Metamodel-Based Metrics for Agent-Oriented Methodologies

Noélie Bonjean* bonjean@irit.fr
Antonio Chella† antonio.chella@unipa.it
Massimo Cossentino‡ cossentino@pa.icar.cnr.it
Marie-Pierre Gleizes* gleizes@irit.fr
Frédéric Migeon * migeon@irit.fr
Valeria Seidita† valeria.seidita@unipa.it

ABSTRACT

A great number of methodologies has been already introduced in the agent-oriented software engineering field. Recently many of the authors of these methodologies also worked on their fragmentation thus obtaining portions (often called method or process fragments) that may be composed into new methodologies. The great advancement in this field, however does not correspond to equivalent results in the evaluation of the methodologies and their fragments. It is, for instance, difficult to select a fragment in the composition of a new methodology and to predict the methodology's resulting features. This work introduces a suite of metrics for evaluating and comparing entire methodologies but also their composing fragments. The proposed metrics are based on the multi-agent system metamodel. The metrics have been applied to the ADELFE and PASSI methodologies, results prove the usefulness of the proposed approach and encourage further studies on the matter.

1. INTRODUCTION

The interest for the concept of agent as the composing element of an autonomous system, capable of interacting with other agents in order to satisfy its design objectives, has grown since the 1980s. Nowadays, a great number of Agent Oriented Methodologies[1] (AOM) have been proposed [1]. These methodologies focus on different aspects and, offering different functionality with different levels of detail, address a scale of Multi-Agent Systems (MAS). The diversity of approaches offers rich resources for developers to draw on, but can also be a hindrance to progress if their commonalities and divergences are not readily understood. Moreover, an attempt from the wide range of AOMs is to benefit from different methodologies by combining their particular features. Several methodologies have been already combined before they were split up into fragments in order to adapt and/or design new methodologies [16]. It is therefore necessary to provide a way of comparison that would help to choose the suited method. In the last few years, the evaluation of MAS software engineering techniques has been one of the most active areas of research and some comparison frameworks have been proposed taking into account what concepts are manipulated, notations used, and development process or pragmatics adopted [2] [23]. Despite this, there is no objective, complete and systematic way to evaluate MAS development methods and tools. Besides, it is difficult to select a fragment in the composition of a new methodology and to predict the methodology's resulting features [8]. In this work, we propose some metrics measuring relevant features of AOM that deal with objective criteria for evaluating and comparing entire methodologies but also their composing fragments.

All methods of evaluation are influenced by the same factors, some affecting the evaluation of the structural features of the methodology itself, some others its enactment performances. The evaluation of static aspects is mainly influenced by the appreciation of: the importance given to designer's experience (a huge availability of guidelines, for instance, minimize that), the work to be done, the artefacts to be produced. These elements correspond to the typical triangle stakeholder-activity-work product considered as central by many design process composition and modelling approaches (for instance SPEM [18]). As regards methodology enactment, factors like problem complexity, designers' experience/number and the development context are crucial. All these variables are connected together. The study of all the variables at the same time is far too complex. Different approaches have proposed separate studies of some of these variables, for instance [4][22][9][15][10].

In the proposed approach, we base the evaluation on the assumption that the MAS metamodel (MMM hence after) adopted in a methodology directly influences the three crucial elements of the methodology (stakeholders, activities, work products) and it is conversely influenced by them. As a result, we think that some useful indications about the methodology features may be obtained by the application of metrics based on the MMM.

For this reason, this work starts analysing the metamodel

*Institut de Recherche en Informatique de Toulouse - IRIT Université Paul Sabatier, Toulouse, France
†Dipartimento di Ingegneria Chimica Gestionale Informatica Meccanica
Università degli Studi di Palermo, Italy
‡Istituto di Reti e Calcolo ad Alte Prestazioni, Consiglio Nazionale delle Ricerche - ICAR/CNR
Palermo, Italy

[1]In this paper we consider the term methodology and design process as synonyms

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

of the fragments obtained by the fragmentation of the methodology. Such fragments (sometimes addressed as process fragments, method fragments or chunks) constitute the root constructs of the methodology itself and they have been extracted by considering a precise granularity criterion: each group of activities (composing the fragment) should significantly contribute to the production/refinement of one of the main artefacts of the methodology (for instance, a diagram or a set of diagrams of the same type). Following this assumption, fragments obtained from different methodologies are based on a similar level of effort (although with some obvious differences among them) and produce results of similar complexity (a work product like a diagram). The approach is also based on some other assumptions we made: (1) The fragments included in a possible loop of methodology process are counted only one time. This ensures that the different methodologies (and related fragments) can be fairly compared. (2) Finally, in order to be comparable, some rules have to be followed in the description of the fragment metamodel. Such rules ensure that different methodologies (and composing fragments) refer to metamodels described with a comparable level of detail.

The proposed metrics are: **fragment input** (measuring the dependency of a fragment on the others), **fragment output** (measuring the complexity of the output work product), **fragment creative effort** (measuring the complexity of the design effort spent on the fragment), **fragment freedom degree** (measuring the degree of freedom offered to the designer while working in the fragment activities). The metrics have been applied to the ADELFE [20] and PASSI [6] methodologies. The results have been used to validate the proposed metrics.

The paper is organized as follows: details about the definition of MAS metamodels and the process fragments are introduced in section 2. In section 3, the proposed metrics are defined specifying the evaluation criteria and the formulas that allow obtaining quantitative results. Section 4 shows and discusses some results coming from the application of the metrics to two methodologies. Some works are related in section 5 before concluding with some prospects.

2. BACKGROUND

Before discussing the proposed metrics we describe all the constructs used for their creation and their application. In this section we first introduce the concept of MAS metamodel and what is its importance in a design process, how we model a design process using the IEEE FIPA styles [13] also including the concept of fragments and finally which diagrams are useful for applying the proposed metrics.

2.1 Multi-Agent System Metamodel

Metamodelling techniques lie in creating a set of concepts used for describing the properties of models. In the same way a model is an abstraction of real world's elements, phenomena, etc. , a metamodel is a further abstraction of a model. A model is always into compliance with its metamodel that rules the way in which a model has to be constructed.

Metamodelling is an essential foundation of Model Driven Engineering (MDE) proposed by OMG[2] where the language for describing metamodels is Meta Object Facility (MOF) [17].

The traditional modelling infrastructure proposed by OMG is made by four layers each one characterised by an *instance_of* relationship with the above layer. The bottom level is the level M0, it contains the user data and is called the **instance model**. The system solving a specific problem is that running on a platform, it represents the elements that exist while the system runs on the real-world platform and manages the user data. In this work, a system corresponds to a MAS. The instance model is created by instantiating what is held in the so called **user model** (level M1); level M2 contains the model of information for instantiating the M1 models and for this purpose it is called **metamodel layer**. Finally level M3 contains the **model information** for creating metamodels; hence the meta-metamodels that is usually reported as MOF.

The MAS metamodel we consider for this work is the one defined in [7]. It presents a multi-layer structure in the same way of OMG and is composed of constructs. We define a *construct* as a general term referring to one of the following entities: *elements*, *relationships*, *attributes* and *operations*. We therefore define a MAS Metamodel Construct (**MMMC**) as one of the previously mentioned entity of the metamodel.

A MAS Metamodel Element (**MMME**) is an entity of the metamodel (M2 level) that is instantiable into an entity of the system model (M1 level). Examples of MMME are: classes, use cases,... Such elements may be instantiated in the corresponding model element.

A MAS Metamodel Relationship (**MMMR**) is the construct used for representing the existence of a relationship between two (or more) instances of MMMEs. For instance, the aggregation relationship among two instances of a MMME class is an instance of the MMMR association.

A MAS Metamodel Operation (**MMMO**) is a behavioural feature of a classifier that specifies the name, type, parameters, and constraints for invoking an associated behaviour.

A MAS Metamodel Attribute (**MMMA**) is a particular kind of elements used for adding properties to MMMEs. A MMMA is a structural feature and it relates an instance of the class to a value or collection of values of the type of the attribute. The attributes type is a MMME. It is worth to note that several metamodels do not list any MMMA. This is the consequence of the fact that, according to MMMA definition, an attribute is a relationship between the class and another element of the metamodel. It is a choice of the designer to represent such relationship using attributes or other strategies (an explicit relationship with the owned element). In order to support all metamodelling style, MMMAs are included in the set of MMMCs supported by our approach.

We claim that every design process is underpinned by a system metamodel, therefore each time a designer is developing her/his MAS (s)he has at his disposal a set of elements, relationships, attributes and operations (s)he can manage for creating the system models; hence for drawing all the work products composing the system model.

In order to provide means for understanding the metrics presented in this work, in the following subsections we will provide an overview on the modelling actions made on the system metamodel constructs during design and the mutual relationships between system metamodel and work product by means of what we call *wp_content diagram*.

[2]http://www.omg.org/index.htm

2.2 Modelling Actions

While composing a work product four different kinds of action can be made on the metamodel constructs; the designer may:

- Instantiate an element in the work product. This corresponds to create a new model element. This modelling action is labelled **Define** (D).
- Instantiate a relationship in the work product. This corresponds to create a new relationship among two model elements according to what is permitted by the metamodel. This modelling action is labelled **Relates** (R).
- Refine an already defined element in the work product. This corresponds to instantiate a new attribute or operation. Refining an element corresponds to performing the following modelling actions: **Refine Attribute** (RFA) or **Refine Operation** (RFO). We note that in both cases it includes the quotation of that element.
- Use an already defined construct in the work product. New work products often relate the new elements to other elements that have been introduced in previous process activities and therefore reported in their corresponding output work products. Reporting an already defined metamodel construct corresponds to performing the following modelling actions: **Quote** (Q), **Quote Relationship** (QR), **Quote Attribute** (QA) or **Quote Operation** (QO).

2.3 Fragment and Relevant Diagrams

When we talk about "portion of work" we explicitly refer to the way of representing design processes established by IEEE FIPA [13] that founds on some underlying ideas: design process is a set of activities performed in order to reach design goals and the whole design process can be divided in phases which in turn are composed of activities and tasks. Each portion of work is devoted to deliver work products, for instance activity is devoted to produce a finer grained artefact, one single work product like a diagram possibly complemented by a text description whereas phase delivers major artefacts, for instance requirements analysis documents. Therefore design process is composed of portion of works to be performed by stakeholders (or process role) in order to deliver work products (models of the system). Each work product represents a set of elements of the whole system metamodel the design process underpins. Moreover a portion of work is usually referred to as "fragment" [12][3][5][19]. The way to describe fragment is still a work in progress and descends from the approved IEEE FIPA standard on process documentation.

What is important now is that the system metamodel assumes a very central role in designing and it is the most important design process component enabling, among others, tracking all the transformations and actions designer does while producing a work product. Besides we have to consider that design process can be seen as a sequence of fragments. In order to calculate the metrics that we propose in this paper it is important to have a look to two important diagrams used in the documentation of design process, hence of each fragment composing it. They are the *Work Product Content diagram* and the *System Metamodel diagram*.

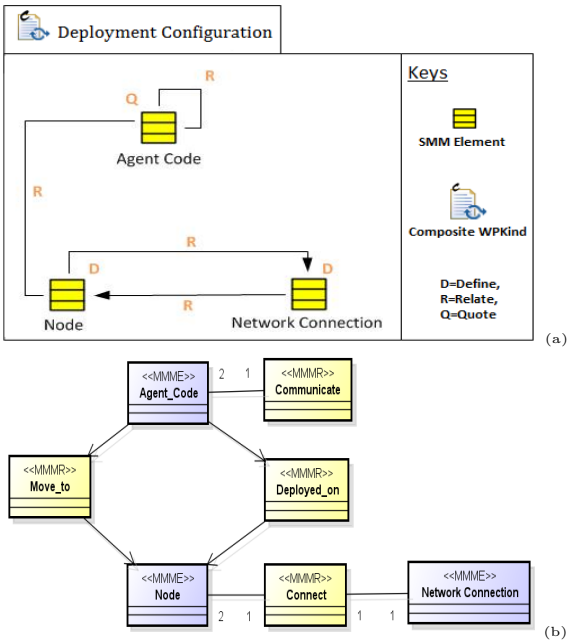

Figure 1: An Example of the Workproduct Content Diagram and of MAS metamodel diagram

The first is a specific kind of diagram (see [7] for more details) devoted to collect all the metamodel constructs that are managed during the design process enactment and are also reported in the work product. It represents the relationships between each work product produced during the design process and all the constructs of the metamodel that are here drawn. An example is given in Fig.1.(a) showing the elements and the relationships between them. They are footnoted with a specific label that represents the kinds of action made on it. The labels correspond to the modelling actions presented previously (cf. 2.2). In addition, the package with an icon on the left uppermost corner points out the work product and its kind (see [21] for the complete list and an explanation on work product kinds).

The second diagram is the MAS Metamodel diagram that shows all the system metamodel constructs that are managed by the designer in using a specific design process. This also includes all the constructs that are accepted as external inputs of the overall process whereas in the work product content diagram only the constructs reported in the work product are shown. Fig.1.(b) shows the MAS Metamodel of the Communication Ontology Description of PASSI.

3. FRAGMENT METRICS

As opposed to object-oriented methodologies, there has not been much work in comparing agent-oriented methodologies because of the intrinsic features in different MASs and their application context. There is therefore a real difficult in evaluating them. The following section presents a set of criteria based on the agent-oriented metamodel of each process fragment. Actually, four metrics are defined: fragment input, fragment output, fragment creative effort and fragment freedom degree.

3.1 Fragment Input

The Fragment Input (FI) is an architectural metric. It represents the input data required by a fragment i.e. a kind of constraints representing a guard condition. It measures the dependency of a fragment to another in a method. Actually, the input of a fragment is the number of its immediately needed constructs. High fragment input identifies fragments with a relevant dependency on other portions of the design process. Besides, a low fragment input is desirable for relating fragments because the probability to find needed constructs is higher. Thus, a fragment can be more reused, which is usually a good objective. For instance, it is common to find fragments with a low FI at the beginning of the process while this number is often higher going on inside the lyfe-cicle.

Let F be all fragments from a design process and I_f be the set of constructs required by the fragment f. We calculate:

$$\forall f \in F, FI(f) = |I_f|$$

Usually, the fragment input shows the most specific fragments of a process, i.e. the fragments that are related each other by tight dependency relationships.

3.2 Fragment Output

The Fragment Output (FO) measures the work product complexity of the output constructs. Actually, work product complexity represents the complexity in reading and understanding a work product. We think this is mainly affected by two numbers: the number of different types of MMMC that can be represented in the work product (according to what specified in the process) and the number of instances of these MMMC actually represented in a specific work product. This latter number belongs to the M1 level of representation and therefore it is out of the scope of our study. The previous number is composed by the number of the different MMMC that, in the specific work product, may be represented, whatever action the designer performs on them in the fragment.

Let F be all fragments from a design process and O_f be the set of constructs in the fragment f. The constructs can be instantiated or quoted or refined. We calculate:

$$\forall f \in F, FO(f) = |O_f|$$

The fragment output shows the most complex fragments of the process. Actually, the more components are outputted in a fragment the more complex is to understand the fragment.

3.3 Fragment Creative Effort

The Fragment Creative Effort (FCE) is a part of the complexity design effort. Actually, the complexity design effort depends on the specific problem, the skills of the designers and the used design process. The problem complexity is inconveniently informal and heterogeneous and it is treated implicitly by folding it into the solution design in software engineering approach. Because some engineers are multidisciplined, process design is divided into different phases. Process design however, is not a formula exercise and as a engineer in any discipline, once the designer has familiarized her/himself with these skills, (s)he will be able to develop her/his own system. Therefore, the skills and knowledge of an engineer influence the design effort of the produced system. Measuring this influence is a complex task that will be studied later. In this first method evaluation, we do not take into account the designer profile, we suppose a unique user. Finally, the analysing method and more specifically the meta-model provides a part of the complexity design effort. Therefore, the fragment creative effort measures the effort done by the design in performing the definition of the portion of system related to a specific fragment. This effort is related uniquely to the introduction of new elements, relationships, attributes and operations in the system model.

The fragment creative effort is measured for each fragment. It is the number of defined or related or refined constructs in a fragment. Let F be all fragments from a method and D_f be the set of defined or related or refined constructs in the fragment f, then:

$$\forall f \in F, FCE(f) = |D_f|$$

The fragment creative effort shows the most complex fragments of the method in term of design. This metric is strongly related to the fragment output but while the FCE addresses the creative part of the work, the FO measures the complexity of the whole resulting work product. In other words, no contribution to FCE comes, for instance, from constructs reported unchanged from previous portions of work, but these constructs are counted in the FO. As a consequence, this formula expresses the relationship between the two:

$$\forall f \in F, FO(f) \geq FCE(f)$$

3.4 Fragment Freedom Degree

The Fragment Freedom Degree (FFD) represents the degree of freedom granted to designer for a fragment. It is calculated by the ratio of the fragment creative effort over the fragment input. This metric enables to define for a fragment if the introduction of the new constructs is strongly conditioned or not. In a method, high fragment freedom degree is ideally required in the first fragments of the method. In fact, these fragments usually imply designers' own mind and creativity. Then, the fragment freedom degree of the following fragments progressively decreases until to obtain a low fragment freedom degree for the last fragments. Actually, the designer is generally strongly conditioned at the end of the process because during the life-cycle, (s)he is guided in order to converge towards a well defined system.

Let F be all fragments from a design process, then:

$$\forall f \in F, FFD(f) = FCE(f)/FI(f)$$

The fragment creative effort shows the process structure. The ideal process structure might present a high value of FFD at the beginning of the process and a progressive decade of it to a low ration.

4. EXPERIMENTAL RESULTS

Currently, these metrics are evaluate on several Agent-Oriented Methodologies: ADELFE, PASSI, INGENIAS [14] and TROPOS [11]. In this section we present the results obtained for every fragment and gathered by ADELFE and PASSI. We also study the results comparatively for the two methods. This experiment enables us to give an analysis on the intrinsic relevancy of the metrics in subsection 4.1. We

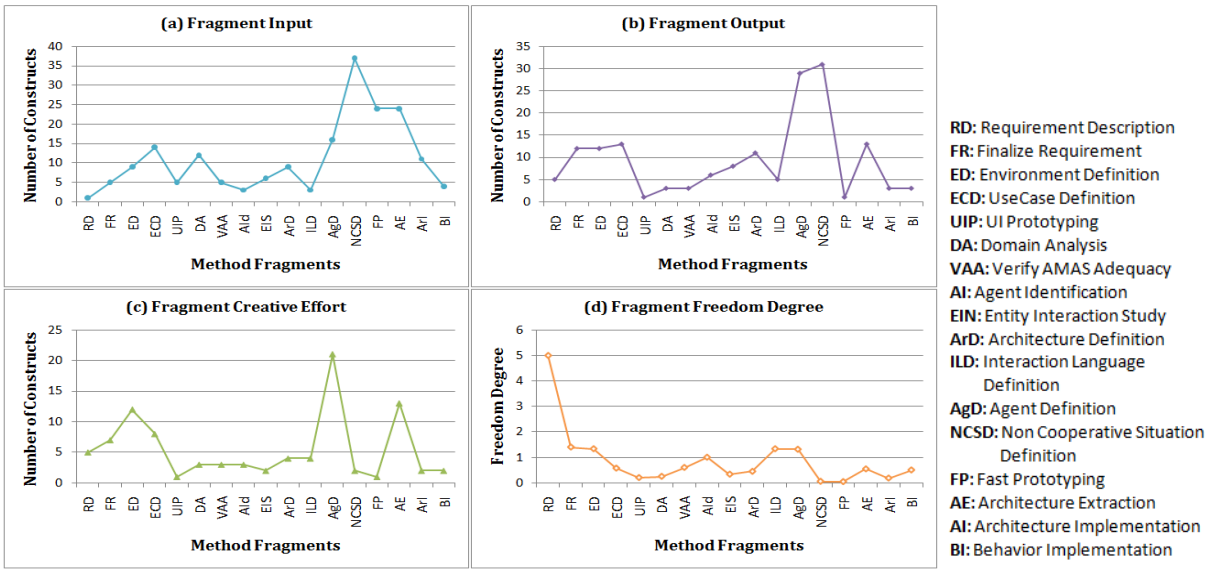

Figure 2: Evaluation of ADELFE

see in subsection 4.2 how process evaluation and fragment evaluation become both possible but are quite different. Finally, we discuss the metrics and the factors that could impact their results.

4.1 Metric Relevancy

It is firstly important to measure the indicators' intrinsic quality and see what kind of correlated analysis they enable. As mentioned in section 3.1, FI gives an idea of the dependencies of each fragment one with another. Therefore, it is natural to find in Fig.2.(a) that higher values are at the ended fragments. It shows their specificity in the ADELFE process and that they will probably induce a difficult reuse. In the same way, for PASSI, we can see in Fig.3.(a) that the first and the fifth fragments (*Domain Requirement Description* and *Domain Ontological Description*) present the lowest values. This is explained by the fact that both analyse the problem domain in terms of the requirements it provides (the first) and of its ontological representation (the second); for this reason, the designer faces a kind of work which is not constrained by other metamodel constructs but it is the result of her/his own observation and reasoning.

Concerning the FO metric, the high values correspond to fragments deliverying heavy weight models of the system. Usually they correspond to the most significant fragments in the process. The usefulness of the fragment low value within the process might be reappraisal. In the ADELFE results in Fig.2.(b), the greatest values are obtained for *Agent Definition* and *Non Cooperative Situation Definition* which indeed are the most significant activities in ADELFE. It is also interesting to note that all fragments settled in the design phase and in the implementation phase (from *Architecture Definition* to *Behaviour Implementation*) have a rather important value except for *Interaction Language Definition* fragment and *Fast Prototyping* fragment. They are two singularities comparatively with the surrounding fragments. It clearly shows the lack in the metamodel of elements to take these activities into account with as much quality as other design or implementation fragments. The PASSI curve reveals an interesting confirmation of these considerations, for instance in the presence of the maximum value for the *Single Agent Structure Definition* fragment which is quite intuitive for an AOSE design process.

The FCE metric also reveals itself to be a very relevant indicator. For ADELFE, three peaks are distinctly drawn above the average value (which is of five constructs) on the FCE plot (cf. Fig.2.(c)). These three peaks correspond to the definition of the most important constructs in ADELFE where the highest efforts are provided to define the environment (*Environment Definition* fragment), the cooperative agent behaviour (*Agent Definition* fragment) and the architecture (*Architecture Extraction* fragment). We will address this discussion in a following subsection. In PASSI, we can note low values of creative effort (cf. Fig.3.(c)) in the *Single Agent Behaviour Description*, in the *Code Reuse* and in the *Code Production*. We were expecting to find these values because these fragments are in the implementation phase of the design process so, because of the nature of PASSI that commits a great effort and a lot of work in the design phase, all the information needed for defining the implementation constructs can be inferred from the previous phase without spending too much effort. Besides another result we imagined is that regarding the *Code Production* and *Code Reuse* creative effort, although these two fragments presents the same value of FO and both the two deals with code concerns, *Code Reuse* has a higher creative effort. During *Code Reuse*, the designer analyses and then reuses patterns of agent and this activity is obviously more demanding than simply writing code on the base of previously drawn structural and behavioural diagrams.

With our experimentation on ADELFE and PASSI, we found these metrics very relevant to show the complexity and the specificity of evaluated methodologies.

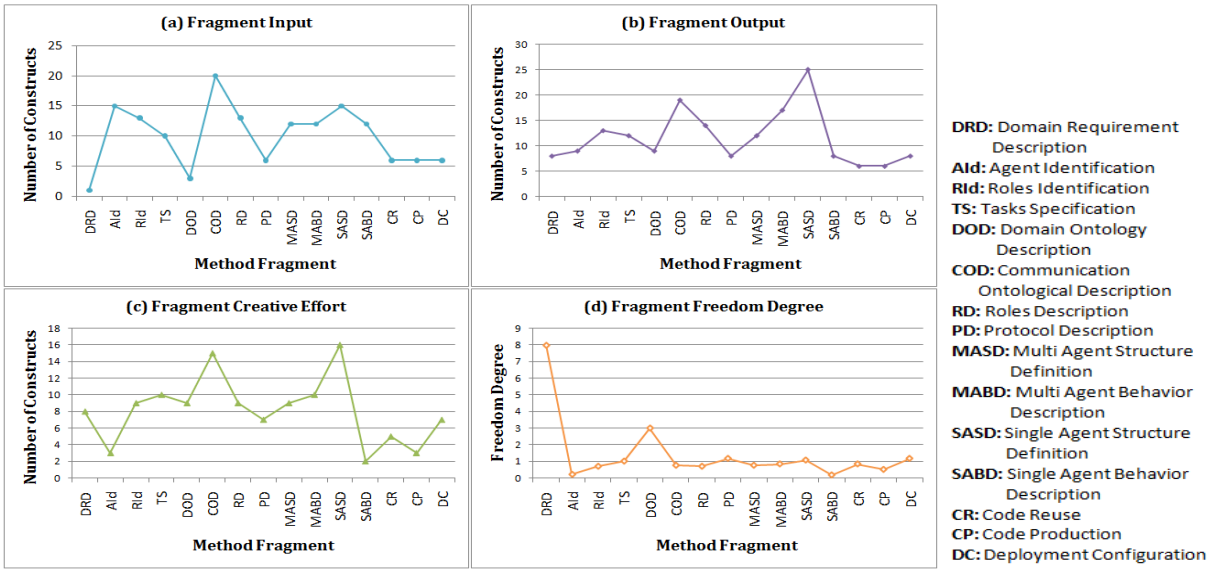

Figure 3: Evaluation of PASSI

4.2 Process Evaluation vs. Fragment Evaluation

As we saw in the previous subsection, the metrics that are proposed in this paper are of a good accuracy to evaluate fragment intrinsic characteristics but the correlated metric values are also of high interest for every single fragment.

A high value of FI is often accompanied by a high value of FCE and of FO, because the more constructs one has to design, and the more constructs are needed, the more is the effort spent in doing the design activity. This general rule is however not always respected, for instance for the *Agent Identification* fragment of PASSI. Although the FI and FO values are rather high, the FCE is rather low, particularly in the PASSI context, because of the fact that this portion of work only requires to group use cases in order to identify agents. All the information is already present in the previous fragment with which, in fact, there is a tight dependency also shown by the value of FI.

As the figures show, the metrics give also information on the global method characteristics. Some singularity points can be observed or comparative values of metrics can only be explained by a global view of the process. Let us illustrate this with some examples. The first one is about the singularities that were already mentioned on some ADELFE fragments that have very low values compared to their neighbourhood. The second one is for FI and FFD values of the *Domain Ontological Description* fragment in PASSI (cf. Fig.3.(a)(d)). As it can be seen, they are offbeat with the rest of the curve; this is because this fragment is not well positioned in the PASSI design process. This can be seen above all from the FFD where we expected to have a decreasing trend for the reasons said before. This result confirms what we thought about the position of this fragment and validate the accuracy of the proposed metrics. A last example is observed on the FFD curve for ADELFE (cf. Fig.2.(d)). As expected, the FFD flatten out at a low level during the requirement analysis phase (ended by the *UI Prototyping* fragment). However the FFD has a slight rise during the *Agent Identification* fragment and the *Agent Definition* fragment. Actually, the *Agent Identification* fragment aims at finding what agents will be considered in the system and the *Agent Definition* fragment aims to define the behaviour: skills, aptitudes, an interaction language, a world representation, etc. for every agent previously identified. In these fragments, designers have a high responsibility in the choice they make. As the figure shows, they have high freedom degree with little constructs already defined to guide them or to constraint them. All these examples find explanation in the relative values of the fragments of a same process rather than in the individual value of each one.

4.3 Discussion

A first difficulty occurring while comparing several methodologies is the lack of normalization. ADELFE is defined with 17 fragments while PASSI only contains 15 ones. And of course, they do not address the same phases with the same granularity. For this reason, we were obliged to normalize the curves in order to present ADELFE and PASSI with comparable values (cf. Fig.4). This was done by aligning fragments according to analysis, design and implementation phases. This alignment may also be needed on the Y axis that is the number of constructs. Obviously, the more constructs the metamodel contains, the higher the values will be in the metrics (except for FFD which is a ratio, naturally). This factor, which can be called the granularity of the metamodel, may happen when methods are using Model-Driven Development. In that case, metamodels are very big with plenty of details needed to tackle with accuracy required by model transformation algorithms. For this reason, it is important to normalize also the granularity with which a process is described and therefore the level of abstraction. None of the metrics takes the presence/adoption of Computer-Aided Software Engineering (CASE) tools into account. However, tools for guiding the engineer during the design are an advantage that should be evaluated by metrics. We will discuss about that in the following section.

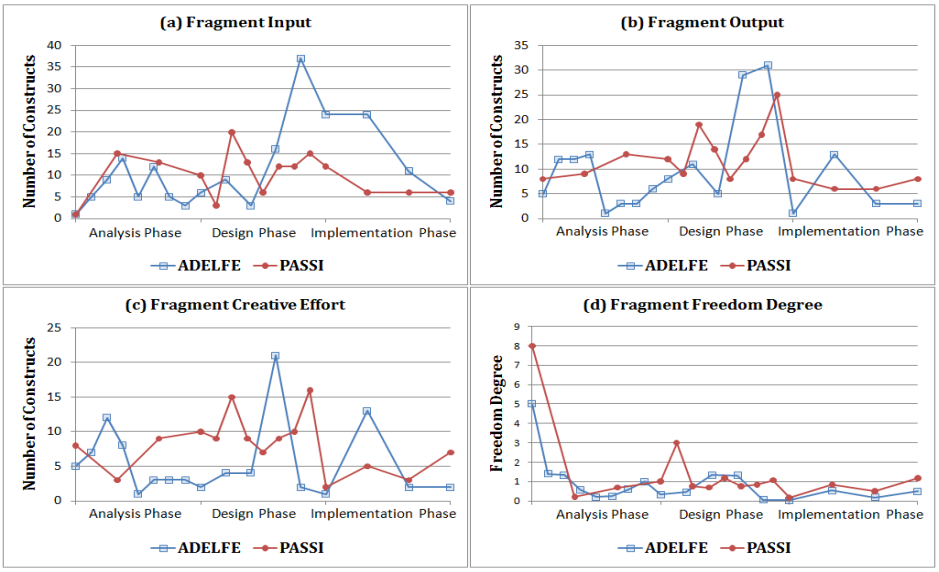

Figure 4: Comparison of ADELFE and PASSI

5. RELATED WORKS

Recently several AOM have been proposed. Thus far, however, software designers have not embraced any single methodology. In order to develop better solutions, designers need to understand advantages and limitations of existing methodologies. Therefore some works focus their efforts on the analysis of methodologies.

Cernuzzi's approach uses goal-question-metric to determine what factors are important to measure for comparing methods [4]. A qualitative analysis followed by a quantitative rating is proposed. Sturm and Shehory [22] develop a catalog of criteria for feature-based analysis of AOSE methodologies. *Agent-Based Characteristics* and *Software-Engineering Criteria* are differenciated. Their set of criteria is suitable to show the drawbacks of methodologies and therefore gives suggestions for further development. The above described approach compares the methodologies by a screening of the criteria. Dam and Winikoff [9] divided their found criteria into four dimensions to examine: (1) concepts and properties, (2) notations and modeling, (3) process, and (4) pragmatics. During the methodology evaluation, the correctness of a notation and the referenced characteristics is difficult to judge. The survey approach proposed here successfully reflects it. Moreover, based on both software engineering process principles and agent characteristics, Lin et al. [15] approach determines whether criteria have been met by the method and provides answers as statements for comparison from questions at the detailed level concerning logical relationships among these criteria.

The main lack of these approaches is that they only evaluate methodologies and do not take into account the portions which compose them. Currently, there is no tool that implements and simplifies the evaluation of the entire method process and their fragments.

6. CONCLUSIONS AND FUTURE WORKS

Despite all papers related to this topic there is no general and commonly adopted evaluation process of method. There is a fundamental need to have evaluation process in order to get a measurement of comparing completed activities of a method. In this paper, four objective dynamic metrics have been defined. These metrics enable analysing and comparing methods process and their fragment. Based on MAS metamodel, for each method fragment, particular numbers of constructs from the modelling actions performed on them are measured. Each measure enables to show some specificities of the fragments or some particularities of the method process. In order to illustrated the metrics, quantitative results of the agent-oriented method evaluation have been presented. This example usage illustrated how to derive method process features from the method fragment metamodels.

Future improvements to the presented metrics may result from an examination whether it is useful to consider different kinds of actions in workflow activity: (i) GUI action which is an activity performed by the designer using a GUI; (ii) automated action which is an activity performed by the tool to create a new constructs e.g. a model transformation; (iii) user action which is an activity not supported by a tool such as using a blackboard.

Currently, these metrics are basically used in a work of designed processes from self-combining method fragments. They enable the designed process evaluation at two levels: the evaluation of similar fragments and the evaluation of different processes.

7. REFERENCES

[1] F. Bergenti, M.P. Gleizes, and F. Zambonelli. *Methodologies And Software Engineering For Agent Systems: The Agent-oriented Software Engineering Handbook*. Kluwer Academic Pub, 2004.

[2] Carole Bernon, Marie-Pierre Gleizes, Gauthier Picard, and Pierre Glize. The Adelfe Methodology for an Intranet System Design. In *International Bi-Conferenystems (AOIS-2002) at CAice Workshop on Agent-Oriented Information SSE'02 (AOIS - SSE),*

Toronto, Ontario, Canada, 27-28 may 2002, page (on line), http://ceur-ws.org, May 2002. CEUR Workshop Proceedings.

[3] S. Brinkkemper, R.J. Welke, and K. Lyytinen. *Method Engineering: Principles of Method Construction and Tool Support*. Springer, 1996.

[4] Luca Cernuzzi, Gustavo Rossi, and La Plata. On the evaluation of agent oriented modeling methods. In *In Proceedings of Agent Oriented Methodology Workshop*, pages 21–30, 2002.

[5] M. Cossentino, S. Gaglio, A. Garro, and V. Seidita. Method fragments for agent design methodologies: from standardisation to research. *International Journal of Agent-Oriented Software Engineering (IJAOSE)*, 1(1):91–121, 2007.

[6] M. Cossentino and V. Seidita. Passi2 - going towards maturity of the passi process. *Technical Report ICAR-CNR*, (09-02), 2009.

[7] M. Cossentino and V. Seidita. Metamodeling: Representing and modeling system knowledge in design processes. Technical Report 11-02, Technical Report ICAR-CNR, 29 July 2011.

[8] Massimo Cossentino, Marie-Pierre Gleizes, Ambra Molesini, and Andrea Omicini. Process Engineering and AOSE (regular paper). In Marie-Pierre Gleizes and Jorge Gomez-Sanz, editors, *Workshop on Agent Oriented Software Engineering (AOSE), TORONTO - Canada, 10/05/2010-11/05/2010*, number 6038 in LNCS, pages 180–190, http://www.springerlink.com, 2011. Springer.

[9] Khanh Dam and Michael Winikoff. Comparing agent-oriented methodologies. In Paolo Giorgini, Brian Henderson-Sellers, and Michael Winikoff, editors, *Agent-Oriented Information Systems*, volume 3030 of *Lecture Notes in Computer Science*, pages 78–93. Springer Berlin / Heidelberg, 2004.

[10] Iván García-Magariño, Massimo Cossentino, and Valeria Seidita. A metrics suite for evaluating agent-oriented architectures. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 912–919, New York, NY, USA, 2010. ACM.

[11] Paolo Giorgini, Manuel Kolp, John Mylopoulos, and Jaelson Castro. Tropos: A requirements-driven methodology for agent-oriented software. chapter II, pages 20–45.

[12] AF Harmsen, S. Brinkkemper, and H. Oei. Situational method engineering for information system projects. In *Methods and Associated Tools for the Information Systems Life Cycle, Proceedings of the IFIP WG8. 1 Working Conference CRISi94*, pages 169–194, 1994.

[13] IEEE Foundation for Intelligent Physical Agents. *Design Process Documentation Template, Document number XC00097A-Experimental*, 2011.

[14] INGENIAS. Home page. `http://grasia.fdi.ucm.es/ingenias/metamodel/`.

[15] Chia-En Lin, Krishna M. Kavi, Frederick T. Sheldon, Kris M. Daley, and Robert K. Abercrombie. A methodology to evaluate agent oriented software engineering techniques. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, HICSS 07, page 60, Washington, DC, USA, 2007. IEEE Computer Society.

[16] Mirko Morandini, Frédéric Migeon, Marie-Pierre Gleizes, Christine Maurel, Loris Penserini, and Anna Perini. A goal-oriented approach for modelling self-organising mas. In Huib Aldewereld, Virginia Dignum, and Gauthier Picard, editors, *Engineering Societies in the Agents World X*, volume 5881 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin / Heidelberg, 2009.

[17] Object Management Group. *Meta Object Facility (MOF) Specification. http://doc.omg.org/formal/02-04-03*, 2003.

[18] OMG. Object Management Group. Software & Software Process Engineering Metamodel. version 2.0. Document number: formal/2008-04-01. 2008, 2008.

[19] J. Ralyté. Towards situational methods for information systems development: engineering reusable method chunks. *Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education*, pages 271–282, 2004.

[20] Sylvain Rougemaille, Jean-Paul Arcangeli, Marie-Pierre Gleizes, and Frédéric Migeon. ADELFE Design, AMAS-ML in Action. In *International Workshop on Engineering Societies in the Agents World (ESAW), Saint-Etienne, 24/09/2008-26/09/2008*, page (electronic medium), http://www.springerlink.com/, 2008. Springer-Verlag.

[21] V. Seidita, M. Cossentino, and S. Gaglio. A repository of fragments for agent systems design. *Proc. Of the Workshop on Objects and Agents (WOA06)*, 2006.

[22] Arnon Sturm and Onn Shehory. A framework for evaluating agent-oriented methodologies. In *Proc. of the Int. Bi-Conference Workshop on Agent-Oriented Information Systems, AOIS 2003, volume 3030 of LNCS*, pages 94–109, 2003.

[23] Quynh-Nhu N. Tran and Graham C. Low. *Comparison of Ten Agent-Oriented Methodologies*, chapter XII, pages 341–367. 2005.