REVISTA COLOMBIANA DE COMPUTACIÓN
Volumen 11, número 2
Págs. 94 - 115

Measurement of Tailored agent-oriented design processes by resorting to flow graphs: A preliminary investigation

Juan C. Garcia-Ojeda[*†]

Fecha de Recibido: 10/08/2010 Fecha de Aprobación: 15/10/2010

Abstract

This work describes the OMaSE-based Flow Graph (OFG). OFG is a tool for assessing the advantages and disadvantages of tailored agent-oriented design process in assembling time.

[*] Autonomous University of Bucaramanga, Department of Systems Engineering,Colombia. jgarciao@unab.edu.co

1 Introduction

Assessing the advantages and disadvantages of non-tailored agent-oriented design processes has proven to be quite difficult to estimate: These design processes are heterogeneous in their composition, structure, and functionality (Cernuzzi, & Rossi, 2002; DeLoach, 2009a; Sturm & Shehory, 2003). In this connection, the estimation of such advantages and disadvantages for tailored agent-oriented design processes could be similarly challenging. In the case of non-tailored agent-oriented design processes, they fail to exhibit conceptual agreement (Cernuzzi and Rossi, 2002). Moreover, the evaluation of these design processes introduces various difficulties (Sturm & Shehory, 2003): (i) They might address different aspects, e.g., beliefs, goals, actions, and ongoing interactions (Jennings, & Wooldridge, 2001), or differ in their terminology; (ii) some of these design processes are influenced by specific approaches such as Object-oriented, Knowledge Engineering, and Requirements Engineering; and, (iii) the design processes' completeness. In addition, the significant benefits of adopting agent-oriented approaches to solve complex problems have not been yet demonstrated; this is attributable to: the lack of industrial strength methods and techniques for developing agent-based applications, the lack of conceptual agreement, and the lack of a common notation and models (DeLoach, 2009a).

Although a number of factors, such as design processes, concepts, and techniques, have been suggested as important elements in the evaluation of non-tailored agent-oriented design processes, the impact of developing design processes appears as a common thread (Sturm, & Shehory, 2004; Yu, & Cisneiros, 2002; Cernuzzi, and Rossi, 2002; Dam, & Winikoff, 2003; Shehory, & Sturm, 2001). This fact gains importance because the agent-oriented community is currently looking for the adoption of method engineering practices (Brinkkemper, 1996) for tailoring agent-oriented design processes (Chella *et al.*, 2004; Henderson-Sellers, 2005; Garcia-Ojeda *et al.*, 2007; Nardini *et al.,* 2007). Unfortunately, the contributions regarding the assessment of the advantages and disadvantages of tailored agent-oriented design processes are scarce.

In this work, we present the OMaSE-based Flow Graph (OFG). OFG is a tool for assessing the advantages and disadvantages of tailored agent-oriented design processes in assembling time. Although the analysis of design processes by resorting to flow graphs is not new (Cardoso, 2005; Cardoso *et al.*, 2006; Polyvyanyy *et al.*, 2008a; Polyvyanyy *et al.*, 2008b), this work specifically advances the state of the art of the agent-oriented software engineering in one crucial aspect: it articulates the role

of flow graphs as a modeling, analysis, visualization, and abstraction tool for measuring tailored agent-oriented design processes. As a modeling tool, OFG allows both a causal and a behavioral design process analysis. As an analysis tool, OFG can be used to verify if the design process proceeds correctly from the beginning to the end, so process engineers can validate the process in fact transform its inputs into the derived output. Also, OFG can be used to derive design process metrics (core of this work). As a visualization tool, OFG improves understanding over the process under construction. OFG enhances the ability of process engineers to picture the relationships between tasks in a process. This fact not only would save time but also would improve learning, speeding up development and saving considerable effort and expenses. Also, OFG can be easily used to find tasks that can be executed in parallel, so process engineers can detect critical path in the process (this feature is not addressed in this work). Finally, as an abstraction tool, OFG provides an abstract representation of the design process, containing only details which provide information about the different components/modules that can be documented at the end of the design process.

The remainder of this work is organized as follows: Section 2 briefly introduces the OMaSE Process Framework. Section 3presents a motivating example used to illustrate the application of OFG. In Sections 4 and 0, this work details the formalism behind the OMaSE-based Flow Graph and the Meta OMaSE-based Flow Graph. In Section 6, a set of metrics is defined. In Section 0, the set of metrics is empirically validated by means of a set of experiments. Finally, Section 0 concludes the work.

2. Background

2.1 The OMaSE Process Framework

The OMaSE Process Framework (OMaSE) helps process engineers to define tailored design processes. In OMaSE, process designers can build their own customized design processes by selecting methods fragments from a repository and assembling them into a complete design process. OMaSE is composed of three basic components: the meta-model, the set of method fragments, and the set of guidelines; and, supported by the agentTool Process Editor (Garcia-Ojeda et. al, 2009). The OMaSE metamodel defines the key concepts needed to design and implement multiagent systems (see). The method fragments are operations or tasks that are executed to produce a set of work products, which may include models, documents, or code. Finally, a set of guidelines define how the method fragments are related to one another (for further details refer to Garcia-Ojeda *et al.*, 2007).

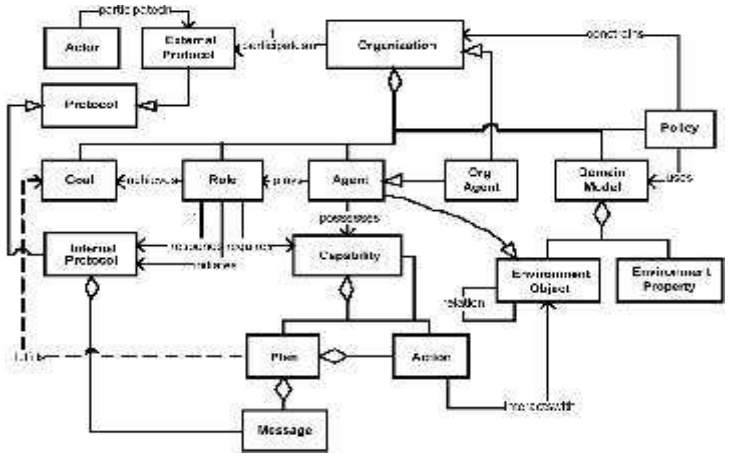
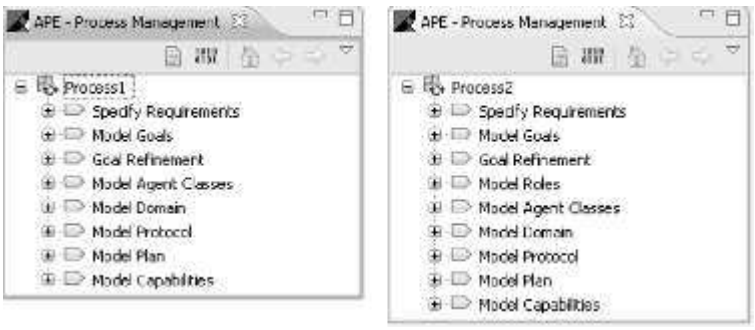


Fig. 1 OMaSe Metamodel

3 Motivating Example

Throughout this work, we use an example from the adaptive sensor networks field to demonstrate the application of OFG in the definition of design process metrics. The example we use is the Adaptive Sensor Network System (ArtS). Basically, an organization is interested in the development of a sensor network system such that its structure self-adapts‡ and overcomes possible sensor (i.e., agents) failures in running time. Therefore, ArtS is required to be highly adaptive and robust. Based on the requirements stated above, let assume that a process engineer assembles two design processes: one generic agent-based design process (see Fig 2 a); and, the other one a role-based design process (see Fig. 2b).



(a) (b)

Fig. 2 . OMaSE-based Compliant Processes

‡Based upon the event that occur in the environment

4 OMaSE-based Flow Graph

This section introduces the OMaSE-based Flow Graph (OFG). Formally, OFG captures OMaSE-based tailored design processes in the form of flow graphs. In OMaSE, processes are assembled choosing the most appropriate tasks from a common repository. These tasks are small jobs performed by one or more producers (i.e., roles). To achieve a particular task, steps (or actions) need to be performed. Likewise, producers create and maintain work products. Work products (or artifacts) are pieces of information or physical entities (e.g., applications, documents, models, diagrams, or code) produced and consumed by tasks. Next, we introduce the formal definition of an OFG

Definition 1 (OFG). An OMaSE Flow Graph is a tuple $OFG = <T, T^{Source}, T^{Sink}, Steps, WorkProducts, Producers, F>$, where:

- T is a finite set of tasks, such that $\forall t \in T$, $t=<name, producer, step, workproduct, input, output>$,
- T^{Source} T is a finite set of source tasks,
- T^{Sink} T is a finite set of sink tasks,
- *Steps* is a finite set of steps, such that $\forall s \in$ Steps, s =<name,description>,
- *WorkProducts* is a finite set of work products, such that $\forall w \in$ WorkProducts, w=<name, description>,
- *Producers* is a finite set of producers, such that $\forall p \in$ Producers, p=<name, workproduct>, where workproducts WorkProducts; and,
- $F \subseteq T \times T$ is the flow relation

The relation F defines a directed graph with nodes T and arcs F. Because OMaSE processes are validated against a set of guidelines[§], we must formally define valid input tasks and output tasks for a given task in an OFG. These constraints are captured by means of Definitions 2 and 3.

Definition 2 (Valid Predecessor Task, $\Rightarrow$t). Given tasks $t_1, t_2 \in$ OFG.T and $f \in$ OFG.F, a *Valid Predecessor Task* $\Rightarrow t$ is defined as $\Rightarrow \mathbf{t} = \{t_2 / t_2 \, f \, t_1 \; t_2.output \cap t_1.input\}$

Definition 3 (Valid Successor Task, $\Rightarrow$ t). Given $t_1, t_2 \in$ OFG.T and $f \in$ OFG.F, a *Valid Successor Task* $\Rightarrow t$ is defined as $\Rightarrow \mathbf{t} = \{t_2 / t_1 \, f \, t_2 \; t_1.output \cap t_2.input\}$

§ In OMaSE guidelines are used to describe how tasks can be combined in order to assemble tailored processes. Guidelines are specified as a set of constraints related to tasks, and work products. These guidelines are formally specified as a set of inputs (i.e., set of work products) that may be used in performing a task and a set of outputs (i.e., set of work products) that may be produced from a task.

On the other hand, Definition 1 allows for OFGs' which are unconnected, without start/end tasks, tasks without any input and output, etc. Therefore we need to restrict the definition to consistent OFGs' (i.e., Definition 4).

Definition 4 (Consistent OFG): An $OFG = <T, T^{Source}, T^{Sink}, WorkProducts, Producers, Steps, F>$ is consistent if:

- $\forall t \in T :\rightarrow \boldsymbol{t} = \emptyset \leftrightarrow t \in T^{Source}$,
- $\forall t \in T :\rightarrow \boldsymbol{t} = \emptyset \leftrightarrow t \in T^{Sink}$,
- (T,F) is a directed, acyclic graph; and,
- $\forall t \in T, \exists t_1 \in T^{Source}, \exists t_2 \in T^{Sink} \mid t_1 \xrightarrow{f^*} t \text{ and } t \xrightarrow{f^*} t_2$

5 Meta OMaSE-based Flow Graph

Given a consistent OFG model, we can even further characterize its structure. Basically, we can make use of the concept of abstraction to generalize collection of tasks with similar purposes into specific components (or modules). This fact is important because project manager/organization can devise the different components that would be modeled at the end of the application of the tailored design process. Also, these components would help us to derive some useful metrics.

Definition 5 (OMaSE-based Components, OComp). A collection of OMaSE-based components, OComp, is the union of seven different components such that $OComp = ReqComp \leftarrow OrgComp \leftarrow FuncComp\ BehComp \leftarrow SocComp \leftarrow NormComp$, where:

- *ReqComp* *OFG.T,* such that $\forall t$ ⌋ *ReqComp,* t.output.name = "System Description" t.output.name = "System Requirement Specification"
- *OrgComp* *OFG.T, such that* $\forall t$ ⌋ *OrgComp,* t.output.name = "Goal Model" t.output.name = "Role Model" t.output.name = "Agent Classes Model" t.output.name = "Model Organization Interfaces"
- *FuncComp* *OFG.T*, such that $\forall\ t$ ⌋ *FuncComp,* t.output.name = "Capability Model"}
- *BehComp* *OFG.T, such that* $\forall\ t$ ⌋ *BehComp,* t.output.name = "Plan Model" t.output.name = "Action Model"
- *SocComp* *OFG.T*, such that $\forall\ t$ ⌋ *SocComp,* t1.output.name = "Protocol Model"
- *DomComp* *OFG.T, such that* $\forall\ t$ ⌋ *DomComp,* t.output.name = "Domain Model"
- *NormComp* *OFG.T, such that* $\forall\ t$ ⌋ *NormComp,* t.output.name = "Policy Model"

Once the set of components is defined, next, we formally define the Meta OMaSE-based Flow Graph (MOFG).

Definition 6 (Meta OMaSE-based Control Flow Graph, MOFG): A Meta OMaSE Flow Graph is a tuple *MOFG = <OComp, MF>*, where

- OComp is a finite set of OMaSE-based Components
- MF ⊆ OComp x OComp is the meta flow relation

The relation M*F* defines a directed graph with nodes *OComp* and arcs *MF*. Definitions 9 and 10 state the conditions for valid flows between OMaSE-based components.

Definition 7 (Valid Meta Predecessor Component, ⇒c). Given c1, c2 ∈ MOFG.OComp and f ∈ MOFG.MF, a *Valid Meta Predecessor Component ⇒c* is defined as ⇒c = *{c2 / ∃ t1∈ c1, ∃ t2∈ c2: c2 f c1 t2.output ∈ t1.input* c1©c2*}*;

Definition 8 (Valid Meta Successor Component, ⇒c). Given c1, c2 MOFG.OComp and f ∈ MOFG.MF, a *Valid Meta Successor Component c⇒* is defined as c ⇒ = *{c2 / ∃ t1∈ c1, ∃ t2 ∈ c2: c1 f c2 t1.output∈ t2.input* c1©c2*}*

Fig. 3 and Fig. 4 show the OFG representation for process 1 and process 2. Likewise, Fig. 5 shows the MOFG representation for process 1 and process 2. Notice they share the same MOFG representation.

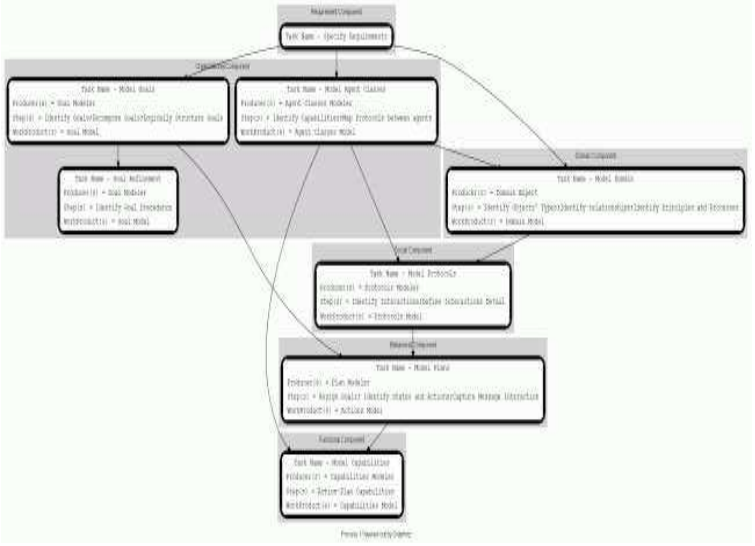


Fig. 3. Process 1 – OFG Representation

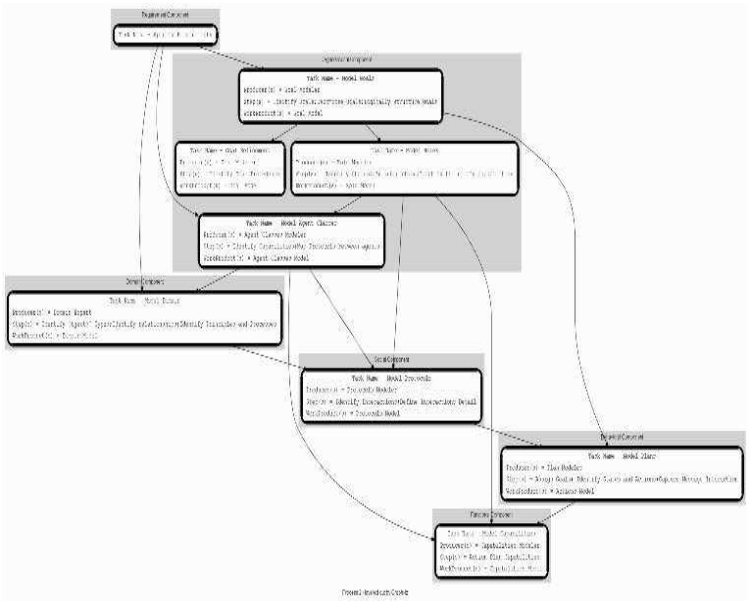


Fig. 4. Process 2 – OFG Representation

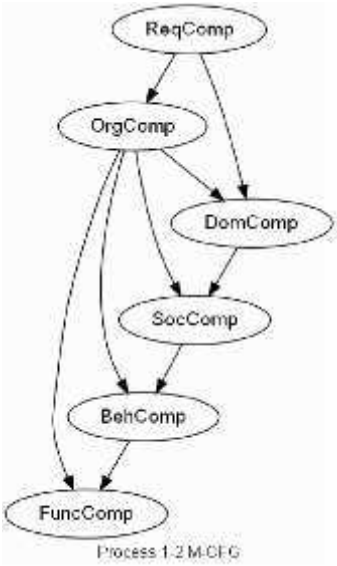


Fig. 5. Process 1 and Process 2
MOFG Representation

6 Metrics Definition

In this section, we introduce a set of metrics derived from the OFG and the MOFG definitions (refer to Sections 4 and0). The aim of these metrics is to provide measurement tools for assessing the advantages and disadvantages of tailored OMaSE-based design process. The set of metrics includes: the size of the tailored design process; and, the expected maintainability, flexibility, and robustness of the system to-be designed. To derive such metrics, this work resorts to the Measurement Information Model (McGarry et al., 2002). The measurement Information Models provides a mechanism for linking defined information needs to software engineering processes and products that can be measured. Next, we explain the set metrics built upon the Measurement Information Model.

6.1 Size of the Design Process

This is the simplest set of metrics. These metrics are based upon the question: what should be measure in order to satisfy the project manager/organization information need regarding the size** of the tailored design process? To address this question, we first identify the process attributes most relevant to the measurement user's information needs. For purposes of this metric, we identified the following attributes of interest: (i) the number of tasks, (ii) the number of steps per task, (iii) people involved in the process; and (iv), the number of OMaSE-based components. Next, we identify the base measures. A base measure is a measure of a single attribute defined by a specified measurement method. In our case, we identify four base measures: OMaSE Component (OC), Process Size (PS), Process Team (PT), and Process Steps (PSt). *OC* is the number of OMaSE Components that can be modeled by using a given process, *PS* is the number of tasks in the process, *PT* is the number of people involved in the process, and *PSt* is the number of steps that need to be performed in the process. Once we have defined the base measures, the next step is to derive the indicators. An indicator is a measure that provides an estimate or evaluation of specified attributes derived from an analysis model with respect to defined information needs. In our analysis, we have defined two indicators: OMaSE Component Workload Index (OCWI) and Process Workload Index (PWI). These two indicators will help project managers/organization to estimate the size of the OMaSE-based tailored process (see Table 1). Thus, by Applying the metrics defined in this section, the project manager/organization may assume that using process 2 in the development of ArtS would imply more money and time than process 1 (see Table 2).

** Size implies time and cost

Indicator	Description	Analysis Model	Decision Criteria
OCWI	**OMaSE Component Workload Index (OCWI)**. This is calculated by establishing the proportion of *OMaSE Components* and *steps* per producer in the process	OWCI = (PSt / OC) * PT	Range [0 ... ∞]The higher this measure is, the greater the size of the process is. (more time and money invested)
PWI	**Process Workload Index (PWI)** . This is calculated by est ablishing the proportion of *steps* and *tasks* per *producer* in the process	PWI = (PSt / PS) * PT	Range [0 ... ∞] The higher this measure is, the greater the size of the process is. (more time and money invested)

Table1 . Metrics Indicator

Process Size Metrics	OC	PS	PT	PSt	OCI	PWI
Process 1	5	7	6	12	14.4	10.29
Process 2	5	8	7	14	19.6	12.25

Table 2 . Derived Process Size Metrics

6.2 Estimating Maintainability, Flexibility, and Robustness of the System to-be Designed

One of the main goals of multiagent systems is to provide solutions for a wide variety of problems (Jennings, 2001). However, indeterministic environments make multiagent systems susceptible to individual failures that can significantly reduce its ability to accomplish its overall goal (DeLoach *et al.*, 2008). Therefore, from the point of view of the project manager/organization, it is extremely important that tailored processes can cope with the development of maintainable, flexible, and robust multiagent systems. Next, we characterize the measurement to be carried-out with the purpose of estimating maintainability, flexibility, and robustness of the outcome of the design processes.

6.2.1 Maintainability

In software engineering, the maintainability of software products or design models is associated with making future maintenance easier. To estimate the maintainability, we rely on the size of the organization. In (Matson, 2008), the author stresses the importance of measuring the size of the multiagent organization. For purposes of this metric, we identify the following attributes: the number of organizational concepts, the number of organizational relationships, the number of functional components, and the number of organizational-functional relationships. Next, we identify the base measures. In this case, we identify four base measures: Organizational Concepts (OC), Organizational Relationships (OR), Functional Concepts (FC), and Organizational-Functional Relationships (OFR). Formula 1 states how OC is calculated. Basically, OC is the number of organizational concepts contained in *OrgComp*. Likewise, Formula 2 gives us the relationships between organizational components. This is obtained by

counting the number of links between elements in *OrgComp*. Also, FC is calculated in Formula 3. In essence, FC can take either the value 0 or 1. Finally, OFR is calculated in Formula 4. Formula 4 gives us the number of outgoing links from elements in *OrgComp* to elements in *FuncComp*. N is the cardinality of the given set.

$$OC = \begin{cases} 0 & ,OrgComp \in MOFG \\ N(C) = \{C \quad OrgComp \mid \forall t_1, t_2 \in C : t_1.output \odot t_2.output \quad t_1.name \odot t_2.name\} & ,otherwise \end{cases} \quad (1)$$

$$OR = \begin{cases} 0 & ,OrgComp \in MOFG \\ N(R) = \{R \quad OFG.F \mid \forall f \in R, \exists t_1, t_2 \in OrgComp : t_1 f t_2 \quad t_1.output \odot t_2.output\} & ,otherwise \end{cases} \quad (2)$$

$$FC = \begin{cases} 0 & ,FuncComp \in MOFG \\ 1 & ,otherwise \end{cases} \quad (3)$$

$$OFR = \begin{cases} 0 & ,OrgComp \in MOFG \\ N(R) = \{R \quad OFG.F \mid \forall f \in R, \exists t_1 \in OrgComp, t_2 \in FuncComp : t_1 f t_2\} & ,otherwise \end{cases} \quad (4)$$

Once we have defined the base measures, the next step is to derive the indicator. For maintainability, we have defined one indicator: Design Process Maintainability Index (DPMI). This indicator will help project managers/organization to estimate the degree of maintainability of the system to-be (see table 3). Because of this, the project manager/organization may assume the maintenance of process 1 would be easier than the maintenance of process 2 (see Table 4).

Indicator	Description	Analysis Model	Decision Criteria
DPMI	**Design Process Maintainability Index (DMPI)**. This is calculated by establishing the nu mber of Concepts and Relationships within the Organizational and Functional Components, as well their relationships.	DPMI = OC + FC + OR + OFC	Range [0 ... ∞] The lower this measure is, the easier to maintain.

Table 3. Design Process Maintainability Metrics Indicator

Process Size Metrics	OC	OR	FC	OFR	DPMI
Process 1	2	1	1	1	5
Process 2	3	2	1	2	8

Table 4. Derived Design Process Maintainability Metrics

6.2.2 Flexibility of the System to-be Designed

In (Robby et al., 2006), authors provide a set of metrics for measuring system flexibility based upon the state-spaces generated by Bogor for particular OMaSE-based organization designs. For purposes of this work, we propose a different approach for measuring the flexibility of the system to-be designed. Since agents depend upon specific

capabilities for accessing and modifying the environment and such capabilities are realized upon plans and actions, we argue that the higher the coupling of the behavioral component with other components is, the more tools the development team would have to model flexible (i.e., able of overcoming failures) systems. Because flexibility is defined in terms of the *BehComp* (Formula 5), if *BehComp* = ⌉ then the flexibility is 0; otherwise, we calculate the In-Degree Function (IDF) value for the behavioral component (see Formula 6). The IDF function calculates the number of incoming relationships to *BehComp*. As result, flexibility's values ranges from 0 to 1. By applying the metrics defined in this section, the project manager/organization may expect the same degree of flexibility either using process 1 or process 2 (see table 5).

$$\text{flexibility} = \begin{cases} 0 & ,\text{BehComp} \sqsubset \text{MOFG} \\ 1 - \dfrac{1}{\text{IDF}} & ,\text{BehComp} \rfloor \text{MOFG} \end{cases} \quad (5)$$

$$\text{IDF} = \left\{ \begin{array}{ll} 0 & ,\text{BehComp} \notin \text{MOFG} \\ N(I)\{I \subset \text{MOFG.MF} \mid \forall f \in I, \exists c_1, c_2 \in \text{MOFG.OComp}, \exists t_1 \in c_1, t_2 \in c_2 : t_1 f t_2 \wedge c2 = \text{BehComp}\} & ,\text{otherwise} \end{array} \right\} \quad (6)$$

Process Size Metrics	IDF	Flexibility
Process 1	2	0.5
Process 2	2	0.5

Table 5 . Derived Design Process Flexibility Metrics

6.2.3 Robustness

In computing, the robustness of a system is associated with the ability of performing without failure under a wide range of conditions. We may assume that if a system that can adapt when external changes occurs (i.e., flexible), then the robustness of the system will increase. For purposes of this work, the robustness of the system to-be designed is given by,

$$\text{robustness} = \begin{cases} 0 & ,\text{flexibility} = 0 \\ \dfrac{\text{flexibility}}{1 - \text{flexibility}} & ,\text{flexibility} > 0 \end{cases} \quad (7)$$

Robustness's values range from 0 to . Applying the metrics defined in this section to the processes depicted in Section 3, the project

manager/organization may expect the same degree of robustness either using process 1 or process 2 (see Table 6).

Process Size Metrics	Flexibility	Robustness
Process 1	0.5	1
Process 2	0.5	1

Table 6 . Derived Design Process Robustness Metrics

7. Metrics Validation

To empirically validate the metrics defined in the previous section, we run two different set of simulations. Next we present the preliminary results.

7.1 Size of the Design Process

For the first experiment, we used a trial version of the iGrafx tool (http://www.igrafx.com/). IGrafx is a discrete event simulator for generic processes simulation. To measure the size of the design process, we simulated the cost and completion time. For every model (i.e., design process), we simulated that for every task in the model, every step was scheduled back to back satisfying the following conditions: (i) service time between steps was normally distributed over an interval of 2 to 4 days, (ii) team members were paid $5.00 per hour; and, (iii) team members used to work 8 hours/day, 5 days/week, and 22 days/month. For empirical analysis, each design process (i.e., model) was simulated for 50 executions. The results of the simulation are shown in table 7 .Notice that results are consistent with the measurements obtained in Section6.1.

Process-related Metrics	Process Completion (Weeks)	Total Labor Cost ($)
Process 1	2.71	$1,368.30
Process 2	3.71	$1,888.70

Table 7 . Size of the Design Process - IGrafx Simulation Results

7.2 System to-be Designed related Metrics

For the second experiment, we designed several ArtS models using process 1 and process 2. Worth noting that the final structure of OMaSE-based designs takes on the form of a graph for computational purposes (Matson, 2008). Because the different models might have goals, roles, capabilities, and agents (each which can grow independently of the

others), we assume that sparse and dense graph specification might provide us the basis for lower and upper bound approximations. Also, for purposes of this work, a generic goal model for the ArtS system is specified (see Fig. 6). Such model consists of four leaf goals: G1.1, G1.2, G2, and G3. The "precedes" relation (*p*) provides the natural ordering of goals achievement in the system. That is, the "precedes" relation between goal x and goal y states that goal *y* cannot be pursued until goal *x* has been achieved (DeLoach *et al.*, 2008). Next we show the preliminary results.

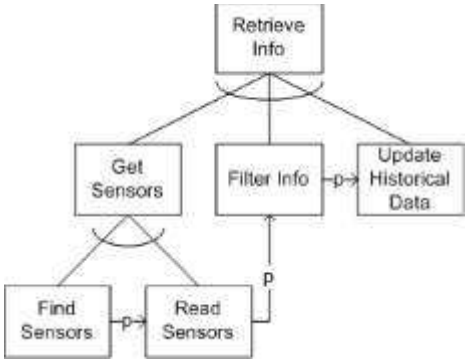


Fig. 6. ArtS Goal Model

7.2.1 Process 1: OMaSE-based Generic Model

For process 1, we documented four generic models (see table 8). Every model consists of four agents, four capabilities, and four leaf goals (i.e., G1.1, G1.2, G2, and G3). For instance, in *Model 1* (see Table 7), every agent possesses one capability and every capability fulfills every leaf goal (via a plan). The size of every model ranges from 32 (i.e., lower) to 44 (i.e., higher). In the case of *Model 1* its size is 32. That is, *Model 1* contains four agents, four capabilities, and four goals, and 20 relationships (i.e., relationships between them). To empirically evaluate the flexibility of designs M1 – M4, we developed a simulation that stepped through the ArtS application. To measure the flexibility, we simulated capability failure (i.e., plans or actions failure rates ranging from 0 to 100%). At each step in the simulation, a randomly assigned goal was achieved. Then, one agent (i.e., sensor) capability was randomly selected and then tested to see whether or not it had failed. We assumed once a capability failed (i.e., plan or action), a capability remained failed for the life of the system. Then, reorganization was performed to assign available sensors to available goals and to de-assign sensors if their capability had failed, and they were no longer able to fulfill their goal. Also to measure the robustness, we calculate the expected functionality of the system under capability

failures. For empirical analysis, each model was simulated for 500 executions. To compare the flexibility and robustness of the models, we looked at the results using each of the models. The results in Fig. 8 and Fig. 9 show that Models 2 and 3 provides more flexibility and robustness than Models 1 and 4.

Model	CG<<via plans>>	AC <<possesses>>	Size
1	Dense	Sparse	32
2	Sparse	Dense	32
3	Dense	Dense	44
4	Sparse	Sparse	20

Table 8. Process 1 – Generic Models

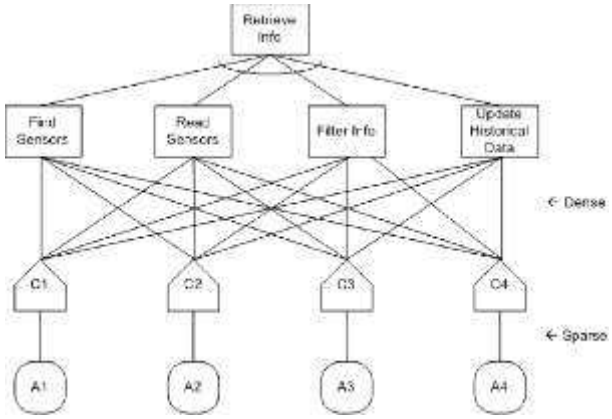


Fig. 7. Process 1 – Model 1

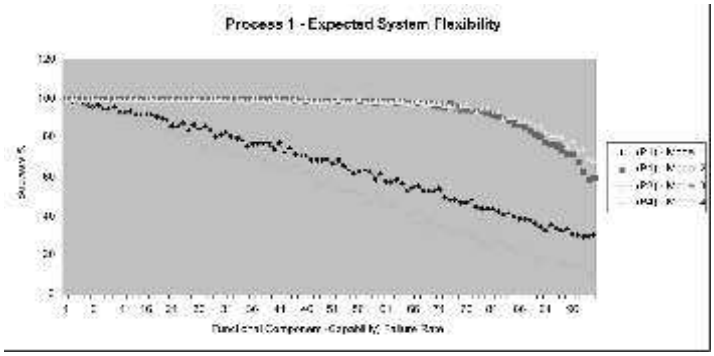


Fig. 8. Process 1 – Expected System Flexibility

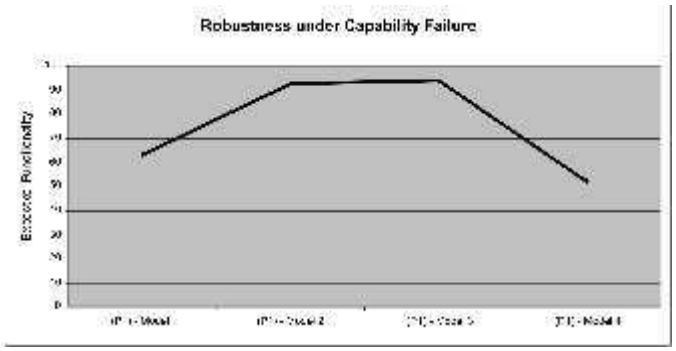


Fig. 9. Process 2 – Robustness under Capability Failure

7.2.2 Process 2: Role-based Model

For process 2, we documented eight generic models (see Table 9). Unlike Process 1, every model in Process 2 consists of four agents, four roles, four capabilities, and four leaf goals (i.e., G1.1, G1.2, G2, and G3). For example, in *Model 1* (see Figura 10) every agent possesses one capability, every role requires one capability and every role achieves one leaf goal. The size of every model ranges from 28 (i.e., lower) to 64 (i.e., higher). In the case of *Model 1* its size is 28. That is, *Model 1* contains four agents, four roles, four capabilities, and four goals, and 12 relationships (i.e., links between them). To empirically evaluate the flexibility and robustness of designs M1 – M8, we used the same approached introduced before (see Section 7.1). That is, to measure the flexibility and robustness, we simulated capability failure (i.e., i.e., plans or actions) and we calculated the expected functionality of the system under capability failures, respectively. However, for every model build using process 2, we simulate the roles played by agents (i.e., sensors). For empirical analysis, each model was simulated for 500 executions. To compare the flexibility and robustness of the models, we looked at the results using each of the models. The results in Figure 11 and Figure 12 show that Models 2 and 6 provides more flexibility and robustness than Models 1, 3, 4, 5, 7, and 8. Also, the reader can notice the expected flexibility and robustness for Models 5 and 7 is 0. This is due to the assumption that if an agent does not possess the set of required capabilities by a given role, then the role cannot achieve its goals (see DeLoach et al., 2008; DeLoach, 2009b). In the simulation, if an agent's capability fails and the role played by the agent requires such capability, the role is no longer able to achieve the assigned goal.

Model	RG<<Achieves>>	RC<<requires>>	AC<<possesses>>	Size
1	Sparse	Sparse	Sparse	28
2	Sparse	Sparse	Dense	40
3	Sparse	Dense	Dense	52
4	Dense	Dense	Dense	64
5	Sparse	Dense	Sparse	40
6	Dense	Sparse	Dense	52
7	Dense	Dense	Sparse	52
8	Dense	Sparse	Sparse	40

Table 9. Process 2 – Generic Models

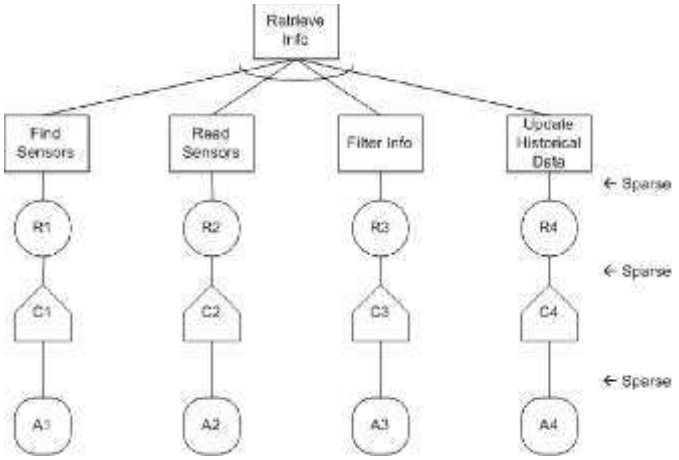


Fig. 10. Process 2 – Model 1

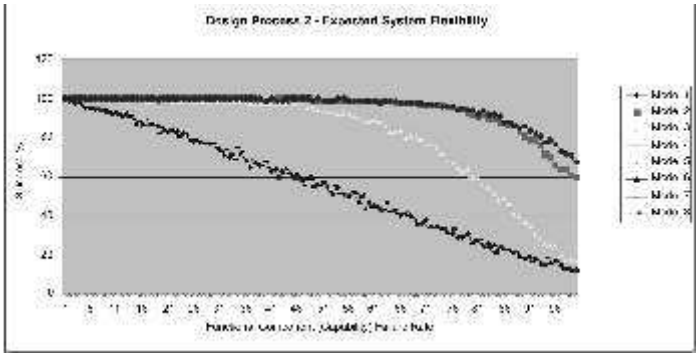


Fig. 11. Process 2 – Expected System Flexibility

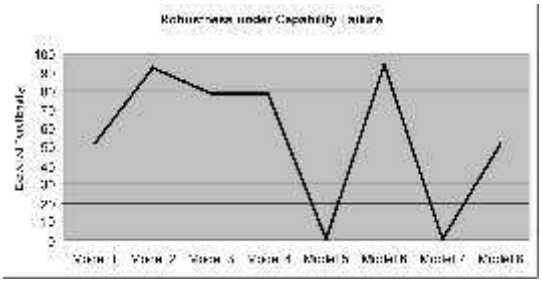


Fig. 12. Process 2 – Robustness under Capability Failure

7.3 Comparing Process 1 and Process 2

In terms of the design process size, we would expect process 1 to be a less expensive and time-consuming process than process 2 (results shown in Section 7.1). In terms of maintainability, flexibility, and robustness (results shown in Section 0), we would expect process 1 to be more maintainable than process 2. Also, we would expect flexible and robust systems either following process 1 or process 2 (see Figure 13 and Figure 14).

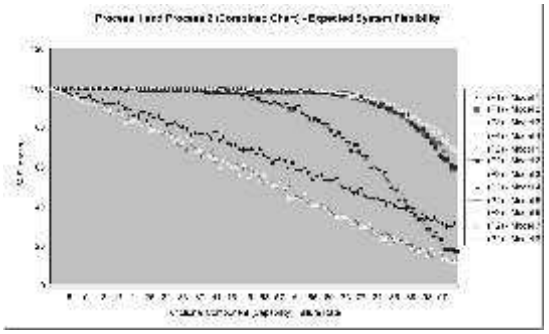


Fig. 13. Process 1 (P1) and Process 2 (P2) - Flexibility (Combined Chart)

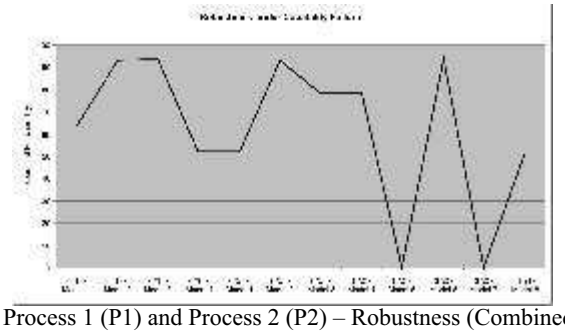


Fig. 14. Process 1 (P1) and Process 2 (P2) – Robustness (Combined Chart)

8 Conclusions

This work presented a flow graph based approach – OFG, MOFG – for assessing the advantages and disadvantages of tailored design processes. In this work, the OFG and the MOFG models were used as a modeling, analysis, visualization, and abstraction tool for measuring tailored design processes. Although this work is a starting point, one interesting question to address is when is a role-based design process a better choice over a generic agent-based design process? From the simulated results, it is clear that the re-organization of the system relies on how well agents and capabilities are coupled.

9 References

[1] [Brinkkemper, 1996] Brinkkemper, S. (1996). Method Engineering: Engineering of information Systems Development Methods and Tools. Information and Software Technology 38, pages 275 – 280. Elsevier Science B.V.

[2] [Cardoso, 2005] Cardoso, J. (2005) Control-flow Complexity Measurement of Processes and Weyuker's Properties. In 6th International Enformatica Confererence, Transactions on Enformatica, Systems Sciences and Engineering, Vol 8, (2005), pages 213 – 218.

[3] [Cardoso et al., 2006] Cardoso, J., Mendling, J., Neuman, G., and Reijers, H.A. (2006). A Discourse on Complexity of Process Models. Business Process Management Workshops (2006), pages 117 – 128.

[4] [Cernuzzi and Rossi, 2002] Cernuzzi, L., and Rossi, G. (2002). On the Evaluation of Agent-Oriented Modeling Methods. Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, pages 21 – 30, Seattle, USA.

[5] [Chella et al., 2004] Chella, A., Cossentino, M., Sabatucci, L., Seidita, V. (2004). From PASSI to Agile PASSI: tailoring a design process to meet new needs. Proceedings of the IAT 2004, pages 471 – 474.

[6] [Dam and Winikoff, 2003] Dam, K. H.., and Winikoff, M. (2003). Comparing Agent-Oriented Methodologies. Giorgini, P., and Winikoff, M. (Eds.) Proceedings of the 5th International Bi-Conference Workshop on Agent-Oriented Information Systems, pages 52 – 59.

[7] [DeLoach et al., 2008] DeLoach, S.A., Oyenan, H.W., and Matson, E.T. (2008). A capabilities-based model for adaptive organizations. Autonomous Agents and Multi-Agent Systems 16(1): 13-56 (2008)

[8] [DeLoach, 2009a] DeLoach, S.A. (2009). Moving Multiagent Systems from Research to Practice. Special Section on Future of Software Engineering and Multiagent Systems. International Journal of Agent-Oriented Software Engineering (IJAOSE). (in Press).

[9] [DeLoach, 2009b] DeLoach, S.A. (2009). OMACS: a Framework for Adaptive, Complex Systems. In Virginia Dignum (ed.) Multi-Agent Systems: Semantics and Dynamics of Organizational Models. IGI Global: Hershey, PA. ISBN: 1-60566-256-9 (March 2009).

[10] [Garcia-Ojeda et al., 2007] Garcia-Ojeda, J.C., DeLoach, S.A., Robby, Oyenan, W.H., and Valenzuela, J.L. (2008) OMaSE: A Customizable Approach to Developing Multiagent Development Processes. In Proceedings of the 8th International Workshop on Agent-Oriented Software Engineering AOSE 2007.

[11] [Garcia-Ojeda et al., 2009] Garcia-Ojeda, J.C., DeLoach, S.A., and Robby. (2009) agentTool Process Editor: Supporting the Design of Tailored Agent-based Processes. In Proceedings of the 2009 ACM Symposium on Applied Computing SAC 2009.

[12] [Henderson-Sellers, 2005] Henderson-Sellers, B. (2005) Creating a Comprehensive Agent-Oriented Methodology: Using the Method Engineering and the OPEN Metamodel. Henderson-Sellers, B., and Giorgini, P. (Eds.) Agent-Oriented Methodologies, pages 368 - 386.

[13] [Jennings, 2000] Jennings, N.R. (2000). On Agent-Based Software Engineering. Artificial Intelligence, 117(2), pages 277 – 296.

[14] [Jennings and Wooldridge, 2001]. Agent-Oriented Software Engineering (2001). Bradshaw, J. (Ed.) Handbook of agent technology, AAAI Press /MIT Press.

[15] [Matson, 2008] Matson, E.T. (2008). Transition in Multiagent Organizations. Ph.D. Disseration, University of Cincinnati, pages 209.

[16] [McGarry et al., 2002] McGarry, J., Card, D., Jones, C., Layman, B., Dean, J., and Hall, F. (2002). Practical Software Measurement: Objective Information for Decision Makers. Addison-Wesley Longman Publishing Co., Inc. 2002.

[17] [Nardini et al. 2008] Nardini, E., Molesini, A., Omicini, A., and Denti, E. (2008). SPEM on test: the SODA case study. In Proceedings of the 2008 ACM Symposium on Applied Computing SAC '08, pages 700-706.

[18] [Polyvyanyy et al., 2008a] Polyvyanyy, A. and Weske, M. (2008). Flexible Process Graph: A Prologue. In Proceedings of the OTM 2008 Confederated international Conferences R. Meersman and Z. Tari, Eds. Lecture Notes in Computer Science, vol. 5331. Springer-Verlag, Berlin, Heidelberg, 427-435

[19] [Polyvyanyy et al., 2008b] Polyvyanyy, A. and Weske, M. (2008b). Hypergraph-based Modeling of Ad-Hoc Business Processes. In Proceedings of the 1st International Workshop on Process Management for Highly Dynamic and Pervasive Scenarios, (Sep 2008).

[20] [Robby et al., 2006] Robby, DeLoach S.A., and Kolesnikov, V.A. (2006). Using Metrics for Predicting System Flexibility. In Proceeding of the Fundamental Approaches of Software Engineeering (FASE'06). 2006.

[21] [Shehory and Sturm, 2001] Shehory, O., and Sturm, A. (2001). Evaluation of Modeling Techniques for Agent-Based Systems. Proceedings of the 5th International Conference on Autonomous Agents, pages 624 – 631. ACM Press.

[22] [Sturm and Shehory, 2003] Sturm, A., and Shehory, O. (2003). A Framework for Evaluating Agent-Oriented Methodologies. Giorgini, P., and Winikoff, M. (Eds.) Proceedings of the 5th International Bi-Conference Workshop on Agent-Oriented Information Systems, pages 60 – 67.

[23] [Sturm and Shehory, 2004] Sturm, A., and Shehory, O. (2004). A Comparative Evaluation of Agent-Oriented Methodologies. Bergenti, F., Gleizes, M-P., and Zambonelli, F. (Eds.) Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering Handbook, pages 127 - 149. Kluwer Academic Publisher.

[24] [Yu and Cysneiros, 2002] Yu, E., and Cysneiros, L. M. (2002). Agent-Oriented Methodologies – Towards a Challenge Exemplar. Giorgini, P., Lesperance, Y., Wagner, G., and Yu, E. (Eds.) Proceedings of the 4th International Bi-Conference Workshop on Agent-Oriented Information Systems, pages 47 – 63.

[25] [Zambonelli and Omicini, 2004] Zambonelli, F., and Omicini, A. (2004) Challenges and Research Directions in Agent-Oriented Software Engineering. Autonomous Agents and Multi-Agent Systems 9, 3 (Nov. 2004), pages 253 – 283.