Towards a Framework for Supporting the Negotiation between Global and Local Business Requirements

Paolo Traverso[1] Marco Pistore[2] Marco Roveri[1] Annapaola Marconi[1]
Raman Kazhamiakin[2] Pierluigi Lucchese[1] Paolo Busetta[1] Piergiorgio Bertoli[1]

[1] *ITC-irst, Via Sommarive 18, I-38050, Trento, Italy*
[2] *Department of Information and Communication Technology, University of Trento, Via Sommarive 14, I-38050, Trento, Italy*
{*traverso,roveri,marconi,lucchese,busetta,bertoli*}*@itc.it* {*pistore,raman*}*@dit.unitn.it*

ABSTRACT

The development of service oriented applications very often needs to address the problem of satisfying two conflicting kinds of business needs: *global business requirements*, i.e., the regulations that dictate the rules of engagement between different organizations, and *local business requirements*, i.e., the rules local to each involved partner which derive from its internal business needs. In this paper, we propose a development process where both global and local service requirements, as well as their behaviors, are incrementally agreed among partners and built through negotiation steps. The development process is supported by the explicit definition of both global and local requirements at different levels of abstraction. We express requirements in a language with a clear semantics, and which allows for explicit links to executable business processes, e.g., written in BPEL4WS. This development process opens up the possibility to adopt a variety of supporting techniques. In particular, automated verification is used to detect design or implementation problems. Automated synthesis of executable business processes allows for a speed up in the development process and reduces development effort. Finally, execution monitoring is able to detect run-time problems with respect to specified requirements.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques; D.2.1 [**Software Engineering**]: Requirements/Specifications; D.2.4 [**Software Engineering**]: Software/Program Verification; I.2.2 [**Artificial Intelligence**]: Automatic Programming; I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms

Design, Languages, Verification

Keywords

Service composition models and languages, Requirements for service-oriented processes

1. INTRODUCTION

In several application domains, service-oriented computing should provide a universal basis for the integration of business processes that are distributed across different entities, e.g., different organizations or companies. In these domains, different organizations must interact and cooperate according to global, shared requirements. At the same time, each organization has its own internal business needs, which are specific to the business it carries out. As a consequence, in these domains, two opposite and often conflicting kinds of business needs have to be taken into account. From one side, the *global business rules*, i.e., the regulations that dictate the rules of engagement between different organizations. From the other side, the *local business rules*, i.e., the rules local to each involved partner and deriving from its own internal business needs.

Several applications have this characteristic. This is the case, for instance, of several e-government applications involving different administrative offices or departments, where global rules derive from national or regional requirements and norms, while local rules derive from each office's responsibilities and internal organization. Another example are business coalitions or market-places, where different companies agree to obey to common market regulations, but still pursue their own distinct profit and interest.

In most of the cases, it is rather natural that global and local business rules have opposite goals and tend to conflict. For instance, a national law may require maximum transparency from a government office towards the citizen, e.g., the citizen should have the possibility to inquire at any time the status of any on-going procedure he is involved in. However, an administrative office may not like to be slowed down in its internal procedures with too many external interactions. A common market regulation may require that an offer evaluation is proposed to a customer after all partners' offers are available, while each vendor's need is to get to know as soon as possible whether the client will buy the product or not.

Dealing with the conflicts between global and local business rules, both valid and well motivated from the two different points of view, is what makes this kind of applications difficult to develop, and what makes them substantially different

from traditional fully centralized applications, where an authority dictates the rules of the game, and fully distributed systems, where each actor has not to deal with general regulations and norms. As a consequence, the development process can hardly be carried out according to classical software development methodologies, which is not able to take into account both the global and the local rules and their natural conflicts. Moreover, software engineering tools that support the life-cycle of distributed business processes should be rethought to support the development process that takes into account both the global and local business needs.

Within Astro, an Italian national project which aims at the study and application of service-oriented computing techniques, we are defining a novel development methodology and the supporting tools necessary to face the challenges outlined above. This paper describes this novel approach.

The central idea is that conflicting global and local business rules should be negotiated within the development process. More precisely, the proposed development process interleaves the phases of specification of both global and local rules with *phases of negotiation between global and local needs*. As a consequence, the *choreography*, i.e., the global view of how different partners interact, the *orchestration*, i.e., the description of how one partner interacts with (some of) the other partners, and the internal business process of each partner, are *incrementally built through negotiation steps*, thus emerging in a commonly agreed choreography and orchestration that, by obeying to global laws and norms, mediates among global and local needs.

To implement this development process based on negotiation, we propose a conceptual framework where the distinction between global and local business rules is explicit in the different phases of the development process, and at different levels of abstraction. We specify global and local business rules both at the level of *strategic requirements*, i.e., business goals and motivations, and at the level of *procedural requirements*, i.e., specifications on how a business should be carried out. The "transparency towards citizens" and the "internal administrative office efficiency" are two examples of, resp., global and local strategic requirements, while a process that does or does not allow for interaction at each step with the citizen is, resp., a global or local, procedural requirements. The proposed methodology also takes into account that not all the local rules of a given partner can be made visible to the other partners. For instance, a customer company may decide to keep confidential its internal business rules on how offers to customers are prepared, in order to keep a competitive advantage.

In this requirements driven development process, global and (external and internal) strategic and procedural requirements are stated with a precise notation and with a clear semantics, and are explicitly linked to the detail design and implementation of business process, e.g., written in standard business process modeling and execution languages, like (abstract or executable) BPEL4WS [1]. This opens up the possibility to provide tools that support the process based on negotiation during the development cycle: *verification tools* that detect specification, design or implementation problems, e.g., the fact the negotiation process leads to a choreography and/or orchestration that actually does not satisfy some global or local rule; *synthesis tools* that suggest solutions, like a business process design or implementation, to speed up the development process and reduce development effort; *monitoring tools*, i.e., tools that monitor the execution of a process to detect run-time problems w.r.t. requirements.

The described approach has been developed guided by a real application domain, investigated inside the Astro project. It consists of service-oriented applications for the public administration. In this domain, procedures involve several different administrative offices, which must follow the strict National and Regional laws and global policies concerning these procedures, but which should also preserve their own autonomy in order to deal with other tasks related to other procedures and to obey to its own internal requirements.

The paper is structured as follows. In Section 2, we introduce the application domain that will be used to explain our approach all along the paper. In Section 3, we describe the proposed development process supporting explicit negotiation phases. In Section 4, we explain how global and local rules can be described with a precise language for requirements specifications, while in Sections 5 and 6 we discuss how all of this opens up the possibility to construct tools that support the development process. We provide a discussion of related works and some concluding remarks in Section 7.

2. THE CASE: PUBLIC ENVIRONMENTAL AGENCY SYSTEM

The Environmental Protection Agency (EPA) is a local agency which deals with a wide range of environmental matters including protecting air, water and soil quality, managing waste, preventing or controlling pollution and promoting sustainable industry. To address these issues EPA has to deal with complex administrative procedures distributed among various actors (administrative offices as well as citizen and industries) and regulated by law: European, National, and Local norms contribute to specialize the same procedure adding new constraints, new actors and goals. Norms can be seen as a collection of goals and activities delegated to specific actors; moreover they specify constraints and obligations concerning for instance minimal and maximal durations of specific steps or of the overall procedure. The definition of a new procedure in the domain of Environmental Protection is a costly and time consuming task, that has to take into account constraints deriving both from norms and from the internal organizational structure of the actors involved in the procedure.

In this paper, we consider a specific licensing procedure for the establishment and operation of a Waste Disposal or Recycling Facility: A citizen or a company submits an application to obtain the license for its waste disposal or recycling facility (incinerator, private landfill,...); the local government, involving various agencies and experts, evaluates the proposal and authorizes it, if it complies with high standards dictated by norms. We will assume that the EPA wants to automate this procedure using web services interfaces, as they offer a platform independent approach for integrating applications.

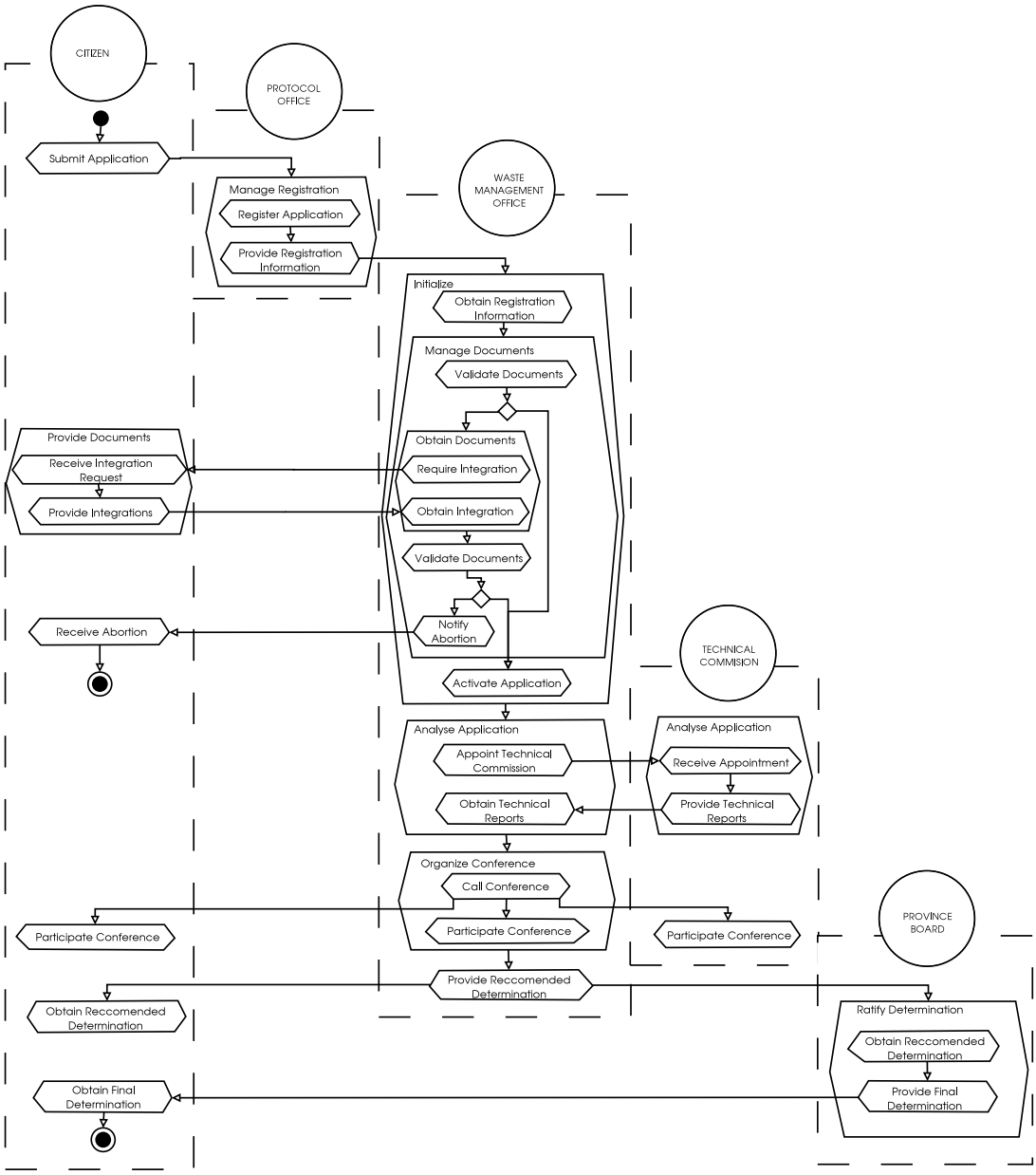


Figure 1: Activity Diagram for the Waste Facility License Procedure.

According to the classical approach, EPA assigns the responsibility of mapping this procedure to a business analyst. Starting from specific regulations and norms ("D.C.I. 27 luglio 1984"; "L.R. 13 aprile 1995 n.59"; "D.Lgs. 5 febbraio 1997") and interacting with the various actors involved, the business analyst models the procedure with the activity diagram depicted in Fig. 1. In this diagram, nested hexagon are used to describe the tasks and sub-tasks assigned to the different parties involved. Any citizen proposing to establish or operate a facility for solid wastes disposal has to apply for a certificate of designation; the application must be accompanied by all documents specified by the specific norms (e.g. an engineering design and operations report). The application is registered by the Protocol Office (PO) and then it is reviewed by the Waste Management Office (WMO) to determine whether the submitted documents are complete. If necessary, the WMO can ask the citizen to provide additional information or clarifications in order to complete the documentation. The validation of the documents must complete within 30 days from the registration of the application. The WMO is then in charge of appointing the Technical Commission (TC), which is composed of various consultants and directors of public agencies (e.g. Sanitary Agency, Water Quality Control Agency, Soil Water and Plant Testing Laboratories, Environment Engineers,...). Each member of the TC has to produce a technical report and send it to the WMO which is responsible to set the Conference and to notify all the participants (TC members, citizen, WMO's responsible for the application,...). The aim of the Conference is to determine whether the facility complies with the norms,

taking into account the submitted information and all the technical reports of the TC members. After the conclusion of the Conference the WMO will be in charge of producing the recommended determination and send it to the Province Board (PB) and to the citizen within 90 days from the Conference Day. The PB will evaluate the recommendations, draft the final determination and finally notify the citizen. Each application for a solid waste disposal site or facility should complete within 150 days from the PO registration. This global process defined by the business analyst is used as a blueprint for the design and implementation of the requested software components. In particular, starting from this global view, each actor involved defines (or adapt, if already existing) its internal processes and implements the web services necessary to carry out its part of the procedure.

The classical development approach outlined above is strongly based on a centralized, "authoritative" design that does not fit the requirements of distributed business processes. In particular, it does not take into account that the process developed by the analyst will have a high probability of being in conflict with the actor's internal requirements and constraints. A critical point is, for instance, the interaction between Citizen and WMO in order to complete the submitted documents. The norms regulating the procedure only require that the validation of the documents should terminate within 30 days. The Citizen would prefer to be able to submit new documentation incrementally within the 30 days, until the validation is successful. On the other hand, this iterative submission of documents would affect the efficiency of the WMO, since the scheduling of its work would depend on the Citizen. According to Fig. 1, the analyst has addressed this conflict by allowing the citizen to submit further information only once. However, from the diagram it is impossible to judge whether this is an acceptable compromise between these conflicting requirements. The negotiation-based development approach discussed in the next section addresses this kind of problems.

3. A DEVELOPMENT PROCESS BASED ON NEGOTIATION

The development process that we are designing is based on two principles: it is *requirements driven* and it is based on the *dichotomy between the choreography and orchestration* in the development of service oriented applications (see also Fig. 2). On the former principle we remark that a clear model of the conflicting requirements is necessary for being able to mediate among them. More precisely, we need to represent requirements at two different levels of abstraction: at a *strategic* level, for representing business goals and motivations, and at a *procedural* level, for describing how a business should be carried out. The activity diagram in Fig. 1 can be seen as a description of the procedural requirements, since it describes the way the procedure should be carried out. Strategic requirements include, e.g., the fact that the Citizen expects the WMO to be collaborative, while the WMO has the goal of reducing interactions. In Section 4 we will define suitable notations for representing them.

The terms orchestration and choreography are often used to refer to the two key aspects of service oriented applications [13]. In *orchestration*, the application is considered from the perspective of one of the business parties. The focus is on the interaction that the party under consideration performs with internal and external web services in order to carry out its tasks inside the procedure. Orchestration is usually private to the business party, since it contains reserved information on the specific way a given process is carried out. *Choreography*, on the other hand, describes the interactions for a global, neutral perspective, in terms of valid conversations or protocols among the different parties. Choreography is usually public, since it defines the common rules for a valid composition of the distributed business processes in the business domain. In our process, we exploit the dichotomy between choreography and orchestration at all levels of the development. We will have both choreographic and orchestration descriptions of strategic requirements, of procedural requirements, and of the implementation based on web services.

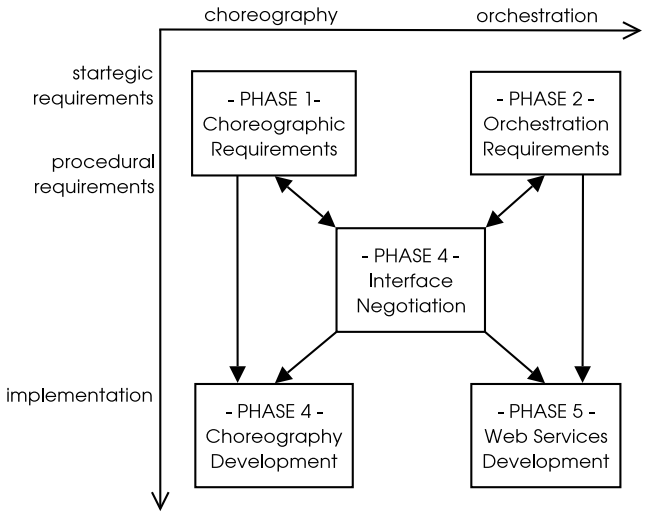


Figure 2: The Proposed Development Process.

The process we have been defining consists of five different phases (see Fig. 3). Taking into account the two principles just described, four phases correspond to the requirements analysis and to the implementation, done both from a choreographic and from an orchestration point of view. The fifth phase consists of the interface negotiation. This is the central phase of the whole process and plays the role of bridging between choreography and orchestration as well as between requirements and implementation. During this phase, the "choreographic" analyst responsible of the procedure and the "orchestration" analysts representing the different partners negotiate the design of the distributed application to be developed, mediating among conflicting goals. This negotiation phase terminates (and development starts) when an agreement has been reached on the services every partner should provide.

4. GLOBAL AND LOCAL REQUIREMENTS SPECIFICATION

Requirements play a fundamental role in the development process discussed in the previous section. Therefore, it is important to adopt flexible notations and methodologies for their specification. Activity diagrams like the one in Fig. 1

Phase 1 – Choreographic requirements

Objective:	Define the requirements for the management of the procedure.
Responsible:	Business analyst in charge of the new procedure.
Input:	Description of the procedure (laws and regulations; discussions with the experts...). Definitions of the existing services that can be exploited in the procedure. Legacy systems that the procedure should reuse (e.g., existing centralized information systems).
Output:	Requirements specification document, covering a strategic (actors involved with their goals, responsibilities, mutual dependencies...) and a procedural (actors' tasks, control and data flows among actors and tasks...) description of the choreographic requirements for the new procedure.

Phase 2 – Actor's orchestration requirements

Objective:	Define the actor's requirements on the services it can provide to support the new procedure.
Responsible:	Business analyst of the specific actor.
Input:	Description of the roles and responsibilities of the actor inside the procedure (laws and regulations; discussions with the experts...). Internal requirements of the actor (i.e., business objectives, internal procedures and organization...). Definitions of actor's services and of other software that can be reused in the new procedure.
Output:	Requirements specification document, covering a strategic (actor's goals and responsibilities; assume/guarantee relations with external actors...) and a procedural (actor's tasks and task decompositions, internal business processes,...) description of the orchestration requirements of the specific actor inside the new procedure.

Phase 3 – Interface negotiation

Objective:	Define the interfaces of the web services provided by the different actors.
Responsible:	Board of the analysts responsible of Phases 1 and 2.
Input:	Requirements specification documents produced in Phases 1 and 2.
Output:	Definition of the web services provided by the different actors (and of the centralized systems) that permit the implementation of the procedure. The services are defined in terms of their interfaces (e.g., in WSDL), of the protocol for interacting with them (e.g., in business process specification languages like BPEL4WS), and, at the strategic level, of the tasks that the services are supposed to perform and the assumptions for their correct behavior.

Phase 4 – Development of the choreography

Objective:	Development of the centralized software systems (e.g., centralized information systems, wrappers for legacy systems...) necessary to support the new procedure.
Responsible:	Analyst / system architect responsible of the new procedure.
Input:	Definition of the choreographic requirements (Phase 1). Definition of the interfaces that the choreographic component should provide (Phase 3).
Output:	Detailed design and implementation of the choreographic system.

Phase 5 – Development of actor's web services

Objective:	Development of the web services of a specific actor.
Responsible:	Analyst / system architect responsible of the actor's software systems.
Input:	Definition of the orchestration requirements of the actor (Phase 2). Definition of the interfaces of the services that the actor should provide (Phase 3).
Output:	Detailed design and implementation of the actor's web services (or adaptation of the existing services and other software components).

Figure 3: The Proposed Development Process Phase-by-Phase.

are fine for representing the procedural requirements, but they need to be completed with a description of the strategic requirements. We exploit the Tropos framework to this purpose. Tropos is a framework for the requirements-driven, agent-oriented development of software [2]. It is based on the premise that during requirements analysis it is important to understand and model the strategic aspects underlying the organizational setting within which the software system will eventually function. By understanding these strategic aspects one can better identify the motivations for the software system and the role that it will play inside the organizational setting. In previous works [5, 16] we have shown how Tropos can be adapted to represent the requirements of service-oriented applications.

Fig. 4 is an example of a Tropos diagram that provides high-level choreographic representation of the requirements of our case study. It describes the actors (circles) involved in the considered procedure with their strategic goals (the ovals attached to the actors). For instance in the diagram we

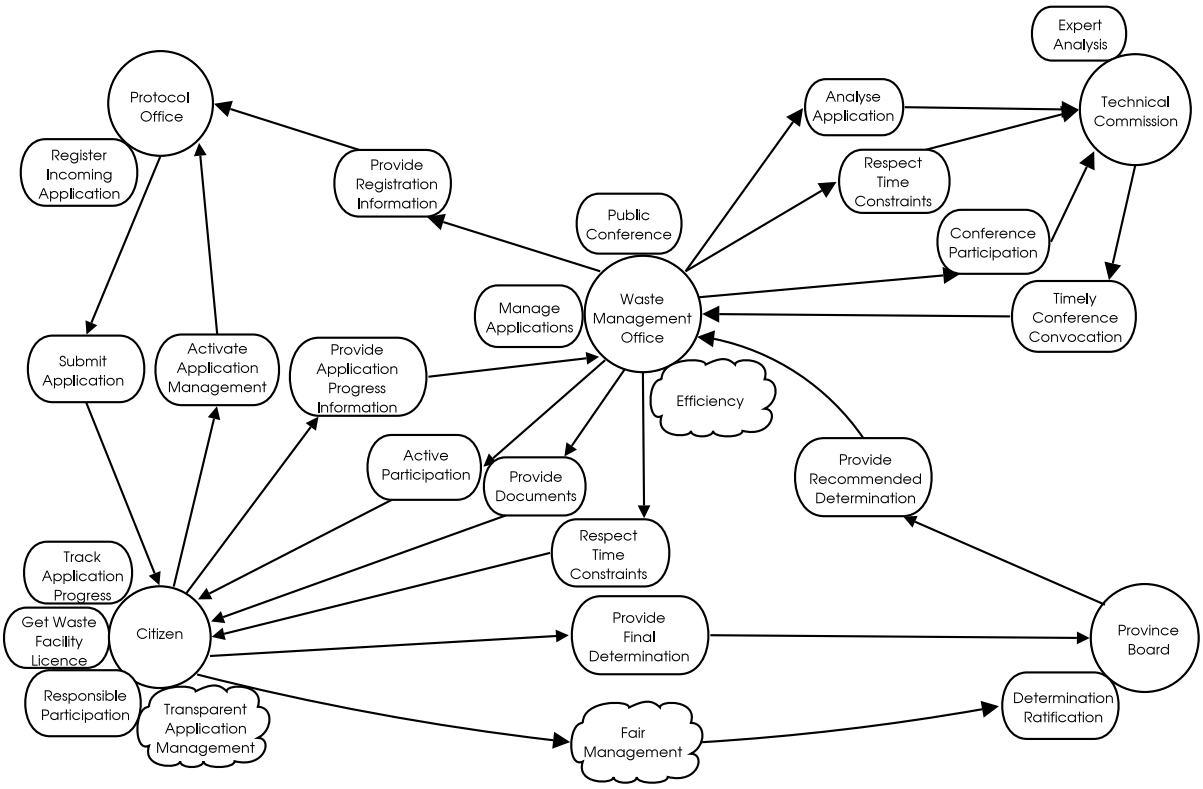


Figure 4: The Choreographic Diagram of Strategic Requirements.

have the Citizen that aims to obtain a waste facility license which is represented with the goal GetWasteFacilityLicence); the WasteManagementOffice that aims to handle with the several applications for getting a license (goal ManageApplication). The Tropos diagram also describes the *interactions* and *contracts* among the different parties. These interactions are represented at a strategic level by means of *dependencies* (the ovals linked to two different actors) that describe intent/offer matchings among actors. For instance the fact that the Citizen depends on the ProtocolOffice for the activation of the application to obtain a waste facility license is represented with the goal dependency ActivateApplicationManagement. Besides goals and dependencies, Tropos permits to represent so called soft-goals and soft-dependencies (clouds). These represent non-functional requirements that will have an impact on how the procedure will be implemented, but whose achievement cannot be defined precisely in terms of clear cut properties (for instance, the appreciation is subjective, or the fulfillment of the requirement can occur only to a given extent). The goal of the Citizen of having a "transparent application management", or the dependency of having a "fair evaluation" from the Province Board are examples of these "soft" requirements.

It has to be noticed that in Fig. 4 and in Fig. 1 we have represented separately the strategic and procedural description of the *choreographic* requirements. However, these two diagrams are interconnected in the actual model of choreographic requirements. Indeed, each activity is linked with the strategic requirements that motivate its presence in the model and that define its expected behavior.

An example of a linked representation of strategic and procedural requirements, is provided in Fig. 5 from the local, "orchestration" point of view of the WasteManagementOffice. This diagram represents not only the global goals of the WMO already represented in Fig. 4 (shaded in the figure), but also its private goals representing the internal needs, requirements, and constraints of the WMO. The goals are organized in a tree structure that refines high-level goals into lower level goals, until they are operationalized into tasks. For instance the goal ManageApplication is refined in two subgoals: ValidDocuments and CompleteApplication. This decomposition is motivated by the requirements of having correct applications, which is captured by soft-goal Correctness. Indeed, in the diagram *contribution* links are used to represent the fact that two sub-goals contribute to the achievement of soft-goal Correctness. The goal ValidDocuments is further refined in the goals CompleteApllicationDocuments and ValidTechicalReports. These two goals are respectively operationalized with task ManageDocuments and ValidateTechicalReports. It has to be noticed that the latter task is not present in the choreography depicted in Fig. 1. Indeed this task is motivated by internal requirements of the WMO. In the diagram, two different kinds of links between goals and tasks are shown. Solid arrows are used to describe that some tasks have been obtained by the operationalization of certain goals, while dashed lines express the fact that the satisfaction of a certain requirement depends on a given task. One can see, for instance, that different tasks are responsible to guarantee the completion in time of the different phases of the procedure.

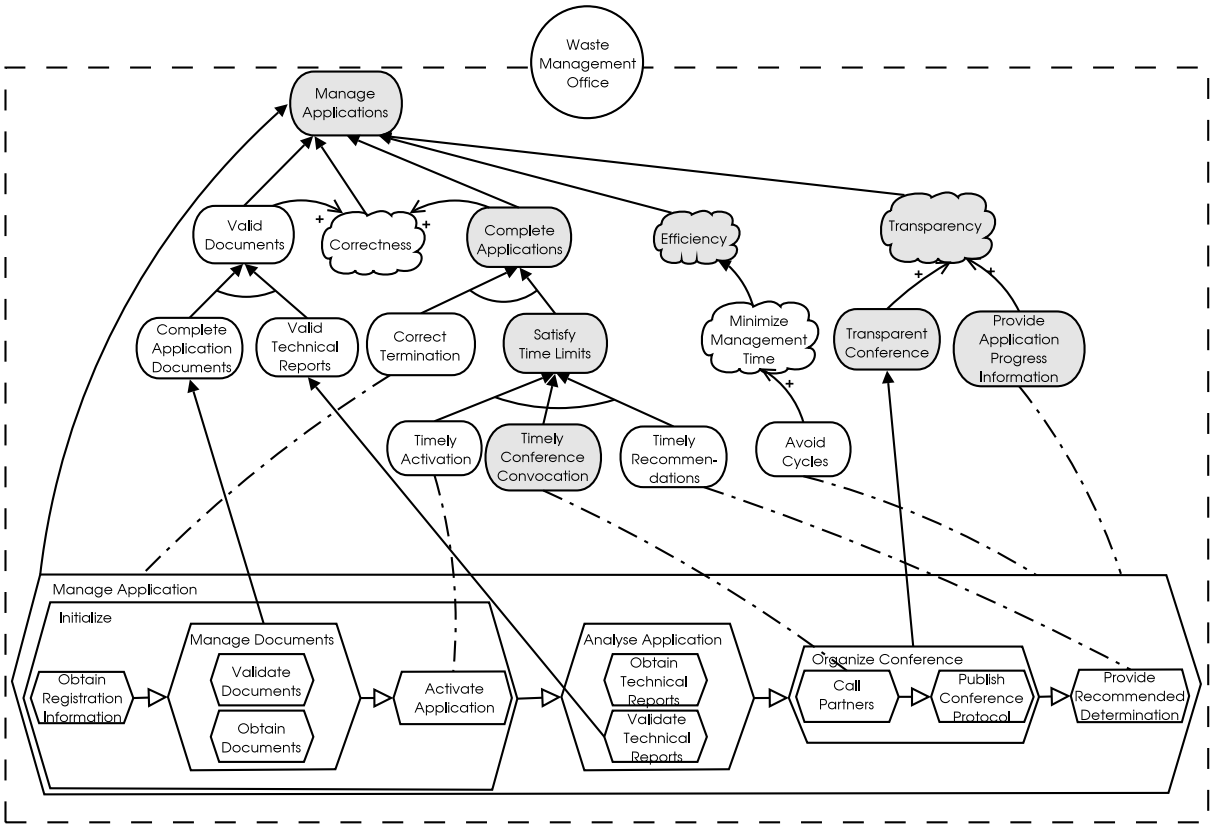


Figure 5: Strategic and Procedural Requirements of the WMO.

The Tropos notations discussed in this section are supported by a corresponding formal language, Formal Tropos [4], which allows for a precise definition of the requirements and of the activity diagrams and enables the usage of verification tools for detecting specification, design, or implementation problems. Formal Tropos permits to specify the valid behaviors and the relations among the different actors, dependencies, goals, and tasks that appear in a Tropos model. At the strategic level the Formal Tropos annotations specify properties like conditions on goal fulfillment, and assume/guarantee conditions on delegations. At the procedural level, they define pre- and post-conditions on tasks and sub-tasks. Even more important, Formal Tropos annotations allow to link together these two levels and the underlying implementation level. The key advantage of Formal Tropos with respect to other approaches is that it defines the dynamic aspects of a model and supports its formal verification already at the requirements level, without requiring an implementation of the specification, e.g., into BPEL4WS processes.

We conclude this section by remarking that, after the negotiation phase has been concluded, the refinement process of the requirements diagrams can further proceed transforming activity diagrams into executable code. In our framework we are adopting BPEL4WS [1] at the implementation level. BPEL4WS is quickly emerging as the language of choice for the description of process interactions. It provides core concepts for the definition of business process in an implementation-independent way, and allows both for the definition of internal business processes and for describing and publishing the external business protocol that defines the behavior of the interaction. Therefore, BPEL4WS permits to describe both the orchestration and the choreography of a business domain with an uniform set of concepts and notations. Most notably, BPEL4WS can be easily obtained by refining activity diagrams like the ones in Fig. 1 or in the bottom part of Fig. 5: see for instance [6] for a Model Driven approach to this refinement. Finally, as shown in [5, 16], links to the requirements can be maintained into the BPEL4WS code, so that requirements traceability is possible.

5. SUPPORTING THE PROCESS: VERIFICATION

The development process described before is accompanied by verification tools that support the different activities necessary to develop correct service oriented applications [5, 16]. These tools allow for verifying the correctness of a model at all levels of abstractions covered by our methodology. At the strategic level verification can be used to validate the requirements and to check their consistency. At the procedural level, verification can be used for proving that the processes are free of anomalies such as “deadlocks” (when a execution is “blocked” and no longer proceeds through the process) and “livelocks” (when an execution gets “stuck” in a never-ending loop), or to check the timing constraints on the different activities. At the implementation level, verification can point out incompatibilities and inconsistencies among

the different web services that need to interact to carry out the procedure. Moreover, verification can be used to check the consistency among the different levels of a specification, that is, the procedural level of requirements should respect all constraints stipulated at the strategic requirements level, and the implemented web services should be a refinement of the activities defined at the procedural level. Finally, verification can be done both from a choreographic point of view, e.g., to check that the defined procedure respects all constraints imposed by the law, and from an orchestration point of view, e.g., to check that the services a party will offer are compatible with its own internal requirements and goal. For lack of space, we cannot give a comprehensive description of all applications of verification inside our process. We focus instead on some specific application scenarios.

A first usage of verification techniques is for validating choreographic requirements. While defining a global choreography, one should deal with partners interactions in terms of intent/offer matches as well as with the business rules common for all the participants of the business process. This makes the definition of this requirements model a complex and error-prone task. In order to catch misunderstandings and inconsistencies in this model one can verify it against set of properties that every execution of the system should satisfy (assertion properties) or some execution may satisfy (possibility properties). Querying the model allows one to check the correctness of the model with respect to the property or to check whether the model is not over-specified and some desirable behaviors are captured by the system. For instance, one property that the choreography should guarantee is that, if all actors carry out their own tasks, as described in the strategic requirements model, then the citizen will eventually get a (positive or negative) answer to the license request. However, a missing goal or dependency in the strategic requirements diagram may falsify this property. Suppose for instance that we remove dependency ActivateApplicationManagement between the Citizen and the PO from the requirements in Fig. 4. Then there is no guarantee that the Protocol Office will eventually forward our application to the WMO after having registered it, and the chain of activities leading to the answer to the citizen is broken. Indeed, if we exploit the verification techniques provided by Formal Tropos to verify that the GetWasteFacilityLicense goal of the Citizen will be eventually fulfilled, we will get a negative answer. Actually, the verification tool provides a counter-example scenario, showing that it is possible for the PO to fulfill all its goals without having to forward to the WMO the citizen's application.

At a lower level of abstraction, verification can be used to detect anomalies like deadlocks in the activity diagrams defining the interactions among parties. For instance, let us assume that, within the negotiation process, we modify the definition of the ManageDocuments activity in Fig. 1 as described in Fig. 6. The intuition is that we want to model a WMO that keeps interacting with the citizen in an iterative, cyclic way until a complete documentation is obtained. If this modification is not reflected into the orchestration activities of the Citizen, a deadlock occurs. Indeed, if the documents provided are not correct also after a first integration, the WMO will ask for further documents. However, according to Fig. 1, after a first integration the Citizen expects either a conference announcement or an abortion of the procedure, so he is not able to provide further documents. The verification techniques we are providing can be used for finding such inconsistencies. For instance, if the analyst designing the services of the Citizen verifies his internal process against the modified choreography described above, the inconsistency is detected and the following scenario leading to the deadlock condition is reported as a witness:

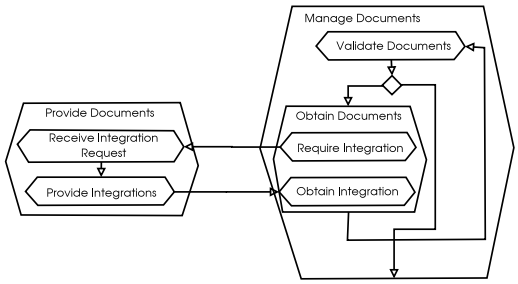
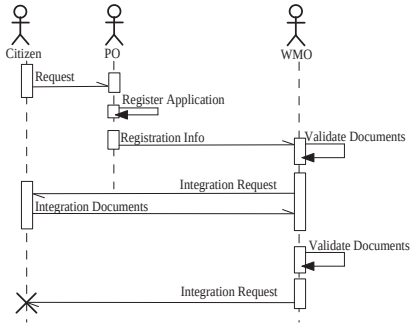
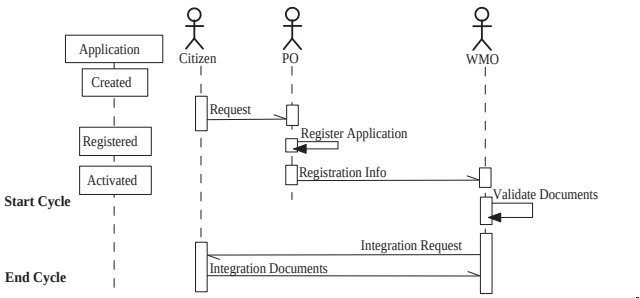


Figure 6: Modified Citizen - WMO interaction.



A last example of verification consists in checking if the choreographic process model is compatible with the local needs and expectations of a specific actor. Let us assume that the choreographic process adopted permits a cyclic interaction between the WMO and the Citizen in order to obtain integration documents, as in Fig. 6. Then the verification tool shows that the internal goal CorrectTermination of the WMO may be violated. Indeed, the formal specification of the goal is that every application submitted to the WMO should terminate with a recommendation or should be eventually aborted by the WMO. This property is violated if the choreography allows for cyclic interactions with the Citizen, and the following example of goal violation is reported by the verification tool:



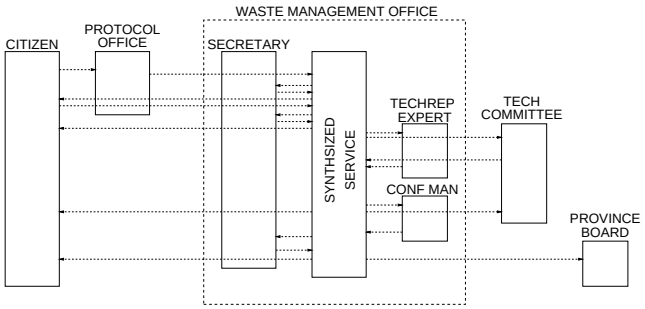


Figure 7: Synthesis of the WMO Service.

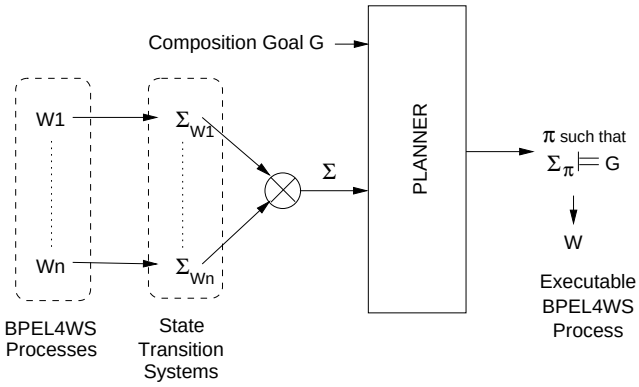


Figure 8: Automated Composition.

6. SUPPORTING THE PROCESS: SYNTHESIS AND MONITORING

In this section we comment on how we can exploit program synthesis techniques to automate the development of web services within our reference process. These techniques come to help after the design and the negotiation of the web services has been done and every participant has to implement his own services. The scenario we are interested in is when the participant already has services (or other software components) available that can be exploited to carry out his activities, but these services have to be adapted and composed in a way suitable to the new procedure ad hand. In our case study, the WMO already has internal services available for managing the standard tasks occurring in the different procedures the office is involved in. These services represent in some sense the 'back-office' of the WMO (see Fig. 7). In our case, the back-office consists of a Secretary service, of a Technical Report Expert and of a Conference Management. The secretary is in charge of evaluating whether the documents provided by the user conform to the requirements, to incrementally file the evolution of the request, and to extract relevant data that have to be communicated to the other parties involved. The technical report expert is in charge to communicate the relevant data to the technical committee, and collect and interpret his responses. The conference management is in charge of organizing the meetings between the participants involved. To participate to the new procedure, the WMO has hence to implement one more service that interacts with the other internal "back-office" services and with the external services of the other participants.

Automating the generation of this new service can be seen as a particular instance of automated generation of web services. By automated composition [14, 20] we mean the task of generating automatically, given a set of available web services, a new web service that achieves a given goal, i.e., that satisfies a given requirement, by interacting with the available web services. Different techniques have been proposed so far which address this problem. In our framework, we exploit the automated task planning techniques described in [14, 20]. According to the approach of [14, 20] (see Fig. 8), we take as our starting point the BPEL4WS specification of the existing internal and external services $(W_1, \ldots, W_n)$. In our case, these descriptions are either already available to the WMO (for the internal services) or they are an outcome of the negotiation phase (for the external services). We encode each of the BPEL4WS specification in a state transition system $(\Sigma_{W_1}, \ldots, \Sigma_{W_n}$ in Fig. 8), which provides a sort of operational semantics to the BPEL4WS model. Each of them describes the corresponding web service as a state-based dynamic system, that can evolve, i.e., change state, and that can be partially controlled and observed by external agents. In this way, it describes a protocol that defines how external agents can interact with the service. From the point of view of the new composed service that has to be generated, say W, the state transition systems $\Sigma_{W_1}, \ldots, \Sigma_{W_n}$ constitute the environment in which W has to operate, by receiving and sending service requests. They constitute what, in planning literature, is called a planning domain, i.e., the domain where the planner has to plan for a goal. In our case, the planning domain is a state transition system Σ that combines $\Sigma_{W_1}, \ldots, \Sigma_{W_n}$. Formally, this combination is a parallel composition, which allows the n services to evolve independently and concurrently. Σ represents therefore all the possible behaviors, evolutions of the planning domain, without any control performed by the service that will be generated, i.e., W. The composition goal G (see Fig. 8) imposes some requirements on the desired behavior of the planning domain. In our case, the goal can be obtained from the "orchestration" requirements model of the WMO. Given Σ and G, the planner generates a plan π that controls the planning domain, i.e., interacts with the external services $W_1, \ldots, W_n$ in a specific way such that the evolutions satisfy the goal G. The plan π encodes the new service W that has to be generated, which dynamically receives and sends invocations from/to the external services $W_1, \ldots, W_n$, observes their behaviors, and behaves depending on responses received from the external services. The plan π is encoded as an automaton and can hence contain complex control constructs, like tests over observations, conditionals, loops, etc. As a final step, we can translate π into process executable languages, like BPEL4WS.

Though still preliminary, the experiments reported in [14, 20], show that the automated synthesis approach described above can deal with cases that are far from trivial. Moreover, an interesting possibility offered by the composition approach described in Fig. 8 is that of obtaining *monitors*, i.e., software components that are able to observe the messages exchanged with (internal or external) services and to report whether they are violating the BPEL4WS protocol that they are supposed to implement. Indeed, we can exploit to this purpose the finite state machines $\Sigma_{W_1}, \ldots, \Sigma_{W_n}$,

that capture the operational semantics of the corresponding BPEL4WS specifications.

7. CONCLUSIONS AND RELATED WORK

In this paper we propose a development process where global and local requirements are incrementally defined within a negotiation process. Requirements are described in a language with a clear semantics, which allows us to define precise links between business requirements and (executable) business processes. This opens up to the construction of tools for the analysis of requirements, the verification of business processes, as well as their synthesis and monitoring. The proposed approach has been inspired by and experimented with a real application we are developing in the public administration field. We see this work as a first step towards the construction of techniques and tools that support the development of distributed services by reducing development time, efforts, and errors.

The Model Driven Architecture [3], backed by OMG specifications such as UML 2.0 [18, 17] , aims to separate business logic from the details of platforms, programming languages and middleware. Developers create platform-independent models (PIMs), which can be semiautomatically transformed to platform-dependent models (PSMs). We share with this approach the need for high level specifications of services; more specifically, in terms of "Model Driven Service Composition" [11], we share the idea that service requirements should be analyzed in a systematic way, and the idea to describe business rules as precise statements. However, none of the previous approaches is based on the idea of incremental definition of the business rules that come out of a negotiation between global and local goals. In our proposal, requirements are structured, analyzed, and negotiated according to a clear distinction between internal business needs for a single business process, dependencies of objectives among different partners, and business rules that are common to a community of services. The development process described in other works, see, e.g., [11, 12], focuses on an important but orthogonal issue, i.e., how high level requirements (e.g., expressed in UML and OCL [11]) for a single process can be classified for service composition, and how they can be refined into executable processes. Some of the model driven approaches advocate for the use of verification techniques, e.g., based on Petri nets [15], or model checking [7, 8]. However, in these approaches, the problem of verifying local versus global rules is not addressed.

In [21] a formalism is proposed, based on Petri nets, which allows for verifying that local implementations of workflows do not create anomalies over organizational borders. However, the considered development process is purely top-down, from community requirements to the local implementation of workflows that have to satisfy the global requirements. There is no global-local requirements negotiation in [21], and thus the problem of verification of local versus global requirements is not addressed.

Different automated planning approaches have been proposed for the composition of web services [22, 10], for the interactive composition of information gathering services [19], and for providing viable plans satisfying specific queries of the user [9]. Within the development process that we propose, we use instead automated planning techniques to generate automatically executable business processes from high level requirements, and to generate automatically at design time monitors that can detect problems at run-time.

Acknowledgments

This work has been supported in part by the FIRB-MIUR project RBNE0195K5 "Astro". The authors want to thank all members of the Astro project for their collaboration and their feedback.

8. REFERENCES

[1] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, F. Leymann J. Klein, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. *Business Process Execution Language For Web Services, Version 1.1*, 2003.

[2] J. Castro, M. Kolp, and J. Mylopoulos. Towards requirements-driven information systems engineering: the Tropos project. *Information Systems*, 27(6):365–389, September 2002.

[3] D.S. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing*. John Wiley and Sons, 2003.

[4] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, and P. Traverso. Specifying and analyzing early requirements in Tropos. *Requirements Engineering*, 2004. To appear.

[5] R. Kazhamiakin, M. Pistore, and M. Roveri. A framework for integrating business processes and business requirements. In *Proc. 8th Int. IEEE Enterprise Distributed Object Computing Conference (EDOC'04)*, 2004. To appear.

[6] J. Koehler, R. Hauser, S. Kapoor, F. Y. Wu, and S. Kumaran. A model-driven transformation method. In *Proceedings of the Seventh International Enterprise Distributed Object Computing Conference (EDOC'03)*, pages 186–197, Brisbane, Queensland, Australia, September 2003. IEEE Computer Society.

[7] J. Koehler, R. Hauser, S. Kapoor, F.Y. Wu, and S. Kumaran. A Model-Driven Transformation Method. In *EDOC 2003*, pages 186–197. IEEE Press, 2003.

[8] J. Koehler, G. Tirenni, and S. Kumaran. From Business Process Model to Consistent Implementation: A Case for Formal Verification. In *EDOC 2002*, pages 96–106, 2002.

[9] A. Lazovik, M. Aiello, and Papazoglou M. Planning and Monitoring the Execution of Web Service Requests. In *Proc. of the 1st International Conference on Service-Oriented Computing (ICSOC'03)*, 2003.

[10] S. McIlraith and S. Son. Adapting Golog for composition of semantic web Services. In *Proc. 8th International Conference on Principles of Knowledge Representation and Reasoning*, 2002.

[11] B. Orriens, J. Yang, and M. Papazoglou. Model Driven Service Composition. In *Proc. of the 1st International Conference on Service-Oriented Computing (ICSOC'03)*, 2003.

[12] M. Papazoglou and J. Yang. Design Methodology for Web Services and Business Processes. In *Proc. of the Technologies for E-Services Third International Workshop (TES'02)*, pages 54–64, 2002. Lecture Notes on Computer Science, Springer-Verlag, Berlin, Germany.

[13] C. Peltz. Web Services Orchestration and Choreography. *Web Services Journal*, July 2003.

[14] M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso. Planning and monitoring web service composition. In *Proc. 11th Int. Conf. on Artificial Intelligence: Methodology, Systems, Architectures*, 2004. To appear.

[15] D. Quartel, M. van Sinderen, and L. Ferrera Pires. Service Creation: a model based approach. In *Proc. of the 7th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'99)*, 1999.

[16] M. Roveri R. Kazhamiakin, M. Pistore. Formal Verification of requirements using Spin: A Case Study on Web Services. In *Proceedings of the 2nd International Conference on Software Engineering and Fomal Methods (SEFM'04)*, Beijing, China, 2004. IEEE Computer Society.

[17] P. Rivett and OMG Group. Unified Modeling Language: Infrastructure - version 2.0, 2003.

[18] B. Selic and OMG Group. Unified Modeling Language: Superstructure - version 2.0, 2003.

[19] S. Thakkar, C. Knoblock, and J.L. Ambite. A View Integration Approach to Dynamic Composition of Web Services. In *Proceedings of ICAPS'03 Workshop on Planning for Web Services*, Trento, Italy, June 2003.

[20] P. Traverso and M. Pistore. Automatic composition of semantic web services into executable processes. In *Proceedings of 3^{rd} International Semantic Web Conference (ISWC2004)*, Lecture Notes Computer Science, Hiroshima, Japan, 2004. Springer Verlag.

[21] W.M.P. van der Aalst. Inheritance of Interorganizational Workflos: How to agree without loosing control? *Information Technology and Management Journal*, 2(3):195–231, 2002.

[22] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S Web Services Composition using SHOP2. In *Proceedings of the Second International Semantic Web Conference (ISWC2003)*, 2003.