AgentSheets – Programming above C-Level

Martin Rausch

German Abstract

In vielen Bereichen, in denen Computer eingesetzt werden, gibt es das Problem, daß Software Entwickler von den Arbeitsvorgängen, die sie erleichtern sollen, zu wenig wissen. Der eigentlich prädestinierten Systementwickler wäre der End-Benutzer. Allein er kennt alle existierenden Anforderungen und Nebenbedingungen seiner täglichen Arbeit.
Mit AgentSheets™ wird hier eine Designumgebung vorgestellt, die es dem End-Benutzer von Applikationen erlaubt, direkten Einfluß auf den Ablauf der Applikationen, Simulationen und Visualisierungen, die mit AgentSheets erstellt wurden, zu nehmen. Dies wird ihm durch eine graphische End-Benutzer-Programmierschnittstelle ermöglicht. Die regel-basierte Sprache Visual AgenTalk™ erlaubt es dem Nicht-Informatiker, das Verhalten von Agenten einzusehen, zu verstehen und gegebenenfalls zu verändern. Dies geschieht mit graphischen Metaphern, die dem Anwendungsgebiet entnommen sind, in dem die Applikation jeweils eingesetzt wird. AgentSheets ist also keine statische Umgebung, sondern erlaubt eine Anpassung des Sprachumfanges an eine spezielle Domäne.
Mit dem Behavior Exchange steht dem Benutzer ein web-basiertes Forum zur Verfügung, wo er sich Lösungen anderer AgentSheets Benutzer ansehen, sie lokal speichern oder Eigene beitragen kann. Der integrierte Java Byte-Code Compiler Ristretto transportiert die mit AgentSheets entwickelten Applikationen in die plattformunabhängige Welt von Java und Web-Browsern.

A Symmetry of Ignorance

AgentSheets™ is a visual design environment for dynamic agent-based simulations, applications, and visualizations. This system was developed in an effort to remedy the »symmetry of ignorance« – the idea that the people who use computer systems and the people who build them do not share the same ideas and concepts about the purpose and the use of the system.

On the one hand, system designers build systems according to their own understanding of the problem and its domain. On the other hand, the customers use programs according to what they understand, which often means guessing as to how the software works. Unfortunately, these guesses are drawn from the customerís own experience which is usually very different from the system designer´s point of view.

AgentSheets approaches this problem by giving end-users the means to shape or even create domain-specific applications themselves, thereby narrowing the semantic gap between system and problem domain. System designers no longer create static applications, but provide end-users with a set of language components that provide basic functionality. With end-user programming interfaces, assembly or reconfiguration of systems should then shift away from system experts and towards domain experts.

Take an information filter as a simple example. Which developer can know, in advance, the user´s preferences, what interests her, or how she wants information to be represented? Or who, other than the scientists themselves, would know what is important and what is not in scientific simulations and visualizations? Should not these scientists be able to change their models and simulations interactively and thereby gain new insights? And all of this without a computer science education!

The problem of user- or problem-centered software becomes apparent especially in the emerging area of software agents. Coming from the field of artificial intelligence, machine learning techniques are applied to automatically extract user preferences and identify iterating tasks that are subject to automation. The drawback of this approach is that learning techniques are fickle, error prone, and slow. They may be useful to identify use patterns that are tacit or subcons-

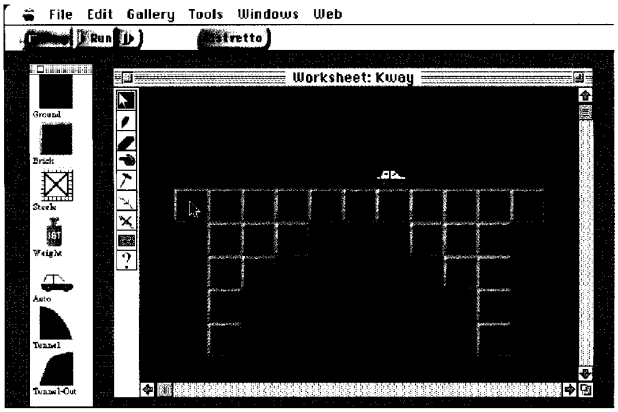


Figure 1: Simulation of forces (red coloration) effecting bricks in bridge constructions

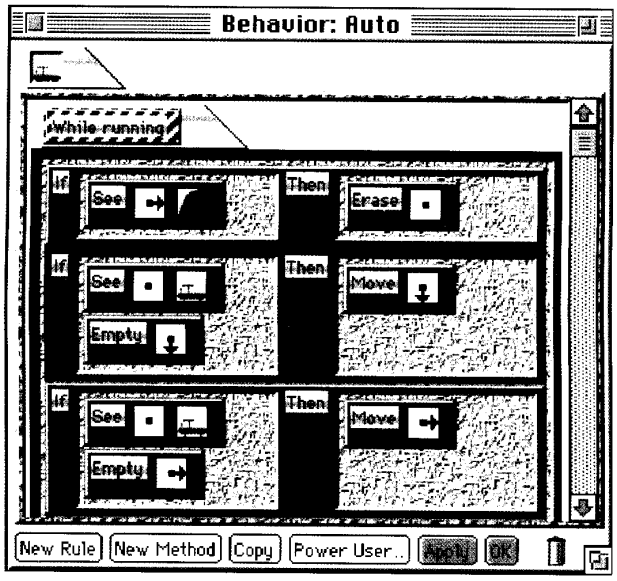


Figure 2: Agent behavior is defined in a collection of visual rules.

cious, but since in most cases users are quite aware of their preferences, one can ask why the user cannot tell its agent what she wants rather than the machine trying to figure that out by itself.

AgentSheets

AgentSheets is a visual design and programming environment that allows users to create agent-based simulations, visualizations, and applications. Figure 1 shows an AgentSheets simulation of gravitational forces to which the bricks of a bridge are exposed. The red coloration denotes the force, and the color intensity its strength, as a car passes over the bridge. All elements of the simulation – in this case cars and bricks – are agents that are arranged in a grid. The grid metaphor was transferred from a successful end-user programming model – the well-known spreadsheet formula language. In spreadsheets, some cells contain formulas that constantly recompute their value with respect to certain user-specified neighbor cells. Most of the time, the cell addressing is relative to the formula cell position. That way, formulas can be copied to other locations and expose behavior that corresponds to the original cell.

Analogously, the software agents expose behavior according to what was programmed into them. Their behavior depends on their environment, i.e. other agents, in their cell neighborhood. The bridge simulation agents shown in Figure 1 expose the following behavior:

- The cars simply move one cell to the right at each step of the simulation. When they do not perceive a solid surface underneath, they fall down. When they encounter the tunnel entry, they disappear.
- Bricks take into consideration things on top of them that could push them down, things on their side (e.g., to which they could be glued), and things below them that could support them. These forces are communicated locally in a diffusion process. They determine the sum of forces on them and colorize themselves accordingly.

The grid into which agents are put is called a Worksheet. In Figure 1, the worksheet contains a specific bridge configuration – an arch. The bridge could be rearranged by the end-user, i.e. bricks could be added and removed. The leftmost window in Figure 1 is called the Gallery. From the gallery, agents are selected and then instantiated in a worksheet.

Visual End-User Programming

AgentSheets defines agent behavior in a way that is significantly different from the spreadsheet formulas mentioned earlier. AgentSheets agents are programmed using a visual programming language called Visual AgenTalk™. The language was designed to maximize domain- and problem-orientation while minimizing the programming language overhead. The result is a rule-based language in which the rule components, conditions, and actions are graphical user interface items, e.g., pop-up menus, check-boxes, text fields. This way, the syntax that causes problems in traditional programming becomes almost trivial. The conditions that are placed – actually dragged and dropped – in the condition box of a rule are interpreted with an implicit AND-conjunction. The condition box as a whole describes criteria that signal the occurrence of a certain situation. The agent reacts with what is specified in the action box of the rule.

In Figure 2, some of the rules that determine the behavior of cars in the bridge simulation are displayed in the Behavior Editor:

- The first rule states that if a car »sees« a tunnel entry to its right, it should erase itself.
- The second rule states that if a car looks at itself and perceives an empty cell beneath it, it should move/fall down.
- The third rule is left to the reader as an exercise.

Many AgentSheets applications share language components, i.e. the Erase-action command (c.f. Figure 2). Other components are specific to one application domain, i.e. retrieving information of web pages. Figure 3 shows a condition that searches a Web page at the specified URL for a term (here »Last«) and retrieves a numerical value if one is found in the immediate vicinity of that term in the Web page. The value is then placed in a named variable of the agent. In this example, a stock quote is read and stored under »quote.«

AgentSheets and Visual AgenTalk represent two innovative

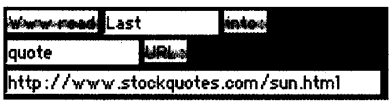


Figure 3: Domain-specific language component to search web pages for numerical values

approaches to close the gap between system design and problem-solving. Empowering end-users to interactively change applications and simulations enhances their system understanding and enables them to give better feedback to system designers. This way of designing applications may still be in its infancy, but it will gain importance quickly in the light of apparent information overload and high-functionality applications.

Examples

Over 1000 applications have been created with AgentSheets in areas including environmental design, education (K-12 to college), fine arts, computer science (i.e. genetic algorithms), architecture, and engineering.

Obvious application areas for AgentSheets are simulations and visualizations of dynamic processes and their factors. The added value is the capacity to interactively change the behavior of the simulation by modifying agents' rules.

The Web-read command that was shown earlier may be employed to build personalized information filters and browsers. The command can retrieve data such as stock quotes, temperatures, or significant company statistics that can be further processed via Visual AgenTalk and AgentSheets. In combination with other functionality, i.e. control of mail clients, task automation in general is another area of employment.

Industrial applications of visual rule-based languages come from areas, such as control of complex machines or devices. Often the circumstances that affect devices change over time, and the control software must be updated to account for the changes. With Visual AgenTalk, such tasks could be performed by the machine operator rather than a software engineer. Adding an additional simulation component, similar to the bridge simulation seen earlier, lets the user test rules before actually applying them to the real device.

The rest of the world

Part of the design philosophy behind AgentSheets is the support of distributed cooperation between AgentSheets users. The Behavior Exchange offers a forum to AgentSheets users where they can share their agents or find agents designed by other users. By offering support for downloading agents from Web-sites, AgentSheets encourages social programming. A more commercial scenario has a provider offering agent templates that provide a certain functionality. These templates could be downloaded from the Web site and then refined and modified by customers to meet their own specific requirements.

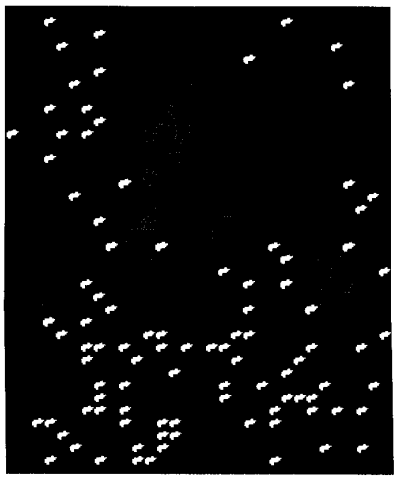
Related to products such as the device control agent mentioned earlier, the Web site could be the source of new releases, updates, or variants of the original product. Lacking any setup overhead, the downloaded agents could immediately be inserted into existing projects.

The Behavior Exchange can be found at http://agentsheets.cs.colorado.edu/Behavior-Exchange/Home.

Java for end-users

Users can make their AgentSheets projects, such as the bridge simulation, into stand-alone applications by using Ristretto™, the Java byte-code compiler integrated into AgentSheets. Ristretto creates a Java applet that runs in any Java-enabled Web browser on all popular platforms. Ristretto and AgentSheets combined are powerful applet-builder tools that enable end-users to access Java technology. Three steps are necessary to create an applet:

- Visually design an application with AgentSheets and Visual AgenTalk
- Run the Ristretto Java byte-code compiler to transform the application into a Java applet.



Simulation of bacteria growth process in glucose medium

- Start the application as Java applet on any Java-capable platform.

Background information

If you are interested in more information, please refer to the following web sources:
AgentSheets Demo & Support Center Europe:
http://www.igd.fhg.de/www/igd-a9/a9-home.html
AgentSheets Inc.:
http://www.agentsheets.com

Points of contact

Dr.-Ing Stefan Noll,
M.Sc. Martin Rausch
Fraunhofer IGD, Darmstadt, Germany
Email: noll@igd.fhg.de
rausch@igd.fhg.de