Agents and Workflow -- An Intimate Connection, or Just Friends?

Martin L. Griss (*moderator*)
Hewlett-Packard Laboratories,
Palo Alto, CA
griss@hpl.hp.com

Gregory A. Bolcer
Endeavors Technology Incorporated,
Irvine, CA
gbolcer@ics.uci.edu

Quiming Chen
Hewlett-Packard Laboratories,
Palo Alto, CA
qchen@hpl.hp.com

Robert R. Kessler
Department of Computer Science,
University of Utah, Salt Lake City, UT
kessler@cs.utah.edu

Leon J. Osterweil
Computer Science Department,
University of Massachusetts, Amherst, MASS
ljo@freya.cs.umass.edu

Abstract

This panel addresses the perspective that agents and workflow can be seen as an evolution of components and scripting. There are several ways in which agents can be used to perform or support a workflow, and several ways in which workflow can be used to orchestrate or control the interactions between agents. The panelists explore several of these connections and various applications of agents and workflow. They discuss some of these relationships between agents and workflow, and propose opportunities for research and practice.

1. Introduction: Components, Scripting, Agents and Workflow *(Griss)*

While there are many definitions of software agents[1,2,3], they can be usefully viewed as an evolution or combination of distributed objects[4], business objects[5] and scriptable components. Agents are components, sometimes mobile, that expose specialized interfaces, are usually driven by goals and plans (rather than procedural code), have "business" or "domain" knowledge, operate autonomously, and quite often interact using a specialized, declarative agent communication language (ACL), such as KQML (rather than direct method invocation). While agents often are used individually (e.g., internet information mining), things become much more interesting when a group or society of agents collaborate together to perform some task (e.g., auctions, or telecommunications provisioning). A number of different agent systems have been developed (e.g., Agent-TCL, Aglets, Voyager, …), focusing on different mixes of mobility, adaptability, intelligence, ACL and multi-language support.

Workflow is a method and mechanism to allow the explicit definition and control of a group of participants in enacting a process of some sort. A variety of different workflow languages and systems have been developed (e.g., HP ChangeEngine[6], Endeavors[7,8], Little-JIL[9,10]), many of which involve a graphical interface to describe allowed

connections between participants, desired and exceptional processing conditions, the assignment of roles, and the request and allocation of resources.

Scripting is an increasingly popular way of gluing components into complete solutions. Scripting languages are typically small, interpretive languages that have good access to the underlying component model (e.g. VB Script or VBA for COM components, Java or Java Script Java for JavaBeans, and TCL or Python). Developers and end-users can create applications by using scripting to combine, control or modify robust components created by domain experts; this is a key strategy that underlies the great appeal of Visual Basic in the enterprise.

Several researchers and practitioners have noticed several interesting connections between agents and workflow. On the one hand, an interesting way of creating a workflow system is to have agents represent the participants and resources, and to have the society of agents collaborate to enact the workflow. In some workflow systems, the participants (human or machine) are in fact referred to as agents. On the other hand, a convenient way of choreographing the interaction between collaborating agents in a multi-agent society is to use workflow as a form of next generation scripting language[11]. Either a new workflow/scripting language will be developed, or workflow primitives will be added to an existing scripting language, such as TCL, VB Script or Java Script.

This panel will discuss these and other relationships between agents and workflow, and propose some opportunities for research and practice.

2. Process Programming Languages As Agent Coordination Specification Vehicles *(Osterweil)*

We consider the definition or specification of a process to be an effective basis for the specification of agent coordination as well. Our view is that a process that is to be executed by the coordinated actions of a variety of agents can and should be specified in such a way that the individual steps of the process can be seen as being allocatable to the different agents. The Little-JIL process programming language demonstrates this[9,10]. In Little-JIL a process is specified as a hierarchical decomposition into steps and substeps. Each step is defined in terms of a variety of characteristics and attributes. But one of the most important of these definition elements is a specification of the type of agent that is to be employed to take responsibility for the execution of the step. In Little-JIL a runtime system is used to determine dynamically the specific agent that meets the specification associated with the step, and to bind that agent to execute the step. In this way, the Little-JIL process specification is also a specification of the way in which the available agents are to be coordinated to execute the defined process.

We have also related our Little-JIL process specifications to an established multiagent scheduling system, called GPGP, developed by the Multiagent Systems Laboratory at the University of Massachusetts. Using Little-JIL and its execution environment we are able to tentatively assign specific agents to the execution of specific steps. This is expected to cause many agents to each have several steps to execute. Because of this it will be useful to provide these agents with guidance about the order in which each should execute its assigned steps. In our work we have devised translation systems that convert Little-JIL process programs into representations in the TAEMS system, which is used as input to GPGP. The result is a set of specifications of the way in which each agent participating in the process is to execute the steps it has been assigned from the overall process. This serves to further

confirm our view that process specifications can be most effective bases for the determination of the ways in which agents should be coordinated.

We are in the process of evaluating these ideas in the context of a larger robot team coordination project, called SAFER, sponsored by DARPA.

3. Intelligent Agents Augment Flexible Workflow *(Bolcer)*

Data agents are small, autonomous software modules that automatically perform a localized task. Intelligent agents, in addition, have the ability to reason about the input data and manipulate it in such a way that it can work in conjunction with other agents without violating any global constraints. Examples of activities that data agents perform range from simple to complex. An agent's activities may include automatic notification via email of the availability of a report, sending a reminder of a scheduled meeting, advising a user of alternate meeting times as the result of a scheduling conflict, or even actively going out and doing research based on some specified criteria. Agent's activities, from the viewpoint of the workflow system, should allow the activity to be accomplished by either a human or non-human agent or both[7]. By allowing the mix of both human and non-human activities, the availability of more intelligent agents allows for the incremental automation of certain activities.

The people participating in the workflow should be allowed to hand off or delegate activities to an agent or take ownership of the activity back from one[8]. Similarly, agents should be allowed to delegate tasks to sub-agents or request the help of human participants as needed. The management of multiple agents allows end-users to accomplish tedious or repetitive tasks with minimal effort and avoids the "keep checking back" syndrome. Agents should allow customization by individuals with or without technical programming knowledge, notification based on existence and changes, constraints on values and appropriate strategies for resolution, and significant event filtering and topic interest registration. At some level of abstraction, an agent and an automated workflow process are interchangeable. Agents historically have been focussed on non-human involved, small-scale coordination activities.

4. Scriptable Agents for Distributed Measurement *(Kessler)*

Much of our work on distributed measurement involves the use of non-mobile agents deployed at startup to the measurement site[12]. These agents, implemented as COM or OCX components in the CWave system, utilize a publish/subscribe bus for communication between agents[13]. As part of this infrastructure, the agents include standard libraries of measurement objects and methods in support of the task at hand; in particular, support for loading and running multi-threaded scripts in VBScript, JavaScript or Perl. Measurements are then initiated, changed, and updated by dynamically sending scripts which are evaluated in the context of the local agent. The script is responsible for performing the measurements and providing any necessary actuation/control of the attached devices. Agents may be logically grouped together to provide a higher-level functionality, which is undertaken via communication and negotiation amongst the agents within the group.

The group communication and coordination in our measurement domain is very similar in style to workflow. We are considering extending our libraries and scripts with workflow primitives. In addition, currently all communication is directly "programmed" in compatible scripts with publishers and subscribers programmatically agreeing on what will be

communicated. This is a different style as compared with agent communication languages that explicitly indicate steps, allocate tasks to agent roles, and negotiate specific tasks and goals. We are exploring how to integrate both workflow primitives and agent communication performatives as an extension of our scripting language. These extensions could lead to an environment in which it in would be easier to set up workflow style collaborations using our scriptable agent technology.

5. Dynamic Agents for Dynamic Workflow Service Provisioning *(Chen)*

We claim that a dynamic-agent infrastructure can provide a shift from static distributed computing to dynamic distributed computing, and we have developed such an infrastructure to realize such a shift[14]. We shall show its impact on software engineering through a comparison with other distributed object-oriented systems such as CORBA and DCOM, and demonstrate its value in highly dynamic workflow system integration and service provisioning.

The infrastructure is Java-based, light-weight, and extensible. It differs from other agent platforms and client/server infrastructures in its support of dynamic behavior modification of agents. A dynamic-agent is not designed to have a fixed set of predefined functions but instead, to carry application-specific actions, which can be loaded and modified on the fly. This allows a dynamic-agent to adjust its capability for accommodating environment and requirement changes, and play different roles across multiple applications.

The above features are supported by the light-weight, built-in management facilities of dynamic-agents, which can be commonly used by the "carried" application programs to communicate, manage resources and modify their problem-solving capabilities. Therefore, the proposed infrastructure allows application-specific multi-agent systems to be developed easily on top of it, provides "nuts and bolts" for run-time system integration, and supports dynamic service construction, modification and movement. A prototype has been developed at HP Labs and made available to several external research groups.

References

[1] M.R. Genesereth, and S.P. Ketchpel: "Software Agents", Communications of the Association for Computing Machinery, July 1994, pp 48-53.

[2] M.N. Huhns and M.P. Singh, "Readings in Agents", Morgan-Kaufman, 1998.

[3] N.R. Jennings and M.J. Wooldridge, "Agent Technology," Springer, 1998.

[4] J. Odell, "Tutorial, Beyond Objects: Agents", OOPSLA98.

[5] O. Sims, Business Objects, McGraw-Hill, London, 1994.

[6] HP Changengine - an Enterprise Process Flow System. (See http://www.ebizsoftware.hp.com/main1.html).

[7] G. Bolcer and R. Taylor, "Endeavors: A Process System Integration Infrastructure," International Conference on Software Process (ICSP4), December, 2-6, 1996, Brighton, U.K. (See http://www.ics.uci.edu/pub/endeavors/endeavors.html).

[8] G. Bolcer and R. Taylor, "Advanced Workflow Management Technologies," Software Process: Improvement and Practice, Dewayne E. Perry and Wilhelm Schaefer, Co-Editors-in-Chief, John Wiley & Sons, Baffins Lane, Chichester, West Sussex, PO19 1UD, UK. http://www.wiley.com/journals/spip/ 1999 (to appear) .

[9] S.S. Sutton Jr. and L.J. Osterweil, "The design of a next generation process programming language," Proceedings of ESAC-6 and FSE-5, Springer Verlag, 1997, pp. 142-158.

[10] A. Wise, "Little-JIL 1.0 Language Report," Technical Report 98-24, University of Massachusetts at Amherst, Sept. 1998.

[11] M.L. Griss, "From components and scripts to agents and workflow," Technical report, HP Laboratories, June 1999 (in preparation).

[12] M.L. Griss and R.R. Kessler, "Building Object-Oriented Instrument Kits," Object Magazine, Apr 1996.

[13] C. Mueller-Planitz, CWave, a Visual Workbench for Distributed Measurement Agents, Ph.D. Dissertation, Computer Science Department, University of Utah 1999 (in preparation).

[14] Q Chen, P Chundi, U Dayal, and M Hsu, "Dynamic Agents for Dynamic Service Provisioning," accepted for publication, Intl. Conf. on Cooperative Information Systems, August 1998.

Biographies

Greg Bolcer received his Ph.D. in Information and Computer Science at the University of California, Irvine (UCI) in 1998, M.S. in Computer Science at the University of Southern California (USC) in 1993, and B.S. in Information and Computer Science at UCI in 1989. He is the CEO and Founder of Endeavors Technology Incorporated, a small California Internet startup dedicated to building tools supporting Web-based workflow, Web-based information systems, and e-business processes. Prior to founding ETI, Bolcer was the project leader for the DARPA funded Endeavors workflow research prototype. He is also one of the key working group participants for the Simple Workflow Access Protocol (SWAP) extensions to HTTP/1.1 and WebDAV. His current research interests include automating business-to-business electronic commerce processes and Internet-scale event notification.

Qiming Chen is a senior computer scientist in Software Technology Lab, HP Labs. He joined HP Labs in 1992, and since then he has conducted research in database, workflow, agent framework and data mining. He designed and developed a Java based workflow system and a dynamic agent infrastructure in HP labs. He was appointed to 13 international technical committees in USA, Europe and Japan, including program committees of 1986, 1989, 1990, 1991, 1992 International Conference on Very Large Databases (VLDB), and editorial committees of international journals and book series. He has authored over 70 technical publications in journals, books and international conferences on databases, knowledge bases and software engineering.

Martin L. Griss is a Principal Laboratory Scientist at Hewlett-Packard Laboratories, Palo Alto, California. At HP since 1982, he has researched software engineering processes and systems, systematic software reuse, object-oriented development and component-based software engineering. He is currently working on model-driven, agent-based application management systems. He created the first HP corporate reuse program, and led HP efforts to standardize UML for the OMG. He was previously director of the Software Technology Laboratory at HP Laboratories, and an Associate Professor of Computer Science at the University of Utah. He is co-author of the book "Software Reuse: Architecture, Process and Organization for Business Success". He has written numerous articles on software engineering and reuse, and lectures widely on systematic reuse and software process improvement. He is a member of the ACM SIGSOFT Executive Committee, and of the joint IEEE/ACM software engineering education project. He is a member of the SSR steering committee, the ICSE99, UML99, OOPSLA99 and TOOLS99 program committees, and the ICSE2002 organizing committee.

Robert R. Kessler is currently a professor and chairman of the Department of Computer Science at the University of Utah. In the early 90's, he founded the Center for Software Science, a state of Utah Center of Excellence. He has also founded several startup companies and is currently involved with an Internet startup company, emWare, as a member of the board. He has served as member-at-large of ACM SIGPLAN and is currently Vice Chairman for Conferences for SIGPLAN. He recently completed a seven year assignment as co-editor-in-chief of the International Journal of Lisp and Symbolic Computation. His current research interests are in agents, software engineering, distributed systems, and visual programming.

Leon J. Osterweil is currently a professor in the Department of Computer Science at the University of Massachusetts, Amherst. Previously he had been a Professor in, and Chair of, Computer Science Departments at both the University of California, Irvine, and University of Colorado, Boulder. He was the founding Director of the Irvine Research Unit in Software (IRUS) and the Southern California SPIN. He has been Program Committee Chair of ICSE 16, TAV 2, ISPW4, and SDE2, and General Chair of FSE 6. He has also presented keynote talks at such meetings as CASE 92 in Montreal, Quality Week 96 in San Francisco, the Inaugural Symposium of JAIST (the Japan Advanced Institute for Software Technology) in Kanazawa, Japan, and ICSE 9 (the Ninth International Conference on Software Engineering) where he introduced the concept of Process Programming. His ICSE 9 paper has been awarded a prize as the most influential paper of ICSE 9, awarded as a 10-year retrospective. He has consulted for such organizations as IBM, Bell Laboratories, SAIC, MCC, and TRW, and SEI's Process Program Advisory Board. Prof. Osterweil is a Fellow of the Association for Computing Machinery.