A Generic Metamodel for Adaptable Service Oriented Systems Modeling using DSM Approach

M. Lethrech, I. Elmagrouni, M. Nassar, A. Kriouile
SIME laboratory, ENSIAS
Mohammed V Souissi University
Rabat, Morocco
{mohammed.lethrech, issam.elmagrouni} @um5s.net.ma
{nassar, kriouile} @ensias.ma

A. Kenzi
ENSAF
Sidi Mohamed Ben Abdellah University
Fes, Morocco
adil.kenzi@gmail.com

***Abstract*—CAC (context aware computing) has recently emerged as a new computing paradigm promising adaptable systems development. Context awareness for services oriented systems (SOS) raises many challenges. Particularly, the challenge of engineering such systems which consists of the definition of modeling approaches, processes, techniques and tools to facilitate construction of these systems. In this paper, we propose a generic metamodel for Adaptable, Domain Specific and Service Oriented Systems. Our metamodel aims to facilitate the creation of Domain Specific Language (DSL) for adaptable and service oriented architecture. For a specific domain the language developer must produce their specific service metamodel based on our generic service metamodel.**

Keywords—Adaptability; CAC; DSM; Meta Modeling; SOC

I. INTRODUCTION

Recently, the paradigms Context Aware Computing (CAC), Service-Oriented Computing (SOC) and Domain-Specific Modeling (DSM) have gained a lot of research attention [2][3][4][5].

SOC has emerged as a new computing paradigm, promising rapid development, with lower cost, of interoperable and scalable distributed applications [1][7]. These services are loosely coupled by nature, allowing addition, modification and suppression of services in a fast and efficient manner.

CAC provides techniques for developing pervasive computing applications that are flexible, adaptable and can react to changes of the context [3].

On the other hand, adoption of DSM (Domain-Specific Modeling) approach in software engineering provides a lot of advantages, essentially a high productivity and a better quality of a generated code [1][8][9]. DSM approach is mainly based on two principles. Firstly, the elevation of abstraction level by modeling the solution with a domain specific language (DSL) − the DSL uses directly the concepts and business rules of a specific domain. Secondly, the full automatic generation of the final solution from the high level specifications. It should be noted that DSL for us is not an UML profile, we have justified this choice in our work [1].

The simultaneous use of these tree paradigms raises many challenges. Particularly, the challenge of engineering such systems [6] which consists of the definition of modeling approaches, processes, techniques and tools to facilitate construction of these systems.

In this paper, we propose a generic metamodel for Adaptable, Domain Specific and Service Oriented Systems. Service adaptability is supported by our approach through the integration of service variability notion to our metamodel. This later aims to facilitate creation of DSL (abstract syntax) for adaptable and service oriented systems. In fact, for a specific domain, the language developer must produce their specific service metamodel based on our generic service metamodel.

This paper is organized as follow; the second section presents the related work, the third is an overview of our Context-Aware, Domain Specific and Service Oriented (CADSSO) modeling approach, the fourth section exposes our generic service metamodel. In the fifth section we integrate the service variability notion to our service metamodel. This later is illustrated via a case study in the sixth section. Finally the article ends with a conclusion and outlook.

II. RELATED WORK

For Vale and Hammoudi [13], a specific context owns its context-aware tasks which are in relation with services. Service variability and adaptation rules are absent in their work.

Also, Boukadi [5] uses Aspect Oriented Programming (AOP) with a UML profile for modeling business services. She uses an ontology for modeling context for inter- enterprise cooperation domain.

Kenzi et al. [7] 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.

Ernst Oberortner, Uwe Zdun, and Schahram Dustdar [16] defined a metamodel for modeling quality of service for various stakeholders.

978-1-4799-3392-1/13/$31.00 ©2013 IEEE

In [20] the authors defined a generic metamodel for multiagent systems (MAS) development based on a lot of extant MAS metamodels.

In our DSL and SOA explorative study [1], we have highlighted a lot of DSLs for SOA, concerning a variety of specific domains; security, orchestration, quality of service…. The main conclusion of our comparative study is that adaptation mechanism is almost absent in all studied DSLs for SOA.

The new version (May 2012) of SOAML [14] doesn't support variability and context awareness.

III. Overview of our CADSSO modeling approach

Our CADSSO modeling approach is divided into five models:

- **(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. We have illustrated our approach with tax domain;
- **(b)** Service variation model: this model represents services variabilities. Each service is adaptable based on its variation points;
- **(c)** Domain specific context model: represent 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 variation model and context model;
- **(e)** Domain specific business rules model: used to model the domain specific business. The domain specific business rules model is modeled by a common modeling language like State machine, interaction diagrams, flow diagrams, petri-nets… [8]. The choice of the modeling language depends on the specific domain.

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 [8].

Separation of concerns is the cornerstone of system flexibility. Our modeling approach ensures a separation between service variabilities, context and service adaptation rules. The figure 1 illustrates our CADSSO modeling approach.

In this paper we focused on the domain specific service and service variation modeling, the other models are out of the scope of this paper.

IV. Generic Service metamodel

There are a lot of service metamodels in the literature [5][13][7][15][16].

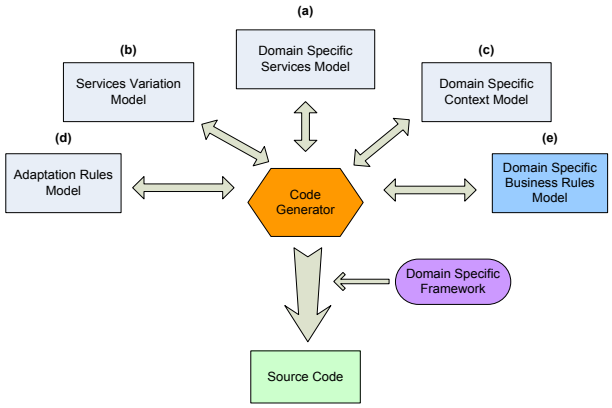

Fig. 1. Overview of CADSSO MDSD approach

Kenzi et al. [7] introduced the notion of ViewServiceInterface and BaseServiceInterface in their service metamodel (VSoaML) which are respectively equivalent in our case to ServiceInterface and BaseServiceInterface. The ServiceInterface make it possible to explicitly model the operations provided by the service using this interface. A service has just one BaseServiceInterface which groups operations provided for all service users.

The ServiceDomaine for Boukadi [5] is equivalent to our SchedulingService which is a composite service. Also LienOrchestration is equivalent to our SchedulingStrategy element.

A service has one or more ServiceOperations and a ServiceOperation has parameters; a service also has one or more service level agreements (SLA). According to service type attribute, 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 a lot of services. We have categorized the service entity into three categories: BusinessService, UtilityService and SchedulingService. The latter simply uses the already implemented business services through ShedulingStrategies. The BusinessServices implement business activities and the UtilityServices constitute non functional activities. A Process is a ShedulingService; it uses BusinessServices through SchedulingStrategies. We use the "process-as-a-service" approach in all the life-cycle of the 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 metamodel is represented in the figure 2.

Our generic metamodel is a template for creating metamodels for adaptable, domain specific and service oriented language. The language developer adds his domain elements to our generic service metamodel to have his domain specific

metamodel. Our extension mechanism is based on specialisation relation. In other words, a general concept can stand for many things. Thus, generality provides extensibility to the metamodel [20][21]. We have illustrated our approach by tax metamodel (see the case study).

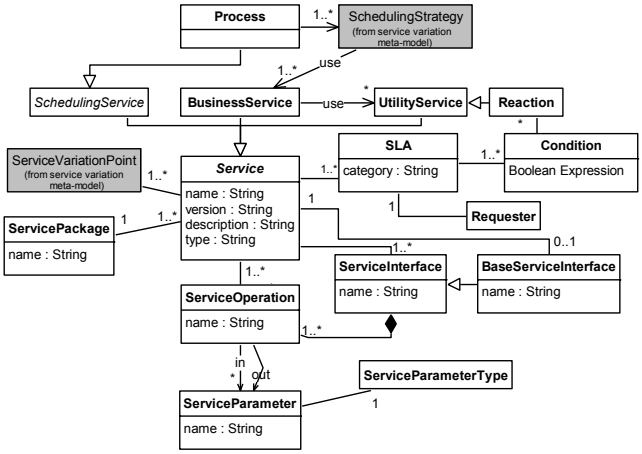

Fig. 2. Generic service metamodel

All our metamodels are in conformity with the MOF (Meta-Object Facility) model [19].

V. INTEGRATION OF SERVICE VARIABILITY TO OUR GENERIC METAMODEL

Integration of service variability into our service metamodel aims to allow modeling of adaptable services.

It is important to put the readers on the same wavelength concerning what adaptation mean for us. There are a lot of adaptation definitions in the literature, for example in [18][17], we find correctional adaptation, adaptive adaptation, evolutionary adaptation and perfective adaptation, providing details of each definition is out of the scope of this paper.

The kind of adaptation treated in our work consists of an automatic service adaptation in response to context change. In other words the system must support the change of the context of use. It is possible that certain service variabilities are not supported in the first version of the application. Thus the service must be easily extended with new variabilities without jeopardizing the service running and availability. Addition of new service variabilities must be done in a transparent manner.

Each service has a lot of variation points [1][10][11][12]. We have categorized this variation points into two categories: FunctionalVariations and TechnicalVariations. 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 invocations 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 user and Persistence, for variation of scheme or physical representation of persistent attributes. The persistence type attribute specifies the way in which the data will be persisted (relational, XML,…). SchedulingStrategy is specialised into two sub-strategies: SequentialSchedulingStrategy and ParallelSchedulingStrategy. Our service variability metamodel is illustrated in the figure 3.

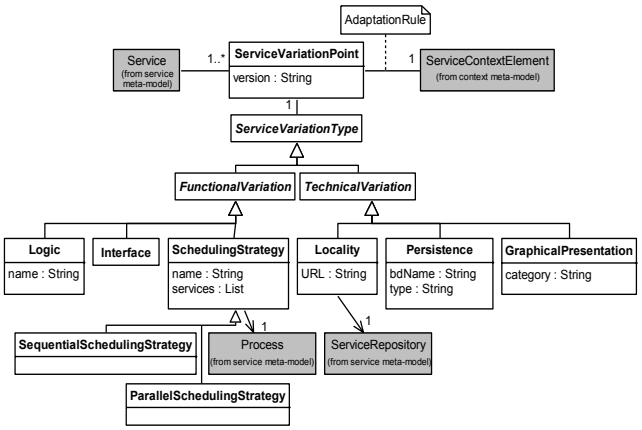

Fig. 3. Service variability metamodel

Essentially our service metamodel is distinguished by:

- The integration of the service variability notion: Each service has at least one ServiceVariationPoint which represents a variability of the adaptable service;
- All studied service metamodels are included in our generic metamodel: here, we aim to create a unifying metamodel for service oriented systems by identifying common concepts of extant service metamodels;
- It's a generic metamodel: For a specific domain the language developer must produce his specific service metamodel based on our generic service metamodel.

VI. CASE STUDY : TAX CALCULATION AND RESTITUTION

A. Tax calculation and restitution metamodel

In this paper, we have chosen the tax domain to illustrate our approach. The tax calculation and restitution algorithms are constantly changing based on the budget law. Therefore, tax information system must be as flexible as possible, so that can respond quickly to business changes.

Firstly, we have to add our tax elements to our generic service metamodel to produce our domain specific metamodel (see figure 4).

1) Domain specific meta-services

a) Tax calculation meta-services

The main BusinessService is "TaxCalculation"; it allows the calculation of the tax value. We have also the TaxDeclaration element, which represent the tax declaration. The UtilityServices are: "GetTaxRate", "GetRatePayerInfo" and "GetRatePayerExemption" they allow getting, respectively, tax rate, information of the ratepayer and ratepayer exemption.

b) 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 …); 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.

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 tax calculation. TaxRestitutionSLA represents the SLA of tax restitution.

Our Domain Specific metamodel is divided into Generic MetaModel, Domain Specific Meta Services and Domain Specific Meta Elements. The figure 4 represents our tax metamodel.

According to our generic service metamodel we can create as much models as taxes. The above example is the model of Calculation and Restitution of Corporation Tax (CT).

B. Corporation Tax calculation and restitution 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 to 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 BusinessServices: CTRestitutionDemand, CTAdvanceRestitution, CTRestitutionFolderVerification and CTRestitution. They have the same role as described in the metamodel but specific to corporation tax. CTRestitutionProcess has its specific SLA called CTRestitutionSLA. Our corporation tax calculation and restitution model is illustrated in the figure 5.

C. Corporation Tax service variations

In our approach each service has at least one service variation point. For a client request, just one service variation point is used for a given service. In this section we will identify the variation point of tax services.

1) CTCalculationService: has four logical variation points:

- CTCalculationWithExemption is corporation tax calculation which takes into account the ratepayer exemptions;

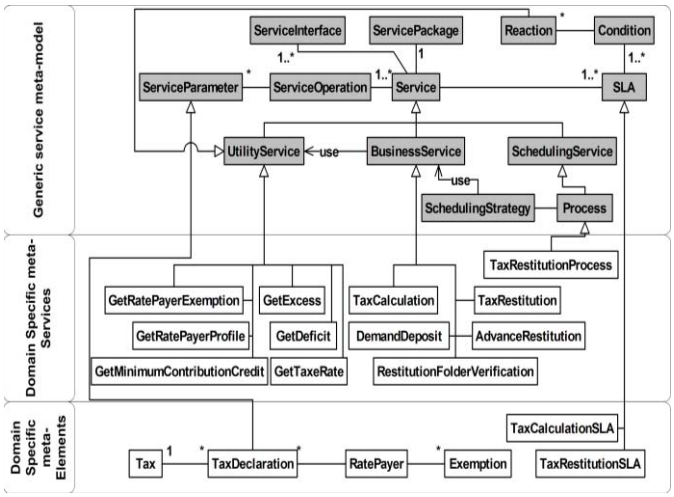

Fig. 4. Tax metamodel

- CTCalculationWithoutExemption is basic corporation tax calculation;
- CTCalculationNotResident is corporation tax calculation for not resident ratepayers;
- CTAutomaticTaxation is based on the previous year corporation tax calculation.

2) GetCTRate: has two logical variation points:

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

3) CTAdvanceRestitution: has two logical variation points:

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

4) 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 6 gathers all corporation tax services variabilities and the figure 7 represents our "CTRestitutionProcess" strategies.

VII. CONCLUSION AND OUTLOOK

We have proposed a Generic metamodel for adaptable Service Oriented systems and DSM approach. This later is the cornerstone of our Context Aware, Domain Specific and Service Oriented (CADSSO) modeling approach.

Our metamodel aims to facilitate creation of Domain Specific Language (DSL) for adaptable service oriented systems. For a specific domain the language developer must produce his specific service metamodel based on our generic service metamodel.

We are working on the production of code generator and framework for our tax calculation domain. Also we will produce the tax calculation concrete syntax with a graphical modeling tool.

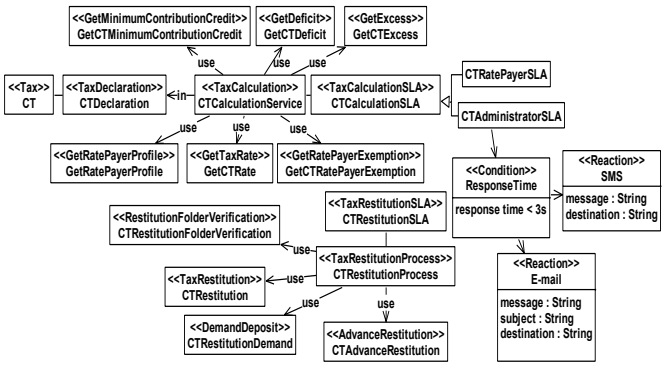

Fig. 5. Corporation Tax calculation and restitution model

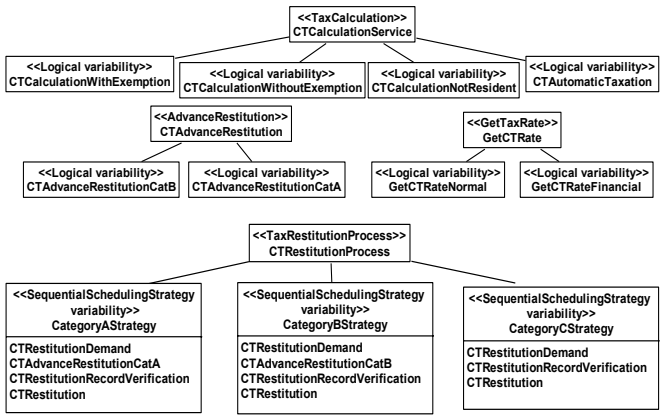

Fig. 6. Corporation tax service variability model

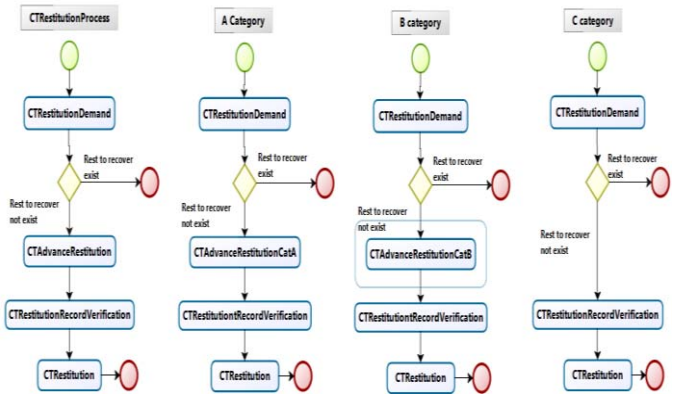

Fig. 7. SchedulingStrategies of CTRestitutionProcess

REFERENCES

[1] M. Lethrech, I. Magrouni, A. Kenzi, M. Nassar, A. Kriouile, "DSL and SOA: an Explorative Study," *in Proc. JDTIC*, 2012.

[2] Robert Hirschfeld, Pascal Costanza, Oscar Nierstrasz. "Context-oriented Programming," *Journal of Object Technology*. Vol. 7, No. 3, March-Aprile 2008.

[3] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, Daniele Riboni. "A survey of context modeling and reasoning techniques," *Pervasive and Mobile Computing Journal*, Vol 6, Issue 2, pages 161-180, 2010.

[4] Thomas Strang and Claudia Linnhoff-Popien, "A Context Modeling Survey," *in proc. The Sixth International Conference on Ubiquitous Computing*, 2004.

[5] Khouloud Boukadi, "On demand inter- enterprises cooperation : A flexible approach based on adaptable services," ENSM, France, 2009.

[6] Papazoglou M.P., Paolo Traverso, Schahram Dustdar, Frank Leymann, "Service-Oriented Computing: a Research Roadmap," *International Journal of Cooperative Information Systems*, Vol 17(2), pp 223-255, 2008.

[7] Kenzi Adil, "Adaptable and Service Oriented Systems Engineering: a Model Driven Approach," ENSIAS, Mohammed V- Souissi University Rabat, Morocco, 2010.

[8] Steven Kelly and Juha-Pekka Tolvanen, "*Domain-Specific Modeling: Enabling Full Code Generation*," IEEE Computer Society Publications, 2008.

[9] Steven Kelly and Risto Pohjonen, "Worst Practices for Domain-Specific Modeling," *IEEE Software*, Vol. 26, No. 4, July/August 2009.

[10] Soo Ho Chang and Soo Dong Kim, "A Service-Oriented Analysis and Design Approach to Developing Adaptable Services," *in Proc. IEEE International Conference on Services Computing,* pp 204-211, 2007.

[11] Soo Ho Chang, Hyun Jung La and Soo Dong Kim, "A Comprehensive Approach to Service Adaptation," *in Proc. IEEE International Conference on Service-Oriented Computing and Applications*, pp 191-198, 2007.

[12] Soo Dong Kim, Jin Sun Her and Soo Ho Chang, "A theoretical foundation of variability in component-based development," *Information and Software Technology*, vol 47, pp. 663–673, 2005.

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

[14] OMG SoaML. Available: http://www.omg.org/spec/SoaML/1.0/.

[15] Hiroshi Wada and Junichi Suzuki, "Leveraging Metamodeling and Attribute-Oriented Programming to Build a Model-driven Framework for Domain Specific Languages," *In Proc. of the 8th JSSST Conference on Systems Programming and its Applications*, 2005.

[16] Ernst Oberortner, Uwe Zdun, and Schahram Dustdar, "Tailoring a Model-Driven Quality-of-Service DSL for Various Stakeholders," *in Proc. MISE '09 ICSE Workshop on Modeling in Software Engineering,* pp 20-25, IEEE Computer Society 2009.

[17] Mohamad-Firas Alhalabi, “Applications modeling and development with an adaptable behavior,” National Institute of Applied Sciences, Lyon, France, May 2009.

[18] Ketfi, A., Belkhatir, N. and Cunin, P.-Y., “Dynamic adaptation, concepts and experiments,” *In Proc. International Conference Software & Systems Engineering and their Applications, ICSSEA* Paris, France, 2002.

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

[20] Beydoun, G. et al, “FAML: a generic metamodel for MAS development,” *IEEE Transactions on Software Engineering* 35 (6) 841-863, 2009.

[21] Henning Berg, Birger Møller-Pedersen and Stein Krogdahl, “Advancing Generic Metamodels,” *in Proc. SPLASH’11,* Pages 19-24, 2011.