Evaluating Information Systems: Constructing a Model Processing Framework

Extended Abstract

João Dias Santa de Vasconcelos Duarte[1]
Professor Orientador: Prof. André Vasconcelos[1]
Co-Orientador: Eng. José Alegria[2]

[1]Instituto Superior Técnico
[2]PT-Comunicações

June 23, 2009

Abstract

Enterprises are growing increasingly hungry for information: the will to empower decisions with carefully digested data, the wish of transform all implicit knowledge into clearly defined, explicit repositories that can be shared by all and the desire to monitor all aspects of their internal dynamics. In the past decade, the rush to technology has created several flaws when it comes to managing computers, applications, middleware and information systems and so organisations struggle to understand how all these elements are behaving.

Even today, as Enterprise Architectures grow in significance and are acknowledged as advantageous artifacts to help manage change, their benefit to the organisation has yet to be fully explored.

We therefore combine our desire for real-time evaluation with the benefits of architecture representation to produce an ontologically supported method of modelling, capturing and evaluating systemic behaviour. We produce a conceptual framework to performing this task, while avoiding the imprecise definitions of quality and quality attributes. According to our abstraction, transfering that responsibility to the end user is essencial since such aspects are subjective and depend on the human context in which they exist. This conceptualisation was materialised in a model-eval-display loop framework, and implemented using Model Driven Software Development practices and tools.

Finally, we tested the prototype against a real-world scenario in PT-Comunicações (PT-C), to observe our conceptual solution in practice.

Keywords: *Evaluation, Information System Architecture, Model Processing Framework, Phenomenology*

1 Introduction

Laudon describes in [9] the importance of knowledge and knowledge management in the today's enterprises. Even though humans have the innate trait of self-awareness through consciousness, organizations — becoming

more and more living complex social organisms — fail to automatically develop similar capabilities of sensing and learning. This task then falls upon the individuals, who must work together, interact and share knowledge and, therefore, engage the task of sense-making as to improve the collective self-awareness.

To transfer knowledge, communication must be done through known shared boundary objects, that are described using a set of concepts and with a semantics which cannot be ambiguous.

The need for Organizational Knowledge is already a widely accepted fact, and in order to improve the holistic view of the Organizational Structure, Enterprise Architectures have been proposed as valuable artifacts due to their communicative nature [11].

With mechanisms to represent organization blueprints, easy-to-use high level notations to describe high-level requirements and real-time processing, firms will become increasingly aware of themselves and their ecosystem, and will be able to make quicker, smarter and better informed decisions.

To increase the collective organizational knowledge, information has to be applied, which in turn can only be produced by processing relevant data in the first place [1]. The ability to collect more data from more organizational sources is there unavoidably valuable. We will focus on the use of data produced by operation systems, middleware, applications and other technological elements to better understand how the technology and information layers of a deployed enterprise architecture behave in real-time.

Regarding architectural representations as boundary objects, we intend to take advantage of their creation and maintenance to encourage stakeholders in an enterprise to discuss, define and perform evaluation through the definition of quality attributes.

This thesis therefore addresses today's inability to evaluate systems in real-time through the "eyes" of Enterprise Architectures.

1.1 Main contributions

This thesis focuses on the problem of providing an unified, **architecture-centric evaluation framework of the real-time behaviour of information systems**. In order to accomplish this objective, several tasks were executed, from which the following contributions emerged:

1. An analogy between the sociological and philosophical domain that studies phenomena, or Phenomenology, and enterprise manifestations and evaluations;
2. A conceptual Model Processing Framework (MPF) that comprises a cycle of Modelling, Instantiation, Data Collection and Evaluation;
3. An implementation of the MPF using openArchitectureWare (oAW) in which:
   (a) We extended the CEO Framework to support our ontology;
   (b) The primitives of the framework were used to populate an oAW's profile diagram;
   (c) We created a template project on oAW that implements each step of the MPF, along with an automation workflow that executes the full cycle of the MPF.

1.2 Paper Organization

The rest of the paper is organized as follows into four sections. Section 2 provides an overview of related work regarding Enterprise Architectures, quality evaluation and modelling instrumentation. Section 3 presents the proposed solution: the conceptual abstraction, the Model Processing Framework and its implementation. Section 4, presents a case study of the implementation of the developed prototype on an industrial environment. Section 5, presents the conclusions and the future work sugestions.

2 Related Work

2.1 Enterprise Architectures

As stated in chapter 1, architectural representation is already as a benefit to enterprises. These enterprise architecture meta models are usually constructed in layers that separate, for example, hardware and software from human actors and even information.

The Unified Enterprise Architecture Modelling Language(and LEAN) [7], is a highly conceptual meta-model for enterprise modelling based on a societal metaphor. It can be understood as meta-metamodel for Enterprise description, from where concepts of Organisation Engineering Center Framework (CEOF) [2] and ArchiMate [8] (for example) can be especialized.

The CEOF stems from the academic domain of Organisational Engineering and defines a clear hierarchy of concepts for Enterprise Architectures, views for different desires, and provides structural metrics for TO-BE architecture comparison.

The ArchiMate language stems from a mix of academic and organisational environments, and defines a taxonomy, along with views that target different stakeholders. Regarding evaluation, ArchiMate also focuses on evaluation of TO-BE architectures, but instead provides analysis of their dynamic behaviour.

What all have yet to provide is the ability to, once the architecture is deployed, perform evaluation of behaviour of this architecture, using real-time data and events. To be able to execute this action we can provide benefits to the architecture life cycle:

- By executing real-time evaluation, we assert the deviation from the before(model) and the implementation, for events specified using model elements and so we perform a "reality-check";
- Perform observations on how the architecture actually behaves after implementation, so that the evaluations such as ArchiMate's can be corroborated (along the temporal axis).
- Understand changes in behaviour, either from infrastructure degradation or unknown change in practices, habits or processes.

2.2 Evaluating Information Technology: metrics and measuring qualities

In [4], Robert E. Filman describes the steps towards achieving "ilities" - typical attributes of systems such as reliability, performance, availability and maintainability - in compositional architectures. These concepts exist in a myriad of domains ranging from Networking to Software Development - and recently, Enterprise Architectures. Even within a given domain, these definitions are not clear. For example, in multimedia domains, the notion of quality of service is subjective since there isn't a widely accepted QoS Framework [3] and so QoS must be always understood in the context of the domain or framework used to measure it.

Filman then states that there is more than one definition for concepts such as reliability; to define reliability is to specify the requirements established in a certain environment and by certain people to attain reliability. The consequence of this looseness is therefore an impossibility to uniquely and globally define ilities. Nevertheless, for one to be able to measure the behaviour of an information system according to a set of qualities, the definition of the set of functional, aesthetic, systematic and combinatoric requirements must be accomplished by stakeholders.

Regarding Enterprise Architectures, as we presented in the previous section, CEOF provides structural metrics, while ArchiMate focuses on predicting behaviour. Archimate's evaluation, however, isn't built on a well defined structured of metrics, which is a problem that we will discuss in the next chapter.

2.3 Instrumentation

Considering that our solution manipulates models and we wish to provide users with an executable representation of the enterprise architecture, we identified several necessities for these tools. **Support for metamodels and metamodel extensions** is essential since Enterprise Architecture Frameworks are metamodel level entities. To augment portability and mobility of this solution, **import and export** of models is desirable. One way of performing validations is through the **verification of constrains** on the model; for UML, the OCL performs this function and therefore it is required for the tool not only syntax validation of the constraint language but also execution. The last but equally important feature is the ability to **chain model transformation actions**, since we're going to perform model edition, validation and execution of custom tasks.

After comparing Enterprise Architect and openArchitectureWare regarding these features, we concluded that openArchitectureWare is better suited for task at hand. Also, by being Open Source, we have the possibility of extending the tool for ajust to our needs.

2.4 On providing quality views over Information System Architectures

We were able to establish that, of the Enterprise Architecture Metamodels, none performs real-time evaluation of deployed architectures. This feature has potential benefits such as helping to reduce the gap between reality and the modelled architecture, provide constant feedback on its behaviour and, finally, reduce the time to

assert the necessity of change.

So too exists a lack of proper instrumentation for performing real-time evaluation but there are, however, tools that can (and will) be used to accomplish this task.

In the next chapter, we will take Gerald Khoury's work on the Lightweight Enterprise Architecture Notation and André Vasconcelos' improved CEOF Architecture Metamodel and construct a conceptual framework for processing models to perform real-time evaluation. Next, we will take openArchitectureWare and provide an implementation of this framework so that we can atest its feasibility.

3 Proposed Solution

The problem we address is the inability to perform modelling of evaluating systems in real-time, using the enterprise architectures as a boundary object and a base for communication. The solution we developed is comprised of five major steps: establishing a strong conceptual base on the evaluation of systems, defining a high level language for this activity, extending an architecture creation pipeline, creating a framework that automates it and finally implementing this framework.

3.1 Establishing the Phenomenological setting

In [14], Recker and Niehaves explained the relevance of philosophy in Information System research by linking several philosophical disciplines such as Ontology, Methodology and Epistemology to IS research paradigms (e.g. positivism, interpretivism). By understanding how we perceive others, what we observe and how we judge, we were able to come up with unified, global, overview of the concepts that are involved when performing evaluation of Information Systems and, therefore, be able to model this domain.

Society has been established as a proper metaphor for enterprises. In order to accomplish our goal, we set out to find a parallel between experiencing, judging, perceiving-capable agents since, inherently, these activities occur inside society and consequently within the metaphor.

Phenomenology is the "study of structures of consciousness as experienced from the first-person point of view." [15]. Its key concepts are: *intentionality* (experience always is directed at something or is about something), *qualia* (sensory data), *bracketing* (putting aside the question of the existence of a real world) and *consciousness* itself. We found this philosophical discipline and these concepts to have a profound link to the Information Systems and Information System research area [14]. In the following subsections of this introduction we describe the mentioned concepts, and then move on to determine in what way the phenomenological dialectic fits the problem at hand.

Since *Phenomenology* imposes **subjectivity** upon the world, we can place its rationale within the *interpretivist* trend [10, 13]. Considering that the phenomenologic concepts and their relations apply to society, we then propose that they can also be mapped into the society metaphor and therefore its target (enterprise systems).

The act of perceiving (*noesis*) the enterprise structure allows the detection of its meaning, therefore producing the boundary object (*noema*) referred in the Organisational Engineering domain as Enterprise Architecture. And by determining what manifestations are observable on this architecture, we perform the *phenomenological bracketing*, for the assumption develops that objects will always be perceived as modelled, regardless of the particularities relative to their nature.

This construction of an enterprise architecture allows the specification of enterprise elements as simply 'being there' - or *noumena* - while by describing their manifestations we define how they appear before the senses - *phenomena*.

Modelling evaluation as an activity that is inherently dependent on observations - *qualia* - we establish a link to the *intentionality* aspect of experience, according to phenomenology. And, finally, by having agents record manifestations as observed (from the first person standpoint) we construct **the observed objects' *facticity***.

Considering this "phenomenological ontology" in a level of abstraction greater than the target of our desired metaphor, we were able to assert its validity and also construct a parallel between evaluation of Information Systems and the philosophical discipline of phenomenology. Many of the central concepts of this domain can be understood under the scope of agent interaction, should they be systems, humans, communities and organisations.

3.2 Producing a High-Level Language Extension

We began by performing the following changes on the LEAN node pairings: (1) added an "observes" relationship between an **Agent and a Resource**, (2) add the **Agent** → **Resource** and **Agent** → **Rule** to the "produces" pairing. These two changes reflect the necessity on bringing Agents and Resources closer. This necessity arose from our closer inspection of the philosophical notions of experience. This close binding also supports the approach of modelling what can be observed on an entity and how to measure that entity's actions.

With a proper metaphor for the act of evaluation, we proceeded to construct our Model Processing Framework. The next (brief) section defines the setting on which the MPF operates.

3.3 Revisiting the Information System Architecture building pipeline

The solution we created further extends ISA creation pipeline (improved in [16]), so that systems can be monitored from the perspective of the final **(AS-IS) ISA**. The revisited pipeline (fig. 1) includes an extra cycle located at its end that feeds the ISA artifact, the manifestations and measurements into a processing framework. This framework is able to evaluate the state of the deployed ISA regarding defined metrics, as well as the ability to initiate this evaluation process at will.

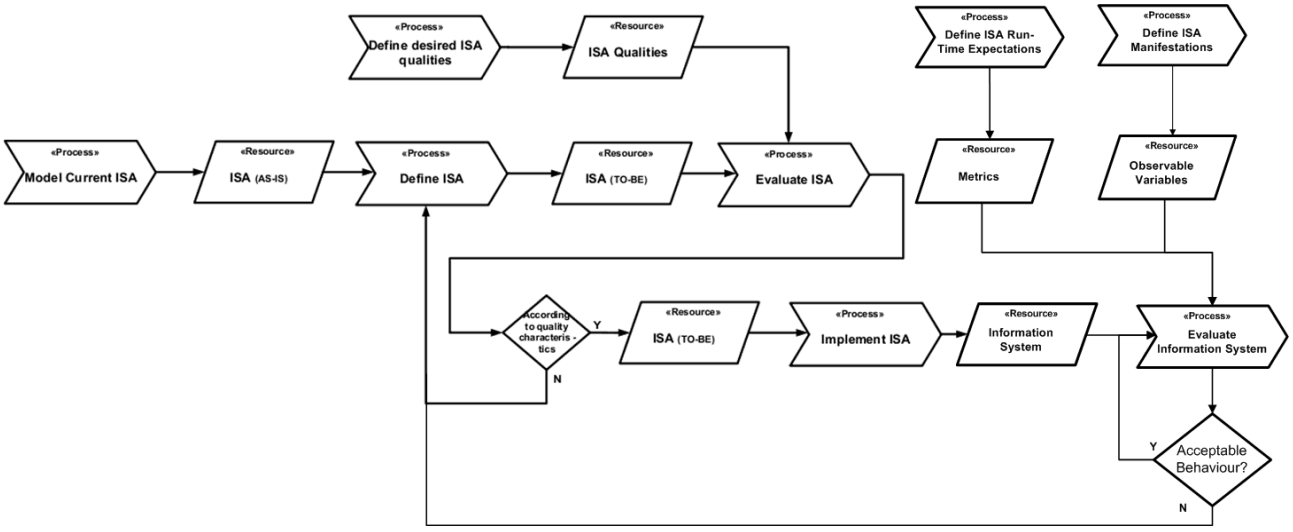

Figure 1: Proposal for pipeline with Information Systems evaluation

3.4 Defining a Model Processing Framework

In [5], the concept of Model Processing is described as a set of tasks, namely creating a model and performing analysis techniques, that are put together in the form of a Model Processing Framework. In the following subsections we describe each of the MPF's components we used as a base for our solution. We explain their tasks and input/outputs and how they're connected but, in order to separate the conceptualization of our framework and the tool support (described in section 3.5), we have kept out all the implementation aspects.

The **Model Editor** supports the creation and edition of models, with proper lexicon and syntactic validation (according to an Enterprise Architecture meta-model). These meta-models must encompass the four concepts of the Lightweight Enterprise Architecture Notation: agent, action, resource and rule. The creation and edition of models must be fully supported by the tool, as to not force the user to understand lower-level programming concepts.

As we explained in section 3.1, evaluation occurs by observing agents while they're performing actions. Therefore, object representation is necessary so that individual characteristics can be modeled. Objects as "things" must be reified within the tool, so that Object diagrams or similar representation of object-level schemas are possible.

The **Model Converter**'s purpose is to act like a pre-processor, transforming the modelled architecture into a format that can be consumed by the Model Analyser. The converter, loaded with the Enterprise Architecture meta-model, has the capacity to navigate a model and generate the code needed to capture the values of monitoring points defined by the modeller.

The core of this MPF is the **Model Analyser**. It is responsible for performing the evaluation of the Informations Systems based on the enterprise architecture model (along with manifestations and measurement specifications) that were defined in the Model Editor. The Model-Level Analyser acts upon metrics that are

defined within the meta-model (don't require user intervention) and, on the other hand, the Object Level Analyser uses continuously stored data and user-defined metrics to judge instance behaviour.

The final stage in the MPF cycle, executed by the **Results Converter**, is the transportation of information produced by the Model Analyser back into the visual representation of the model, as in the Model Editor. The values (of data collected) are displayed in each of the objects' manifestation attributes and the generated warnings by the Model Analyser are displayed to the user.

3.5 Implementing the Model Processing Framework

An abstract framework for evaluating enterprise models was constructed and, to attest its viability, we describe in this section a possible implementation for the MPF, using an existing architecture meta-model and an UML tool.

To sum up our objective, the aim was to construct a model driven evaluation framework for the real-time manifestations of deployed enterprise architectures. For this implementation, an infrastructure comprised of a central repository and agents that capture (through its senses) certain dynamic attributes of another agent, is assumed to exist.

3.5.1 Extending the Meta-Model

The framework chosen was the Center of Organisational Design and Engineering Framework (CEOF) since it draws a clear line between the several layers of the Enterprise Architecture, and describes the entities that belong to each.

We have extended this framework at the Information System Architecture level, through its Block and Service elements since, in this architecture, the concept of agent as action-performer entity does not exist. Thinking back to our LEAN extension and Vasconcelos' [16] taxonomy, Blocks closely represent Agents and therefore we must define how (or through what) they manifest themselves in reality. For this meta-model, we changed the manifestation semantics slightly in order to better align out concepts with CEO's. The ODE loop [12] shows the concepts "Variables" and "Monitoring Points" as the meta-level inputs for Domain Monitoring. To better suite ODE's syntax we used the latter concept "Monitoring Points" to refer to the observation standpoint of a certain variable of an Agent's behaviour. Monitoring points are therefore the equivalent of the Manifestations, but add a perspective semantic to it. This characteristic is useful due to the nature of the Case Study monitoring platform, that relies on monitoring agents located near systems that observe their manifestations.

Looking back into the ODE loop, Domain Analysis is performed adding models (structure) and heuristics (evaluation) to the results of Domain Monitoring (observable data). To the heuristics we will call Metrics. Essentially the Metric concept represents an expectancy that is based on collected data from Monitoring Points.

In CEO Framework, the Service is a collection of operations made available by architectural blocks. We

therefore apply the metric concept to the Service element, so that we can evaluate actions, using manifestations from the actor, according to LEAN and our LEAN extension.

3.5.2 Instrumenting the MPF cycle

The tool selected was openArchitectureWare since, of all those reviewed in chapter 2, it presents the better support for the needed characteristics: graphical interface, UML2, UML Profiles, Code Generation, XMI import and export, OCL checking and for being Open Source.

The **Model Editor** requires easy-to-use, UML2 class and object diagram support as well as support for UML profiles. openArchitectureWare includes the Graphical Modelling Framework[1] (from the Eclipse Modelling Project) that provides this tool with a graphical environment for creating diagrams. We've recreated the Information System Architecture of the CEO framework on a UML Profile (using a Profile diagram in the graphical interface). We have also inserted our meta-model extension in this diagram, by creating two Stereotypes named *Monitoring Point* and *Metric* as described in section 3.5.1. Having the CEOF profile and its extension defined, it is now possible to create an UML model (using the Class Diagram), apply the profile, and model an architecture with the correct taxonomy. To model object instances in oAW, the eclipse UML2 implementation supplies the uml::InstanceSpecification element, which is a M1-level representation of the M0 object.

The next block in the MPF workflow, the **Model Converter**, plays the role of converting the UML model into a format than can be used by the Model Analyzer. A XMIReader component was used (so that other components can manipulate the UML model) along with the Generator component to perform code generation. The CEOF meta-model was recreated in Ruby using an Object Oriented approach so that, when the Generator is invoked, it creates two ruby source files: one for the model, and another for the instances, and each level is supported by the upper.

For the core of this MPF, the **Model Analyser**, openArchitectureWare supports OCL expressions and validations. These are used to calculate Service metrics by navigation InstanceSpecifications, accessing Monitoring Point values in Blocks, performing calculations and finally comparing the output with a threshold which warns the user. They are also used to perform checks at the model-level for malformedness, according to the meta-model.

Finally, the **Results Converter** transforms the manipulated model (now possessing values for the monitoring points) back into UML. This processing block also has the responsibility to guarantee that the end of the processing cycle leaves the openArchitectureWare in a state where the process can be run again. To be able to show the user the values used for calculating metrics, the model UML(.uml) file is rewritten, and special care is given so that in subsequent executions of the workflow only changes these Values.

[1] http://www.eclipse.org/modeling/gmf/

4 Case Study

In this chapter we explain how the solution was applied to a real-world situation. This thesis was developed in an enterprise context, therefore our testbed consisted of the enterprise PT-Comunicações and its technological infrastructure.

PT-C has been concerned over the years with measuring IT performance, availability and errors. For that purpose, the Pulso platform was developed to sustain mechanisms of near-realtime measuring, monitoring and storing basic performance indicators for major systems that support key business processes.

However, the evaluation procedures are still *ad-hoc* since there's no technical or conceptual framework that allows - in a simple, declarative and semantically robust way - to define what entities are being targetted and how to specify and calculate the relevant indicators for either Quality of Service, Quality of Protection or even Quality of Maintenance.

The EDS[2] division of PT-Comunicações is responsible for this monitoring platform and, as such, provided an environment that was both resourceful and accessible for us to experiment with our solution since we were given access to an infrastructure of monitoring, data collector agents attached to every relevant server, database and network link. By "relevant set" we mean the entities that are, at the time of this writing, critical to the business.

In this chapter we describe a "proof-of-concept" implementation of an existing scenario. We start by describing the Pulso platform in section 4.1, where we explain its architecture, its infrastructure of monitoring agents, and its current evaluation mechanisms. Our purpose is to demonstrate the steps needed to construct a cycle of the MPF using the implementation we proposed. For this, in section 4.2, we describe a fraction of PT's Call Center system (sufficient for an example) and an evaluation scenario using the CEOF metamodel and openArchitectureWare implementation.

4.1 Pulso Architecture

The Pulso monitoring framework [6] is comprised of system monitoring agents (e.g. Linux, Windows, HP-UX, Oracle) and network probes (client-server and server-server).

Agents are system-specific software whose job is to periodically capture basic behaviour readings of their hosts. For instance, Linux agents capture observable variables such as CPU usage, CPU load (for 1, 5 and 15 minutes) and memory usage. For databases, agents designed for Oracle SGBDs capture waiting times for query processing queues. Depending on the business demands, the agents are further developed to produce more observable variables.

Probes passively monitor network traffic between either servers and clients or between servers. They capture data regarding to bandwidth consumption and protocol-specific traffic (e.g. SQL transaction requests and errors, IIS communication).

[2] Eficiẽncia, Disponibilidade e Segurança de TI/SI

In the Pulso platform the definition of Enterprise entities (ranging from disk partitions to business processes) is constructed using a graph of "things" and the links between then. Therefore, there are no classes, no type hierarchy and no architectural layers. This approach provided some benefits due to the lack of constrains on the evaluation process, making it is faster to implement. However, by allowing any entity to be related to another and by providing full flexibility and no constraints, this solution has also been proven to be hard to maintain as systems grow in complexity. Since there are no *a priori* validations that can be performed on the Technology Infrastructure model, the structures evolve into large graphs, inherently difficult to debug and complex to change in the long term.

To store the data produced by agents and probes a centralized repository was created in the form of a database, where all events are dispatched after being sent to a central logging server. These events have a common header format that includes an identification of the monitored entity, what is observed attribute and the timestamp of the event. The header helps a dispatcher determine where to insert the event in a database where the timeseries for each event type are stored, for future analysis.

4.2 Call Center

As a case study, we chose to analyse a fraction of the Helpdesk Call Center that provides assistance to PT's internet, television and phone clients. The infrastructure that supports consists of:

- Interactive Voice Response - is an interactive technology that allows a computer to detect voice and keypad inputs. It is used to channel the client to the correct pool of call center agents by a telephone key driven menu;
- Automatic Call Distributor - is a device or system that distributes incoming calls to a specific group of terminals which are use by agents;
- Computer Telephony Integration - allows interactions on a telephone and a computer to be integrated or co-ordinated;
- Customer Relationship Management - manages client data such as personal information and account data. It is supported by Siebel software.

The typical flow of a client's call starts when the client dials the number and is picked up by the IVR. The client is asked to specify one of the pre-selected classes of problems and, after the IVR has the necessary data, it transfers the call to the ACD. The distributer selects an available agent, from the correct pool one agents, and the conversation can now begin.

4.2.1 Setting up openArchitectureWare

Our first step of the implementation was to create an openArchitectureWare project and properly configure it. This involves (1) creating an empty openArchitectureWare project and configuring it for UML2, (2) loading the CEOF profile and (3) creating an empty UML model and a UML Class diagram.

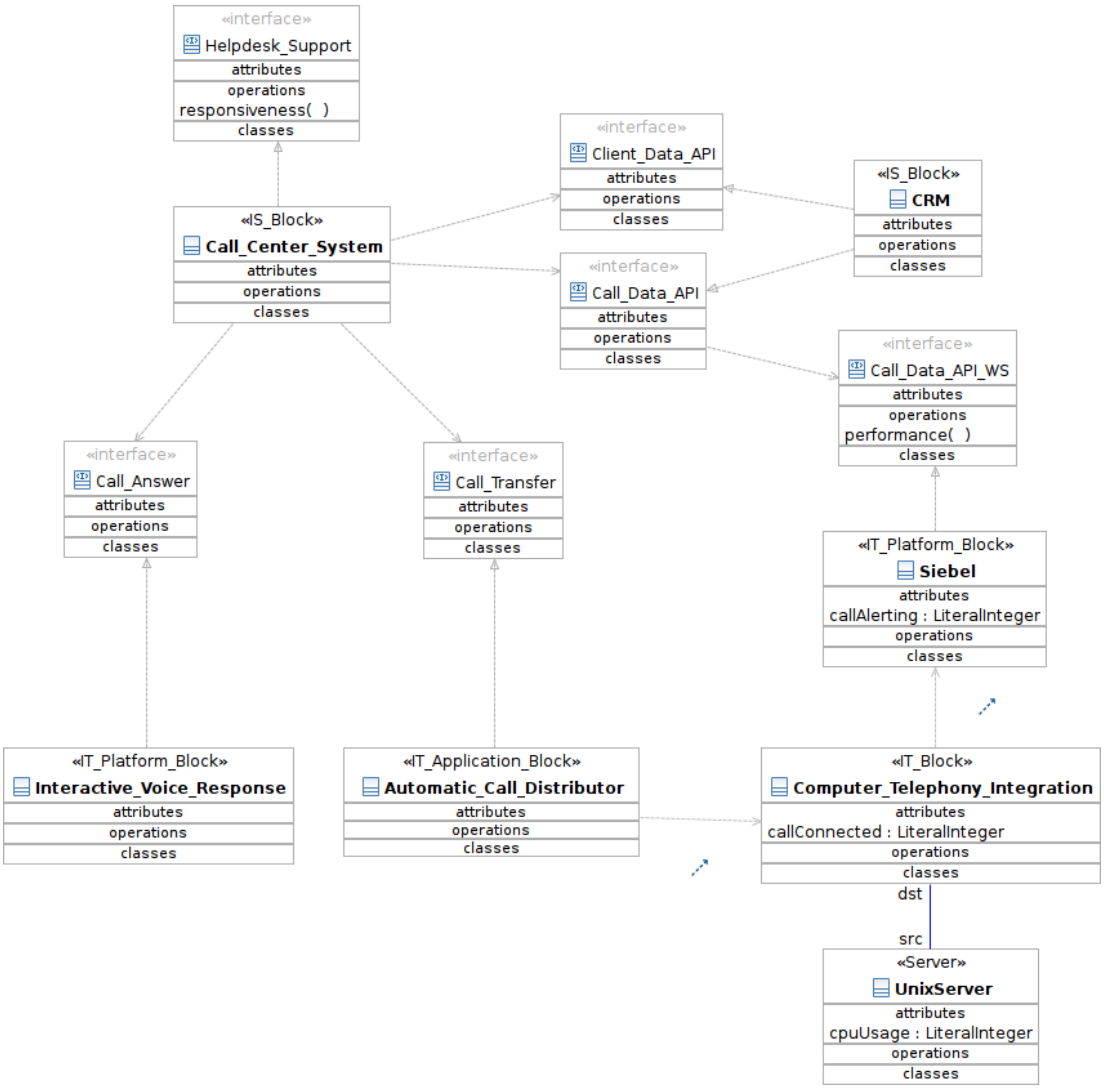

Figure 2: UML Class diagram for the ISA with added Monitoring Points and a Metric

Having set up the base project, we started modelling the Call Center into the oAW application. The creation of the Information System Architecture is essential to our modelling of the phenomena produced by these systems and so our next step was to describe it. While modelling this fraction of the ISA we only included the blocks and services we were able to identify from the existing documentation and therefore elements such as databases or even software components were possibly left out (the resulting Class diagram can be seen in the fig. 2).

To stereotype either classes (Blocks) or interfaces (Services), the context menu of that element provides an "Apply Stereotype" entry where only the appropriate stereotypes can be applied (e.g. Block stereotypes can't be applied to interfaces).

Since what we desire is to capture the dynamic behaviour of this architecture, we also need to construct the corresponding Object diagram. This diagram contains the instances of the Class diagram that exist and

will be the target of our evaluation.

To add the Object diagram onto our solution, we used the existing UML Class diagram where we added the representation of the objects through UML's InstanceSpecification primitive and "Usage" relationships (the resulting object diagram can be seen in the MPF execution below, fig. 3).

The next step of the configuration was to establish what we can observe on each of the Blocks and how do we wish to evaluate the services on which they are implemented. As observable variables of the existing blocks we have (1) "callAlerting" on the Computer Telephony Interaction IT Block, (2) "callConnected" on the Siebel Platform Bock and (3) "cpuUsage" on the UnixServer.

We chose to model the measurements as expectancies (that evaluate to true or false) for this fits better with the OCL paradigm and provides a way to warn the user of this MPF if an expectancy isn't met.

The *responsiveness* metric was created to calculate the difference between the two moments of the call alerting and the moment the operator picks up the call. For the purpose of this proof-of-concept, we assumed the last timestamp of each moment and the metric simply determines if the difference is greater than a certain number. Since timestamps' measurements don't have milliseconds and most of the systems are on automatic answering, we do not want them to be greater than 1 second. The *performance* metric is a simple threshold checker to see if the unix server's cpu is not being overused, degrading the system's performance. The following code listing shows how this performance and responsiveness metrics was defined, using oAW's implementation of OCL. The metrics are defined using the model-level elements and also the thresholds that define if the metric passes or fails according to the expectations for the instances.

Listing 1: Responsiveness and Performance Metrics

```
context InstanceSpecification if hasMetric(this, "responsiveness")
  WARNING "Last call took too long to pick up":
  (getValue(this, "Computer_Telephony_Integration", "callConnected") -
   getValue(this, "Siebel", "callAlerting")) < 1;

context InstanceSpecification if hasMetric(this, "performance")
  WARNING "Last call took too long to pick up":
  getValue(this, "UnixServer", "cpuUsage") < 95;
```

4.2.2 Resulting ISA and Execution

Adding monitoring points to Blocks is done by adding an attribute to a Block stereotyped Class and appying the CEOF::Monitoring Point. Similarly, adding operations to Services defines metrics, after applying the stereotype (of the same name). We applied three Monitoring Point and the two Metrics to the existing ISA (fig. 2).

Having our class and object diagrams, manifestations and metrics modelled, we were now able to execute the Model Processing Framework's workflow and observe its output. As seen in figure 3, at the end of the cycle, the instances that possess monitoring points now have values associated, and the Console View at the

bottom of the oAW screen has launched a warning since the last registered call took 1 second to be answered (the difference of timestamps between the two moments).

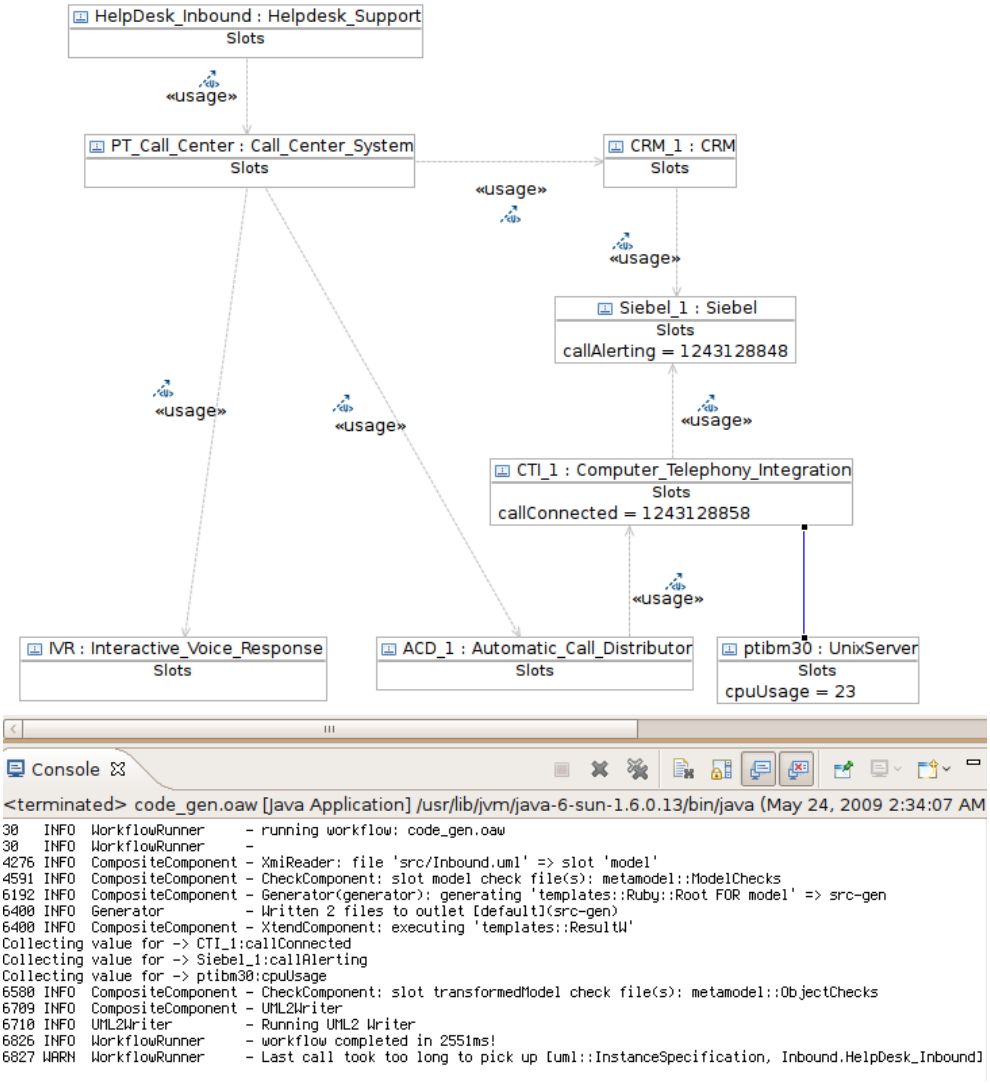

Figure 3: Results after a run of the MPF cycle.

5 Conclusions

The Information System Research is still in a very imature state [7]. Several contributions have started to emerge during the last 20 years and the interest on the topics such as Information System modelling, Enterprise Architectures and Organisational Engineering and Design are growing fast.

In this thesis we attempted to further develop the works of Gerald Khoury and André Vasconcelos by providing some insight on how to perform architecture-centric, ontologically supported, evaluation of Information System behaviour. We began by finding the appropriate mindset to approach this problem

through a higher level abstract analogy to our problem. We turned into the metaphorical view of societies as a superset of enterprise systems, and searched within this metaphor for guidelines on how to model a solution.

We encountered a parallel between the field of Phenomenology (that studies human and group experience and phenomena) and the acts of judging the behaviour of architectural components by transforming their observable variables.

To help modelers, analysts and every participant in the enterprise community, we proprosed an automated cycle of evaluation using the Enterprise Architecture as the standpoint and communication medium between all parties.

The Model Processing Framework was sucessfuly implemented using the openArchitectureWare tool, and finally we constructed an example taken from a real-world enterprise situation and we were able to measure a simple metric from observed manifestations.

5.1 Shortcomings and Future Work

Regarding the phenomenological aspect that we provided in this domain, we think that much more can be done to improve the holistic view on Enterprises by further understanding the key ideas of this philosophical domain. This requires, however, a less traditional engineering approach and willingness to nearly "decypher" some of the cornerstone books of this discipline.

The Model Processing Framework implementation has some limitations that can be overcome with a harder focus on software development:

- The cycle cannot be repeated automatically (the user has to press the "Run" button again). Investigation has already been done (also the community forum is very active) regarding this issue, and it can be solved with a Java Ant task that executes the workflow as a normal Java application;
- The helper functions that abstract the metric design from the Eclipse/Ecore model navigation can be further developed to provide a friendlier experience;
- The presentation of negatively evaluated metrics can be improved by, for example, marking the diagram elements with color codes;
- Not all meta-model checks of the CEO framework were implemented.

Finally, being "unified" an aspect of our ontology extension and conceptual MPF, we did not attempt to apply our solution to another architecture meta-model such as, for example, Archimate. We do, however, consider the solution to be perfectly compatible with Archimate, for it also has a well-defined notation, even though not based upon UML. openArchitectureWare is also prepared to handle such flexible notations as Javabeans-based models and XML-based models, so this should not present a major challenge.

References

[1] Ackoff, R. L. From data to wisdom. *Journal of Applied Systems Analysis 16* (1989), 3–9.

[2] André Vasconcelos, Pedro Sousa, J. T. Enterprise architecture analysis - an information system evaluation approach. *Enterprise Modelling and Information Systems Architectures 4*, 1 (2008).

[3] Aurrecoechea, C., Campbell, A. T., and Hauw, L. A survey of qos architectures. *Multimedia Systems Journal, Special Issue on QoS Architecture 6* (1996), 138–151.

[4] Filman, R. E. Achieving ilities, October 1999.

[5] Fomel, S. Object management group. uml profile for schedulability, performance, and time specification. *OMG Document* (2002), 02–03.

[6] José Alegria, Tiago Carvalho, R. R. Uma experiência open source para "tomar o pulso" e "ter pulso" sobre a função sistemas e tecnologias de informação.

[7] Khoury, G. R. *A unified approach to enterprise architecture modelling.* PhD thesis, University of Technology, Sydney. Faculty of Information Technology., 2007.

[8] Lankhorst, M. *Enterprise Architecture at Work: Modelling, Communication and Analysis.* Springer, December 2005.

[9] Laudon, J., and Laudon, K. *Management Information Systems: Managing the Digital Firm*, 10 ed. Prentice Hall, December 2006.

[10] López-Garay, H. Extending checkland's phenomenological approach to information systems. *Critical reflections on information systems: a systemic approach* (2003), 46–64.

[11] Magalhães, M. R., Sousa, P. M. M. V. A. d., and Tribolet, J. M. N. S. The role of business processes and entreprise architectures in the development of organizational self-awareness. *TECKNE - Revista de Estudos Politecnicos 6*, 9 (July 2008), 9–30.

[12] Matos, M. G. d. Organizational engineering: An overview of current perspectives. Master's thesis, Instituto Superior Técnico, july 2007.

[13] Mingers, J. Embodying information systems: the contribution of phenomenology. *Information and Organization 11*, 2 (2001), 103–128.

[14] Recker, J. C., and Niehaves, B. Epistemological perspectives on ontology-based theories for conceptual modeling.

[15] University, S. Stanford encyclopedia of philosophy.

[16] Vasconcelos, A. *Arquitecturas dos Sistemas de Informação: Representação e Avaliação.* PhD thesis, Instituto Superior Técnico, 2007.