2009 Sixth International Conference on Information Technology: New Generations

# Towards a Goal-Based Service Framework for Dynamic Service Discovery and Composition *

Luiz Olavo Bonino da Silva Santos, Eduardo Gonçalves Silva,
Luís Ferreira Pires, and Marten van Sinderen

Centre for Telematics and Information Technology, University of Twente
P.O. Box 217, 7500AE, Enschede, The Netherlands
E-mail: {olavol,e.m.g.silva,pires,m.j.vansinderen}@ewi.utwente.nl

## Abstract

*Service-Oriented Computing allows new applications to be developed by using and/or combining services offered by different providers. Service discovery and composition are performed aiming to comply with the client's request in terms of functionality and expected outcome. In this paper we present a framework for dynamic service discovery and composition. This framework is based on goals and tasks as the means to represent the client's expected outcome and functionality, respectively. The framework encompasses a goal-based service ontology, a set of domain and task ontologies and a supporting service platform with a service matching and composition algorithm. The client informs the platform about the goal to be fulfilled. The platform's matching algorithm searches in the repository for services that can fulfill the client's goal. Moreover, the platform gathers client's contextual information to use as inputs for the services and thus, reduce the need for client interaction. If no single service is able to fulfill the user's goal, a service composition is then performed.*

## 1 Introduction

Service-Oriented Computing (SOC) has emerged as a computing paradigm that uses the concept of service as the basic construct for distributed applications. SOC has a promising vision of a world of cooperating services where cooperation relations can be dynamically created to form applications and business processes [5]. In a simple service setting we have a service client that requests the execution of a service to a service provider. In case of a small amount of services and service providers, it is straightforward for the service client to decide which pair of service and service provider best fits its needs.

However, in environments where a large number of service clients and service providers are present, it becomes difficult to manually and individually match the client's requests with the available services and their providers. In such environments, a supporting service platform is necessary. A service platform can support service clients in activities such as service discovery, selection, composition and invocation. The support can also be extended to service providers by providing a mechanism for rapid creation, deployment and advertisement of services. The addition of semantics to service descriptions and to messages exchanged between service clients, service providers and the supporting service platform enables complex reasoning tasks [9]. Among these reasoning tasks are the interpretation of service providers' capabilities and service clients' requirements. Assuming that the participants share the same conceptual model and that the terms in the exchanged messages are mapped to this model, semantic interoperability becomes possible. In our framework we use ontologies to provide this shared conceptualization.

Commonly, service client's requirements are expressed in terms of inputs, outputs, pre-conditions and effects, also known as IOPE. End-users, i.e., human service clients, may have difficulties to express such requirements as they would have to deal with technical issues such as the request language and the type, format and coding of the IOPE. To tackle application scenarios where end-users are not technology literate we propose the use of goals to express what the end-user wants to be accomplished by the service. The use of goals aims at raising the definition of service client's requirements to a higher abstraction level, therefore facilitating its use by end-users.

In this paper we present a framework for dynamic service discovery and composition. This framework is based on the concept of goal to express the user's service requirement

*The present work is partly funded by the Freeband Communication project A-Muse (http://a-muse.freeband.nl). A-Muse is sponsored by the Dutch government under contract BSIK 03025.

978-0-7695-3596-8/09 $25.00 © 2009 IEEE
DOI 10.1109/ITNG.2009.27

and it uses context information to reduce the need for user interaction. Our framework is composed by a goal-based service ontology which is used to define domain specifications and by a supporting context-aware service platform. These domain specifications are used to semantically annotate services and the exchanged messages between service clients, service providers and the supporting platform.

This paper is further structured as follows. Section 2 gives an overview of the architecture of our goal-based service framework. Section 3 details the proposed goal-based service ontology. Section 4 presents the supporting service platform and the matching algorithm. Section 5 gives conclusions and identifies topics for future work.

## 2 Goal-Based Service Framework

Goal-based analysis has been used in different areas of Computer Science to identify stakeholders' objectives, determine requirements for software systems and guide system's behavior. As a representation of a service client's objectives, goals are used in Service-Oriented Computing to indicate the desired outcome of a service. In the Service-Oriented Computing literature we can find initiatives for service discovery and composition based on goals such as the Web Service Modeling Ontology (WSMO) [1], GoalMorph [10] and the approach presented by Zhang et al in [11]. Although these initiatives do not agree on what is a goal they have in common the assumption that goal definitions are already available and have been previously identified and modeled. However, these initiatives either do not clarify how these goals are gathered and modeled or do not detail how the goal descriptions relate to concrete services. We claim that prescriptions of how to model goals are still missing and that they should be addressed when developing a dynamic service discovery and composition framework based on goal gathering and modeling.

Our framework to support dynamic service discovery and composition is based on goal modeling, and assumes that the involved stakeholders (service clients, service providers, supporting platform) share the same conceptual models, i.e., the same set of ontologies. This requirement is necessary because the approach relies on the availability of domain-specific ontologies. For each domain, the concepts of the domain and the valid goals of the domain are identified together with the tasks required for their fulfillment. Figure 1 depicts the main elements that comprise our framework:

- *Goal-based service ontology (GSO)*. This ontology defines domain-independent concepts such as service, stakeholder, organization, goal and task, and their relations. These definitions are further used and specialized in the domain and task ontologies.

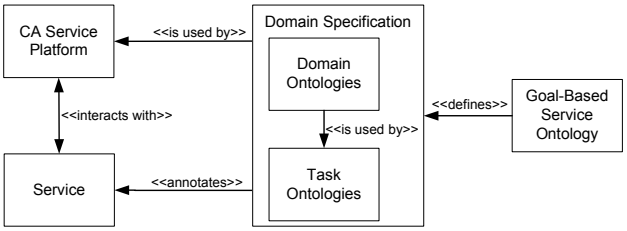


**Figure 1. Main components of the Goal-Based Service Framework**

- *Domain specification*. Based on the goal-based service ontology concepts, domain specifications are defined providing a shared knowledge about particular domains. The domain specification is divided into:
  - Domain ontologies. These ontologies define domain-specific concepts, the relations among these concepts and the valid goals for that domain.
  - Task ontologies. A task ontology is associated with a domain ontology and provides domain-specific definitions of valid tasks in that domain and how they related to the domain goals.
- *Context-Aware Service platform*. The context-aware service platform supports the interaction between service providers and service clients. From the service provider's perspective, the platform supports the publication of service descriptions. From the service client's perspective, the platform provides mechanisms for service discovery, composition, invocation and monitoring, among others. Moreover, the context-aware components of our supporting platform provide user's contextual information that is used (i) to select which of the tasks that support a given goal will be used in the service discovery and composition procedures and, *(ii)* as input data for the discovered services. The context information gathering reduces the need of direct user input and, thus, reduces also the need of user's interaction supporting a more autonomic behavior of the platform.

The minimal deployment of the framework includes the GSO and the CA Service Platform. Domain specifications can be deployed together with the other two components or added by domain specialists. A set of domain specifications are being developed in the scope of our work for the purpose of validating the framework as a whole and checking the suitability of the GSO. The GSO and the supporting CA Service Platform are the focuses of this paper and are

discussed in the sequel. Examples of domain specification based on GSO are the subject of forthcoming papers.

## 3 Goal-Based Service Ontology

The concept of goal has several different definitions depending on the domain the term is used, e.g., Philosophy, Sports, Economy, among others. Narrowing down to the Computer Science domain, a variety of definitions of the goal concept can also be found. In the Artificial Intelligence (AI) realm, goal is defined as a "description of a world state that is expected to be realized" [7]. Among the several definitions for goal in the agent-oriented computing community, in [4] goal is defined as a "state with highest utility and an agent must choose the course of action to reach that goal" and in [6] goal is defined as a "final state that the agent tries to achieve by moving from its initial state through a defined and finite sequence of intermediary states".

For the purposes of this framework, we adopt and extend the goal definition presented in [3] and define goal as the *propositional content of a service client's intention*. In this definition a service client has an intentional moment of the type ***Intention***. Other types of intentional moments include ***Belief*** and ***Desire***). Desire and intention express a will of an intentional agent towards a state of affairs in reality. The difference between intentions and desires is that by intending something, an intentional agent commits at pursuing it. Therefore, by having a goal a service client commits to pursue the fulfilment of that goal. Using this definition we can have that alternative state of affairs can satisfy (satisfy in the logical sense of having a proposition representing that propositional content) the goal. This opens the possibility of using Fuzzy Logic to assess partial satisfaction (if necessary).

### 3.1 Tasks, goals and services

A task is defined here as *the means to fulfill a goal*, i.e., it defines a process that transforms the world from a setting *A* into a setting *B*. When the state of affairs derived by the outcome of task *T* satisfies the propositional content of goal *G*, we say that task *T* supports goal *G*. The object diagram in Figure 2 depicts the relations between goals, tasks and the related agents. In we derive two other relations:

- *Fulfillment*. If *AgentB* can execute *TaskA* and *TaskA* supports *GoalA*, it implies that *GoalA* can be fulfilled by *AgentB*.
- *Delegation*. If *AgentA* owns *GoalA* and, for some reason it cannot fulfill the goal itself, a delegation is performed to an agent that can fulfill *GoalA* (*AgentB* in this example). Here, the ownership relation entitles the owner agent to delegate the fulfillment of the goal to another agent.

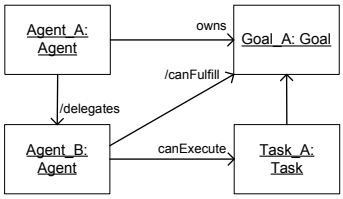


**Figure 2. Relations between goals and tasks**

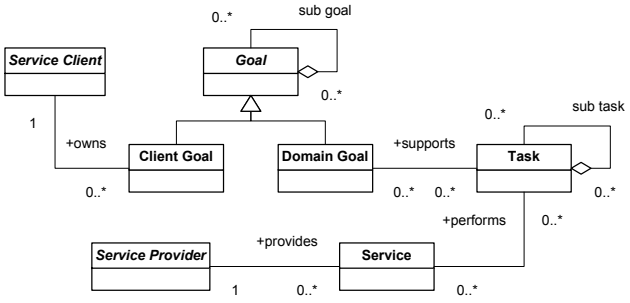


**Figure 3. Definition of goal, task and service**

The ownership, delegation and fulfillment relations among agents, tasks and goals defined in our framework are inspired by the Tropos [14] framework.

In our approach we consider service as the concrete realization of a task. In other words, a task is an abstract definition of activities (in the sense that it does not have a direct implementation) whose outcome matches the state of affairs proposed by a goal. This separation between a definition of activities and the actual implementation of these activities allows the distinction of administrative domains for tasks and services. While services are typically defined by service providers, tasks can be defined by service clients, domain specialists or service providers. Figure 3 depicts an excerpt of the goal-based service ontology that presents the concepts of *Goal*, *Task* and *Service*. Here, the concept of *Goal* is specialized into *Client Goal* and *Domain Goal*. While a *Client Goal* is defined and owned by a *Service Client*, a *Domain Goal* is defined by a domain specialist in the scope of a domain ontology. Both *Goals* and *Tasks* can be structured in a hierarchy of sub-goals and sub-tasks, respectively. By traversing the hierarchical tree of *Goals* the platform can search for tasks that support the main goal (the higher level goal) or the sub-goals. Similarly, the platform traverses the task tree searching for services that can perform the main task (the higher level task) or the sub-tasks.

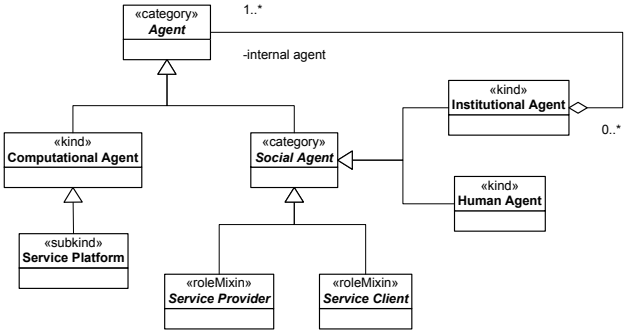


**Figure 4. Agent definition**

## 3.2 Agents

For the specification of our goal-based service ontology, we related the aforementioned definitions of goal and task with the concepts of the Service-Oriented realm. The concept of agent, as used in Figure 2 is specialized in terms of the concepts of service client, service provider and service platform. Figure 4 depicts an excerpt of the goal-based service ontology that defines the concepts of service platform, service client and service provider. These concepts are classified using the typology for universals defined in the Unified Foundational Ontology (UFO) [2].

By relating a concept of our ontology to an UFO universal we commit to the principles of identity supplied by the universals. Among the types of universals defined in UFO that are of particular interest for our ontology we include the concepts of *kinds*, *roles*, *role mixins* and *categories*. We used these types of universals to define our main concepts such as the service client, service provider and the service platform. The concept of *Service Platform* is stereotyped as a kind and, therefore, is defined as a rigid entity type. A type $T$ is rigid if for every instance $x$ of $T$, $x$ is necessarily (in the modal sense) an instance of $T$ [2].

The concepts of *Service Provider* and *Service Client* are stereotyped as the UFO universal role mixin which is defined as an anti-rigid type. A type $T$ is anti-rigid if for every instance $x$ of $T$, $x$ is possibly (in the modal sense) not an instance of $T$ [2]. The different categorization between service platform, service client and service provider is based on the nature of their possible instances. While an instance of a service platform is in every situation of the type Service Platform, an instance of service client can be of the type Service Client in one situation and of the type Service Provider in another situation.

As happens in real world, service clients and service providers can represent either individuals or institutions, e.g., *XYZServices* is an institutional service provider and *Bob* is an individual service client. Therefore, our ontology further specializes the concepts of Service Client and Service Provider into the sub-concepts of *Human Service Client*, *Institutional Service Client*, *Human Service Provider* and *Institutional Service Provider* as depicted in Figure 5. In a, a Service Client is further specialized into Human Service Client and Institutional Service Client which are also sub-types of the concepts Human Agent and Institutional Agent, respectively. b shows that the concept Service Provider is specialized into Human Service Provider and Institutional Service Provider which are also sub-types of the concepts Human Agent and Institutional Agent, respectively. These four sub-concepts are stereotyped as the UFO universal role which is defined as an anti-rigid type. Each one of these four roles is a sub-type of a kind and a roleMixin. This modeling solution of multiple disjoint allowed types follows the ontological design pattern described in [2].

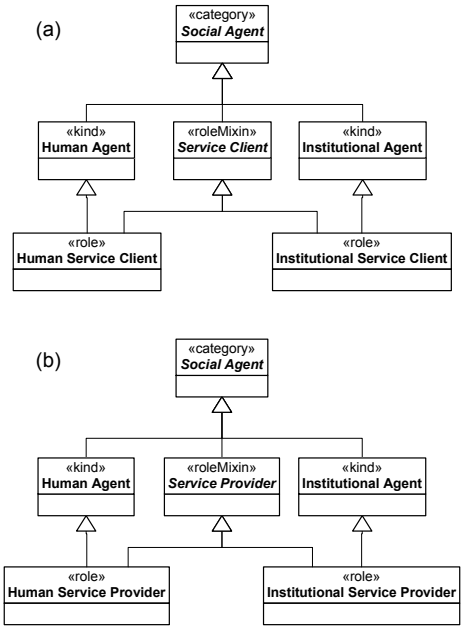


**Figure 5. Classification of service client and service provider**

# 4 Context-Aware Service Platform

Our platform, depicted in Figure 6, aims at supporting non technical service clients (end-users) in finding a suitable service, i.e., a service that fulfills their requested goals. Apart from service clients our framework also provides support to service providers on the process of service publication. When a service provider creates a new service, he semantically annotates it (e.g., its IOPEs, behavior descrip-

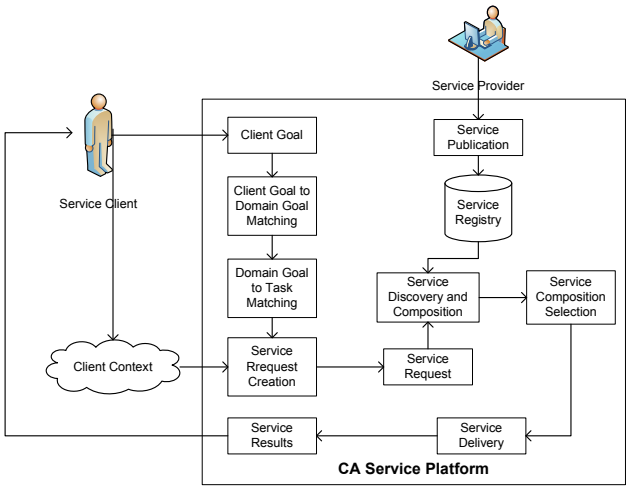


**Figure 6. Context-aware service platform**

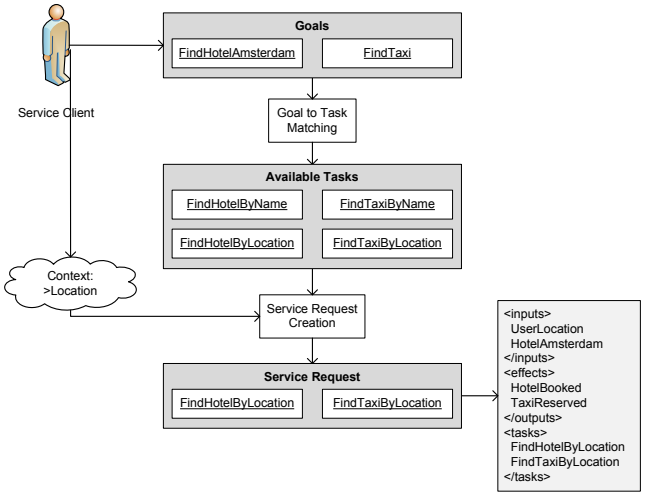


**Figure 7. Service client request**

tion, quality properties, etc.) using the ontologies in the domain specification. These annotations are then used by the platform to perform the service publication. This allows other entities to discover and possibly compose services to match a service client's request. Below we focus on the framework support to service clients, assuming that all existing services are created, described using the available domain and task ontologies and published to the platform.

The platform allows a service client to express his goals. The client goals are then matched against the available domain goals. Given this set of goals, the platform determines the set of tasks in the task ontology that support the specified goals. Provided with this set of tasks, the platform refines the service request. The service request contains the properties that define the different tasks, consisting on a set of inputs, outputs, preconditions, effects, goals, and non-functional properties. Not all these properties have to be always present in a service request. In our framework the service request is further optimized, by means of the context information available for the user that requested the service. This optimization consists on filtering the previously created service request inputs and preconditions, considering only the inputs and preconditions that can be delivered by context sources of the user or the inputs the user specifies in his request. This will allow abstracting the user from the process of invoking a service, by delegating the gathering of the required information to the platform, through the available context sources.

Figure 7 shows an example of creating a service request. In this example the client has goals of finding a taxi and a hotel in Amsterdam. The client has a device that provides his current location. Given this location information, the platform creates a service request which consists of tasks that support the goals (*FindTaxi* and *FindHotelAmsterdam*) and can be executed without the direct intervention of the user, i.e., using the user context information (location) and the user specified hotel location (Amsterdam). This reflects that tasks that do not have as effects having a hotel and a taxi booked in Amsterdam are not considered for the final service request, i.e., only the *FindHotelByLocation* and *FindTaxiByLocation* tasks are considered in the service request creation. For the illustrative purposes of this example, goals and tasks are only represented here by their names.

Provided with this service request, the dynamic service composition platform can be invoked to discover, and possibly compose, services that match the specified service request. The first action performed by the dynamic service composition platform is to discover services that match the requested goals. This means, in our example, services that semantically match the tasks *FindHotelByLocation* and *FindTaxiByLocation*. The set of discovered services are retrieved and stored in a matrix. If there is a service that matches all the service request parameters, then the service is directly retrieved, otherwise a further step is performed, aiming at the creation of a service composition that matches the service request. The process of service composition is performed by a graph-based algorithm for automatic service composition [8]. In the graph a node represents a service and an edge represents an output-input semantic relation. The algorithm starts by creating a graph with services that provide the request's outputs and effects. Then, in each iteration the algorithm matches the inputs of the graph's services with the outputs of the services from the set of discovered services organized in the matrix. The process continues until the service request's inputs and preconditions are matched, and the client goal is fulfilled by the com-

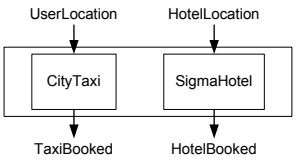


**Figure 8. Service client request**

posed services. Input-output matching, and goal matching, are performed using the domain ontologies. This allows exact, plugin, subsume and intersection semantic matching. In our example, two services were discovered that match the service request goal: *CityTaxi* and *SigmaHotel*. Then the platform delivers a composite service that combines both services in a single service composition, as shown in Figure 8.

## 5 Conclusions and future work

In this paper we presented our framework for goal-based dynamic service discovery and composition. This framework is primarily target on application on scenarios where the service clients are end-users without technological training and scenarios where the service clients require reduced interaction with the services. For this purpose we propose the use of goal to express the service clients' requirements and the use of context-awareness to gather information to be used as inputs for the services. In this manner the service clients have a higher level of abstraction way of expressing what they want to be accomplished by the services (by using goals) and a reduced need to interact with the services (by using context information).

Moreover, we presented and discussed the ontological foundations of the main terms defined in the framework, i.e., goal, task, service client, service provider and service platform. This ontological foundation provides a solid underlying conceptualization and supports the semantic definition of the terms used throughout our framework.

The distinction between the abstract description of activities (the task) and the concrete realization of activities (the service) has been shown useful in our framework to support dynamic service discovery. The framework assumes the previous existence of domain and task ontologies defined by domain specialists. This assumption makes the framework suitable for environments where the domain is clear and well known. Examples of suitable domains for our framework are Ambient Intelligence (AmI) and mobile pervasive applications' domains where users are not able to interact often with the computational devices. Additionally, the CA Service Platform has been presented together with an overview of the service discovery and composition algorithm. The platform has been implemented and tested with a limited amount of services and concepts of the ontologies.

As future work we have: *(i)* definition of techniques, guidelines and tool support for client's goal specification and domain specification based on our goal-based service ontology; *(ii)* use of model transformation techniques for automatic transformation of goals and tasks models into service requests; *(iii)* test the platform with more complex domains and larger amount of services; *(iv)* definition of evaluation criteria for the framework and; *(v)* comprehensive evaluation of the framework based on the defined criteria.

## References

[1] S. Arroyo, E. Cimpian, J. Domingue, C. Feier, D. Fensel, B. Knig-Ries, H. Lausen, A. Polleres, and M. Stollberg. Web service modeling ontology primer. W3C Member Submission, June 2005.

[2] G. Guizzardi. *Ontological Foundations for Structural Conceptual Models*. PhD thesis, University of Twente, 2005.

[3] G. Guizzardi, R. Falbo, and R. S. S. Guizzardi. Grounding software domain ontologies in the unified foundational ontology (ufo): The case of the ode software process ontology. In *1th Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS'2008)*, Recife, Brazil, 2008.

[4] M. N. Moghadasi, A. T. Haghighat, and S. S. Ghidary. Evaluating markov decision process as a model for decision making under uncertainty environment. In *Proceedings of the 2007 International Conference on Machine Learning and Cybernetics*, volume vol. 5, pages p.p. 2446–2450, 19-22 Aug 2007.

[5] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing research roadmap. Technical report, European Union Information Society Technologies (IST), Directorate D, 2006.

[6] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. Artificial Intelligence series. MIT Press, July 1994.

[7] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition edition, 2002.

[8] E. Silva, J. Martinez Lopez, L. Ferreira Pires, and M. van Sinderen. Defining and prototyping a life-cycle for dynamic service composition. In *Proceedings of the 2nd International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing (ACT4SOC 2008)*, July 2008.

[9] K. Sycara, M. Paolucci, J. Soudry, and N. Srinivasan. Dynamic discovery and coordination of agent-based semantic web services. *IEEE Internet Computing*, 8(3):66–73, 2004.

[10] M. Vukovic and P. Robinson. Goalmorph: Partial goal satisfaction for flexible service composition. In *Proc. International Conference on Next Generation Web Services Practices NWeSP 2005*, page 6pp., 22–26 Aug. 2005.

[11] K. Zhang, Q. Li, and Q. Sui. A goal-driven approach of service composition for pervasive computing. In *Proc. 1st International Symposium on Pervasive Computing and Applications*, pages 593–598, 3–5 Aug. 2006.