Formalisation of Mediation Protocol for Web Services Composition with ACME/ARMANI ADL

Raoudha Maraoui
Faculty of Sciences of Monastir
Tunisia
maraoui.raoudha@gmail.com

Mohamed Graiet
MIRACL, ISIMS, Tunisia
Mohamed.graiet@imag.fr

Mourad Kmimech
MIRACL, ISIMS, Tunisia
mkmimec2@iutbayonne.univ-pau.fr

Mohamed Tahar Bhiri
MIRACL, ISIMS, Tunisia
tahar_bhiri@yahoo.fr

Béchir El Ayeb
Faculty of Sciences of Monastir
TUNISIA
Ayeb_b @yahoo.fr

***Abstract*—SOA (Service Oriented Architecture) defines a new Web Services cooperation paradigm in order to develop distributed applications using reusable services. The handling of such collaboration has different problems that lead to many research efforts. In this paper, we address the problem of Web service composition. Indeed, various heterogeneities can arise during the composition. The resolution of these heterogeneities, called mediation, is needed to achieve a service composition. In this paper, we propose a sound approach to formalize Web services composition mediation with the ADL (Architecture Description Language) ACME. To do so, we first model the meta-model of composite service manager and mediation. Then we specify semi formal properties associated with this meta-model using OCL (Object Constraint Language). Afterwards, we formalize the mediation protocol using Armani, which provides a powerful predicate language in order to ensure service execution reliability.**

Keywords- Web Services Composition; Mediation; Transactional Web Services; Formalization; ACME/ARMANI ADL; reliability.

I. INTRODUCTION

The recent evolution of Internet technologies expands the role of the Web from a simple data support to a middleware for B2B (Business to Business) applications. This new Internet wave is guided by the concept of Web services. However, it is necessary to combine a set of atomic service to answer for more complex requirements [1]. The problem we are interested in is how to ensure a reliable Web service composition. By reliable, we mean any compositions where all instances are correct in the sense that they meet designer's requirements, and especially in case of component failure. But, despite the organization of the composition into steps, the Web services composition has many heterogeneity problems. The resolution of these heterogeneities, called mediation, is needed to achieve a reliable service composition. In this paper, we formalize a reliable service composition based on non-functional Web Services properties. To do so, we describe the protocol mediation using the ACME of architectural concept style and Armani [17], to detect architectures software disparities.

This paper is organized as follows. In Sections 2 and 3, we present the Web services modeling related works, and then describe our formalization approach of Web services composition, respectively. In Section 4, we study the Web services meta-model and we propose a new composite service meta-model. Afterwards, we present in Section 5 the informal and semiformal specification of transactional properties. In Section 6, we propose a new architecture. In Section 7, we present our case study: a travel agency application. Finally, we conclude the paper by summarizing the main results and describing our futures woks.

II. RELATED WORKS

Many efforts have been provided to allow a usable and acceptable Web services composition. These efforts have been implemented by several composition standard and approaches and vary between those that aspire to become industry standards to those that are much more abstract. There are several formalisms for modeling Web services composition. We can cite the Petri nets, contracts, graphs, [2] , [3] UML (Unified Modeling Language), and ADLs. Each approach has advantages and disadvantages. For example, modeling using Petri nets is sound, has an intuitive graphical representation, and very visual. This approach is relevant but does not use the power of Petri nets for the composition verification. It does not model inputs and outputs of services. Another approach [4] used the concept of contracts, which

 ISBN: 978-1-61208-105-2

are graph transformations rules. They are specified by assertions expressing the the parties' obligations and rights. This approach remains inadequate if we want to make a dynamic or semi-automatic service composition. In our work, we try to formalize Web services compositions with ADL, an architecture description language which describes such formal process. It is recognized that UML does not describe software architecture within the meaning of ADL [5]. Even if you can use profiles to give the ADL characteristics [6], this approach limits his strong reusability property. Therefore, our approach is inspired by ADL. Yet most approaches ignore the specification of non-functional properties such as security, dependency, or transaction management. We try in this work to formalize Web services compositions with an architecture description language by implementing the protocol mediation and encouraging a large proportion of non-functional properties namely transaction management. In the next section, we present our method of formalization that derives from an MDE (Model Driven Engineering) approach which is based on the use of the ADL ACME / Armani.

III. PROPOSED APPROACH

In order to check the Web services composition, we use an MDE-based approach (Fig. 1).

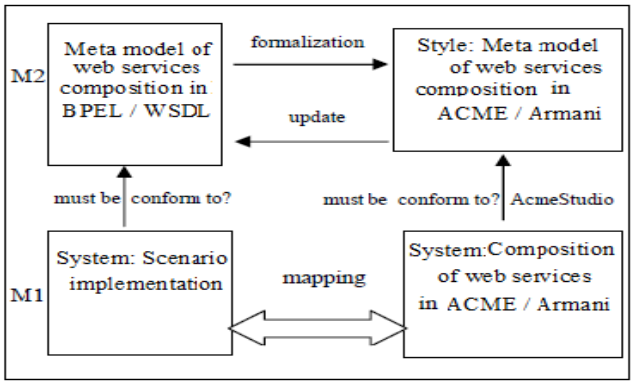

Figure 1. An overview of our services composition checking approach applied to the web service model.

Indeed, we distinguish two levels M2 and M1. The M2 level describes the Web services composition meta-model and its formalization in Acme/Armani while the M1 level describes the services model. We aim to check its conformity with its meta-model.

For that purpose, we transform this service model into Acme/Armani through the M2 level formalized in Acme/Armani. The M1 level is conform to the M2 level if it checks the coherence of rules described in the M2 level and the specific rules described in the M1 level. This is checked thanks to the AcmeStudio environment, which enables the evaluation of the Armani constraints [8].

Indeed, to achieve the formalization of web service composition in ACME and check the consistency of this composition, we describe the meta-model of web service composition (M2) using the concept of architectural style of ACME. A web services composition in M1 is described using the concept of ACME system. Level M1 is said to be conform to M2 if it satisfies the consistency rules described in M2 in addition to specific rules outlined in M1.

Our approach of components assembly checking has several advantages:

- It could be applied to several components models.
- It allows validating (see the labeled arrow updated on Fig. 1) the coherence rules described on the M2 level of the considered component model. Indeed, the completeness of these rules must be considered as well on the theoretical level as on the practical level through a test activity. Representative test models based on functional testing can be established in order to validate the coherence of the suggested rules thanks to the AcmeStudio execution environment.
- The expressiveness power of Acme/Armani is higher than the UML/OCL which is considered as an alternative to our approach.

IV. META-MODELING OF COMPOSITE SERVICE

In this section, we offer an overview of the services composition that defines a meta-model of composite service. This meta-model reifies all reliable characteristics of a service composition. It identifies their interdependencies, allows a comprehensive understanding of the mechanism composition and provides the ability to reuse our meta-model, which is independent of application domains or specific technologies. The construction of our meta-model is based on the modification of various properties of a service composition. Each of its properties is clearly identified and defined. Moreover, our meta-model is built as an extension of the meta-service model of OASIS (Organization for the Advancement of Structured Information Standards) [10] and W3C (World Wide Web Consortium). As, an atomic service, a composite service inherit all properties [11]. A composite service is a composition of one or several services: services: the services' constituents.

We allocate these services constituents to business services and management services of the composite (Fig. 2):

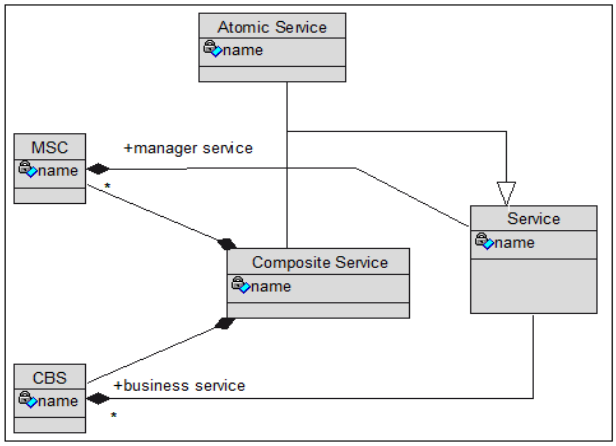

Figure 2. A meta-model of composite service.

- Business services: These services provide their functionality without global knowledge of the composition. The business services are grouped in the composite service business or CBS.
- Manager Services Composite MSC: These are specialized services in the management of the composition logic. They manage the other components and services, which have a comprehensive understanding of the composition. The service managers are grouped in the manager service composite or MSC [12].

The MSC meets all services managers who are totally transparent to users. It is the invisible part of the composite, in charge of the composition logic. Inspired by services composition existing work, we can abstract four main roles that are described in Fig. 3:

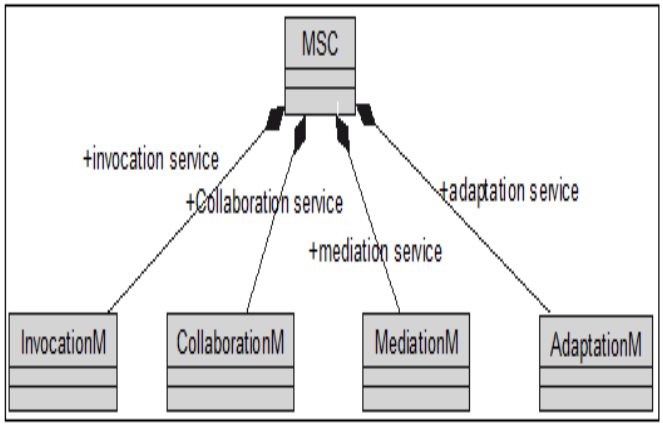

Figure 3. A meta-model of Manager Service Composite.

We focus mainly on the definition of MSC and more specifically on the mediation manager.

A. *Web services mediation*

The resolution of heterogeneities between Web services is critical to the achievement of the composition of these services. Indeed, the composition would lead most of the times to failure without a mediation between the functioning of services and data exchanged between them. In general, mediation is to resolve conflicts between stakeholders to ensure successful interactions. Furthermore, no current approach offers a comprehensive solution to the mediation protocol for Web services composition. Our work aims to answer to this lack of clarity. We are interested in a classification proposed by [7]:

- The integration level of Web services: aims to resolve all the heterogeneities between the non-functional properties.
- The adaptation level interface: aims to resolve all the heterogeneities of the service properties described in a WSDL document
- The data level mediation: aims to resolve all the heterogeneities of the service of data exchanged between the composed Web services.

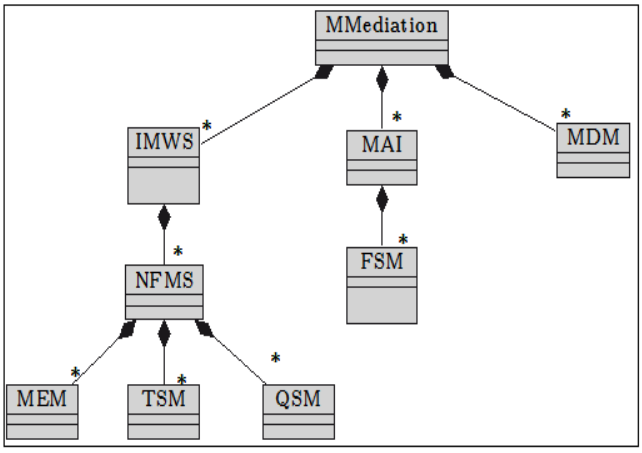

Figure 4. A meta-model of manager mediation

However, we can go further into the analysis of the meta-model and extract other properties to solve all kinds of heterogeneity. These properties included the specific non functional properties such as:

- The sequences message exchange (MEM).
- The transactional properties: They are managed by the Transaction Service Manager (TSM).
- Quality of service (QSM): This term includes nonfunctional properties, such as availability, speed, and cost

B. *The transactional patterns mediators*

Moreover, we introduce in our mediation the concept of transactional pattern, which is a point of convergence between workflow patterns and ATMs (Advanced Transactional Models) [14], one can express the logic of business processes, and the other can define the reliability of the executions. We also show their use to define and ensure service reliability compounds. For example, we use the ANDJoin pattern [15] that describes a class of interactions where a service will be activated after the termination of other services (Fig. 5).

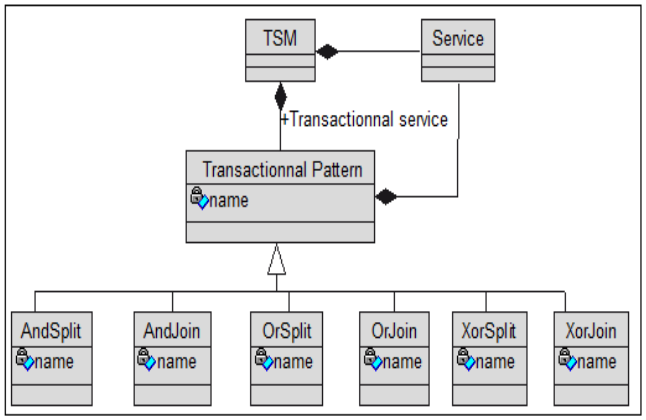

Figure 5. Transactional patterns.

 ISBN: 978-1-61208-105-2

C. *Composition of transactional Web services*

In this section, we show how we combine a set of transactional Web services to offer a new more complex value-added service. To manage the coordination of service components of a Transactional Composite Service (TCS), a composed service defines preconditions for external transitions (Fig. 6). These preconditions specify how the service responds to state of other services and how it can influence their behavior. Thus, a transactional web service can be set up as the couple of all components of its services and all preconditions set on their external transitions [13].

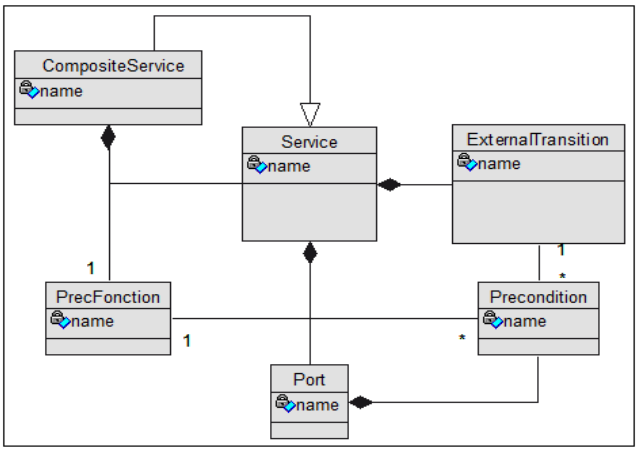

Figure 6. Definition of a transactional composite service.

Then, we show in Fig. 7 how these preconditions can express a level of abstraction above dependencies between services. These dependencies in turn define the control flow and the transactional flow of the service compound.

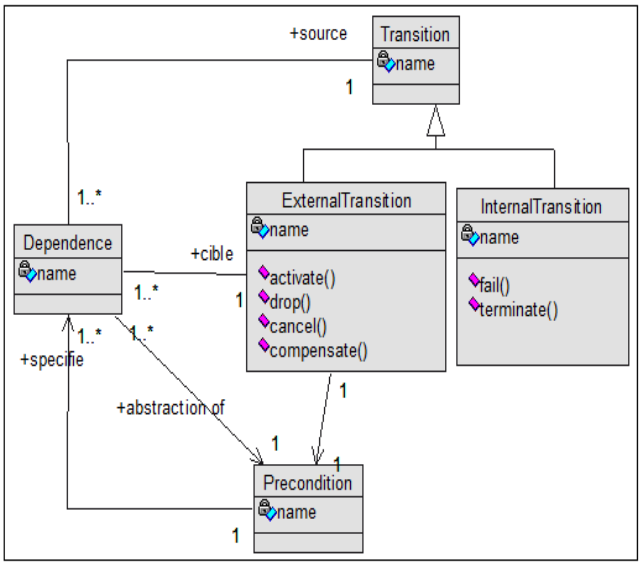

Figure 7. The preconditions to express a level of abstraction above dependencies between services.

The internal transitions that we consider in our approach are fail (), terminate () and external transitions are activate (), drop (), cancel () and compensate ().

- *Dependencies between services components of a TCS:*

The preconditions express the form of dependency relations (successions, alternative, etc) between service components, that is to say how services are coupled and how the behavior of some services can influence the others. In general, a dependency of S1 on S2 exists if the initiation of a transition (internal and external) of S1 can be triggered from external transition of S2. The management of these dependencies includes the definition of 5 types of dependencies: activation, alternative, abandonment, compensation and cancellation.

V. SPECIFYING PROPERTIES OF WEB SERVICES FROM THE PROPOSED META-MODEL

A. *Non functional transactional properties*

It is necessary to make a choice among various nonfunctional properties for each system as it is often impossible to fully satisfy all. We have chosen to highlight the transactional approach by the interest it provides. In addition if you want to move towards more rigorous, it is possible to complete this vision chart needs through the appropriate use of pre and post conditions expressed textually with OCL [16].Thus the semi-formal specification of some OCL constraints described informally as follows:

- In the component type AndSplit mediator, any port service type must have a pre-condition equal to active.

Context MedAndSplit

InvPortServiceTerminate:Self.ports ⟶ oclIsTypeOf (PortTWSService)implies Forall(p : PortWSServicejp:P rec == activate)

B. *Structural properties*

Although our framework focuses on the specification of transactional properties related to non-functional mediation for web service composition, it is clear that the formalization of these properties generates other properties related to the structure and the operation of composed Web services. Among the structural properties of our style, we can cite:

- Every component in the system must satisfy to be made a Web service client, mediator or service.

Context System

InvServiceType:Self.service ⟶ oclIsTypeOf (CompTWSClient) ORoclIsTypeOf(CompTWSService) ORoclIsTypeOf(CompTWSMediator)

- In the Component CompTWSMediator, there must at least two ports, a port of entry and an output port.

 ISBN: 978-1-61208-105-2

Context CompTWSMediator
Inv AtLeast2ports:Self.port ⟶ size () >=2

C. Fonctional properties

A specific style shows sequences of operations. Among the functional properties of our style, we can cite:

- A mediator AndSplit type specifies tha a set of services will be activated after the termination of another service.

Context AndSplit
Pre:SCN.PortTWSClient.Prec==terminate
Post:FB.PortTWSService.Prec==activateAND
HR.PortTWSService.Prec==activate

VI. A NEW ARCHITECTURE STYLE:WSM

By studying the deployed systems, there is a number of architecture which are not limited to one style only use. This is the case for our style that works in client/server roles style and symmetrical drawing some specific pipe/filter style. This WSM style (Web Service Mediation) has three components: clients, servers and mediators. They all play the role of a service with certain features. The Ombudsman is the link between the actors who are clients and servers. Clients and servers can communicate only with the mediators. There is no direct connection between the different clients of the system or between different servers. They use SOAP (Simple Object Access Protocol) as the communication protocol in order to exchange structured data regardless the programming languages or operating systems. The WSM style is an interaction model application implementing connections to perform a Web services composition. This style is not specific to a domain, it is rather generic in order to increase the level of reuse and adapt it to any field. In fact this advantage goes to the ACME ADL that allows these users to formalize their own styles.

A. The ADL ACME

The ADL ACME [17] [18], developed at Carnegie Mellon, is a common foundation for architecture description languages. It aims to enable the exchange of architectural specifications across different ADLs. ACME is based on seven types of entities to describe architecture: components, connectors, systems, ports, roles, representations, and rep-maps (map representation). Moreover, it provides a rather powerful predicates language called Armani [19] with functions appropriate to the field of software architecture. The Armani language allows describing architectural properties in the invariant or heuristics forms attached to any architectural element (component, family, system, connector, etc.). Such properties are achievable within the AcmeStudio environment [20]. In the same way, the ADL Acme supports the type concept. One can define types of architectural elements (component type, connector type, role type, port type and style type). The concept property of Acme used in the type and instance levels allows attaching nonfunctional properties to the architectural elements. Lastly, Acme provides basic types (int, float, boolean and string) and type builders (enum, record, set and sequence).

B. Formalisation of the mediation service for the Web services composition with ACME

Our work began with the improvement of an existing style. We have studied the work of [21] dealing Web services composition without mediation approach, or control over the execution of flow of services. The added mediation approach is used to increase the interactions reliability between services and ensured proper implementation through transactional patterns and connectors that represent mediators. We define in our WSM style five types of connectors that inherit from ConnTWS which is connector type of Web service and represents the five types of dependencies mentioned above. The connector ConnTWS contains rules that detect inconsistencies and show that the connector should have only two roles. Fig. 8 shows an example of an activation connector, which specifies a fundamental property to ensure the activation dependency. This property ensures that for any role r1 attached to a port P1, and for any role r2 attached to a port p2 , the two roles are different, the port p1 must be a precondition equal to ”terminate”. Therefore, to ensure this property the port p2 should be equal to a pre-condition ’activate’ and vice versa.

```
46. // Definition of Activation Connector
47. Connector Type ConnTWSAct extends ConnTWS with
    {
48. Rule CondActivation = invariant forall r1 : Role in
    self.ROLES |
49. Forall r2 : Role in self.ROLES |
50. Forall p1 : PortTWSClient in r1.ATTACHEDPORTS |
51. Forall p2 : PortTWSService in r2.ATTACHEDPORTS|
52. (r1 != r2 AND attached (r1, p1) AND attached (r2, p2))
    -> (p1.Prec == terminate AND p2.Prec == activate) OR
    (p2.Prec == terminate AND p1.Prec == activate) ;}
```

Figure 8. The ACME descriptions of the activation connector.

In addition, this style cans be used to detect the mismatches between web services. Thus, rules are defined, illustrated in Fig. 9. The first rule states that all the elements found in a system of this style must meet the requirement of being one of three component types CompTWSClient, or CompTWSService or CompTWSMediateur.

The second rule checks that if two components are connected one of them must be of mediator type and the third shows that the control flow is formalized as a composition between the AndSplit mediator on one hand and activation connector on the other. Indeed, if the component AndSplit exists it must necessarily be attached to an activation connector.

 ISBN: 978-1-61208-105-2

VII. CASE STUDY

We present in this section a scenario to show how this style can be used in ACME Studio to detect inconsistencies. The example shows a web travel organization application. The client specifies its requirements in terms of destination and choice of accommodation through the activity ”Specification of Client Needs” (SCN). This specification is then passed through mediation service AndSplit that describes how the services “Flight Booking” (FB) and ”Hotel Reservation” (HR) will be activated after SCN termination.

The organization Travel Service Online (TSO) described above, specifies a dependency of activation between SCN and HR services, denoted depAct (SCN,HR) under the activation condition, HR CondAct (HR) = SCN.terminate(). So HR will be activated after the termination of SCN. But the client component SCN has only Client type port according to the WSM specification style. In addition the mediator, AndSplit has an input service type port that can be assembled with the client port component SCN having a pre-condition “activate”.

```
143. /// Configuration of few rules
144. Rule rule33 = invariant forall comp: Component in self.
     COMPONENTS satisfiesType(comp, CompTWSClient)
     OR satisfiesType(comp, CompTWSService) OR
     satisfiesType(comp, CompTWSMediateur)
145. Rule rule34 = invariant forall c1: Component in self.
     COMPONENTS | forall c2: Component in self.
     COMPONENTS|connected(c1,c2) ⟶
     (satisfiesType(c1, CompTWSClient) AND
     satisfiesType(c2, CompTWSMediateur)) OR
     (satisfiesType(c1, CompTWSService) AND
     satisfiesType(c2, CompTWSMediateur)) OR
     (satisfiesType(c2, CompTWSClient) AND
     satisfiesType(c1, CompTWSMediateur)) OR
146. (satisfiesType(c2, CompTWSService) AND
     satisfiesType(c1, CompTWSMediateur))
147. Rule rule35 = invariant exists c: Component in self.
     COMPONENTS | declaresType(c, MedAndSplit) AND
     forall conn : ConnTWS in self.CONNECTORS |
     attached(c, conn) ⟶ (satisfiesType
     (conn,ConnTWSAct)); }
```

Figure 9. The The ACME descriptions of few rules.

It also has two ports as client having”terminate” as pre-condition. A fundamental property was described in the activation connector and specifies that any assembly with a client port service must satisfy a dependency of activation, i.e., a precondition ”activate” and pre-condition ”terminate” on both sides of the connected ports. So given these properties checked during assembly AndSplit mediation service that has a service port ”activate” pre-condition with the SCN client service, it can only have one client port pre-condition ” terminate”. As a result, we check the function of a listed mediator AndSplit, which is to complete a service that is SCN client service. On the other side the mediator has the same role to enable other service that are the HR Service and FB using the same process as the AndSplit mediation service which can be linked with an activation connector. However, the different dependencies of activation, alternative, and cancellation have been fulfilled with the ADL ACME / Armani and fostered a reliable Web service composition through mediation. We note that Acme Studio puts warning triangles in architecture during the inconsistency detection process. These triangles are superimposed on pre signaling components or connectors, which indicate that one or more constraints are not met. In this case, it means that an architecture inconsistency has been detected and is localized around the connector or component as in Fig. 10.

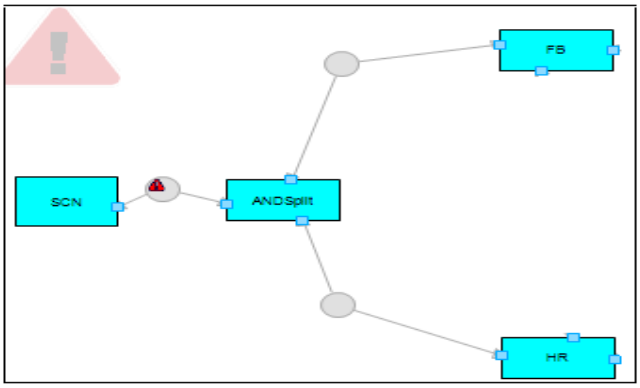

Figure 10. The initial system architecture with warning triangles showing where mismatches have been detected.

A triangle does not indicate what type of asymmetry is. This is why we should select the connector in question to find the reported failed rules. Fig. 11 shows this point of view of the activation connector between FB and services ANDJoin. The rule states that the activation connector fail to evaluate to true as shown in the figure and as consequence the activation dependence is failed, which then leads to failure of the entire system.

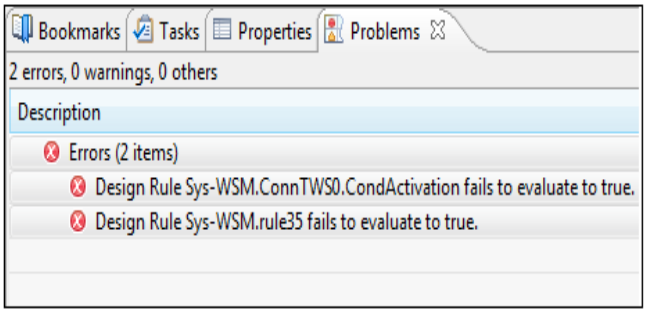

Figure 11. The initial system architecture with warning triangles showing where mismatches have been detected.

To process the ANDJoin mediator, it is necessary to satisfy the activation condition in the connector between FB and ANDJoin. To correct the detected inconsistency, we have to establish a good activation condition between these

 ISBN: 978-1-61208-105-2

components by associating a precondition to enable ConfirmReqReserv port of the ANDJoin component.

VIII. CONCLUSION AND FUTURE WORKS

This work presented in a general framework to ensure a safe design and execution of software architectures specifically the web services composition. We could formalize this composition mechanism by implementing the mediation protocol and ensuring reliability advocated by specifying non-functional properties. To do so we use Acme to check assembling consistency of Web service composition. We address this issue by describing the Web services composition Meta-model (M2 level) using Acme style architecture. The checking of the structural and non-functional properties of the composition models exploits the AcmeStudio features of verifying invariants of an Acme model. In our future works we are considering the following perspectives:

- Using existing techniques developed by the Semantic Web initiatives to promote the automation of messages and the selection of mediator models.
- Using external analysis tools associated to AcmeStudio environment in order to reason on Web services composition structures: processing global properties from local properties.
- Developing systematic translation rules of Web service composition architecture through the M2 level provided in Acme style (WSM style) which would call upon an MDE approach.

REFERENCES

[1] F. Curbera, I. Silva-Lepe, and S. Weerawarana: On the integration of heterogeneous web service partners, IBM T. J. Watson Research Center, August, 2001. [retrieved: June, 2010]. http://www.research.ibm.com/people/b/bth/OOWS2001/curbera.pdf

[2] R. Hamadi and B. Benatallah: A Petri Net-based Model for Web Service Composition, in School of Computer Science and Engineering, The University of New South Wales, In Proceedings of the 14th Australasian Database Conference (ADC'03), CRPIT 17, pp. 191–200, Australian Computer Society, Adelaide, Australia, February, 2003.

[3] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella: Automatic Composition of e-Services, Proceedings of the First International Conference on Service-Oriented Computing (ICSOC), pp. 43–58, 2003.

[4] R. Heckel: Towards contract based testing of web service, in Electronic Notes in Theoretical Computer Science 116, pp. 145–156, 2005.

[5] N. Medvidovic and N. R. Taylor: A classification and comparison framework for software architecture description languages. IEEE Transactions on Software Engineering, 26 (1): pp. 70–93, January 2000.

[6] M. Graiet: Contribution à une démarche de vérification formelle d'architectures logicielles, thèse de doctorat, Université Joseph Fourier, 25 Octobre 2007.

[7] M. Mrissa: Médiation Sémantique Orientée Contexte pour la Composition de Services Web, thèse de doctorat, Université Claude Bernard Lyon I UFR Informatique, pp. 15–36, 2007.

[8] M. Kmimech, M. Tahar Bhiri, M. Graiet, and P. Aniorté: Checking component assembly in Acme: an approach applied on UML 2.0 components model, In 4nd IEEE International Conference on Software Engineering Advances (ICSEA'2009), Portugal, IEEE Computer SocietyPress, Septembre 2009.

[9] M. Rouachid: Une approche rigoureuse pour l'ingénierie de compositions de services Web, thèse de doctorat, Université Henri Poincaré, Nancy, pp. 31–34, 2008.

[10] OASIS (2008), Service component architecture assembly model specification version 1.1. http://www.oasis-opencsa.org/. [retrieved: August 10, 2010].

[11] OpenGroup (2009). Soa source book. http://www.opengroup.org/projects/soa-book. [retrieved: June, 2010].

[12] M. Oussalah: Vers une meilleure compréhension de la composition de services par Méta Modélisation d'un service composite, 4th Francophone Conference on Software Architectures, CAL'2010, Pau-Paris, March 2010.

[13] S. Bhiri, C. Godart and O. Perrin: Patrons transactionnels pour assurer des compositions fiables de services web, Technique et Science Informatiques 28(3): pp. 301–330, 2009.

[14] S. Bhiri: Reliable Web services composition using a transactional approach, International Conference on e-Technology, e-Commerce and e-Service (EEE'05): pp. 22–30, 2005.

[15] W. M. P. van der Aalst, A. P. Barros, A. H. M. ter Hofstede, and B. Kiepuszewski: Advanced Workflow Patterns. In O. Etzion and Peter Scheuermann, editors, 5th IFCIS Int. Conf. on Cooperative Information Systems, number 190 in LNCS, pp. 18–29, Eilat, Israel, September 6–8, 2000.

[16] J. Warmer and A. Kleppe: The Object Constraint Language: Precise Modeling with UML, AddisonWesley, 1998.

[17] D. Garlan, R. T. Monroe, and D. Wile: Acme: An Architecture Description Interchange Language, Proceedings of CASCON 97, Toronto, Ontario, November, pp. 169–183, 1997.

[18] D. Garlan, R. T. Monroe, and D. Wile: Acme: Architectural Description of Composed-Based Systems, Gary Leavens and Murali Sitaraman, ed.s Kluwer, 2000.

[19] D. Garlan, R. Monroe, and D. Wile: Acme: Architectural Description of Component-based. Capturing software architecture design expertise with Armani. Technical Report CMU-CS, pp. 98–163, Carnegie Mellon University School of Computer Science, 2001.

[20] Group 2006, http://www.cs.cmu.edu/~acme/Acme Studio/ [retrieved: August 12, 2010].

[21] C. Gacek and C. Gamble: Mismatch Avoidance in Web Services Software Architectures, Journal of Universal Computer Science, vol. 14, no. 8, pp. 1285–1313, 2008.

 ISBN: 978-1-61208-105-2