VERSAG: Context-Aware Adaptive Mobile Agents for the Semantic Web

Kutila Gunasekera¹, Arkady Zaslavsky¹, Shonali Krishnaswamy¹, Seng Wai Loke²
¹Faculty of Information Technology, Monash University, Australia
²Department of Computer Science & Computer Engineering, La Trobe University, Australia
{kutila.gunasekera, arkady.zaslavsky, shonali.krishnaswamy}@infotech.monash.edu.au, S.Loke@latrobe.edu.au

Abstract

Software agents roaming around and accessing services is an important part of the vision of the Semantic Web. The need to engage in diverse activities in rapidly changing environments makes it essential that these agents are able to adapt to varying situations. We propose a novel approach to engineer adaptive software agents for such scenarios. Our agents have the ability to exchange their capabilities with peers, support multiple forms of adaptation, enable software reuse through a component-based infrastructure and provide fine-grained and efficient agent mobility. We describe our solution, the first implementation and identify further research issues.

1. Introduction

Agents roaming around the Internet and accessing services was an important part of the original semantic web vision [1]. In an increasingly pervasive environment these agents have to roam on heterogeneous devices and platforms, some with severe resource restrictions. As it is not possible to predict in advance the environments it may have to encounter, an agent has to be flexible to adapt to different environments. This situation has brought about the need to build adaptive software agents. There are several complementary approaches in which software agent systems can be made adaptive. Multi-agent systems exhibit adaptive behaviour by their interactions and agents can adapt by learning new skills, as individuals or in groups. Also it is possible for agents to adapt by changing their internal structures and acquiring components. This compositional adaptation approach is the focus of this paper.

We propose a novel approach to engineer adaptive software agents. Our agents are autonomous entities which execute on heterogeneous platforms, acquire new software components at runtime, and adapt according to environmental conditions. This compositional architecture for agents allows an agent to acquire any particular behavioural model during its lifetime. The ability to change behaviour dynamically makes them more versatile compared to conventional agents.

In previous research which looks at compositionally adaptive agents, components are usually acquired from a designated provider service or a repository, and agents have limited control of the adaptation process. While in most cases agents adapt to carry out new functionality, in recent research we see agents adapting to suit new environments too. Further details of related work are described in [2]. We believe adaptation through component change is the most powerful since it effectively allows an agent to completely change its behaviour while maintaining identity. However, to be truly useful in unpredictable environments, agents should have more freedom in adaptation. Next, we describe our proposed solution which aims to bridge this gap.

2. Versatile Self-Adaptive Agents

VERSAG (VERSatile Self-Adaptive Agents) is a lightweight agent architecture where an agent is seen as a carrier of software components. Behaviours of an agent are implemented as portable software components which can be changed at runtime. Salient features of the proposed solution are: 1) the ability of agents to exchange behaviours with peer agents; 2) an agent’s ability to adapt based on contextual input and 3) component level mobility with efficient migration mechanisms. From an engineering point of view VERSAG can also be considered a layer built on top of existing agent toolkits which provides a “toolkit independent” approach to build agents.

We term the software components carried and exchanged by VERSAG agents as “capabilities”. A capability is formally defined as a tuple \(<id, F, credentials, env>\) where, \(id\) is a unique identifier, \(F\) is the set of functions in the capability, \(credentials\) represent meta-data about the capability and \(env\) is a set of environments the capability supports. A function can be either a primitive function or a nested function. A primitive function would be typically coded in a programming language whereas a nested function would be made up as a combination of other functions. Informally, a capability is a software component which can be attached to and detached from a software agent to provide the agent with a particular
application-specific behaviour such as a data mining behaviour or a BDI like agent model.

A VERSAG agent is guided by an itinerary which specifies the locations it has to traverse and the activities to carry out at these locations. The agent decides when and from where the necessary capabilities are acquired based on criteria such as availability, number of locations a particular capability is required at, network cost and resource constraints at locations. An agent also has the ability to pass on its capabilities to other agents when requested, making all VERSAG agents potential capability providers. Capability descriptions should allow agents to search for and reason about them and may include details such as security certificates, information about platform specific optimizations, processing needs etc. which would help the decision making process. We also note that an agent should be able to handle issues such as non-availability of a capability or location and task failures.

Capability exchange among peer agents has a number of advantages over runtime component loading from a designated provider. With all agents being providers, the need for a single high-performing provider is eliminated and an individual agent need not be as robust in its provider role as it is no longer the sole provider. Also, acquiring capabilities from nearby peers instead of distant providers reduces network traffic.

An agent has to be able to deal with changes in the environment. VERSAG agents are therefore envisaged as possessing context-awareness to guide their activities.

Capability carrying agents are an additional layer of code mobility on top of the agent toolkit which allows us to make agents adaptable and implements a finer-grained form of code mobility. There is scope for improvement on code migration strategies as research has identified that most existing agent toolkits implement simple and inefficient migration strategies [3]. We expect to use adaptive migration strategies for capability transmission in VERSAG agents, providing efficient migration on top of standard agent toolkits.

Security and trust issues, ever present in mobile agents, exist in VERSAG too. While the importance of these issues is acknowledged, we do not investigate them in detail as they are not within the scope of the research.

An implementation of VERSAG is in progress using the FIPA compliant JADE agent toolkit as basis. A VERSAG agent is a thin layer built on top of a JADE agent and could be ported to different agent platforms. Programmatically VERSAG agents are homogeneous as all agents are instantiated from the same class and differences arise from the carried components, which are seen as data.

An agent’s tasks are specified in an itinerary. To carry out these tasks, an agent requires different skills which are available in the form of capabilities. Agents are cooperative in that they are always willing to provide their capabilities to other requestors. Figure 1 below shows the structure of an agent.

Figure 1: The structure of a prototype VERSAG agent

The itinerary management module contains methods for parsing and executing an itinerary. The capability management module’s tasks are to locate and fetch capabilities from peers, execute them when requested by the itinerary management module and also to serve capability requests from peers. Capability discovery in the prototype is done with the JADE Directory Facilitator as a mediator. The capability repository is where an agent stores its capabilities. Importantly, capabilities are independent of the agent toolkit (JADE) and can be used on other toolkits or environments where necessary execution infrastructure is available.

3. Conclusions and Outlook

We have proposed a novel approach to develop context-aware adaptive mobile agents suitable for semantic web environments. The proposed versatile self-adaptive agents have the salient feature of being able to dynamically adapt by exchanging software components among peer agents. With the current prototypical implementation of VERSAG, we have successfully demonstrated the feasibility of capability exchange among agents and the potential to reduce network traffic by using localized capability exchange [2]. Our future work includes research on capability description languages, reasoning and discovery methods.

4. References