Distributed workflows: the OpenKnowledge experience

Paolo Besana¹, Vivek Patkar², David Glasspool¹, Dave Robertson¹

¹ University of Edinburgh, ² UCL Department of Oncology

Abstract Software systems are becoming ever more complex, and one source of complexity lies in integrating heterogeneous subsystems. Service Oriented Architectures are part of the answer: they decouple the components of the system. However normally SOA is used in a centralised perspective: a single process invokes remote services, unaware of being part of a workflow. We claim that the centralised, or orchestration-based, approach cannot scale well with increasing complexity and heterogeneity of the components, and we propose an alternative distributed, or choreography-based, approach, that forces developers to think in terms of actors, roles and interactions. We first present the OpenKnowledge framework, designed according to choreography-based principles and then show how a complex, distributed model for managing the triple assessment of patients suspected with breast cancer can be easily implemented using this framework.

1 Introduction

Software systems are getting more and more complex. An important source of the complexity is the need to integrate different, heterogeneous subsystems. Service oriented architectures decouple the components of complex systems: every component exposes services that are accessible through the network using standard methods. Decoupling is a good software engineering practice, as it reduces the interdependencies between components and, if well designed, simplifies reusability of the services in different systems.

Complex systems can be pulled together, invoking services belonging to different and possibly external systems using workflow languages like BPEL or YAWL. These framework are based on a centralised, imperative paradigm: a central process controls everything, and the other services are passive, unaware of being part of a workflow.

We claim that this approach does not scale well with the growing complexity of systems: we advocate a different paradigm, based on the choreography of actors. We believe that the design of systems can gain from this paradigm, independent of the deployment technique. In fact, the choreography paradigm forces the developers to think in terms of the actors, their roles and their interactions in complex scenario, making them explicit.

The OpenKnowledge¹ project has allowed us to developed a fully distributed, peer-to-peer framework, focussed on shared interaction models that are executed

¹ www.openk.org
by peers. Based on this framework, we have tested the paradigm in different scenarios of varying level of heterogeneity and complexity. In this paper we will focus on a choreography-based implementation, based on the OpenKnowledge framework, of the assessment procedure followed by a patient suspected of breast cancer. In Section 2 we explain our claims about the choreography-based architecture, at the core of the OpenKnowledge project, presented in Section 3. In Section 4 we describe the triple assessment scenario, comparing the centralised and the distributed models. Finally, in Section 5 we show how we applied the OpenKnowledge framework to the assessment scenario.

2 Choreography- and Orchestration-based architectures

Our claim, as stated in the introduction, is that a choreography-based architecture forces the developers to think distributed applications from a different perspective that scales better with an increasing number of interacting actors. A distributed application becomes an set of interactions between actors that take different roles. The paradigm is independent from actual implementation and deployment: the implementation can be a complete peer-to-peer, fully open architecture, or a more traditional, closed architecture where every component is certified, and deployed by certified administrator, or it can be something in between. What is different is how the application is conceived, and the type of middleware that connects the pieces.

For example, an online pharmacy that provides a service for buying prescribed drugs can be designed using an orchestration-based language such as YAWL [10] or using a choreography-based approach. Figure 1 shows a YAWL workflow for the service. It runs as a central process started by the reception of a request from a customer. The process invokes a remote service for verifying the prescription, invokes another remote service for ordering the delivery of the drug and calls another service for charging the cost (it can be the national health service, a private insurance or a direct payment system). The roles are implicit: the focus is on the flow of activities. Figure 2 shows a distributed workflow for the same application: it focuses on the actors’ roles and on their interaction. The choreography-based approach forces us to analyse more explicitly the domain of the problem. In writing this simple example, I quickly found out that I needed to specify the ID of doctor to contact, and the ID of the funding body. Thinking about this, it was evident that the information was connected to the customer ID. The same information could be obviously defined in the orchestration-based architecture, but because the roles are implicit, the analysis is independent from the representation.

The choreography model, especially if implemented with an open, peer-to-peer architecture, requires designers to address issues of heterogeneity and brokering. Peers are likely to be different and they need to understand each other. The same services may be available from many peers, and the search and discovery process can be complex, especially if it needs to be perform at real time.

3 OpenKnowledge
The OpenKnowledge kernel [9] provides the layer that assorted services and applications can use to interact using a choreography-based architecture able to deal both with the semantic heterogeneity of the actors and with their discovery.

The framework allows a direct translation of a choreography oriented design, such as the activity diagram in Figure 2, into an executable application. The core concept is the interaction models, performed by different applications and service providers. These actors are the participants of the interactions, and they play roles in them. In an interaction all the roles have equal weight; the behaviour of all the participants and in particular their exchange of messages are specified. The roles in the interaction models are played by the participants, called peers.

Interaction models are written in Lightweight Coordination Calculus (LCC) [7,8] and published by the authors on the distributed discovery service (DDS) with a keyword-based description [5]. LCC is an executable choreography lan-
guage based on process calculus. An interaction model in LCC is a set of clauses, each of which defines how a role in the interaction must be performed. Roles are described by their type and by an identifier for the individual peer undertaking that role. Participants in an interaction take their entry-role and follow the unfolding of the clause specified using a combination of the sequence operator ('then') or choice operator ('or') to connect messages and changes of role. Messages are either outgoing to ('⇒') or incoming from ('⇐') another participant in a given role. A participant can take, during an interaction, more roles and can recursively take the same role (for example when processing a list). Message input/output or change of role is controlled by constraints defined using the normal logical operators for conjunction and disjunction. In its definition, LCC makes no commitment to the method used to solve constraints - so different participants might operate different constraint solvers (including human intervention). Figure 3 shows the LCC clause for the pharmacy role described in the interaction of Figure 2. The clause highlights how close is the LCC transposition from the specifications represented with a UML diagram.

The peers that want to perform some task, such as buying a medicine or providing prescription verification service, search for published interaction models for the task by sending a keyword-based query to the DDS. The DDS collects the published interaction models matching the description (the keywords are extended adding synonyms to improve recall) and sends back the list.

Interaction models and peers are designed by possibly different entities, and therefore the constraints and the peers' knowledge bases are unlikely to be perfectly corresponding. The heterogeneity problem is dealt splitting the task in three phases and limiting its scope. We have already seen the first phase, performed by the DDS: the interaction descriptions are matched using a simple query expansion mechanism. Then the peers compare the constraints in the received interaction models with their own capabilities, and finally the peers need to map the terms appearing in constraints and introduced by other peers [2]. The scope of the matching problem is limited to the specific interaction model in the second phase, and to the specific interaction run in the third phase.

The peer capabilities are provided by plug-in components, called OKC. An OKC exposes a set of Java methods that are compared to the constraints in the interaction models. The comparison is performed between the signatures of the constraints and of the methods, transforming them into trees and verifying their distance [3,4]. The signatures can be annotated with the semantics of each
Constraint annotations

@Annotation( @role(pharmacy), @variable(M),
  drug(name, dose) )
@Annotation( @role(pharmacy), @variable(P),
  patient(name, surname, date_of_birth, address(street, post_code)) )

Java method annotation

@MethodSemantic( language="tag",
  params="\{patient(family_name, birthday, street.post_code)\",
  "medicine(name, dose)\"")

public boolean deliverMedicine(Argument P, Argument M) {...}

Fig. 4. Annotations for the constraint deliver(M, D) and for a corresponding method

Fig. 5. Adaptor for constraint deliver(M, P)

parameter, which can be structured terms, as shown in Figure 4. The comparison process creates adapters, that bridge the constraints to the methods, as shown in Figure 5. An adaptor has a confidence level, that reflects the distance between the constraint and the best matching method: the average of all the confidences of constraints gives a measure of how well the peer can execute an interaction, and it used to select the most fitting one. Once the peer has selected an interaction, it advertise its intention of interpreting one of its roles to the discovery service by subscribing to it. Figure 6 shows the state of network when roles in an interaction are subscribed by at least one peer.

When all the roles are filled, the discovery service chooses randomly a peer in the network as coordinator for the interaction, and hands over the interaction model together with the list of involved peers in order to execute it.

The coordinator first asks each peer to select the peers they want to interact with, forming a mutually compatible group of peers out of the replies and making the task of selecting the best team for a task a distributed activity. The selection is not always necessary: peers can subscribe signalling that they interact with everybody.

While different implementations are possible, in the OpenKnowledge kernel the coordinator executes the interaction, instantiating a local proxy for each peer. The remote peers are contacted only to solve constraints in the role they have subscribed.

4 Medical guidelines

Gaps between medical theory and clinical practice are consistently found in health service research. Care procedures can often differ significantly between
different health centres, with varying outcomes for the patients, and many medical errors could be avoided if standard procedures were followed consistently. One of the causes of discrepancies in care is the difficulty in distributing and sharing efficiently the large amount of information that is continuously produced by medical research.

These issues have pushed the development of clinical practice guidelines. Several studies have shown that published guidelines can improve the quality of care. However, most such guidelines are provided as booklets, often hundreds of pages long, and covering relatively narrow fields or specific pathologies. Hundreds of guidelines are available, and generalist doctors are expected to be aware of, and follow, the guidelines relevant to each patient. The result is that guidelines are rarely followed, and inconsistencies in medical treatments are not reduced as much as hoped.

Information technology can improve the situation. Many clinical guidelines provide informal descriptions of workflows and rules that can be translated into formal representations, executable by machine, and research has suggested that computerised clinical supports can improve practitioner compliance with the guidelines [1]. One such formal model, of a diagnosis workflow for patient suspected of breast cancer, is presented in [6].

4.1 Breast cancer

Breast cancer is the most commonly diagnosed cancer in women, accounting for about thirty percent of all such cancers. One in nine women will develop breast cancer at some point in their lives. In the UK, women with symptoms that raise suspicion of breast cancer are referred by their GP to designated breast clinics in local hospitals. To increase the accuracy of diagnosis, a combination of
clinical examination, imaging and biopsy - known together as triple assessment - is recommended for qualifying women.

The first element of triple assessment consists of gathering the patient details and clinical examinations, done by breast surgeon. If the clinical examination reveals an abnormality then the patient is referred to a radiologist for imaging which consist of either ultrasound, mammography or both. If either the examination or imaging findings warrant it then a specific type of biopsy is performed, either by a radiologist or a surgeon, and the tissue is sent to a pathologist for an examination. The collective results from all three tests influence the final management of the patient. A small number of "worried well" patients may not qualify for either imaging or biopsy and could be discharged straight away. As the entire clinical process is distributed among three different disciplines and involves a number of different clinicians, a very close co-ordination and good communication between those involved is essential for the smooth running of the clinic.

4.2 Centralised model

The model presented in [6] is designed according to a centralised principle, and the abstract workflow is shown in Figure 7. The workflow is executed on a server, and it is focussed on the activities to perform and their relations. As described in the previous section it involves different clinicians. However, this is not explicitly represented in the model; roles are enforced by requiring different permissions to access the activities. An activity queues a task to the todo list of a user, who finds it when they log in. When they finish the task, the activity is marked as completed and the workflow proceeds.

![Figure 7. Centralised representation of the triple assessment.](image)

4.3 Distributed model

The distributed nature of the triple assessment procedure is not very clear from the centralised workflow of Figure 7. A more thorough analysis is shown in the the UML activity diagram of Figure 8. The four main participants: the breast surgery service (BSS), in charge of the first three activities in the workflow in Figure 7, the breast imaging service (BIS), responsible for the imaging decision and for the two alternative possible examinations, the breast pathology service (BPS),

---

3 A demo is available at: http://www.acl.icnet.uk/lab/tallis/Sample06_Triple_Assessment_workflow.htm
in charge of the biopsy, and the multi-disciplinary team (MDT), responsible for the final decision together with the surgery service.

5 Triple assessment in OpenKnowledge

The aim of implementing the triple assessment process using the OpenKnowledge kernel was to obtain a proof of concept: verify if LCC was expressive enough for the task, and determine how the choreography approach influences the design of the system. Given the similarity between the activity diagrams as specification language and the LCC formalism, the task was rather straightforward.

The first step was the conversion from the activity diagram in Figure 8 into a LCC protocol. The work of extracting the activity diagram from the flowchart of the activity had already cleared the identity of the actors in the interaction. However, the activity diagram in the figure is the result of working on a specification intended for a centralised model and thus it includes only actors whose activities are represented in workflow. The conversion of the diagram into an LCC interaction model showed the need to include the patient as a participant.

In the test, the interaction model is published on the DDS by the peer playing the breast surgery service role, that directly subscribes to it after receiving
acknowledgement of the publication. The other peers search for interaction models for triple assessment, and receive the interaction previously published by the BSS. They compare the constraints in the role they want to perform with their local OKCs, and if the match is good enough, they subscribe. In complex interactions like this a role may include many subroles, some shared with other roles: a peer needs to be able to solve the constraints in all the subroles reachable by the entry role. Figure 9 shows a fraction of the exchanged messages in a run of the interaction.

In the real world situation, the services subscribe first - and more than one service can subscribe to the same role - and the patient, or possibly her generalist doctor, subscribes to the patient role when required to go through an assessment after being identified as at risk. The other services remain always subscribed: every time a new patient subscribes, a new interaction is started. The patient is able to select the services she wants to interact with, possibly based on geographical distance or previous experiences.

As we said in Section 3, the way constraints are solved is independent from the kernel: a method in an OKC can simply be a wrapper to a web service, a query to a database, a call to an external legacy application, etc. The execution of the constraints is asynchronous: the coordinator of the interaction sends a message to the peer currently active, that forwards the request to the OKC bound to the constraint. As we said earlier, the binding is found during the comparison between interactions and available OKCs.

The peers can be stand-alone applications on the practitioners’ computers, that show messages to the users to alert them of an incoming patient, or wait for the users to fill up partially pre-compiled forms. Because solving constraints is decoupled from the interaction models, the peer could also be a servlet running on a web server, similarly to the current implementation of the process: the user logs in and finds the task to perform, such as a form to compile, or an update to her calendar. Similarly, the peer can be a process on a server, and the constraints can be solved sending SMS messages to a mobile phone.

6 Conclusion

In this paper we have seen how the design and implementation of complex systems, such as the application for managing patients with suspected breast cancer,
can gain from a distributed, choreography-based paradigm. While the paradigm is independent of the actual implementation and deployment, OpenKnowledge provides an operational framework for quickly setting up distributed systems. In OpenKnowledge systems are composed around interaction models, that coordinate the peers' behaviours by specifying the roles they can take, the exchange of messages between the roles and the constraints of the messages. Peers participate in the interaction taking one (or more) roles: in order to participate they need to compare the constraint in the roles with their available services and subscribe to the interaction on a distributed discovery service, that initiates interactions when their roles are filled.

As shown in the medical example, interaction models map rather accurately their specification as activity diagrams, making the implementation straightforward. Moreover, discovery of interaction and semantic matching between constraints and peers services allows reusability of components.

References


