divided into two parts: modeling stage and code generation stage.

6.1 Modeling Stage

We can divide the modeling stage into two steps: 1- Meta-models definition (abstract syntax): domain specific service meta-model as an extension of our generic service meta-model and domain specific business meta-model. 2- Models specification: modeling of our five models.

6.1.1 Meta-models definition

6.1.1.1 Domain specific service meta-model definition

For a specific domain, the language developer must produce his domain specific service meta-model as an extension of our generic service meta-model (defined in the section 4.2). Thus, we have to add our tax elements to our generic service meta-model to produce our tax calculation and restitution service meta-model. This latter is divided into Domain Specific Meta-Services and Domain Specific Meta-Elements (see figure 5).

6.1.1.1.1 Domain specific meta-services

6.1.1.1.1.1 Tax calculation meta-services

The main business service is “TaxCalculation”; it allows the calculation of the tax value. Their utility services are: “GetTaxRate”, “GetRatePayerInfo” and “GetRatePayerExemption” they allow getting, respectively, tax rate, information of the ratepayer and ratepayer exemption.

6.1.1.1.1.2 Tax restitution meta-services

For tax restitution we have the following BusinessServices: DemandDeposit, AdvanceRestitution, RestitutionRecordVerification and TaxRestitution. The DemandDeposit service allows deposit of restitution demand; AdvanceRestitution handles an automatic restitution based on the ratepayer category (A, B or C). The RestitutionFolderVerification service performs verifications of the restitution folder (invoices …etc); TaxRestitution service performs the final tax restitution.

In addition to the BusinessServices we have three UtilityServices: GetDeficit, GetExcess, and GetMinimumContributionCredit which allow getting, respectively, deficit, excess and Minimum Contribution Credit of the ratepayer.

6.1.1.1.2 Domain specific meta-elements

A Tax can have a lot of TaxDeclarations (Declaration and Bundle for corporation tax); a Ratepayer can file a lot of TaxDeclarations and a RatePayer can have a lot of Exemptions.

Also, we have the TaxCalculationSLA which is a SLA of the tax calculation business service. TaxRestitutionSLA represents the SLA of tax restitution. The figure 5 represents our tax calculation and restitution service meta-model.

According to our tax meta-model, we can create as much models as taxes. The example illustrated in the figure 6 represents the service model of Corporation Tax (CT) Calculation and Restitution.

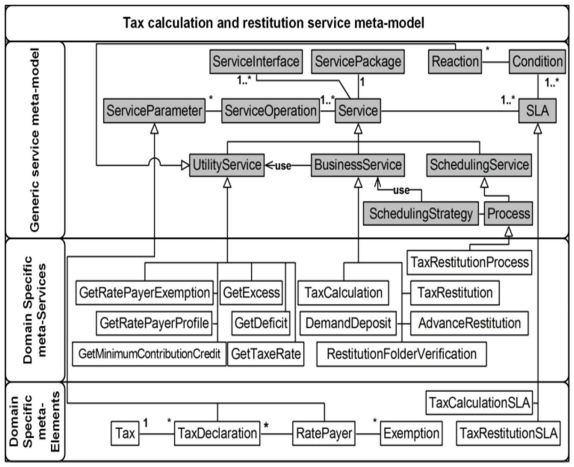


Figure 5: Tax calculation and restitution service meta-model

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6.1.1.2 Domain specific business meta-model definition

For domain specific business meta-model we have used Business Process Model and Notation (BPMN) Object Management Group (OMG) standard [34] and mathematical formulas.

6.1.2 Models specification

6.1.2.1 Corporation tax calculation and restitution service model (conforms to our domain specific meta-model)

The CTCalculationService handles the calculation of corporation tax. It uses the GetCTRate, GetRatePayerProfile, GetRatePayerExemption, GetCTDeficit, GetCTExcess and GetCTMinimumContributionCredit UtilityServices. They allow getting respectively the corporation tax rate, ratepayer profile, the ratepayer exemptions, corporation tax deficit, excess and minimum contribution credit. CTCalculationService takes an in CTDeclaration parameter and has a CTCalculationSLA. This latter is specialized into two sub-SLAs, one for administrators and the other for ratepayers. The response time of CTCalculationService for an administrator must be lower than three seconds; otherwise an e-mail and a SMS must be sent to the exploitation service manager.

The restitution of corporation tax is a SchedulingService named CTRestitutionProcess. It uses four business services: CTRestitutionDemand, CTAdvanceRestitution, CTRestitutionFolderVerification and CTRestitution. They have the same role like described in the meta-model but specific to corporation tax. CTRestitutionProcess has its specific SLA called CTRestitutionSLA. Our corporation tax calculation and restitution service model is illustrated in the figure 6.

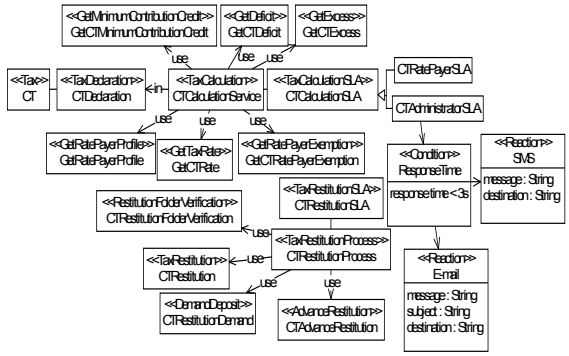


Figure 6: Corporation Tax calculation and restitution service model

6.1.2.2 Corporation tax service variability model (conforms to our service variability meta-model)

In this section we will identify the variation points of each service. Each service has at least one service variation point. For a client service request, just one service variation point is used.

CTCalculationService: has four logical variation points:

- CTCalculationWithExemption: is a corporation tax calculation which takes into account the ratepayer exemptions;
- CTCalculationWithoutExemption: is the basic corporation tax calculation;
- CTCalculationNotResident: is a corporation tax calculation for not resident ratepayers;
- CTAutomaticTaxation: is based on the corporation tax paid for the last four years.

GetCTRate: has three logical variation points:

- GetCTRateNormal: is used to retrieve the corporation tax rate for not financial resident ratepayers;
- GetCTRateFinancial: is used to retrieve the corporation tax rate for banks, insurance and resident ratepayers;
- GetCTRateNotResident: is used for not resident ratepayers.

CTAdvanceRestitution: has two logical variation points:

- CTAdvanceRestitutionCatA: is used for taxpayers whose category is "A". Allows 80% of advance restitution;
- CTAdvanceRestitutionCatB: is used for taxpayers whose category is "B". Allows 50% of advance restitution.

CTRestitutionProcess: has three sequential scheduling strategies. The three strategies are differentiated by the use of CTAdvanceRestitution service:

- CategoryAStrategy: uses the CTAdvanceRestitutionCatA variation point of the CTAdvanceRestitution service;
- CategoryBStrategy: uses the CTAdvanceRestitutionCatB variation point of the CTAdvanceRestitution service;
- CategoryCStrategy: doesn't have advance restitution.

The figure 7 gathers all corporation tax service variabilities.

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

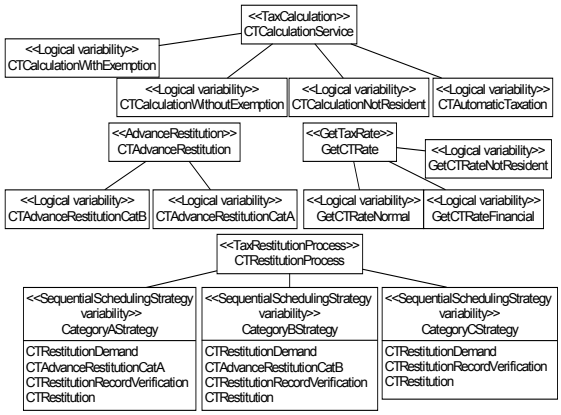


Figure 7: Corporation tax service variability model

6.1.2.3 Corporation tax context model (conforms to our context meta-model)

Our TaxContext model is composed of one subContext: CTContext for corporation tax. Obviously we can add other sub-context for other taxes.

The CTContext is composed of two ContextElements, RatePayer and CTDeclaration. The former is composed of four ContextParameters: Resident, Exemption, TaxRegim and Category. The latter contains just one ContextParameter named WithAnnex. The corporation tax context model is represented in the figure 8.

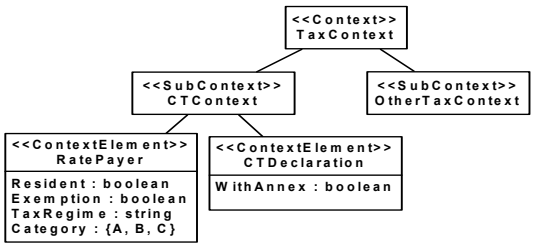


Figure 8: Tax context model

From the above model, and for corporation tax, we have identified for each adaptable service a ServiceContextElement: CTCalculationServiceSC, GetCTRateSC, CTAdvanceRestitutionSC and CTRestitutionProcessSC. Each ServiceContextElement is responsible, respectively, of variations of CTCalculationService, GetCTRate, CTAdvanceRestitution and CTRestitutionProcess services. Our ServiceContextElements are illustrated in the figure 9.

6.1.2.4 Adaptation rules model

In this section we will model services adaptation rules.

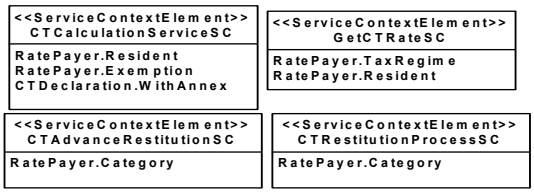


Figure 9: Corporation Tax ServiceContextElements

"*CTCalculationService*" adaptation rules:

Adaptation of this service depends on two context elements, RatePayer and CTDeclaration. We have four logical service variation points: with exemption, without exemption, for non-resident and automatic taxation (see table 1). The conditions possible values are (Yes,No). The indifferent value (-) mean that the condition value can take one of the condition possible values.

Table 1: "CTCalculationService" Decision Table.

CTCalculationService		R1	R2	R3	R4	ELSE
C1	An Exempt Ratepayer	Y	N	N	-	
C2	A Resident Ratepayer	-	Y	N	-	
C3	Is Declaration Deposited	N	Y	Y	N	
C4	Is Annex deposited	N	-	-	-	
A1	CTCalculationWithExemption	X				
A2	CTCalculationWithoutExemption		X			
A3	CTCalculationNotResident			X		
A4	CTAutomaticTaxation				X	
A5	ERROR					X

"*GetCTRate*" adaptation rules:

If the ratepayer is resident and the context parameter "RatePayer.TaxRegime" equals to "Bank" the "GetCTRateFinancial" variation point will be used. If it is a not resident ratepayer, the requester will use the GetCTRateNotResident (see table 2).

Table 2: "GetCTRate" Decision Table.

GetCTRate		R1	R2	R3
C1	Ratepayer's Regime	Other	Bank	-
C2	A Resident Ratepayer	Y	Y	N
A1	GetCTRateNormal	X		
A2	GetCTRateFinancial		X	
A3	GetCTRateNotResident			X

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

"*CTAdvanceRestitution*" adaptation Rules:

For corporation tax restitution we have two different treatments, one for ratepayer A category and the other for B category (see table 3).

Table 3: "CTAdvanceRestitution" Decision Table.

CTAdvanceRestitution		R1	R2	ELSE
C1	Rate payer category	A	B	
A1	CTAdvanceRestitutionCatA	X		
A2	CTAdvanceRestitutionCatB		X	
A3	NoRestitution			X

"*CTRestitutionProcess*" adaptation Rules:

Also restitution process has three SchedulingStrategies, one for each ratepayer category (A, B or C) (see table 4).

Table 4: "CTRestitutionProcess" Decision Table.

CTRestitutionProcess		R1	R2	R3
C1	Rate payer category	A	B	C
A1	Category A strategy	X		
A2	Category B strategy		X	
A3	Category C strategy			X

6.1.2.5 Tax calculation business model: corporation Tax

For domain specific business modeling, the domain developer must produce a domain specific language. In our case we have used Business Process Model and Notation (BPMN) Object Management Group (OMG) standard [34] and mathematical formulas.

Corporation tax calculation business process:

The corporation tax calculation business process shows, functionally, when each logical variation point of CTCalculationService is used (see the figure 10).

The activities Get Corporation Tax Excess, Get Corporation Tax Deficit, Get Corporation Tax Exemption and Get Corporation Tax Credit Minimum Contribution use respectively the utility services GetCTExcess, GetCTDeficit, GetCTRatePayerExemption and GetCTMinimumContributionCredit.

The "Get corporation tax Rate" sub-process (see the figure 11) allows getting the corporation tax rate. In other words it shows how to use the three service variabilities of the GetCTRate service.

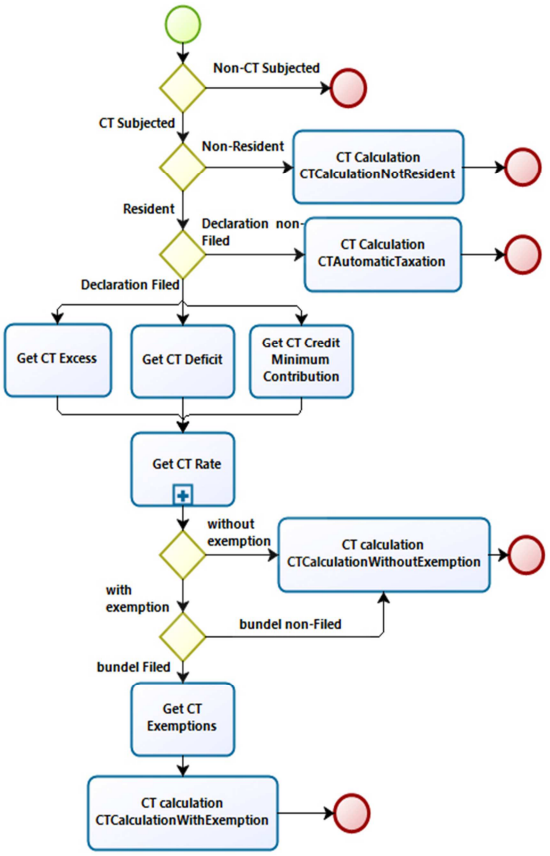


Figure 10: Corporation tax calculation business process

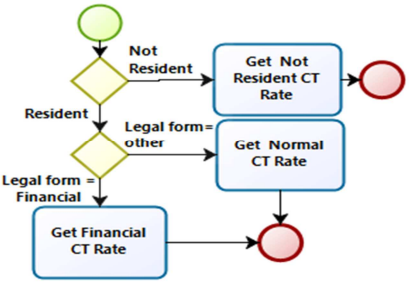


Figure 11: "Get corporation tax rate" sub-Process

CT calculation mathematical formula:

- CT Calculation without exemptions (normal formula):

$$CT = MAX\left(\frac{((E - D) \times CTR) - (MCC + S),}{TT \times MCR}\right) \quad (1)$$

- CT Calculation with exemptions:

$$CT = MAX\left(\frac{((E - E' - D) \times CTR) - (MCC + S),}{TT \times MCR}\right) \quad (2)$$

$$E' = \frac{TE}{TT} \times E \quad (3)$$

- CT Calculation automatic taxation:

$$CT = \frac{\sum_{i=N-1}^{i=N-4} Ei}{4} \times CTR \quad (4)$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- CT Calculation for nonresident:

$$CT = E \times CTR \quad (5)$$

E = Earning **E'** = Exempt Earning
N = Taxation year
CTR = Corporation Tax Rate
MCC = Minimum Contribution Credit (N-1)
MCR = Minimum Contribution Rate
TE = Exempt Turnover **D** = Deficit (N-1)
S = Surplus (N-1) **TT** = Total Turnover

6.2 Code Generation Stage

6.2.1 Graphical concrete syntax

To illustrate our code generator, we used in this paper only two meta-models: domain specific service meta-model and service variability meta-model.

The abstract syntax of our domain specific service and domain specific service variability languages was done by the Eclipse Modeling Framework (EMF) [35][36].

The model used to represent models in EMF is called Ecore. Both EMF and MOF are conceptually very similar and express in reality analogous meta-modeling concepts [37]. More specifically, the current version of the specification, that is MOF 2.4.2, introduces a subset of the MOF that is called the Essential MOF (EMOF). The EMOF meta-model is identical to the Ecore meta-model of the EMF, whereas the differences are predominantly on naming rather than conceptual. Therefore, the EMF can effectively read and write serializations of the EMOF meta-model [38].

We have chosen a graphical representation of our DSLs concrete syntax (corporation tax service and service variability modeling tools, figures 12 and 13), which is more comprehensible than the tree-based view, as the saying goes: a graphical model is worth a thousand words. We have used the Graphical Modeling Framework (GMF) [39][40] to create our DSLs concrete syntax.

Our Corporation Tax service Model designed by our tax service modeling tool is illustrated in the figure 12. The CTCalculationService has two ServiceInterfaces: CTCalculationServiceInterface1 and CTCalculationServiceInterface2. The first ServiceInterface has one ServiceOperation named cTaxCalculation. This latter has two inServiceParameters: ratePayer1 and cTaxDeclaration1, it also has one outServiceParameter named ctCalculationResult.

The same for our corporation Tax service variability model designed by our tax service variability modeling tool (see the figure 13). Our CTCalculationService has four logical Service VariationPoints. The figure 14 shows how to join a service to its variation points.

6.2.2 Models to code transformation

DSM aims to generate code directly from the models without having to modify generated models or code. In other words, while creating a DSM solution the objective is that after generation, additional manual effort to modify or extend the generated code is not needed [3]. With this in mind, our code generator is performed by direct transformation of the abstract domain models to the corresponding source code without any model to model transformation. The table 5 illustrates the mappings between CADSSO approach models and Java API for XML Web Services (JAX-WS) code elements. This mapping is performed through a set of transformation rules.

Table 5: CADSSO approach models and JAX-WS code elements mappings

CADSSO Model elements	Java
Package	Package
ServiceInterface	Interface (@WebService)
TaxCalculation	Class (@WebService)
ServiceOperation	Method (@Override)
OutServiceParameter	method return value
InServiceParameter	method parameter
RatePayer	Class
TaxDeclaration	Class
UtilityService	Class (@WebService)
Process	Class (@WebService)
ServiceVariationPoint	Method

The first version of our code generator generates only the class skeletons. The service operations code will be generated in the next version based on our domain specific business model, domain specific context model and adaptation rules model.

We have chosen the Acceleo Model to Text Language (MTL) to generate our source code. Acceleo is a pragmatic implementation of the OMG MOF MTL standard [41]. The target of our code generator is a web service implementation of JAX-WS.

We have used two Acceleo templates, one for services implementations (generateImplementations, see figure 15) and the other for services interfaces (generateInterfaces, see

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

figure 16). The result of our code generator is represented in the figures 17 and 18.

7. RELATED WORK

In our DSL and SOA explorative study [42], we have studied a lot of DSLs for SOA concerning a variety of specific domains: security, orchestration, quality of service QoS, etc. The main conclusion of our comparative study is that adaptation mechanism is almost absent in all studied DSLs for SOA.

In [6] the context is modeled via an ontology-based context model developed by using the Web Ontology Language (OWL). In order to capture vagueness (fuzzy information) in the context representation, Madkour and al. propose an extension of OWL by adding the concept "concept property" which is a group of properties. The concept property includes a specification of the degree for each property. The context represents situations. A situation is inferred using the semantic Web Rule Language (SWRL). A task ontology allows connecting a situation to specific task, and then specific tasks to specific service to be recommended. In addition, they propose a context-aware service composition framework based on Artificial Intelligent planning.

Heorhi and al. [43] and Bucchiarone and .al [28] propose a comprehensive framework for adaptable context-aware service-based business processes. They represent the context in the form of state-transition diagram of a set of entities. They defined the concept of abstract activity, which is a goal to achieve. This latter, is in the form of context configuration to be reached. At runtime a specific service composition will be generated to achieve this goal. This work treats the composition of services at run time and the adaption rules modeling is out of its scope.

Hafiddi and al [24] defined a pattern for modeling context-awareness of services, based on an UML profile, AOP and a key value context meta-model.

Also Bucchiarone and al. [44][45] defined a context-driven adaptation modeling process for the on-the-fly adaptation of the service-based applications (SBA). The context modeling is done by an XML representation of the context components. This latter is refined by adding new context dimensions if necessary. Then the relationship (named adaptation triggers) between SBA elements (service, process…) and context dimensions has been done via a mapping table. Then they have determined the adaptation strategies (service substitution, re-execution, re-composition…) and their mapping with the SBA elements. These mapping is done via a service/adaptation-strategy table. The design and realization of monitors (able to detect changes in the context dimension) and platform (responsible of the context analysis and trigger the corresponding adaptation action) have not been illustrated in this paper.

Kazhamiakin and al. [7] Defined an adaptation taxonomy with three dimensions: the "why" dimension which define adaptations kinds, the "what" dimension for subject and scope of adaptation, and the "how" dimension for adaptation strategy, decision mechanism and adaptation implementation. We conclude that all studied approaches do not focus on the specific role of various contextual properties in the adaptation process.

Yahyaoui [46] proposes context-aware service policies to achieve adaptable web services. He extends the WSPL language (Web Service Policy Language) making the policies rules context-aware. The context, as he has shown, is represented by a single simple variable. Based on this last, the policy rules change. In addition to a very simplified representation of the context and the absence of adaptation rules modeling, there is a strong coupling between context and policy rules.

Kenzi and al [32] defined an UML profile (VSoaML) for modeling adaptable service oriented systems. They treat the service adaptation with multiview service concept. Context modeling is out of the scope of their work.

In [22], the authors present an interoperable architecture for the development of context-aware services based on Model Driven Engineering and ontologies. Their context meta-model is based on the OMG's ODM (Ontology Definition Metamodel) and supported by OMG's MDA (Model Driven Architecture). For them a specific context owns its context-aware tasks which are in relation with services. Adaptation rules are absent in their work.

Achilleos [25][38] defined a model-driven petri net based framework for pervasive service creation, he deals essentially with the dynamic nature of pervasive services (service behavior), in addition, he treats only the service presentation variability.

Boukadi [20] uses Aspect Oriented Programming (AOP), with an UML profile, for modeling services. She uses an ontology for context modeling of inter-

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

enterprises cooperation domain. She did not deal with service adaptation modeling.

In [23], the authors also defined an UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services. They defined the element CAObject (Context Aware Object) which is a generalization of service and operation.

In [21], the authors specify a service adaptation approach based on UML, AOP and MDA with a context meta-model. They use context and aspect techniques to achieve service adaptability using a model driven approach. Through Model Driven Development, context models are built as independent pieces of application and at different abstraction levels then attached by suitable transformation techniques called parameterized transformation. Context model specify contextual entities that are involved in a given context aware application. From a context model, an aspect model is derived. This aspect model specifies the behaviors linked to the context model.

Soo Ho Chang et al. [29][30][31] focused on service variability modeling. Service and context modeling are out of the scope of their work.

The current version of SOAML (1.0.1) [47] does not support variability and context awareness.

8. CONCLUSION AND OUTLOOK

To facilitate the development of adaptable service oriented systems for a specific domain, we have proposed the CADSSO development approach. To produce an adaptable system, the variability must carefully be analyzed and designed in the first stages of a modeling approach. Indeed, our approach has planned a model specific to the variability specification. To take advantage of contextual information, the context of use was also modeled via a context model. In addition and for better concerns separation, adaptation rules have been modeled separately with decision tables. The domain business modeling aims to complete the generation of the dynamic code, so allowing being close to full code generation.

Our approach is divided into two stages. The first one is modeling stage. Likewise, it is divided into two steps: 1- Meta-medels definition: (abstract syntax of the DSLs) for domain specific service (as an extension of or generic service meta-model) and for domain specific business. 2- Models specification. The second stage consists of the development of the modeling tools (graphical modeling syntax) and code generator for a specific target platform.

We are currently finalizing our modeling tool and code generator by incorporating the domain specific business model, domain specific context model and adaptation rules model into our toolbox. Model validation is also planned, which will help to avoid semantic errors in the generated code.

REFRENCES:

[1] Bauer, F. L., Bolliet, L., & Helms, H. J. "Report on a Conference Sponsored by the NATO Science Committee", In *NATO Software Engineering Conference 1968,* 1968 (p. 8).

[2] Capers Jones, Software Productivity Research. Available : http://www.cs.bsu.edu/homepages/dmz/cs697/langtbl.htm, 1997.

[3] Kelly, S., & Tolvanen, J. P. (2008). *Domain-specific modeling: enabling full code generation*. John Wiley & Sons.

[4] Steven Kelly, "Domain-Specific Modeling 76 cases of MDD that works," *MetaCase*, 2009.

[5] MetaCase, "Domain-Specific Modeling with MetaEdit+: 10 Times Faster than UML", *White paper*, 2012.

[6] Madkour, M., El Ghanami, D., Maach, A., & Hasbi, A. "Context-Aware Service Adaptation: An Approach Based on Fuzzy Sets and Service Composition", *Journal of Information Science and Engineering*, *29*(1), 2013, 1-16.

[7] Kazhamiakin, R., Benbernou, S., Baresi, L., Plebani, P., Uhlig, M., & Barais, O. "Adaptation of service-based systems", In *Service research challenges and solutions for the future internet,* 2010, (pp. 117-156), Springer Berlin Heidelberg.

[8] Schilit, B., Adams, N., & Want, R. "Context-aware computing applications", In *First Workshop on Mobile Computing Systems and Applications,* 1994 December , pp. 85-90, IEEE.

[9] Chen, G., & Kotz, D. "A survey of context-aware mobile computing research", *Technical Report TR2000-381*, Dept. of Computer Science, Dartmouth College, Vol. 1, 2000, No. 2.1, pp. 2-1.

[10] Davis, J. *Open source SOA*. Manning Publications Co., 2009.

[11] Ric Merrifield, Jack Calhoun and Dennis Stevens, Harvard Business Review. Available : https://hbr.org/2008/06/the-next-revolution-in-productivity# , 2008.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[12] Kumar, B. V., Narayan, P., & Ng, T. *Implementing SOA Using Java EE,* Pearson Education, 2009.

[13] Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. "Service-oriented computing: a research roadmap", *International Journal of Cooperative Information Systems*, vol. 17, 2008, no. 02, pp. 223-255.

[14] Lethrech, M., Elmagrouni, I., Kenzi, A., Nassar, M. & Kriouile, A. "A generic metamodel for adaptable service oriented systems modeling using DSM approach", In *3rd International Symposium ISKO-Maghreb*, Marrakech, 2013, November, pp. 1-6, IEEE.

[15] Lethrech, M., Elmagrouni, I., Nassar, M., Kriouile, A., & Kenzi, A. "Domain Specific Modeling approach for context-aware service oriented systems", In *International Conference on Multimedia Computing and Systems (ICMCS),* 2014 April, (pp. 575-581). IEEE.

[16] DEY, A. K. "Understanding and using context," *Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing,* 2001, 5, 1.

[17] Strang, T., Linnhoff-Popien, C., & Frank, K. "Cool: A context ontology language to enable contextual interoperability", In *International conference on Distributed applications and interoperable systems*. 2003, Springer, pp. 236-247.

[18] Wada, H., Suzuki, J., Takada, S., & Doi, N. "Leveraging metamodeling and attribute-oriented programming to build a model-driven framework for domain specific languages", In *the 8th JSSST Conference on Systems Programming and its Applications*, 2005, March.

[19] Oberortner, E., Zdun, U., & Dustdar, S. "Tailoring a model-driven quality-of-service dsl for various stakeholders", In *ICSE Workshop on Modeling in Software Engineering*. 2009, pp. 20-25, IEEE.

[20] Khouloud Boukadi , "On demand inter-enterprises cooperation : A flexible approach based on adaptable services", ENSM Ecole Nationale Superieure des Mines, 2009, november.

[21] Valérie Monfort and Slimane Hammoudi, "Towards Adaptable SOA: Model Driven Development, Context and Aspect," *Proc ICSOCServiceWave*, 2009, pp. 175–189, Springer.

[22] Samyr Vale, Slimane Hammoudi, "An Architecture for the Development of Context-aware Services based on MDA and Ontologies," *Proc of the International MultiConference of Engineers and Computer Scientists* Vol I, 2009, Hong Kong.

[23] Sheng, Q. Z., & Benatallah, B. "Contextuml: a uml-based modeling language for model-driven development of context-aware web services", In *International Conference on Mobile Business*. 2005, pp. 206-212, IEEE.

[24] Hafiddi, H., Baidouri, H., Nassar, M., & Kriouile, A. "An aspect based pattern for context-awareness of services". *International Journal of Computer Science and Network Security*, vol. 12, no. 1, 2012, pp. 71-78.

[25] Achilleos, A., Yang, K., & Georgalas, N. "Context modelling and a context-aware framework for pervasive service creation: A model-driven approach", *Pervasive and Mobile Computing*, vol. 6, no. 2, 2010, pp. 281-296

[26] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A. & Riboni, D. "A survey of context modelling and reasoning techniques", *Pervasive and Mobile Computing*, vol. 6, no. 2, 2010, pp. 161-180

[27] OMG, MOF, Available : http://www.omg.org/mof/, 2014.

[28] Bucchiarone, A., Marconi, A., Mezzina, C. A., Pistore, M., & Raik, H. "On-the-fly adaptation of dynamic service-based systems: incrementality, reduction and reuse", In *Service-Oriented Computing,* 2013, pp. 146-161, Springer Berlin Heidelberg.

[29] Chang, S. H., & Kim, S. D. "A service-oriented analysis and design approach to developing adaptable services", In *IEEE International Conference on Services Computing*. 2007, pp. 204-211, IEEE.

[30] Chang, S. H., La, H. J., & Kim, S. D. "A comprehensive approach to service adaptation", In *IEEE International Conference on Service-Oriented Computing and Applications*. 2007, pp. 191-198, IEEE.

[31] Kim, S. D., Her, J. S., & Chang, S. H. "A theoretical foundation of variability in component-based development", *Information and Software Technology*, vol. 47, no. 10, 2005, pp. 663-673

[32] Kenzi, A., El Asri, B., Nassar, M., & Kriouile, A. "Engineering adaptable service oriented systems: A model driven approach", In *IEEE International Conference on Service-Oriented Computing and Applications*. 2009, pp. 1-8, IEEE.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[33] Pooch, U. W. "Translation of decision tables", *ACM Computing Surveys (CSUR)*, *6*(2), 1974, 125-151.

[34] OMG, BPMN. Available : http://www.omg.org/bpmn/, 2014.

[35] Eclipse EMF. Available: http://www.eclipse.org/modeling/emf/, 2014.

[36] Steinberg, D., Budinsky, F., Paternostro, M., & Merks, Ed. *EMF: Eclipse Modeling Framework*. Addison-Wesley Professional, 2008.

[37] Mohamed, M., Romdhani M., & Ghedira, K. "MOF-EMF Alignment", in *the International Conference on Autonomic and Autonomous Systems*, Athens, Greece, 2007, June, pp. 1-7.

[38] Achilleos, A. "Model-Driven Petri Net based Framework for Pervasive Service Creation", Unpublished doctoral dissertation, School of Computer Science and Electronic Engineering, University of Essex.

[39] Eclipse GMF. Available: http://www.eclipse.org/modeling/gmp/, 2014.

[40] Richard C. Gronback. "Eclipse Modeling Project: A Domain Specific Language (DSL) Toolkit", Addison-Wesley, 2009.

[41] Eclipse Acceleo. Avalable: http://www.eclipse.org/acceleo/, 2014.

[42] Lethrech, M., Elmagrouni, I., Kenzi, A., Nassar, M. & Kriouile, A. "Dsl and SOA: an explorative study", In *JDTIC, Doctoral days in information and communication technologies*, Casablanca, 2012.

[43] Raik, H., Bucchiarone, A., Khurshid, N., Marconi, A., & Pistore, M. "Astro-captevo: Dynamic context-aware adaptation for service-based systems", In *IEEE Eighth World Congress on Services (SERVICES),* 2012, June pp. 385-392, IEEE.

[44] Bucchiarone, A., Kazhamiakin, R., Cappiello, C., Di Nitto, E., & Mazza, V. "A context-driven adaptation process for service-based applications", In *Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems,* 2010, May, pp. 50-56. ACM.

[45] Bucchiarone, A., Cappiello, C., Di Nitto, E., Kazhamiakin, R., Mazza, V., & Pistore, M. "Design for adaptation of service-based applications: main issues and requirements", In *Service-Oriented Computing ICSOC/ServiceWave Workshops,* 2010, January, pp. 467-476. Springer Berlin Heidelberg.

[46] Yahyaoui, H., Wang, L., Mourad, A., Almullah, M., & Sheng, Q. Z. "Towards context-adaptable Web service policies", *Procedia Computer Science*, *5*, 2011, 610-617.

[47] OMG, SoaML. Available: http://www.omg.org/spec/SoaML/1.0.1/, 2014.

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

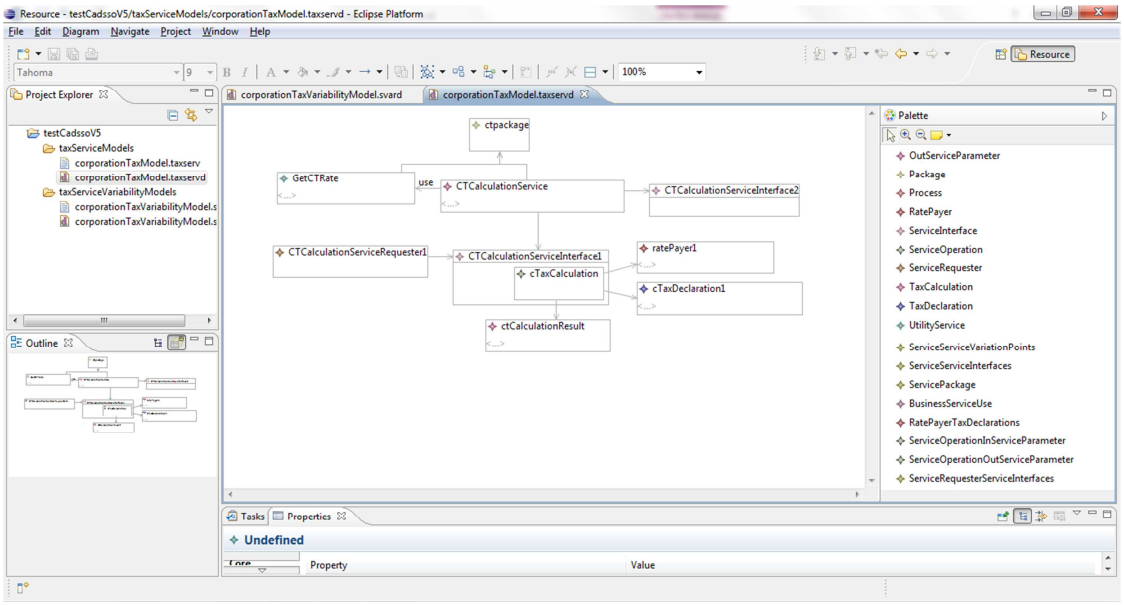


Figure 12: Corporation Tax Service Model Designed By The Tax Service Modeling Tool

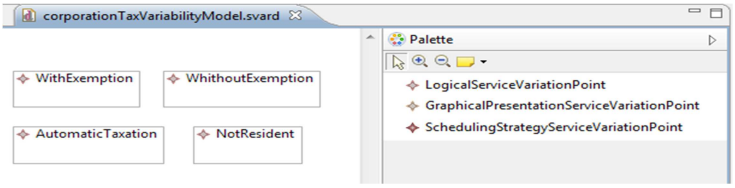
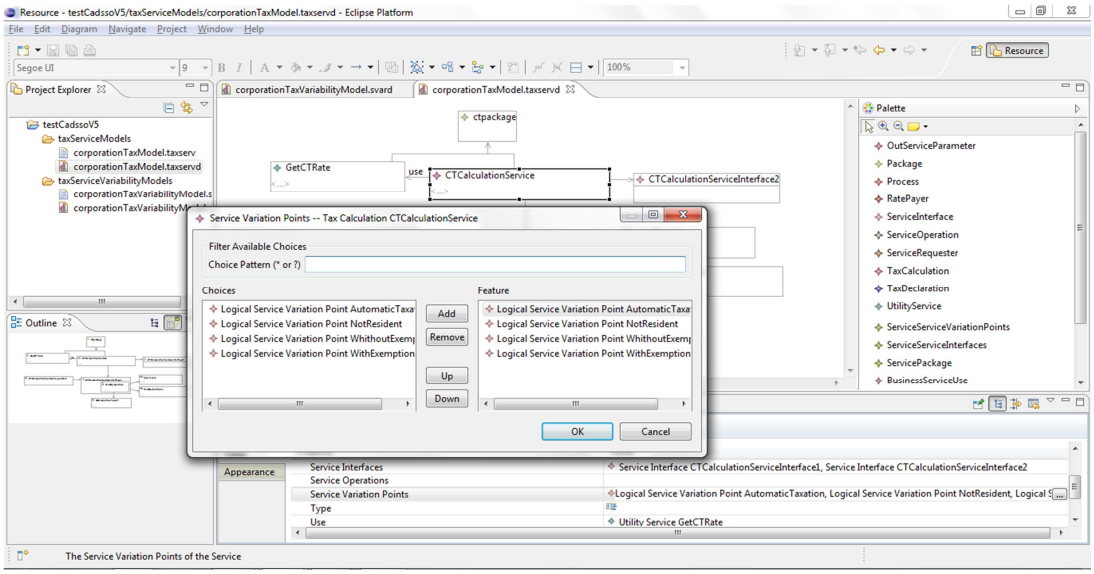


Figure 13: Corporation Tax Service Variability Model Designed By The Tax Service Variability Modeling Tool



ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 14: Ctcalculationservice Logical Service Variation Points

```
[template public generateImplementations(anAdaptableSOA : AdaptableSOA)]
[comment service implementations generation /]
[for (s: Service | anAdaptableSOA.services) ]
    [file (s.name.toUpperFirst().concat('.java'), false)]      …
public class [s.name.toUpperFirst()/] implements [for (si: ServiceInterface | s.serviceInterfaces) separator (',
')][si.name/] [/for]{
            [for (svp: ServiceVariationPoint | s.serviceVariationPoints) ]
                [for (si: ServiceInterface | s.serviceInterfaces) ]
                    [for (so: ServiceOperation | si.serviceOperations) ]
             @Override
             public [so.outServiceParameter -> at(1).type/] [so.name.concat(svp.name) /](
             [for (insp: InServiceParameter | so.inServiceParameter) separator (', ') ] [insp.type/]      [insp.name/] [/for]
                                 ){
                        [so.outServiceParameter -> at(1).type/] [so.outServiceParameter -> at(1).name/] = 0;
                                                    // business
                        return [so.outServiceParameter -> at(1).name/];
                                 }
                        [/for]
                    [/for]
            [/for]
             }    …
```

Figure 15: The "Generateimplementations" Acceleo Template

```
[template public generateInterfaces(anAdaptableSOA : AdaptableSOA)]
[comment service interfaces generation/]
[for (s: Service | anAdaptableSOA.services) ]
            [for (si: ServiceInterface | s.serviceInterfaces) ]
                    [file (si.name.toUpperFirst().concat('.java'), false)]
package [s._package.name/];    …
public interface [si.name.toUpperFirst()/] {
   [for (svp: ServiceVariationPoint | s.serviceVariationPoints) ]
      [for (si: ServiceInterface | s.serviceInterfaces) ]
        [for (so: ServiceOperation | si.serviceOperations) ]
          @WebMethod
          [so.outServiceParameter -> at(1).type/] [so.name.concat(svp.name) /](
          [for (insp: InServiceParameter | so.inServiceParameter) separator (', ') ] [insp.type/] [insp.name/] [/for]    );
        [/for]
      [/for]
    [/for]  …
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 16: The "Generateinterfaces" Acceleo Template

```
package ctpackage;
import javax.jws.WebService;
@WebService(endpointInterface = "ctpackage.CTCalculationServiceInterface1")
public class CTCalculationService implements CTCalculationServiceInterface1 , CTCalculationServiceInterface2 {
    @Override
    public double cTaxCalculationAutomaticTaxation( TaxDeclaration cTaxDeclaration1 , RatePayer ratePayer1 ){
            double ctCalculationResult = 0;
             // business
             return ctCalculationResult;
    }
    @Override
    public double cTaxCalculationNotResident(TaxDeclaration cTaxDeclaration1 , RatePayer ratePayer1 ){
             double ctCalculationResult = 0;
             // business
             return ctCalculationResult;
     }
     @Override
     public double cTaxCalculationWhithoutExemption(TaxDeclaration cTaxDeclaration1 , RatePayer ratePayer1 ){
             double ctCalculationResult = 0;
             // business
             return ctCalculationResult;
      }
     @Override
     public double cTaxCalculationWithExemption(TaxDeclaration cTaxDeclaration1 , RatePayer ratePayer1 ){
             double ctCalculationResult = 0;
            // business
            return ctCalculationResult;
      }
}
```

Figure 17: Code Generation Result Of The Acceleo Temple Generateimplementations

```
package ctpackage;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;
@WebService
@SOAPBinding(style = Style.DOCUMENT)
public interface CTCalculationServiceInterface1 {
            @WebMethod
            double cTaxCalculationAutomaticTaxation( TaxDeclaration cTaxDeclaration1 , RatePayer ratePayer1);
            @WebMethod
            double cTaxCalculationNotResident( TaxDeclaration cTaxDeclaration1 , RatePayer ratePayer1);
            @WebMethod
            double cTaxCalculationWhithoutExemption(TaxDeclaration cTaxDeclaration1 , RatePayer ratePayer1);
            @WebMethod
            double cTaxCalculationWithExemption(TaxDeclaration cTaxDeclaration1 , RatePayer ratePayer1);
}
```

Figure 18: Code Generation Result Of The Acceleo Temple Generateinterfaces