Flexible Coordination of Service Interaction Patterns

Christian Zirpins, Winfried Lamersdorf, Toby Baier
Distributed Systems and Information Systems Group – VSIS
University of Hamburg, Hamburg (Germany)
{Zirpins, Lamersdorf, Baier}@informatik.uni-hamburg.de

ABSTRACT
Service-oriented computing is meant to support loose relationships between organisations: Collaboration procedures on the application-level translate to interaction processes via Web Services. Service composition deals with the specification and enforcement of such processes. Its main focus is on service *orchestration* where workflow management is utilised for proactive coordination. In such an approach, coordination process and interaction logic are usually captured in the same workflow – which leads to deficiencies in recognising the possible impact of operational coordination on the interaction logic. In this paper, we claim that the choice of coordination alternatives impacts the quality of the composed service and has to be customised to each specific service case. As a consequence, we outline a solution that is based on *service interaction patterns* where the paradigms of patterns and idioms are applied to interaction procedures and orchestration processes. This allows studying a) reusable interaction patterns typical for service relationships and b) for each pattern a range of possible coordination idioms. Finally, we sketch a technique that refines the service logic based on analysis of its interaction patterns and utilisation of suitable coordination idioms selected by rules in terms of changing service context.

Categories and Subject Descriptors
J.7 [Computer Applications]: ADMINISTRATIVE DATA PROCESSING.

General Terms
Management, Design, Economics.

Keywords
Service-oriented Computing, Service Composition, Interaction- and Coordination Process Patterns, Rule-based Process Refinement.

1. INTRODUCTION
Web Services are software components that provide self-contained functionality via Internet-enabled, interoperable interfaces and publish a common description of their characteristics to be dynamically discovered, selected and accessed by clients. They provide fundamental building blocks for *Service-oriented Computing (SOC)* [1] that aims to support service relationships between organisational participants. However, a single Web Service is almost never capable of representing a complete *application-level service* (e.g. a flight booking service). On the one hand, even a basic application service normally includes a non-trivial *bilateral interaction procedure* between a client and a service provider (e.g. book → encash) that includes communication endpoints on both sides and clear *conversational logic*. On the other hand, an application service typically splits into functional parts (search flight offers, book flights) of multiple providers (e.g. flight broker, airline) and includes their *composition logic*, resembling a *multilateral interaction procedure*.

In either case, the field of *Web Service composition* provides means to assemble basic Web Services into composite ones that constitute a considerable step towards application services. In particular, service composition deals with the coordination of composite services by means of *service orchestration processes*. Orchestration languages like BPEL4WS adopt concepts of *workflow* to specify flows of control and data between Web Service operations. As service-oriented computing focuses on cross-organisational relationships, service composition typically includes multiple interconnected orchestration processes controlled by different participating organisations. Therefore, *cross-organisational workflow* is an area that is closely related to service composition. It focuses workflows that span multiple organisational domains. The central problem is the decomposition of single workflows with respect to the set of participating organisations. A straight forward solution for service realisation is the mapping of interaction procedures to orchestration processes (i.e. cross-organisational workflow).

However, it has to be minded that service interaction procedures and cross-organisational workflow differ in some subtle aspects like change frequency of participants and additional facets of their interrelation [2]. In this paper, we propose to address one such facet concerned with coordination: grounding interaction procedures on cross-organisational workflow implies a fixed decision on operational coordination (i.e. decomposition, refinement and distribution of local workflows). This is not desirable for services because it impacts their characteristics and should be rather treated as a separated aspect that can be decided on dynamically (at provision-time) and independently from the core service logic. Our solution involves *coordination idioms* that get dynamically selected and applied to *interaction patterns* in the interaction procedure by *rules* based on the actual service context.

The rest of this paper motivates and outlines our service coordination concept. Initially, we review service-oriented computing

ICSOC'04, November 15–19, 2004, New York, New York, USA.

concepts and technology for supporting application processes in section 2. Then we detail the coordination problem of service composition in section 3. Subsequently, we propose patterns and rules as an approach to achieve flexibility of service coordination in section 4. In section 5 we discuss related work before concluding in section 6.

2. SYSTEM SUPPORT FOR APPLICATION-BASED SERVICE PROCESSES

As a basis for our main proposal, the following section introduces the underlying concepts of process-oriented application services and their relation to common techniques of service-oriented computing. First, it introduces Web Services as building blocks for distributed information systems. Then, it details the specific class of application services which is strongly related to interaction processes between organisational participants. This is followed by a survey of major process-oriented Web Service extensions. Finally, the relation to service composition is drawn.

2.1 Web Services as Component Technique for Distributed Information Systems

The mainly industrial driven standardisation of the *Web Service* system-technology architecture [3] is the latest evolutionary step of support for distributed software applications on the Internet. In particular, this is true for cross-organisational cooperative information systems [4]. Web Services are software components that offer self-contained functionality to potential users of interoperable applications in a distributed environment, using open, standardised interfaces and protocols. Service *providers* are realising the functionality of their services and publish service characteristics in a standardised form via service *brokers*. This allows different service *clients* to discover (maybe even at runtime), select (according to their criteria), bind, and access them [1]. Essentially, Web Services can be seen as components of distributed applications that are only abstractly defined in the beginning. Subsequently, a client can dynamically change providers to realise the concrete Web Service functionality at runtime. Therefore, the *service* of a Web Service provider consists of supplying compatible software components for external clients. Existing metamodels for Web Services (e.g. WSDL) provide the means to describe services of this kind.

An interesting variant is the case of a distributed software application that represents a service on application-level (referred to as *application service*), building on functionality of separate Web Services as components. An example for this is a composed, distributed flight reservation service with all of its sub services which can be bound to different locations or organisations for different clients or situations. While basic Web Services in their current state do support technical access to the functionalities of such a composed application service, their service model does not suffice completely to adequately describe the resulting service on the application-level: beyond its lack of means to express several (functional and non-functional) application aspects, such as transaction support, authentication security, quality of service, service accounting and service billing, a strong deficit is the lack of a process-oriented perspective of application services. Such a viewpoint allows covering the interaction procedure (which is in most cases quite complex) between the different, distributed and for the most part heterogeneous service participants. Better system-level support for such interactive and process-oriented aspects of open Web Service-based applications is a main research goal in service-oriented computing.

2.2 Application Services and Service Interaction Processes

In particular, an interactive application service often includes a *bilateral interaction procedure* between client and provider which is not covered by the Web Service model. Such so called *conversations* between two service participants require operational interfaces (i.e. Web Services) as communication endpoints on *both* sides as well as explicitly exposed and mutually agreed conversational logic (e.g. the flight booking example mentioned above could start with a request message from a passenger to a travel agency, followed by a flight confirmation message and finally an invoice message back from the travel agency to the passenger).

Concerning application services with multiple service providers (such as flight booking via a flight information agency and an airline), it is often the case that *multilateral* interaction procedures have to be focussed in addition to bilateral interaction procedures. In such a case, the correct interrelation of all the functional parts (i.e. retrieval of flight information from the agency followed by flight reservation at the airline) requires a specification of composition logic for service execution – i.e. an instruction how an application service is constructed from other application services including all procedures and protocols on application-level.

Within a process-oriented perspective of cooperating Web Services the representation of a (composite) application service not only requires to specify separated communication endpoints of service participants by a set of their Web Service interfaces but also to capture the associated service *procedure* that represents their underlying *interaction logic*.

2.3 Process-oriented Web Service Extensions

In the course of ongoing development, the Web Service architecture was – both technically and conceptually – expanded to more adequately support the manifold requirements of distributed applications. Some of these extensions are already geared towards process characteristics of Web Services and Web Service-based applications – namely Web Service Conversation, -Choreography and Orchestration (see figure 1).

First to mention, *Web Service conversation* revolves around the extension of operational Web Service interfaces by protocols of their correct call sequences (see figure 1, left). Those approaches, which are often based on finite state automata (e.g. WSCL by Kuno et al [5]), mainly target bilateral conversation with single Web Services.

For the coverage of complex interaction procedures that go beyond single Web Services, superordinate protocol specifications are currently being developed which are collectively known as *Web Service choreography* (see figure 1, right). Languages like the W3C working draft *Web Services Choreography Description Language (WS-CDL)* [6] define the publicly observable behaviour of collaborations, each consisting of invocations of different Web Services, from a neutral point of view. Choreographies are well suited for the characterisation of general service processes and allow verifying the consistency of potential participants based on their conversation descriptions.

Furthermore, concepts and technologies to glue Web Service components together and make them available as a single composite service are subsumed under the term *Web Service composition*. The unification of design and specification of composite Web Services constitutes a basis for system-level support or even automation of general management operations – like for example the discovery and binding of suitable providers and clients as well as their enactment during service execution.

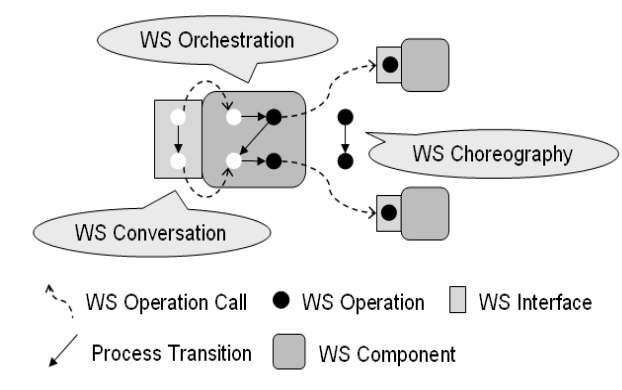

Figure 1. Process-oriented extensions of basic Web Services

The most commonly used approach to service composition is the so called *Web Service orchestration* (e.g. [7]): Its first step is to specify the causal dependencies of functional Web Service calls as well as the dependencies of the herein used parameters as a process (see figure 1, middle). Subsequently, this process specification represents the *coordination logic* of the composite Web Service and furthermore serves as the instruction for automated coordination of the service application flow. The concepts of the respective process description languages or *orchestration languages* (e.g. BPEL4WS [8]) are mainly based on those for the representation of *workflows* (i.e. automated, collaborative process flows [9] that can also be used to model control- and data flow of a set of Web Services), enabling execution support by *Workflow Management Systems (WfMS)*.

2.4 Application Service Composition

A comprehensive representation of application services in open, distributed environments – that is a major research goal of service-oriented computing – includes revealing and exposing all relevant relationships between enterprises or organisations that need to be considered in order to enable open application-level interaction. Part of these efforts is the composition of application service components to *value added services*. However, in contrast to the composition of Web Services, the perspective is here much broader and more application-oriented [2].

A central aspect in the composition of value added application services is the process-oriented viewpoint onto the interaction logic of the service participants. A direct approach to support this view on a system-level is its implementation as Web Service orchestration. The interaction logic of an application service is then expressed by the control and data flow of a workflow. The participants of the workflow are represented using a role model and their mutual communication of messages are mapped to activities which, in turn, are realised as Web Service operations (see figure 2).

Since service-oriented computing mainly aims at relationships that span multiple different organisations, such a service composition typically consists of multiple networked orchestration processes which can be enforced by different participants. For this reason, cross-organisational workflow techniques (see e.g. [10]) that support workflows over multiple organisational (and technical) domains naturally have to be considered. A central problem of such workflows is the distribution of a central workflow to different organisations that take part in the execution. Facets of this problem include, among other things, questions of process meta-models and modelling, analysis, classification, consistency and verification of processes, as well as their execution and runtime architectures.

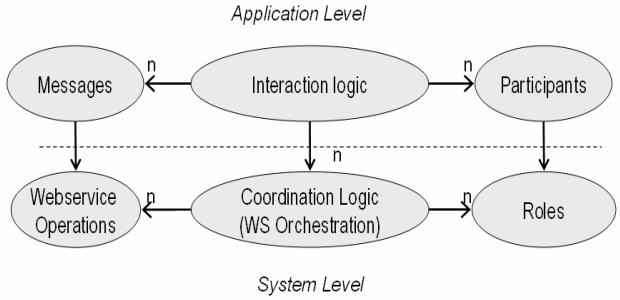

Figure 2. Mapping of service procedure to Web Service techniques

However, compared to cross-organisational workflows there are additional requirements in terms of system dynamics when realising application services. This results from dynamically changing participants and, thereby, the resulting flexibility of the (application) processes. Furthermore, some additional (application-specific) facets of the relationships are hard to cover [2] (one of those will be presented in more detail later and results in the essential problem behind this work).

In contrast to the dynamics of service-specific interaction processes, cross-organisational workflows (e.g. for the realisation of collaborative relations in virtual organisations) are often tailored to previously known participants and have to be changed only rarely (apart from system evolution). However, participants of composite application services change often– even during runtime – since providers offer their service to several clients and clients can switch their providers, respectively. Therefore, appropriate support for this natural system property is vital. Accordingly, representations for the interaction logic of application services need to initially capture the characteristic service dependencies only in a universal ("*abstract*") form (i.e. still without precise specification of the organisations who will participate). When instantiating *concrete* service relationships (also called *service cases*) where the participating organisations are ultimately appointed, this abstract interaction logic has to be refined with respect to the requirements of the hence completely known execution context. In doing so, also facets of the service relationship must be specified, that can not be appropriately expressed within the framework of general workflow models (e.g. requirements for quality or organisational structure of the procedure).

Workflow-based composition of application services is at the heart of various current research approaches: For example the system *"eFlow"* of Casati et al [11] particularly enables the flexi-

ble composition of application services as workflows that can be changed dynamically. The project *DySCo* [12] also introduces a process-based service notion and develops means to derive the partial process of each participant from it. Another workflow-based system of similar kind is *SELF-SERV* from Dumas et al [13]. Some approaches also consider extended application requirements: Yang et al. [14] define the *"Service Composition Lifecycle"* that is a methodology for flexible and dynamic construction of application services. Based on that, Orriëns et al [15] employs business rules for dynamic process construction.

So far, the impact of operational coordination of service instances on the characteristics of the underlying composite application service is an aspect of process-based service composition that draws little attention. However, we will see that this specific facet of service relationships introduces further differences to cross-organisational workflow approaches that open up a considerable potential for optimisation.

3. THE COORDINATION PROBLEM OF APPLICATION SERVICE PROCESSES

As discussed in the last section, the current mainstream approach to the composition of application services is based on a process-oriented design of the interaction logic together with a workflow-based realisation as Web Service orchestration process. While this approach has considerable merits (e.g. in terms of the intuitive representation and possible visualisation of processes) it has also drawbacks, partly caused by inherent characteristics of the workflow concept itself and partly by different requirements of cross-organisational workflow on the one hand side and service-oriented computing on the other [2].

A specific class of problems arises from the *coordination* aspect of application service processes: Here, we distinguish the *logical dependencies* that are modelled by the interaction logic from the *operational coordination* that refers to the procedure or method that is utilised to enforce the logical dependencies. While cross-organisational workflow processes represent the logical dependencies of interactions (i.e. causal and data relationships of message exchanges) they simultaneously act as instructions for their coordination on the execution-level by distributed WfMS[1]. Note that hereby the coordination procedure emerges only implicitly as a "side-effect" of dependencies from the interaction logic and not because for application-specific reasons.

On the other hand, there are in most cases multiple concrete alternatives for the enforcement of the abstract application-oriented interaction logic. A reason for this is the multiplicity of possibilities to split the dependencies of the interaction logic into different partitions as well as the variety of alternatives to delegate parts of a partition to executive organisations for operational coordination. This ambiguity is important because the choice of the operational coordination structure affects certain (non-functional) characteristics of the cross-organisational workflow. Colombo, Francalanci and Pernici [16] describe this effect in terms of the *organisational structure* of inter-enterprise relationships. Furthermore, we reckon an impact on additional *non-functional characteristics* that affect the *quality* of an application service (QoS).

[1] For the sake of precision, it has to be noted that process engines usually transform the workflow process representations that are used for modelling into an equivalent representation that is optimized for execution purpose (e.g. ECA rules for active DBMS).

3.1 An illustrative example

In order to illustrate the issue, we discuss the example of a simplified *flight booking service*. The interaction logic of this service is shown in figure 3.

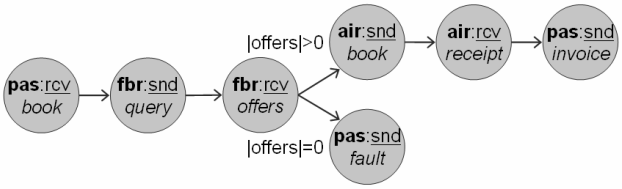

Figure 3. Flight booking service: Interaction procedure

The pseudo-notation – that is geared to usual workflow models – contains *communication steps* (circles) and *transitions* (arrows). Communication steps represent the sending of messages to an endpoint (e.g. *book*) either originating from (rcv) or going to (snd) a *role* that represents a participant (**tra** = travel agency, **pas** = passenger, **air** = airline, **fbr** = flight broker). The example informally expresses that a passenger accesses a booking service and a flight broker is consequently queried for an airline to forward the request to. If an offer is found, a booking request is issued to the airline, and, after its confirmation, an invoice is sent to the passenger. Otherwise, a fault is reported.

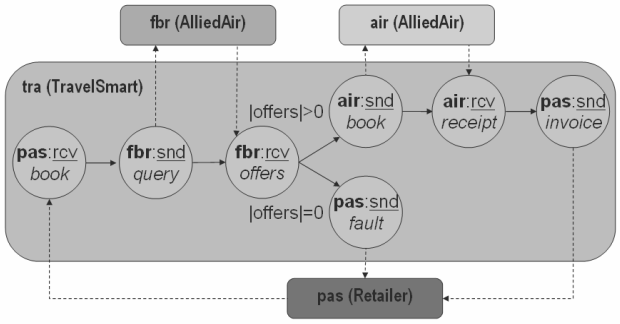

Figure 4. Coordination alternative A: central orchestration

Up to now, the interaction logic tells nothing about the participant(s) that enforce the dependencies between the message exchanges at runtime. Figure 4, figure 5 and figure 6 show three alternative refinements (A-C) in terms of operational coordination. For each of them, the interaction logic is partitioned into interrelated interaction processes that are assigned to individual roles. Subsequently, interaction processes can be translated to a Web Service orchestration language (whereby message communications translate to Web Service operations) that allows their distributed execution. While alternative partitionings only impose minor process changes which lead from the abstract interaction logic to the concrete coordination logic, also deeper modifications are possible – as long as the original dependencies of the interaction logic are preserved. Thus, the examples demonstrate that the coordination logic of a service realisation can vary substantially while the respective interaction logic remains exactly the same.

However, different coordination alternatives affect various non-functional characteristics: On the one hand, differences can emerge on the system-level: For example, alternative A features a centralisation of the coordination that results in lower overhead and thus reduced costs (because there is less hand-over of control), but might suffer from common drawbacks like, e.g., a bottleneck. The alternatives B and C feature decentralised coordination that leads to more complexity but allows for local optimisations and increased parallelism.

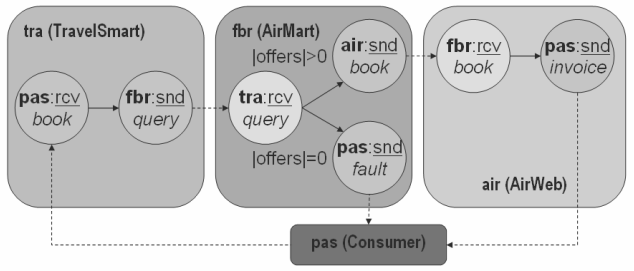

Figure 5. Coordination alternative B: Decentralised orchestration with an implicit change of control

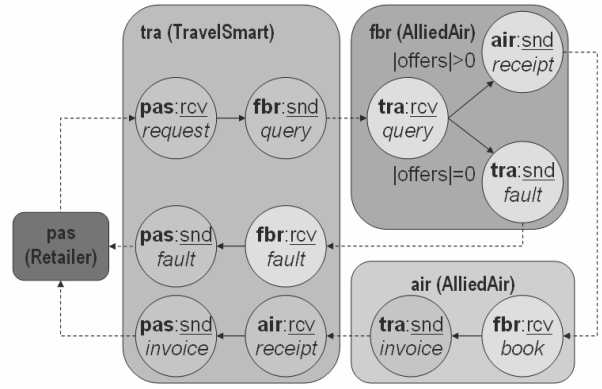

Figure 6. Coordination alternative C: Decentralised orchestration with explicitly preserved control

On the other hand, differences appear on application-level: While alternative A empowers TravelSmart to coordinate (and control) the whole interaction procedure, alternative B delegates a certain level of coordination and control to other participants. This results in an explicit shift of rights and responsibilities as well as an implicit shift of autonomy. Such differences may be crucial to specific business constraints of an organisation. While this is in some cases acceptable (e.g. alternative B: TravelSmart delegates short-term consumers to public accessible sub-services because customer retention is considered unimportant.) it is not acceptable in others (e.g. alternative A: TravelSmart acts as an integrator of non-public sub-services for a retailer to which TravelSmart wants to keep an exclusive long-term relationship.). However, it is possible to compensate such effects partly by refining the operational coordination (e.g. alternative C: In spite of delegated coordination, TravelSmart regains control as well as the direct and exclusive relationship to its customer.).

3.2 Inflexible service coordination

The concrete choice of coordination alternatives depends on the specific requirements of the organisations that eventually participate in the execution of the application process. Because application services are faced with a continuous change of participants the concretisation of the specific coordination logic should be kept open until the participants (together with their specific and dynamically changing requirements) of a specific application service *case* are ultimately known. Only then, a refinement should be made that is optimal for this specific case and complies with the abstract (functional and non-functional) requirements of the application service as specified in the interaction logic.

The principal problem of common workflow approaches to describe such distributed, interactive applications in open heterogeneous environments is due to the fact that flexibility of operational cooperation is usually not supported. In particular, general workflow models lack the appropriate abstractions to adequately express the necessary coordination aspects. Moreover, general workflow techniques lack appropriate methods (beyond general approaches to flexibility) to vary the operational coordination dynamically at runtime depending on case-specific criteria.

4. A PATTERN-BASED APPROACH FOR FLEXIBLE SERVICE COORDINATION

As presented in the last section, the currently dominant approach to realise distributed applications in heterogeneous environments by workflow-based composition of Web Services still shows severe shortcomings in case of service-oriented cooperative information systems (application services).

In terms of coordinating application service processes, workflows pinpoint the concrete coordination process without taking into account qualitative aspects (e.g. the global organisation structure) and do not provide appropriate means for a flexible control of coordination at provision-time. Overall, the impacts of coordination in terms of service-quality can neither be expressed nor dynamically adapted to the requirements of varying participants or other dynamically changing system states.

Therefore, the goal is to provide system-level support for optimised enforcement of application services by dynamic control and optimisation of non-functional service properties based on flexible coordination of service-specific cross-organisational interaction procedures with utilization of alternative coordinative variations at provision-time.

In this section, we first summarize challenges in the service composition lifecycle that originate from the coordination problem. Thereafter, we sketch our approach to solving some of them: a framework for flexible service coordination that is based on patterns of service interaction. Finally, we accentuate the conceptual part of this framework which strives for a base of empirical knowledge of specific real-life patterns.

4.1 Coordination Related Challenges in the Service Composition Lifecycle

Before outlining the main ideas of our approach, we summarize the challenges that emerge from last section's observations in the context of the *service composition lifecycle* proposed by Yang and Papazoglou [14]. Such a lifecycle is structured into five phases: *1) planning* (synthesis of service logic), *2) definition* (abstraction of service composition), *3) scheduling* (analysis of possible composition refinements in the context of a new service case), *4) con-*

struction (assembly of concrete composition for the case), and *5) execution* (enforcement of concrete composition).

In this work, we suppose service interaction logic (i.e. conversation- and composition logic) as external input (e.g. from business process (re-)engineering) and do not directly interfere with its synthesis. Thus, our methodology can be classified as *semi-fixed composition* [14] and considers phases 2-5 where we face the following problems:

- ***Phase 2) Definition phase:*** specification of coordination-independent interaction procedure. Here, an *interaction process meta-model* is needed that can represent given conversational- and composition logic in terms of dependencies between abstract service components (messages) and abstract participants (roles) and does not imply any constraints for operational coordination. Additionally, the model needs to provide means for representing possible coordination choices and policies for their selection.
- ***Phase 3) Scheduling phase:*** analysis and evaluation of coordination choices. Here, exact information is needed about a) the range of possible coordination choices for a given interaction procedure, b) the range of relevant service characteristics and the effects of individual choices on them, and c) the service case's context, i.e. the group of possible participants together with their characteristics and requirements. The analysis has to consider all this information and prepare it for evaluation. Finally, evaluation requires a pre-defined metric of qualitative measurements and a formal framework for automated reasoning in terms of coordination policies.
- ***Phase 4) Construction phase*:** refinement of interaction procedures into service orchestration processes. Here, model transformation capabilities are needed from the interaction process meta-model to a cross-organisational workflow meta-model of choice. Additionally, structural transformations of cross-organisational workflows are needed to refine the dependencies of the interaction procedure to the choice of operational coordination. This does not only require the capability to merely do any such transformation but rather exact knowledge of how to realise each specific choice of cooperation.

There are no specific problems in terms of this discussion w.r.t. the execution phase because orchestration processes resemble standard (cross-organisational) workflows that are handled by a respective WfMS as usual.

4.2 Generic Mechanisms for Flexible Service Coordination Based on Patterns

A concept to meet these challenges has to deal predominantly with the management of relationships between the abstract application-level (= interaction logic) and the concrete implementation-level (= coordination logic) of application services. In particular, such a concept has to a) free the interaction logic from undesired effects on the coordination logic and b) leverage relevant implications that result from a concrete coordination variant onto the abstract level of interaction logic. While a) is a question of appropriate techniques (i.e. mainly analysis, optimisation, and transformation of respective process models), b) requires a-priori knowledge of concrete use cases: What coordination choices are there and to which forms of interaction procedures do they apply? What are the implications of specific coordination choices in terms of which service characteristics? Those questions arise because, in terms of application-specific effects of coordination alternatives, also non-technical aspects have to be considered that need to be empirically analysed beforehand.

Our approach considers both aspects in a joint technical and conceptual framework: On the one hand side, the optimisation of coordination-sensitive service properties is generally enabled by generic mechanisms for flexible workflow-based coordination of process-oriented application logic. On the other, a concrete catalogue of reusable interaction patterns is used as a basis for the utilization of the generic mechanisms for the support of specific service interactions.

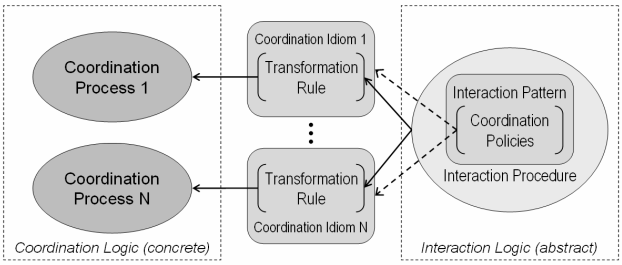

Figure 7. Conceptual overview of the approach

The basic principle of the technical solution (see figure 7) consists of a combination of *design*- and *implementation patterns* (the latter are referred to as *idioms*) with rule-based control by so called *policies*. Thereby, the interaction logic of an application service is not only expressed by a process model (like in usual workflow-based approaches) but is kept abstract as far as possible (i.e. in particular with respect to non-functional parameters that are unknown until provision-time). The abstract parts of the respective *interaction procedures* are expressed by *interaction patterns* that initially only specify the generic process characteristics. In contrast, the refinement of runtime aspects that are needed for the concrete coordination of services is firstly specified (i.e. dynamically) at provision-time (e.g. a pattern could initially specify an abstract payment procedure where different coordination alternatives impact the security properties of the application service that are decided on at provision-time).

Possible refinement alternatives are anticipated in *coordination idioms* that represent the coordination alternatives. For each abstract interaction pattern, there exist a set of such concrete coordination idioms (e.g. there could be a centralised and decentralised alternative for the coordination of the payment procedure mentioned above). Each idiom specifies the realisation of a concrete coordination process by *transformation rules* from the abstract interaction procedure of the pattern.

The criteria for the choice of the respectively most appropriate coordination idiom are specified as part of the interaction logic by *coordination policies*. A coordination policy describes the effect of a coordination variant in terms of specific non-functional service properties and thereby controls the choice of alternatives with respect to current participants (e.g. a policy for the payment procedure could state that the requirement of a participant for the non-functional service property of *secure payment* has to lead to the choice of the centralised coordination idiom).

For the preferably fast, comfortable and reliable development of application services, basic pre-analysed patterns are offered. They

provide means to model the interaction logic at design-time and serve as basis for analysis and optimisation of coordination logic at runtime. Therefore, preferably generic and reusable patterns are collected in a *pattern catalogue*.

4.3 Towards a Conceptual Framework of Concrete Service Patterns

Despite the fact that, by now, there is a large interest in the composition of application services, the field of research is yet in an initial stage and consolidated knowledge is still scarce. Thus, up to now only few generic forms of service-specific interaction logic are known and those few are merely characterised in an informal way. Beyond the *brokerage pattern*, which is probably best-known and some other forms like *delegation* or *migration* that all emanate from the context of system technology, the context of application-oriented and -specific interactions is a potential source of additional relevant forms.

Our framework is intended to provide the necessary means to gather those generic interaction patterns in a systematic way. Thus, the framework comprises a catalogue of concrete generic interaction patterns, coordination policies and coordination idioms to subsequently build system support techniques upon. Here, recurring (and thus reusable) patterns are collected and classified. With a basic set of such patterns, application-level interactions can be designed and analysed to finally optimise and support respective process coordination with system software techniques.

The conceptual challenge is to identify a range of relevant service patterns to fill the catalogue. The first step is to assemble a collection of typical interaction patterns. For that purpose, patterns are collected that are generic and as little as possible domain-specific. In addition to a basic set of such patterns, a taxonomy of service-specific interaction patterns is needed for the classification and structuring of the collection. The idea is to foster the identification of concrete patterns with appropriate use cases. After interaction patterns are informally identified, they are analysed in terms of the underlying generic procedure and interaction logic. Also, relevant non-functional properties (e.g. performance, robustness, security) together with an appropriate metric are determined.

In contrast to interaction patterns, the collection of coordination idioms can be done in a systematic way. The identification is based on one interaction pattern respectively that already determines a scope in terms of the structure of the underlying processes and thus allows deriving of possible coordination alternatives. Each such coordination variant has to be analysed in terms of its detailed process, its possible prerequisites, and its effects on interaction-specific non-functional service parameters. The gathered information is finally pinpointed as transformation rules of the coordination idiom and extension of the interaction pattern with additional coordination policies.

The rationale for these investigations is to gain consolidated knowledge about the coordination of service interactions. Concrete patterns and idioms become part of the system's knowledge base and can be applied in the lifecycle. The existence of such knowledge is crucial for the application of our approach and forms the conceptual part of our framework. Its investigation constitutes an important and distinctive part of our research.

5. RELATED WORK

In principle, our approach to service coordination is part of the field of workflow-based service composition (see sec.2 for general work in this area) and copes with coordination problems that are partly rooted in the workflow itself and partly originate from the different requirements of service composition. In particular, we adopt techniques of patterns and rule-based transformation in the context of workflow.

The general use of *rules in workflow management* is quite common: Apart from the integration of rules as elements inside workflows, rules have been used on meta-level for *workflow adaptation* [17, 18]. In this case, rules govern modifications that are applied to a workflow either statically at design-time or dynamically at runtime to add flexibility. More recently, business rules have been proposed to construct BPEL4WS service composition processes [15]. This approach has particular similarities to ours in terms of its use of rules to conduct different transformations of service orchestration processes during the service composition lifecycle. Though, it does not consider pattern mechanisms or investigate concrete process structures.

Patterns provide means to conserve and reuse knowledge about the solution of a generic problem. They range from informal guidelines used in system design (*design patterns*) to customizable code fragments (*implementation patterns* or *idioms*). In particular, patterns are widely known for their use in object-oriented design and architecture [19]. Pattern concepts have also been applied to workflow management. Foundational studies on basic control-flow structures, named *workflow patterns* [20], can be used to examine and compare general workflow languages. In [21], a formal model is proposed for rule idioms that can be instantiated as rule elements in workflow schemas in the Chimera-Exc language. In [16], high-level design patterns for organisational coordination and control structures are proposed and related to corresponding cross-organisational workflows.

Recently, pattern-based approaches have been proposed for service composition. The authors of [22, 23] propose architectural design patterns that give indications on starting points for our investigation of interaction- and coordination patterns. Another proposal is to use design patterns of service composition logic [24]. To the best of our knowledge, work on process patterns for service interaction or -coordination has not been published yet.

6. CONCLUSION

In the emerging research field of service-oriented computing and, especially, in the area of service composition, many approaches are closely related to workflow concepts. In particular, concepts of cross-organisational workflow are often used to model and execute composite services. However, first doubts appear on the appropriateness of workflow concepts for service composition as the latter is believed to imply more complexity, more dynamics and more facets in the relation of participants. Additional doubts emerge from findings from research in the area of cross-organisational workflow that indicate shortcomings of current concepts as regards the support of coordination aspects. The problem is that most efforts concentrate on developing generic techniques to solve problems of an application area that is generally not well understood yet. Only few approaches aim at investigating concrete characteristics (e.g. specific classes of problems, their requirements and solution strategies etc.) of composite services.

In this paper we propose to investigate the specific facet of coordination aspects in service composition. We stress the relevance of coordination alternatives for the enforcement of service composition dependencies, as the choice of such an alternative rebound on service characteristics. To address this point, we propose generic mechanisms that allow representing relationships (coordination policies) between the abstract service composition logic (interaction patterns) and its concrete coordination choices (coordination idioms). The ability to model patterns of composition logic and their idioms of coordination enables us to formulate and structure knowledge of a range of concrete problems and solutions of service composition that we intent to examine. Ultimately, the generic mechanisms together with the concrete knowledge translate into a framework to support the lifecycle of service composition.

7. REFERENCES

[1] M. P. Papazoglou and D. Georgakopoulos, "Service-oriented computing: Introduction", *Communications of the ACM*, vol. 46, pp. 24-28, 2003.

[2] J. Yang, W.-J. v. d. Heuvel, and M. P. Papazoglou, "Tackling the Challenges of Service Composition in e-Marketplaces", in *12th Int. Workshop RIDE-2EC*, 2002.

[3] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services - Concepts, Architectures and Applications*: Springer, 2004.

[4] G. D. Michelis et al. "Cooperative Information Systems: A Manifesto", in *Cooperative Information System: Trends and Directions*, M. P. Papazoglou and G. Schlageter, Eds.: Academic Press, 1997.

[5] H. A. Kuno, M. Lemon, A. H. Karp, and D. Beringer:, "Conversations + Interfaces = Business Logic", in *2nd Int. Workshop, TES 2001*, F. Casati, D. Georgakopoulos, and M.-C. Shan, Eds.: Springer, 2001, pp. 30-43.

[6] N. Kavantzas, D. Burdett, and G. Ritzinger, "Web Services Choreography Description Language Version 1.0", W3C Working Draft 27 April 2004.

[7] C. Peltz, "Web Service orchestration and choreography: a look at WSCI and BPEL4WS - Feature", *Web Services Journal*, vol. 3, 2003.

[8] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana, "Business Process Execution Language for Web Services, Version 1.0", BEA, IBM, Microsoft 31 July 2002.

[9] D. Georgakopoulos, M. Hornick, and A. Sheth, "An overview of workflow management: from process modeling to workflow automation infrastructure", *Distributed and Parallel Databases*, vol. 3, pp. 119-53, 1995.

[10] P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig, "CrossFlow: cross-organizational workflow management in dynamic virtual enterprises", *Computer Systems Science and Engineering*, vol. 15, pp. 277-90, 2000.

[11] F. Casati and Ming Chien Shan, "Dynamic and adaptive composition of e-services", *Information Systems*, vol. 26, pp. 143-63, 2001.

[12] G. Piccinelli, A. Finkelstein, and S. L. Williams, "Service-oriented work-flows: the DySCo framework", in. *29th Euromicro Conf.*, 2003.

[13] M. Dumas, B. Benatallah, and Z. Maamar, "Definition and Execution of Composite Web Services: The SELF-SERV Project", *Data Engineering Bulletin*, vol. 25, 2002.

[14] J. Yang and M. P. Papazoglou, "Service Components for Managing Service Composition Lifecycle", *Information Systems*, vol. 29, pp. 97-125, 2004.

[15] B. Orriëns, J. Yang, and M. P. Papazoglou, "A Framework for Business Rule Driven Service Composition", in *4th Int. Workshop, TES 2003*, B. Benatallah and M.-C. Shan, Eds.: Springer, 2003, pp. 14-27.

[16] E. Colombo, C. Francalanci, and B. Pernici, "Modeling Coordination and Control in Cross-Organizational Workflows", in. *CoopIS/DOA/ODBASE 2002*, R. Meersmann and Z. Tari, Eds.: Springer, 2002, pp. 91 ff.

[17] C. Zirpins and G. Piccinelli, "Evolution of Service Processes by Rule Based Transformation", in *IFIP Int. Conf. I3E 2004*, W. Lamersdorf, V. Tschammer, and S. Amarger, Eds.: Kluwer, 2004, pp. 287-305.

[18] G. Joeris and O. Herzog, "Managing Evolving Workflow Specications with Schema Versioning and Migration Rules", University of Bremen TZI Technical Report 15, 1999.

[19] F. Buschmann, R. Meunier, H. Rohnert, and P. Sommerlad, *Pattern-Oriented Software Architecture - A System of Patterns*: Wiley and Sons Ltd., 1996.

[20] W. M. P. v. d. Aalst, A. H. M. t. Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow Patterns", *Distributed and Parallel Databases*, vol. 14, pp. 5-51, 2003.

[21] F. Casati, S. Castano, M. G. Fugini, I. Mirbel, and B. Pernici, "Using Patterns to Design Rules in Workflows", *IEEE Trans. Software Eng.*, vol. 26, pp. 760-785, 2000.

[22] O. F. Rana and D. W. Walker, "Service Design Patterns for Computational Grids", in *Patterns and Skeletons for Parallel and Distributed Computing*, F. A. Rabhi and S. Gorlatch, Eds.: Springer, 2003, pp. 237-264.

[23] B. Benatallah, M. Dumas, M. C. Fauvet, and F. Rabhi, "Towards Patterns of Web Services Composition", in *Patterns and Skeletons for Parallel and Distributed Computing*, F. A. Rabhi and S. Gorlatch, Eds.: Springer, 2003, pp. 265-296.

[24] D. Edmond and M. T. Tut, "The Use of Patterns in Service Composition", in *Int. Workshop, WES 2002*, C. Bussler et al. Eds.: Springer, 2002, pp. 28-40.