Using Goal Oriented Requirements Analysis in the Design of Product-Service Artifacts

Javier Martínez Silva*, José Reinaldo Silva*

*Department of Mechatronics Engineering, University of São Paulo, São Paulo, Brazil

Emails: javsilva@usp.br, reinaldo@usp.br

Abstract— Intelligent manufacturing demands more research effort to the design of systems that fit new distributed architectures suitable to new business process and to new control architectures. In this work we address the problem of a distributed architecture service-oriented, PSA, focusing on its design process and mainly in the process of requirements analysis. We claim that goal-oriented methods could be used to reach a better performance, compared with traditional object-oriented methods.

Keywords— manufacturing service, goal-oriented requirements, KAOS, intelligent planning

1 Introduction

During the last few years, both practitioners and researchers works in automatic planning divided its interests in two main lines. On the one hand seeking for new planning and scheduling approaches using Artificial Intelligence formal methods. The other hand, looking for new methods to treat real problems and also take some feedback from already consolidated formal methods. Although both tendencies converge to improve the performance of planners, trying to optimize the search algorithms and the general solution, in the second line there is a clear need to propose a design process to deal with real and complex applications. Generally, such methods focus on the design process and specifically on the requirements analysis, aiming to improve the knowledge acquisition and domain modeling of planning and scheduling problems (Basbaum et al., 2013).

It is important, when dealing with real world problems, to have a design process. Such process is defined by a sequence of sub-process that can lead the designer to create a more adherent model following a model-driven approach, where requirements analysis could be done over a sound model. Therefore, it would be necessary to make a transference from a schema that represents requirements to another (formal) where it is possible to make property analysis and flow analysis.

Extracting a sample for the domain knowledge is a difficult task since it is done in the early phase of the process which is characterized by having an incomplete knowledge about the problem. Therefore, an obstacle in this stage is exactly to model incomplete knowledge. Moreover, it is not required to represent each single domain feature but just those concerning the planning problem and that affect the choice of actions to compose such plan. Therefore, requirements for the planning problem could be written in a formal description while the domain would require a representation using some ontology language, which make translations among different knowledge represen-
ments based on a goal-oriented method, KAOS (Lapouchnian, 2005) (Van Lamsweerde, 2009), still using Petri Nets as a framework for formal requirements analysis.

This new method is supported by ReKPlan (Requirement Engineering using KAOS for Planning Problems) framework, which currently is not one hundred percent ready to perform the KAOS/UML/GHENeSys/PDDL net translation, but it will be during the development of this project. The approach is coupled over PSA clustering architecture, which is introduced as a contribution in this work related AI Planning techniques in single manufacturing services.

This paper is organized as follows: Section 2 introduce the main PSA architecture issues. Section 3 presents aspects of Service-Oriented approach. Section 4 presents some definitions and conceptual notions of KAOS, a Goal Oriented Requirement Method (GORE) and some practical issues to use this method over real case of study. Finally, we analyze the process as a whole and add some concluding remarks on prospective contributions in further works.

2 A Product-Service Architecture

By the end of the 90’s a discussion emerged in the academy about the possibility of a paradigm shift where the product-oriented approach widely used would be replaced by services - which in some situations could also include the use of artifacts.

Today the idea of servicitization evolved to define the concept of Product–Service System (PSS) (Sakao, Oluand and Matzen, 2009). Companies reach a different level of success by investing in product-services selling (Garetti et al., 2012), specially in the computer business. A currently accepted definition says that PSS stand by “Product(s) and service(s) combined in a system to deliver required user functionality in a way that reduces its impact on the environment” (Baines, 2007).

Research works mention several existing projects towards a modeling and simulating method to PSA. More recently, a proposal emerged that relates the dynamic behavior of such systems with knowledge information captured in the early phase of design - functional requirements, non-functional requirements, domain knowledge, etc. (Sakao, Shimomura, Sundin and Comstock, 2009).

However, there is not a consensus - neither in academy or in the industry - on how to proceed a formal development for PSS systems. For instance, it is a challenge on how to design a new enterprise service architecture, or on how to choose a proper design approach. The same doubt can be spread to the design of PSS business processes and consequently, on how to make a coupling between its business process and system design requirements.

Simultaneously to this evolution, collaborative e-Work has been developed over the past decade as a framework for design, engineering, and control of next-generation manufacturing systems, providing collaborative, computer-supported and communication-enabled productive activities, fundamentally transforming the ability of highly distributed enterprises to improve e-Production and e-Service (Nof, 2006) (Nof, 2007). e-Work theory and models provide foundations for augmenting abilities of organizations and all agents (human, computer, robot, etc.) to interact and collaborate through certain protocols, in order to accomplish their goals.

An approach that seeks such a good design based on the new framework concentrates on a combination of available and recent design methods and tools (Dutra et al., 2013) (Silva, 2014).

Manufacturing services are normally arranged as a cluster of different industrial branches. For instance, the automotive industry can collaborate by exchanging information, requirements, products, and services, in mutual interest, to converge to a more generic business process that deals with individual or collective transportation (Silva and Nof, 2015). As depicted in Fig.1, we call each industrial branch a component, where each component may belong to different corporations - in charge of a service resulting in some product, tuple base of PSA architecture.

In such environments, e-Work can be a good way to interact and collaborate within a cluster of such companies. Normally, the interacting companies form a configuration that looks like an object-oriented cluster: Each component keeps all requirement and design information to itself, while shares only the information strictly necessary to keep the cluster working well. Collaboration is effective but strict, which is considered normal in a competitive market, especially one that is eager for competitive innovation.

More recently, in (Tao et al., 2012) (Zhang et al., 2014) introduced a new configuration where components define agents interacting with a new entity which is a cluster itself, materialized in a cloud system. Ownership is still a key issue and
so far it is not clear how the main business process relates to the cloud or grid system. However, the agility of this new arrangement is already clear, which also facilitates resource sharing, and demand capacity sharing (Yoon and Nof, 2011) (Moghaddam and Nof, 2014) (Seok and Nof, 2013) (Seok and Nof, 2014). This new clouding approach is represented in Fig.1.

However, independently of the formal model chosen, it is also clear that, for real systems it will be necessary to have tools and frameworks to help product-service design. itSIMPLE (Integrated Tool and Software Interface to Model Planning Environment) could be a good alternative.

3 AI for design of services in Product Service Architecture

itSIMPLE is an open-source, Java-based tool that has been applied to several real planning applications since 2005(Vaquero et al., 2005). The main initial purpose is to reduce the gap between real planning applications which are seldom represented directly in PDDL and state-of-the-art AI planners.

Treating a large number of problems, increases the importance of the early stages of modeling process, which is characterized by the transformation of informal requirements to in a mix of UML (Unified Modeling Language) and OCL (Object Constraint Language) notation and then to Petri Net for later analysis of this requirements (See Fig.2). At the end, the problem instance is transformed to PDDL (Planning Domain Definition Language), which is understandable by various planning software. Currently, itSIMPLE involve fourteen of the main different planers (Vaquero et al., 2013).

Figura 2: itSIMPLE key mission: Convert requirements in modeled planning (scheduling) problem using knowledge techniques

itSIMPLE is being improved to manage more efficiently the first knowledge acquisition. The key challenge on this improvement is addressed to express the main features of the planning and scheduling problem as requirements using formal or semi-formal methods - even for real complex problem- in an attempt to bridge the gap from linguistic expression to formal modeling techniques: a schematic representation for the analysis, verification and validation leading from initial to more consistent model, which would be transferred to the automatic planners at a later stage.

The tool upgrade include techniques for modeling and designing service-oriented process, that have the similar properties of plans: a initial state, a final goal (the service delivery), and a set of applicable actions or sub-processes leading until this final goal.

Another improvement which is being included is related with model analysis using high-level Timed Petri Net. These new functionality will enable it to handle realistic and distributed/decentralized scheduling problems, considered to be among the most demanding problems in the new product-service architecture.

Due to the importance of early stages of the design process, we propose the use of KAOS method for requirements elicitation, which results in a first formal specification with basis in temporal logic formalism.

4 KAOS method for the early design of New Manufacturing Architecture

To provide a direct definition of KAOS (Keep All Objectives Satisfied) method, we should first emphasize the notion of goal-driven (Van Lamsweerde, 2004). A goal is a prescriptive state declaring the purpose of some (existing or to-be) systems whose satisfaction generally arise from the collaboration between agents with some responsibility over the system. Therefore, goals should drive the requirements elicitation process resulting in domain-specific requirements, avoiding a difficult balance between functional and no-functional requirements.

In this context, GORE (Goal Oriented Requirements Engineering) methods emerged to justify the “why”, “what” and “who” dimensions, and to ensure that requirements meet the system objectives - including those of safety and security (Van Lamsweerde, 2009). GORE methodology, even being a recent approach, has justified its use in comparison with another mechanisms such as UML, which could be inefficient in the initial stages of a design process, since the semi-formal diagramatic approach could turn requirements analysis very difficult.

KAOS is a GORE method which provides mechanisms based on formal logic LTL and graphical modeling representations for requirements of a problem in terms of goals. These goals express stakeholders needs and viewpoints of, in conflict or not, but eventually manageable (similar to other representation of requirements). Thus, KAOS is described as multi-paradigm framework that com-
bines different levels of reasoning: semi-formal, for modeling and structuring goals; and formal, based in the linear time logic formalism (Van Lamsweerde, 2009).

KAOS ontology incorporates main concepts for a distributed system whose components may change from state to state: objects can be entities, relationships or events. In order to specify the dynamic behavior, system operations are defined as input-output relations over objects, where state/transition relations are declared by signatures over objects - with pre, post, and trigger conditions. Agents are active components such as humans, devices, legacy items or eventually software components, and must play some role to achieve a goal satisfaction.

Goals are graphically represented by hierarchies of AND/OR refinement abstractions. A refinement ends when every sub-goals become realizable by some individual agent (people, other software devices, robots, etc), which means that the goal must be expressible in terms of conditions that are monitored and controlled by the agent. Parent goals are obtained by abstraction whereas child goals are obtained by refinement. Basically, a refinement is a process that results in a tree where leaf goals are assignable to single system agents.

An AND-refinement defines a conjunction of parent conditions which are validated if its own children is satisfied (graphically represented by black solid filled circle), while an OR-refinement is a disjunction of sub-goals where only the achievement at least one is enough to satisfy a parent goal (graphically represented by no fill circle).

Leafs are defined as requirements or assumptions depending whether they are assigned to the software-to-be or to an environment agent, respectively. Fig.3 shows how graphically are represented goals, requirements and assumptions in KAOS. All of them use parallelograms, but requirement borders are drawn in bold line; assumptions are filled yellow, and agents are represented by hexagons.

The Fig.4 shows an example of a partial model for a goal “Students registered in time when the period is started” where Academic Committee, Web System Manager and Secretary are agents involved.

Goals can be classified according to its behavior, covering a maximal set of intended behaviors in a declarative and implicit policy. In this scenario, a behavior is defined as a sequence of system/transition. Thus, a general behavior violates the goal specification if it does not belong to the set prescribed in the requirements.

Formally a behavioral goal can be defined using temporal logic sentences:

\[ C \Rightarrow \Theta T \]

Table 3: Main elements of goal model

Figura 3: Main elements of goal model

Figura 4: A partial goal model applying AND/OR refinements

where \( C \) is the current condition, \( T \) is the target condition and \( \Theta \) is one of the LTL operators:

- \( \square T \): In the next state \( T \) is satisfied.
- \( \Diamond T \): \( T \) is satisfied eventually in the future.
- \( \Box T \): \( T \) is satisfied always in the future.

Time stamps (\( d \)) are used for quantifying this operators. Therefore, \( \Diamond_d T \), means that, from the current state, \( T \) is eventually satisfied in the future before deadline \( d \); and \( \Box_d T \), means that \( T \) is always satisfied in the future up to deadline \( d \).

Goals in the Fig.4 are preceded by \( Ach \) term, which is derived from “Achievement”. There is also a term meaning “Maintain” which compose the close set of goal categories.

5 A simple manufacturing service architecture: a car assembling problem

The ROADEF is an important system competition provided by RENAULT corporation (Nguyen, 2003) where are published a variety of interesting
problems. The car sequencing is one of these problems where customers orders are sent to the manufacturing facility in real-time in a typical e-work process. The knowledge domain (delivery dates, resources capabilities, efficiency level, quality of the final product) is composed of goals to be maintained.

The requirements issue focus on three main services provided by the factory: body shop, paint and assembly where the primary goal is to optimize the paint solvent used to wash spray guns each time a color is changed.

The Fig.5 shows the first proposal of a car assembling problem over architecture PSA.

We can use the itSIMPLE approach to model the domain knowledge including all relations and dependencies in a car assembling factory. RekPlan (Requirements Eng. and KAOS for Planning) framework will provide a model based on abstract knowledge model balancing functional and non-functional requirements. Besides being abstract, this model is enough to establish all relationships between the assembling line and its context. Sub-goals are formulated as consequence of AND-refinement, while sub-objectives are generated by OR-refinement were only one can turn to be a feasible alternative (Martinez and Silva, 2015).

Returning to itSIMPLE a good operational plan of activities for the painting problem could be provided - as for all other manufacturing activities, completing a PSS.

6 Conclusions

The introduction of KAOS as a requirement engineering approach can bring some light to the design of a Product-Service Architecture. Based on the results we can claim that goal-driven methods - specially if object-oriented - can improve the requirements analysis for manufacturing services. Responsibility analysis and agent assignment suits better the design of service systems producing better results than using UML (Vaquero et al., 2013) or a similar object-oriented approach.

From each service, structure objects are derived forming a clustering of object which linked with goals represent all outstanding knowledge of PSA.

Future work point to an automatic process to transfer the results from ReKPlan to the language PDDL which can be read by itSIMPLE and sent to the automatic planners (so far this process is made manually). However to better fit a design process for PSS scheduling and real time systems should be covered. To fit that demand a more substantial analysis should be performed using model checking and Timed Petri Nets (Silva, 2014) (Silva and Nof, 2015) and a new version of itSIMPLE (the itSIMPLE-S) using a interface language with the planners that includes time: RDDL. Naturally, a new set of planners/schedulers must be included.

Acknowledgements

We thanks CAPES agency for partially financing one of the authors of this work.

Referências
