Dynamic orchestration of distributed services on interactive community displays: the ALIVE approach

Gómez-Sebastià, I.*, Manel Palau, Juan Carlos Nieves*, Javier Vázquez-Salceda* and Luigi Ceccaroni****

* Universitat Politècnica de Catalunya, C/Jordi Girona 1-3, E08034, Barcelona, Spain, e-mail: {igomez, jcnieves, jvazquez}@lsi.upc.edu

** Tech Media Telecom Factory, C/Marina 16-18, Torre Mapfre 28 A E08005 Barcelona, Spain, e-mail: {manel.palau, luigi}@tmtfactory.com

Abstract Interconnected service providers constitute a highly dynamic, complex, distributed environment. Multi-agent system design-methodologies have been trying to address this kind of environments for a long time. The European project ALIVE presents a framework of three interconnected levels that tackles this issue relying on organisation and coordination techniques, as well as on developments in the Web-services world. This paper presents initial results focused on a high-tech, real use case: interactive community displays with touristic information and services, dynamically personalized according to user preferences and local laws.

1 Introduction

In our cities, urban information services are provided in ways which have not changed much in a century: bus-stop shelters, metro-stations panels and screens, maps, and urban furniture. This scenario brings up the opportunity to improve the information services that a city provides to citizens and tourists not only making it dynamic but also with the novel possibility of providing ubiquitous access to informational and other services. In this paper we will focus in a specific scenario, played in a dynamic and changing environment. The scenario proposed shows the difficulty to offer (on real time) personalized recommendations adapted to user profiles and contexts.

We propose a solution to cope with this scenario by means of intelligent agents, capable of reorganising themselves to compose and control the execution of dynamic and flexible services. The technologies and mechanisms concerning coordination and cooperation among agents (such as communication, negotiation and monitoring techniques) are key in order to get the right content with the right agent and in the right location. Furthermore, in open and complex agent societies [10], concepts such as organisational meta-models and self-organisation [7], trust,

reputation, norms, obligations and joint-intentions [6] are required in order to effectively model uncertain and competitive environments.

The ALIVE framework (see Section 2) presents a new approach to the engineering of distributed software systems based on the adaptation of coordination and organisation mechanisms, often seen in human and other societies, to service-oriented architectures. In this paper we show how this framework facilitates the interaction between content and service providers (e.g., city administrators, advertisers, restaurants), and service integrators in the development and maintenance of distributed systems. Section 3 of the paper presents a scenario based on the dynamic orchestration, organisation and coordination of services: a personalised recommendation tool that brings city services closer to residents and tourists by interconnecting service providers, locations and people. Finally, in Section 4 the proposed system is compared with existing work.

2 The ALIVE Architecture

New generations of networked applications, based on the notion of software services that can be dynamically deployed, require profound changes in the way in which software systems are designed. New approaches are needed, which allow integrating new functionalities, and thus new services, into existing running systems of already active, distributed and interdependent processes.

The ALIVE framework combines *Model Driven Design*[1] (MDD) with coordination and organisational mechanisms, providing support for "live" (that is, highly dynamic) and open systems of services. ALIVE's multi-level approach helps the maintenance of distributed systems by reorganising and adapting services. This is achieved via a set of intelligent agents responsible of adding control and flexibility to service composition and execution, effectively providing intelligent service orchestration. These agents are fully aware of the organisation's structure and goals, and thus can reflect them on service composition and execution, such as serving with higher priority requests coming from an actor higher on the organisation's hierarchy.

2.1 ALIVE Levels

The ALIVE framework extends current trends in engineering by defining three levels (see Fig. 1) as follows.

[1] See [http://www.omg.org/mda/], visited on October 29, 2008

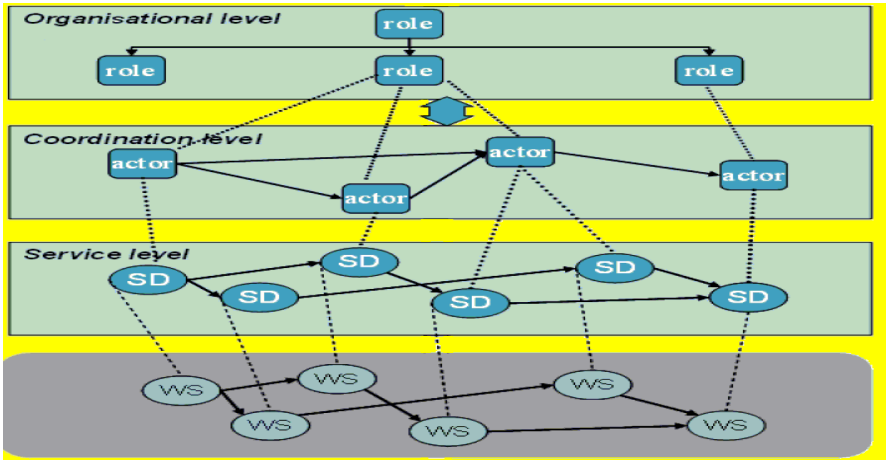

Fig. 1. The ALIVE architecture

A *Service Level* augments and extends existing service models in order to make components aware of their social context and of the rules of engagement with other services by making use of semantic Web technologies. This comes in handy when highly dynamic and always changing services (WS in figure) are present in the system, as the *meta-information* contained in each service description (SD in figure) eases the process of finding substitute services if a given service does not respond to invocations.

A *Coordination Level* provides the means to specify, at a high level, the patterns of interaction between services, using a variety of powerful coordination techniques from recent European research in the area [5, 11]. This is very useful when the system has to react to failures or exceptions inherent to the main processes (*e.g.*, failing payment or booking systems).

Finally, an *Organisational Level* provides context whether directly or not for the Coordination and the Service Levels, specifying the organisational rules that govern interaction and using recent developments in organisational dynamics [16] to allow the structural adaptation of systems over time. This is important when frequent changes on rules and restrictions are expected, as adapting the other two levels without this information would be a tough task.

The ALIVE multi-level framework is specially fit for scenarios where changes can occur at either abstract or concrete levels, and where services are expected to be continuously changing, with new services entering the system and existing services leaving it at run-time. For example, when there is a significant change on a high level (*e.g.*, a change in the organisational structure), the service orchestration at lower levels is automatically reorganised. Another example is the automatic adaptation of higher levels when lower ones suffer significant changes (*e.g.*, the continuous failure in some low-level services).

3 Application of the ALIVE framework

In order to demonstrate the applicability of the ALIVE architecture, we describe a scenario about multimedia distribution of information and orchestration of services, where *interactive community displays* (ICDs) [3] are used by TMT Factory (a software company) to provide a hyperlocal Web[2] in urban environments. The services and information provided, and how users' information is stored, processed and distributed, are subjected to various municipal, national and European regulations. The challenge is to achieve the interaction among content providers, service providers and integrators, to provide personalized information and recommendations.

For this scenario we consider the case of an underage user interacting with an ICD in search of entertainment and cultural options around the city. In this case, the system identifies (if possible) the user; gathers ratings and reviews about available services, along with user preferences and location; and finally offers personalized recommendations. In the following sections details are given about the system's model and operation.

3.1 System model and components

In the simplified model in Fig. 2, general components are shown together with components specific to the case, such as a map system, an information finder, museums, public transport information, the cinema reservation system and a public museum information service, provided by the city. It is important to notice the coexistence among local services and services offered via the Internet, and among public and commercial information.

Fig. 3 depicts the main components of the ALIVE architecture. In the organisation level, the *Domain Ontology* includes all domain-specific terms in the system, such as "Cinema" or "Museum". The *Organisational Model* (Org Model) describes the role each stakeholder in the system should fulfil. This model can be maintained by a set of tools, including a *Model Checker*, which can verify the consistency of the model (if consistency is important). The *Utils* component receives high-level exceptions and handles them at run-time, updating the organisational model as required.

In the coordination level, the *MAS Generator* component receives information from the *Utils* component to (re)generate the agents that will populate the *MAS*. In the scenario presented, entities such as "Cinema agent & transport" or "Map agent" will be generated. These entities are the agents responsible of monitoring service executions and composing available services when required. The *Work-*

[2] The difference between the semantic Web and the hyperlocal Web is that the ontologies of the latter one contain not only semantic information, but also geographic coordinates.

flows component contains the workflows to be executed by the agents when performing individual tasks, including workflows such as "Cinema reservation" or "Museum information composition". The *Monitor* keeps track of the execution of the workflows to detect anomalies or deviations from expected behaviour as soon as possible, enabling the system to react to them.

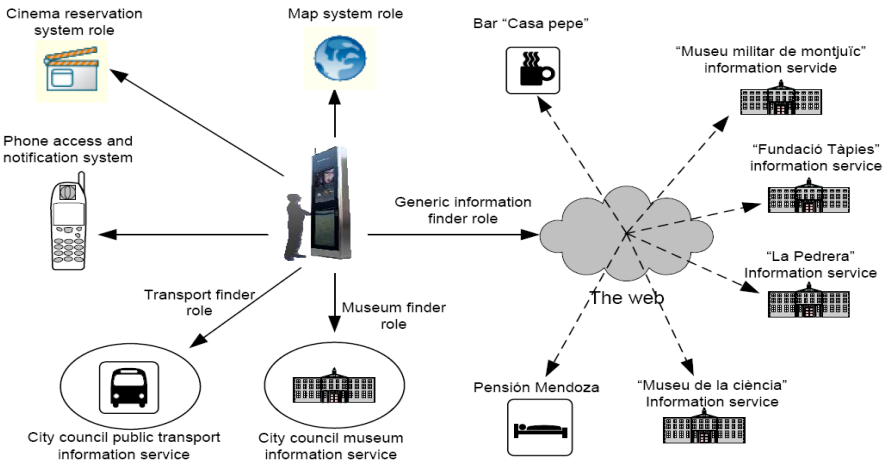

Fig. 2. System organisational model

In the service level the *Execution Engine* is used by agents to execute the workflows. The *Workflow Builder* allows dynamically (re)creating workflows that orchestrate the execution of the agents in order to reach some organisational goal. *Service Templates* are made available for the prototypical services to be offered; in this scenario, this component contains, for instance, templates for the "Cinema" service that enable the system to make ticket reservations on different theatres, regardless the concrete booking service.

3.2 Use case scenario in detail

In the scenario presented, if the user wants to visit some museums, she can ask the system for museum information, including a brief description and location on the map. In the current organisational structure of the system, this corresponds to the *museum information searching* task of the *museum information finder* role, which is performed by an agent, residing at the coordination level, which invokes the City council's museum information service through the service level. Let us suppose, however, that this service has become unavailable, so in order to be able to retrieve museum information upon the user request, the system has to find alternative ways of performing the task that was assigned to the *museum information finder*.

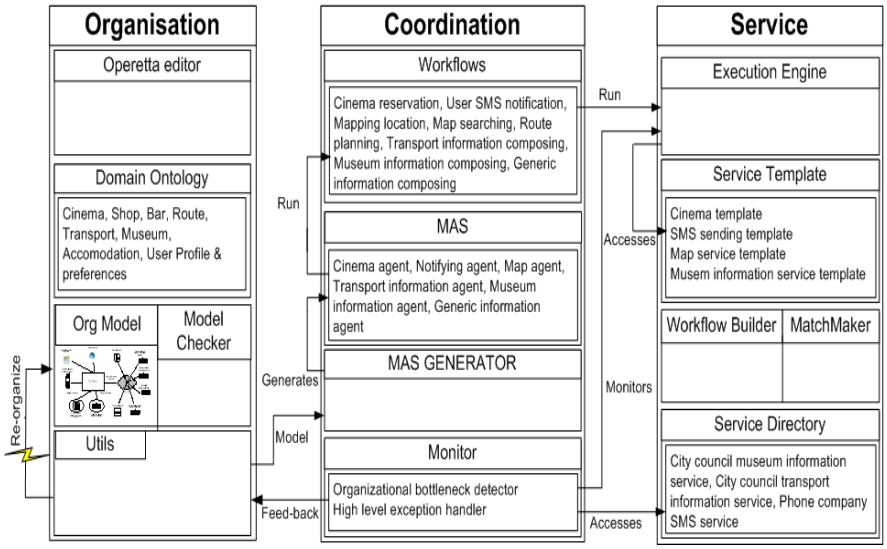

Fig. 3. Main components of the ALIVE architecture used in the scenario

As there is no alternative service to be invoked in the current workflow in the service level, an exception is passed to the *Monitor* at the coordination level, which looks for alternate ways of enacting the workflow. In this case, there are no such alternatives, though. Consequently, the notification of the exception goes up to the organisational level, which reorganises the model in order to perform the task requested by the user, triggering changes in both coordination and service levels.

After these changes, the system responsible of retrieving museum information is an agent able to gather and compose museum information from data found on the Internet. As a result, the actor responsible of enacting this role has changed, and the new actor, residing at the coordination level is invoked. This actor requests the service level to find out concrete museum services, access them, and come back with the required information. Then, the agent is able to compose (i.e. put together and format) this information, providing opening times, prices and brief descriptions for each available museum. When the information is composed, it is returned to the user, who gets what she requested even if one of the core components of the system was not available (see Fig. 4).

Once the user has information about all museums, she requests the system to locate them on the map. The components at organisational level look for an actor responsible of performing this task: in this case, an agent which can generate queries to existing map servers. This agent, residing at the coordination level, requests a map of a given location to the service level. The service level searches for existing services which implement the mapping service template. Three services are found: one specialised in road maps, which does not have accurate maps of most

museums' areas; a second one that requires a payment for the maps; and a third one which is suitable for the required tasks.

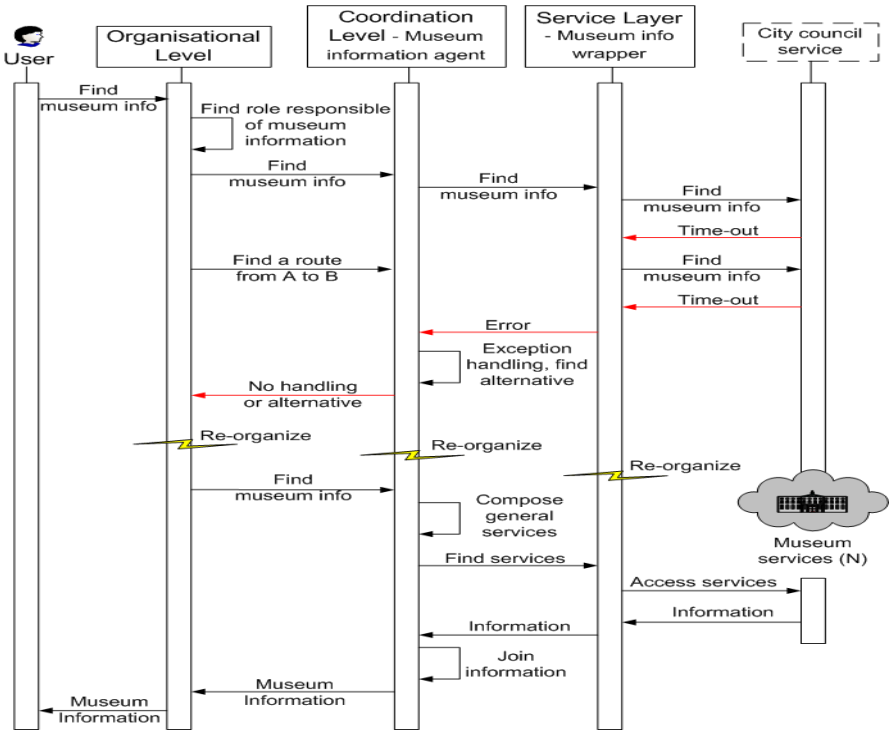

Fig. 4. Part of the scenario execution workflow showing reorganisation at all levels

After checking the museums information, the user decides that she would rather leave museum visits for another day, and go and try the new 3D-cinema instead. Therefore, the user asks the system to book two tickets. As this entertainment facility is quite far from the user's actual location, the system suggests a route on public transportation to the cinema.

For this, the organisational level finds the actor responsible of performing this task. In this case, it can be an agent for booking the cinema and finding a route. This agent finds out the actions required to enact the workflow associated to the requested task, and realizes that, for this, it needs to access a cinema booking service on the Internet, and request two tickets. Consequently, it commands the service level to perform this task. Once it is done, it needs to find a route from user's location to the cinema via a public transportation system, so the next task in the workflow is to ask the service level for information about such system. Once the service level provides this information, the agent is able to find out means of reaching the cinema. The user is presented this information on the ICD screen: the booked session, the bus to be taken, buses' route, and the nearest bus stop on a

map of the zone. At this point, the user can log out of the system and head to the bus stop.

Later on, the system might realise that changed traffic condition would delay the bus long enough for the user to miss her session. If necessary, it proceeds in cancelling current cinema reservation, and informing the user about the situation via SMS. Then, the user can, for instance, reply to the SMS choosing to book a ticket for the following session and ask for some bars or shops near the cinema to pass the time before the newly booked session starts, receiving all this information on her cell-phone.

As shown in the scenario description, the coordination level, via its monitoring system, can detect that a given workflow is continuously failing or been executed in a rather inefficient way due, for instance, to the presence of one agent responsible of too many tasks which is becoming a bottleneck. In this case, the coordination level can suggest changing the organisational level in order to reallocate task responsibilities to avoid such bottleneck.

4 Related work

Main lines of research about incorporating agent-based methods and techniques into Web-service contexts [14] include the extension of communication with more flexible patterns or protocols [1], which can support service-to-service negotiation [4, 11] and dynamic service composition. For the latter, most of the proposed approaches are based on either pre-defined workflow models or in AI planning techniques.

In the case of the pre-defined workflow models approach, a composite service is pre-defined by the designer by means of abstract workflows where nodes are not bounded directly to services but to search recipes [2, 15]. At runtime these recipes are used to find and bind to concrete services. In our solution we use a similar approach by means of the Service Templates, which are used to dynamically bind the concrete services at runtime. The difference is that workflows do not need to be predefined, but can be dynamically generated from the organisational objectives.

In the case of the AI planning approach, existing methods tend to be based on the assumption that each Web service method is an action or task which alters the state of the world. Typically, input and output parameters in service methods act as preconditions and effects. Examples of this approach include the use of existing planning languages, such as Golog [12] or SHOP2 [18], or the synthesis of services based on theorem proving methods [17]. An important remark to be done is that most approaches use centralised planning, where a specific component (the planner or the theorem prover) is the one that dynamically generates the full plan based on its (centralised) perceptions of the state of the world. In the ALIVE approach we make use of distributed planning, based on the TAEMS framework [9],

where the agents at the coordination level build partial plans that are then shared to generate a partial global plan.

An extra distinction of our approach over existing ones [13] is that it provides organisational context (mainly organisation's objectives and structure) that can be used when selecting and invoking services, providing an organisation awareness to some components (such as agents on coordination level or matchmaker on service level), which can direct the decision making in order to move towards the higher-level organisational objectives. One of the effects is that exceptions can be managed at the low-level, searching for a service (or composition of services) to substitute services that happen to be non-operational. This kind of exception handling is common in other SOA architectures. However, exceptions can also be managed at the high-level, looking for alternative ways to fulfil a workflow, or executing some tasks when certain states are reached. This high-level exception handling is not commonly seen in other SOA architectures.

5 Conclusions

The ALIVE framework aims to design and develop distributed systems suitable for highly dynamic environments, based on model-driven engineering, and three interconnected levels: service, coordination and organisation. The project fulfils the needs of TMT Factory (a software company) for more flexible, adaptive technologies to be used in interactive community displays, focussing on providing an easy to maintain system.

Due to the connection among levels, a change in the service level can trigger changes on both coordination and organisation levels; whereas a change in the organisation level can also trigger changes in the other two. These changes can be automatically carried out (with the assistance of ALIVE tools). This provides the system with high dynamicity, making it capable to react to changes on the environment.

Through the monitoring and exception system in the coordination level, workflow errors can be detected and treated properly. As shown in a use case, this provides the system with high stability, enabling it to fulfil its goals when a core component fails. The fact that service templates can abstract the service level and organisation level from concrete Web services makes workflow specifications more stable through time, whereas service discovery provides continuous system adaptation to a highly dynamic environment.

Finally, the high-level organisation, connected to the coordination and, indirectly, to the service level, ensures system's compliance with organisational structure, as available services will be composed and used according to organisation objectives, roles, norms and restrictions.

Acknowledgments This work has been partially supported by the FP7 project ALIVE IST-215890, which is funded by the European Commission. The authors would like to acknowledge

the contributions of their colleagues from the ALIVE consortium (http://www.ist-alive.eu). The views expressed in this paper are not necessarily those of the ALIVE consortium.

References

1 Ardissono, L., Goy A., Petrone, G.: Enabling Conversations with Web Services. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 819-826. Melbourne (2003)
2 Casati, F., Ilnicki, S., Jin, L.: Adaptive and Dynamic Service Composition in eFlow. In: Proceedings of the 12th International Conference on Advanced Information Systems Engineering. Springer, Stockholm (2000)
3 Ceccaroni, L., Codina, V., Palau, M., Pous, M: PaTac: Urban, ubiquitous, personalized services for citizens and tourists. To appear in: Proceedings of the Third International Conference of Digital Society. Cancun (2009)
4 Ermolayev, V. et al.: Towards a Framework for Agent-Enabled Semantic Web Service Composition. Int. J. of Web Services Research, 1(3):63--87 (2004)
5 Ghijsen, M., Jansweijer, W., Wielinga, B.B.: Towards a Framework for Agent Coordination and Reorganization, AgentCoRe. In: Coordination, Organizations, Institutions, and Norms in Agent Systems III. LNCS, vol. 4870, pp. 1--14. Springer, Heidelberg (2008)
6 Jennings, N.R.: Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions. Artificial Intelligence, 75 (2), pp. 195-240 (1995)
7 Juan, T., Sterling L.: The ROADMAP Meta-model for Intelligent Adaptive Multi-agent Systems in Open Environments. In: Agent-Oriented Software Engineering IV. LNCS, vol. 2935, pp. 231--253. Springer, Heidelberg (2004)
8 Klusch, M.: Semantic Web Service Coordination. In: Helin (eds.) CASCOM - Intelligent Service Coordination in the Semantic Web. WSSAT, pt. I, pp. 59--104. Birkhäuser Basel (2008)
9 Lesser, V. et al.: Evolution of the GPGP/TÆMS Domain-Independent Coordination Framework. In: Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems, vol. 9, pp 87–143. Kluwer Academic Publishers (2004)
10 Luck, M., McBurney, P., Shehory, O., Willmott, S.: Agent Technology: Computing as Interaction. A Roadmap for Agent Based Computing. AgentLink III (2006)
11 Matskin, M. et al.: Enabling Web Services Composition with Software Agents. In: Proceedings of the Conference on Internet and Multimedia Systems, and Applications. Honolulu (2005)
12 McIlraith, S., Cao Son, T.: Adapting Golog for Composition of Semantic Web Services. In: Proceedings of the 8th International Conference on Knowledge Representation and Reasoning. Toulouse (2002)
13 Papazoglou, M.P., van den Heuvel, W.J.: Service oriented architectures: approaches, technologies and research issues. J. VLDB , vol. 16, pp. 389--415 (2007)
14 Singh, P., Huhns, N.: Service-Oriented Computing Semantics, Processes, Agents. Wiley (2004)
15 Thandar, M., Edmond, D.: The use of patterns in service composition. In: Web Services, E-Business, and the Semantic Web. LNCS, vol. 2512, pp. 28--40. Springer, Heidelberg (2002)
16 van der Vecht, B., Dignum, F., Jules, J., Meyer, Ch., Dignum, V.: Organizations and autonomous agents: Bottom-up dynamics of coordination mechanisms. In: The Fifth Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems. Estoril (2008)
17 Waldinger, R.: Web Agents Cooperating Deductively. In: Formal Approaches to Agent-Based Systems. LNCS, vol. 1871, pp. 50--262. Springer, Heidelberg (2001)
18 Wu D. et al.: Automating DAML-S Web Services Composition Using SHOP2. In: Proceedings of the 2nd International Semantic Web Conference. LNCS, vol. 2870, pp. 195--210. Springer, Heidelberg (2003)