Editor: **Terry R. Payne,** University of Liverpool, t.r.payne@liverpool.ac.uk

Adaptable, Organization-Aware, Service-Oriented Computing

Huib Aldewereld, *Utrecht University*
Julian Padget, *University of Bath*
Wamberto Vasconcelos, *University of Aberdeen*
Javier Vázquez-Salceda, *Polytechnic University of Catalonia*
Paul Sergeant, *Calico Jack*
Athanasios Staikopoulos, *Trinity College Dublin*

Service-oriented computing is the new wave emerging from maturing Web services and the adoption of elements from Semantic Web technology. More sophistication, in response to business requirements, does of course not make it easier to use or to control. In particular, business processes demand resilience and real-time adaptation in the face of changing business requirements, incorporation of alternative services, and finding suitable substitutes when those needed are unavailable.

The European Union-funded Alive project (http://www.ist-alive.eu) is prototyping ideas, driven by commercial and industrial use cases, that utilize research in organizational modeling, software agents, model-driven engineering, artificial intelligence, the Semantic Web, and Web services to construct tools and demonstrators to address these needs. This article outlines the Alive architecture for service-oriented computing, describes some of the innovative tools we have developed and illustrates it all with a detailed run-through from one of our use cases.

New Trends in Service-Oriented Applications

The promising advances in service orientation have spawned a vision for the technologies that will power the Internet over the next few years and the new functionalities they will require. From Web 3.0 to the future Internet, from cloud computing to the Internet of things, we can see an emerging trend of distributed networked applications, some of them based on software services. Such applications can be dynamically deployed, modified, and composed so as to create radically new types of distributed software applications that will shape the Web of the future.

To fulfill the vision, these applications must be able to communicate, reconfigure at runtime, adapt to their environment, and dynamically combine sets of simple, specialized, independent services into more complex, added-value business services. This requires profound changes in the way we design, deploy, and manage software systems, replacing existing waterfall-like engineering techniques with approaches that integrate functionalities and behaviors into running systems that consist of active, distributed, interdependent processes.

Approaches (and associated methodologies and tools) for designing and engineering the new generation of open software should, we believe, possess several key features:

- They should scale up to tackle large-scale applications consisting of hundreds or thousands of components.
- Due to the sheer size of the applications being built, they should support self-governing software—that is, the engineered applications should "look after themselves." Approaches should thus explicitly factor in feedback loops that enable the connection of runtime phenomena with design-time models and artifacts.
- To increase application transparency, approaches should accommodate humans in the feedback and governance loops, allowing for potential human intervention in the software governance processes.

1541-1672/10/$26.00 © 2010 IEEE
Published by the IEEE Computer Society

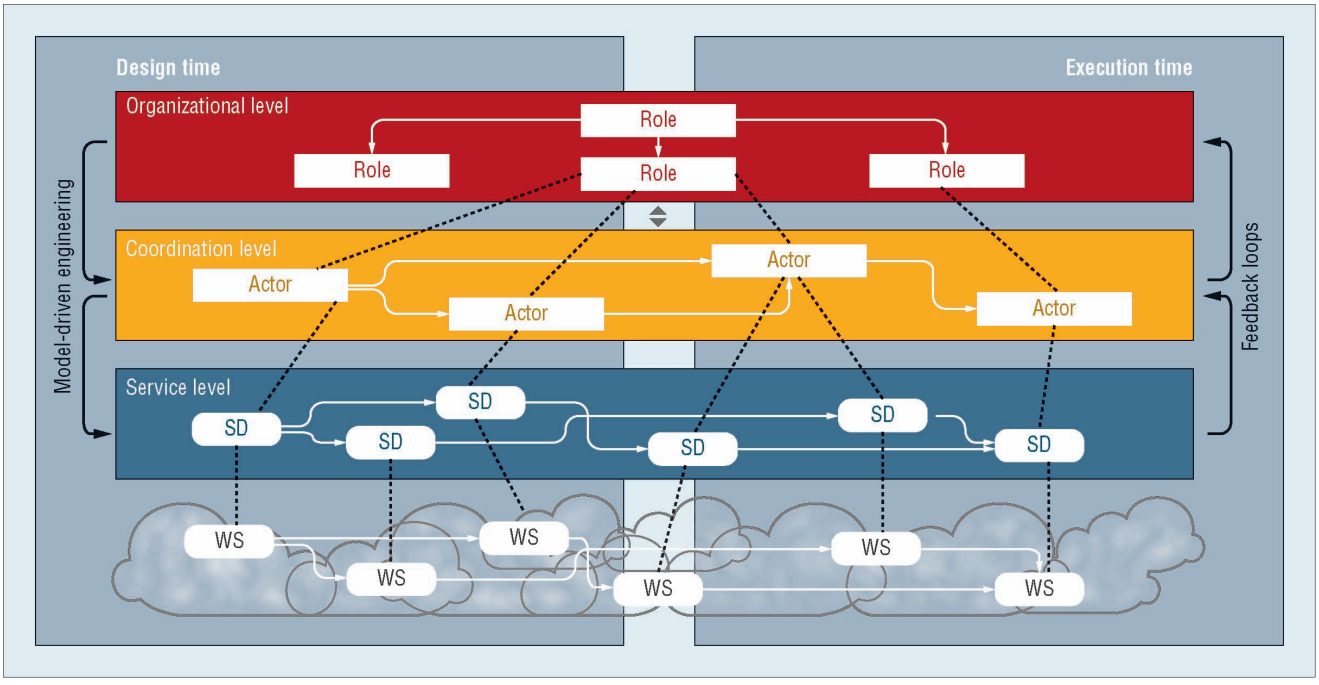

Figure 1. The Alive framework. We use model-driven engineering (MDE) to bind the service, coordination, and organization levels of distributed system design and management.

- Approaches should allow alternative points of entry in the design process, both to accommodate existing systems—and let developers add missing parts gradually—as well as different development styles and needs.
- They should provide a methodology to support and guide the use of (semi-)automatic tools.
- They should factor in and incorporate existing open standards, allowing for extensions to be easily integrated.

The goal of the Alive project is to put forward new solutions to address these challenges.

The Alive Approach

The Alive architecture combines model-driven development (MDD) with coordination and organizational mechanisms, providing support for highly dynamic and open systems of services. Alive's approach extends current trends in engineering by defining three levels in the design and management of distributed systems: the service, coordination, and organization levels (see Figure 1).

The *service level* supports the semantic description (SD) of services and the selection of the most appropriate Web service (WS) for a given task based on the semantic information contained in the service description. This effectively supports higher level, dynamic service composition. For highly dynamic services, the semantic description eases the process of finding equivalent services when a previously identified service is unavailable or when more suitable services are registered subsequently. This level is also responsible for monitoring service activity.

The *coordination level* specifies the patterns of interaction between services, transforming the organizational representation (including information flows, constraints, tasks and agents) coming from the organizational level into coordination plans. Using proven planning tools, we provide a mechanism to automatically synthesize (linear) plans to achieve organizational goals. We enact these plans in a distributed fashion, making use of the service level, and analyze via logs the plans already executed to give feedback to the planning process and subsequently the organization level.

The *organizational level* provides context for coordination and services through an explicit representation of the system's organizational structure. We achieve this by modeling the organizational stakeholders and their relationships. Hence, we can derive formal goals, requirements, and restrictions governing actors. Additionally, there are tools and mechanisms for the verifying and analyzing organizational specifications. We can also handle changes to the organizational structure (such as reorganization or changes to the rules of governance) arising from proposals by either the coordination or the service level. Finally, this level provides methods for norm-oriented organizational design, supporting flexible behavioral governance in scenarios where traditional approaches do not fit well.

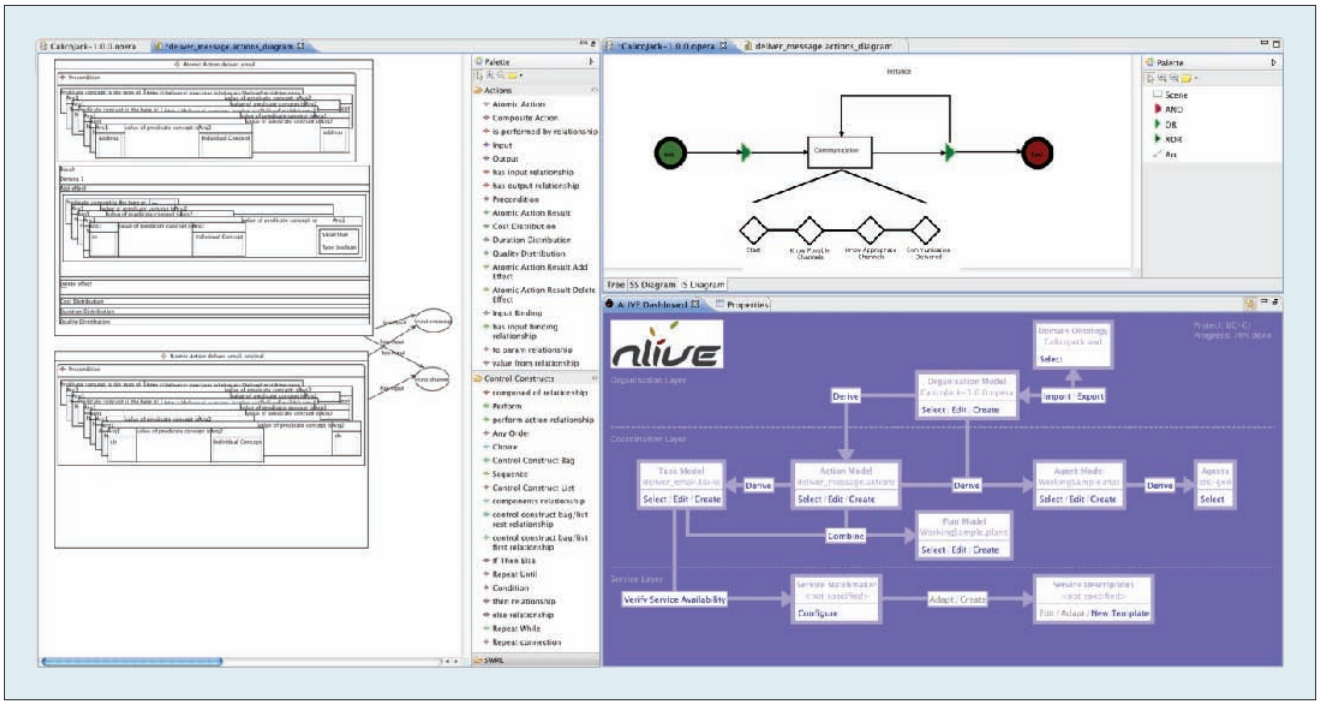

Figure 2. The ALIVEclipse design dashboard. The dashboard guides system engineers through the design and modeling processes. The landmarks of the communication process include start, know possible channels, know appropriate channels, and communication delivered.

We bind these three levels together using model-driven engineering (MDE), both at design time and execution time. This provides automatic transformations from the models at each of the three levels to multiple target platforms, realizing a form of feed-forward from design to implementation. At execution time, connections between the levels in the opposite direction, realize feedback from implementation to design, with events at the service level feeding into the coordination (leading to replanning) and organizational (leading to reorganization) levels.

This multilevel approach makes the Alive framework especially suitable for scenarios where changes are likely to occur at either an abstract or concrete level. It also meets the needs of highly dynamic services, with new services entering the system and existing services leaving it over the lifetime of the service composition's execution. For example, when there is a significant change in, say, the organizational structure, the service-level orchestration is automatically reorganized, effectively combining the existing services in new ways to reflect the organizational changes. Another example is the automatic adaptation of more abstract levels when more concrete ones suffer significant changes (such as due to the repeated failure of a service). Furthermore, the Alive framework lets more concrete levels adapt within themselves, while keeping the system's overall goals and objectives clear.

The Alive Tools

To support the Alive architecture, we used the Eclipse Modeling Framework to create a range of tools, which we have combined into a single package called ALIVEclipse. The toolset integrates design tools for model creation and the specification of system functionalities, coupled with the runtime tools required to deploy an Alive system. ALIVEclipse contains editors for modeling organizations (to provide context), actions (to indicate system capabilities), tasks (to indicate action relations and dependencies), plans (to represent abstract workflows), agents (to represent the acting components on the coordination level), and service descriptions (to detail the available functionalities provided by a service).

Figure 2 shows an example screenshot taken from the design phase. The development methodology is directly supported by a dashboard that guides system engineers through the design process (see the bottom-right pane). Because all the models in Alive are metamodel-based, translations between the different models are relatively easy. The toolset contains a full set of model-to-model transformations to generate parts of the models based on other models available. Also, the toolset contains model-to-text transformations to, for example, generate the code for the agents and services. The upper-right pane shows the interaction structure (a workflow on a high -level of abstraction that is part of the organization model). The left-hand pane shows a part of the

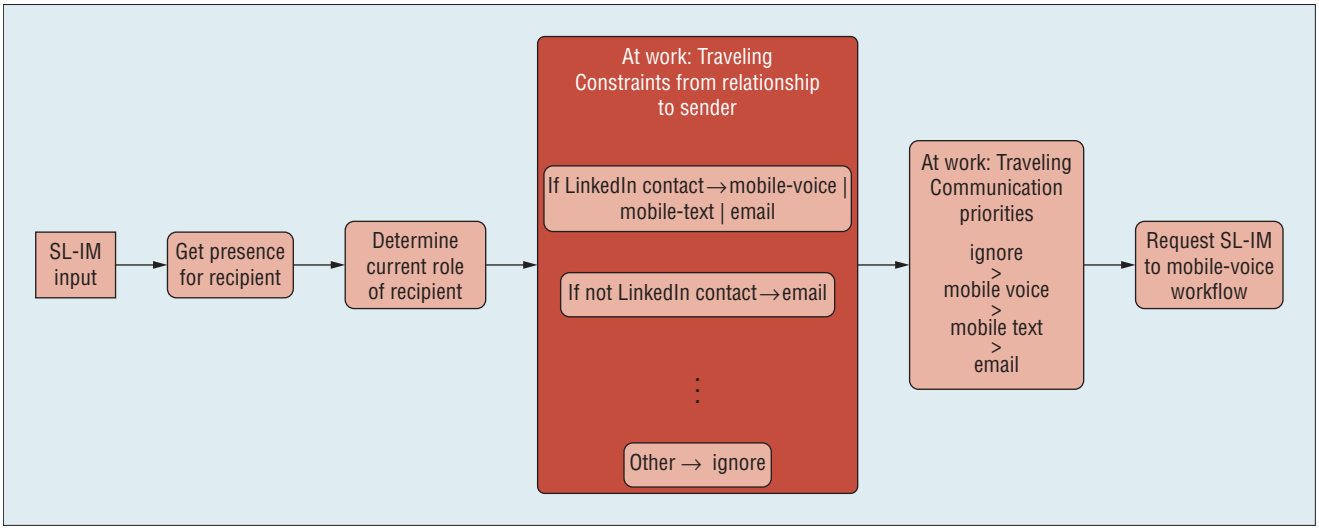

Figure 3. Service communications router. Dynamic reconfiguration allows for the expansion of communication landmarks. In this case, because Bob and Alice have an established LinkedIn relationship, his instant message (IM) will be rerouted to Alice's real-life email through his real-life email.

actions model, specifying the details of a deliver messages action.

For execution, ALIVEclipse supports the remote set-up, deployment, and running of Alive systems. Moreover, it contains components to evaluate and manage a running system by means of event logs and monitoring (to view the output of a running system) and the workflow visualization and analysis tools (to visualize and evaluate the system functionalities).

An Alive Scenario

To illustrate aspects of the Alive approach, we examine a service communications router (SCR) taken from a scenario provided by Alive partner Calico Jack. In the sample scenario, Bob is in Second Life. He wants to contact Alice so he uses Second Life's instant messaging (IM) facility. Alice is not currently in Second Life, but she is in Bob's LinkedIn network. She's working away from the office. How do we deliver the message?

At the organizational level, the SCR requires expressions of roles and access permissions for communication channels, such as those allowing in-game friends access to in-game voice over IP (VoIP) and IM, and real-world telecommunications. At the coordination level, the SCR requires reasoning to determine service selection for specific situations. This reasoning might introduce external resources, such as a representation of the user's diary and professional network. At the service level, the SCR requires specification of both in-game and real-world communication channels and redundancy—for example, using multiple SMS delivery services, IM, VoIP, email, and so on.

More concretely, at the organizational level in our example, Alice allows in-game friends to communicate with her via in-game IM, SMS (on her real-world mobile phone), and real-world email. At the coordination level, a communication request within Second Life from Bob to Alice is routed to Alice's alternative channels because she is not currently in Second Life. At the service level, if Bob cannot send Alice an SMS message via VoIP (for example, due to a lack of software credits), then a dynamic reconfiguration is required to reroute his IM to Alice through Bob's real-life email and on to Alice's real-life email (see Figure 3).

Our framework uses the recipient's presence information to determine his or her likely current role. This lets us use the normative roles governing communication with the recipient in that role (in this case working but traveling). In this example, the salient rules are based on relationships in the LinkedIn business social network—being a contact within this network will permit mobile communication while the recipient is in the role "At work: Traveling." Originators who are not LinkedIn contacts will only be permitted email communication.

As Figure 3 shows, once the set of possible communication channels have been determined (mobile voice, mobile text, and email), the most suitable becomes the highest prioritized option and then a request is made to construct a workflow for delivering the message from originator's SL-IM channel the recipient's mobile-voice channel.

The upper-right pane in Figure 2 shows the interaction structure diagram for this example and the expansion of the landmarks of the communication process. Landmarks, a key part of the modeling process, identify sets of conditions that hold at that point in the process. Figure 3 expands on these landmarks, capturing finer-grained actions, including the constraints on the communication imposed by the relationship between the sender and recipient.

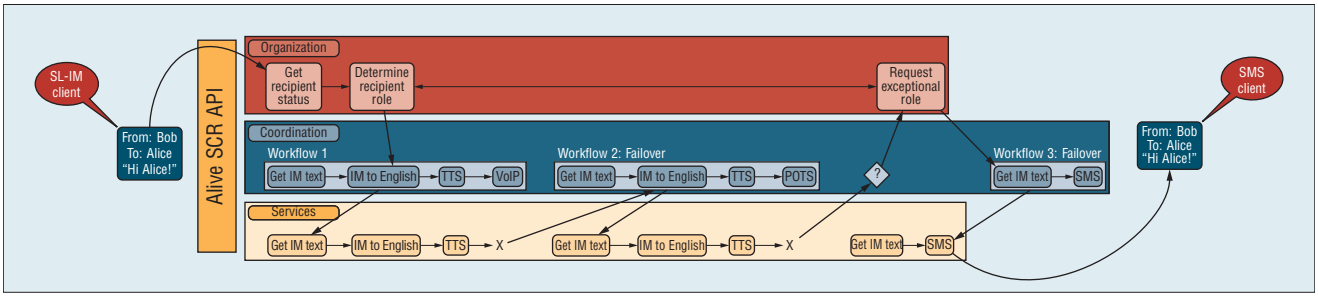

Figure 4. Service communications router. The Alive framework maps workflows onto a changing real-world environment.

Figure 4 illustrates how we map the requirements to the levels of the Alive architecture, starting from the high-level organizational workflow, the identification of candidate workflows at the coordination level, and finally the actual execution at the service level, including the fail-over from workflow 1 that uses VoIP to workflow 2 using plain old telephony services (POTS). Here an exception occurs because the recipient is away from the office, which the coordination level resolves using the SMS delivery workflow—which is permitted because Bob and Alice have a LinkedIn relationship—and the task is completed. This process shows how the Alive architecture demonstrates resilience and adaptation in mapping designs onto a changing real-world environment.

The Alive Consortium

The Alive consortium consists of the Polytechnic University of Catalonia (Spain), Utrecht University (the Netherlands), University of Bath (UK), Trinity College Dublin (Ireland), University of Aberdeen (UK), Thales B.V. Nederland (the Netherlands), Tech Media Telecom Factory SL (Spain), and Calico Jack (UK). In addition to the use-case scenario we describe here, the project explores interactive community displays to demonstrate dynamic orchestration of services for citizens and dynamic crisis management to evaluate crisis management policies. ■

Acknowlegdments

The Alive project (Coordination, Organization and Model Driven Approaches for Dynamic, Flexible, Robust Software and Services), FP7-215890, is funded by the European Union's Framework 7 Program (between February 2008 and October 2010).

Huib Aldewereld is a post-doctoral researcher at Utrecht University. Contact him at huib@cs.uu.nl.

Julian Padget is a senior lecturer at the University of Bath. Contact him at jap@cs.bath.ac.uk.

Wamberto Vasconcelos is a senior lecturer at the University of Aberdeen. Contact him at w.w.vasconcelos@abdn.ac.uk.

Javier Vázquez-Salceda is a post-doctoral researcher at the Polytechnic University of Catalonia (UPC). Contact him at jvazquez@lsi.upc.edu.

Paul Sergeant is the CEO of Calico Jack. Contact him at paul@calicojack.co.uk.

Athanasios Staikopoulos is a post-doctoral researcher at Trinity College Dublin. Contact him at athanasios.staikopoulos@cs.tcd.ie.