Goal-Directed Design for Proactive and Intelligent Device Collaboration

Michael VanHilst[1] and Martin Griss[2]

[1] Florida Atlantic University
mikev@cse.fau.edu
[2] University of California, Santa Cruz
griss@soe.ucsc.edu

Abstract. Tomorrow's ubiquitous computing environment will be filled with intelligent devices: on your person, in your office, car, and home, on the street, in stores, museums and restaurants, ..., just about anywhere you can think of. These devices could potentially talk to one another in ad-hoc networks. But what will they talk about? This paper describes a goal directed design paradigm within agent oriented programming for intelligent devices that respond to and make use of other like-minded devices, without presupposing what is out there.

1 Introduction

Future devices and systems will exhibit emergent intelligence - the seeming ability to act intelligently by integrating a variety of information and resources pertinent to the user's current situation. Consider the following scenario involving a cell phone, a bookstore, and a printer.

A businesswoman is walking down the street when her cell-phone rings. Her colleague in the office is calling to discuss a last minute change to the proposal she is about to present to an important client. The businesswoman tells the colleague to make the changes and call her when it is ready.

While she is waiting for her colleague to call again, the businesswoman enters a used bookstore. As she passes through the door, her cell phone vibrates briefly. The message on the cell phone's screen shows the name of a novel the businesswoman has been looking for and the location: "aisle 2 third shelf". The businesswoman decides to have a look.

The businesswoman is still in the bookstore when her colleague calls again. This time the cell phone does not ring, but only vibrates. Her colleague has the revised proposal ready. The businesswoman decides to buy the book and then find a place to print the new proposal.

When the businesswoman approaches the front cashier, the cell phone vibrates again and its light flashes. The message on the display reads, "45 cents, print now?" The businesswoman asks the cashier about the printer. She points to a printer behind the counter. The businesswoman presses OK and the printer behind the counter prints the new proposal.

This scenario illustrates the kind of proactive behavior possible with "intelligent" software agents in the mobile and connected world of ubiquitous computing. But how difficult is it to create the applications needed to make this scenario come to life?

1. How did the cell phone communicate with the bookstore?
2. Why didn't the cell phone ring when it was inside the bookstore?
3. How did the cell phone find the printer, and why was the printer available to customers?

In this paper we describe how to design systems based on goals and goal-seeking behavior. Goals are decomposed into smaller goals and assigned to specific components in the system. Each system component, or sub-component agent performs behaviors in pursuit of its goals. Large systems with complex behavior are formed by collections of narrowly focused agents, each acting in its own self-interest.

In the next section we describe the design principles that guide goal oriented design while walking through the design of the components for the above scenario. We also discuss an approach for the issue of privacy. In the following section we describe the relevant mechanisms provided by agent standards and platforms. In the third section we provide a larger scenario illustrating a richer set of behaviors and describe how they map to goal-oriented components and again walk through the components of the design. We conclude with some discussion of related work.

2 Goal Directed Design

In appliance applications, we don't always know what functions and resources will be available. But it is likely that the same goals will apply regardless of the available devices and resources. Agents allow us to design in terms of goals, decomposing higher goals into lower goals, and allocating goals to specific behavioral units. An agent watches for opportunities to achieve its goals or for changes and failings that impact a goal state it seeks to maintain.

To understand how a goal-oriented design can be implemented, let's look at part of the cell phone behavior. The high level goals for an appliance will typically fit the following categories: perform core functions, enhance user experience, promote revenue generation, comply with regulations, and support aggregation. The cell phone has a number of core functions including staying connected and notifying the user of incoming calls and other status related events. It enhances user experience by providing the highest quality connectivity available and the best user interface available at any given time. For the cell phone's parent company, opportunities for revenue generation increase when the user has compelling reasons to use the cell phone, especially for billable resources and services. Knowing the cell phone's location, in part, satisfies a regulatory requirement. Being able to function in collaboration with other devices supports aggregation and greatly increases opportunities for goal satisfaction. Annoying the user, or compromising the user's privacy would work against the users experience and discourage cell phone use.

In the above scenario, the store has implemented two services with the goals of enhancing user experience and promoting revenue generation. They do this by notifying the user's personal device of opportunities to satisfy it's own goals, and, not

coincidentally, spend money in the store. The printer manufacturer has its own goal in designing the printer to interact with customers: more pages to print means revenue generating ink or toner will be consumed at a higher rate.

Let's look more closely at the case of the printer. The printer has a primary function, it prints documents, and a corresponding goal to maintain a high quality of printing. Some printers already monitor various parameters and offer to order supplies from the manufacturer. Future printers may also have regulatory goals related to the printing (e.g., for a fee) of copyrighted material. In our scenario, the printer has a goal to enhance revenue generation for both its owner and the manufacturer by seeking documents to print. To achieve this goal, the printer leverages an ability to communicate with other devices in close proximity through some form of short-range wireless network. The goal thus has a sub-goal to communicate with other devices as they enter the range of its neighborhood and identify those with an active goal to print a document. A second sub-goal would then be to reach an agreement, through negotiation, to actually print the document. In some cases, the negotiation may involve a third sub-goal, to find a service that can convert the document to a format the printer can handle.

When the businesswoman entered the used bookstore, her cell-phone interacted with the store's agent. Like the printer, the store agent also has a goal of promoting revenue generation. In this case, the store seeks customers with an interest in books - especially books that it has in stock for immediate purchase. As with the printer, this goal has a sub-goal to communicate with other devices as they enter the range of its neighborhood and identify those with an active goal - in this case, to buy a book. The negotiation leads to the customer revealing that it wants the book, and the store revealing that is has the book. But customers may not want to reveal their personal preferences, or the contents of their shopping list, and the store may not want to reveal its entire inventory, since a competitor might be listening. The negotiation takes an iterative approach involving less revealing information, such as the first letters of an author's name or parts of ISBN numbers, in order to reduce the amount of personal information that is ultimately revealed by either party.

The store has another goal, which is to enhance user experience. This goal has several parts. One part that we see in the scenario, is to assist the user in locating items in the store. We will have more to say about location services and navigation in a later section. But another goal in enhancing the experience of the store's customers, is to maintain a quiet environment inside the store. In the scenario, that goal translates to a goal that customer devices should not ring or beep, and the store deploys an agent specifically for that goal. A museum might have a similar goal that cameras not use flash. The store's agent seeks other agents that can ring or beep, or have a goal to do so. Two mechanisms may be applied. In one approach, the device sends notifications of intent before it actually performs an act. It receives a request from the store agent to not perform that act. In the other approach, a store agent gives the device the goal to not use ringing, which the device adds to the set of rules it uses to decide how best to get the user's attention.

Now let's look at the cell phone. The cell phone has a goal of enhancing user experience and revenue generation by enabling the user to perform resource-consuming activities, such as sending and receiving photos and documents. The ability to receive more documents is dependent on the cell phone having sufficient

memory to store them. Thus one of the goals of the cell phone is to find opportunities to unload items from memory. When the businesswoman first received the document from her colleague, the cell phone offered her the ability to forward the document to another device, such as her office or home PC. Since the businesswoman did not choose this option, and no printer was currently in the businesswoman's personal area network, the phone acquired the goal to print the document when an opportunity became available. Like the printer, the phone now had a sub-goal to communicate with other devices as they enter the range of its neighborhood and identify those that are able to print, or otherwise consume, the document. When the printer appeared, the phone entered into negotiations about the feasibility and terms for printing the document and then presented the choice to its user.

3 Agents, Goals, Matching, and Planning

The agent paradigm is a level of abstraction above objects and components. Agent systems are designed as communities of goal-directed autonomous agents interacting to form teams and perform complicated tasks. By autonomous, we mean that agents are driven by their goals to initiate and react to events without being prompted by an external source. Agents are said to "sense their environment"[1] and "behave intelligently"[2]. The agent paradigm is well suited to systems that must dynamically adapt to, and thrive in, new and unknown contexts.

In our design, each proactive device or component in the system is characterized by the goals it is trying to achieve plus its abilities to perform specific services. Thus a significant part of the design effort and subsequent implementation must deal explicitly with describing, representing and manipulating goals.

3.1 Agents and Agent Platforms

Goal-directed agents leverage the infrastructure and middleware services provided by common agent platforms. These services include message services, discovery of other agents, message parsing based on an ontology, negotiation, and matching and planning among goals and abilities. All of these services are either provided by common agent platforms, or can easily be added as plug-in services or separate service agents [3].

Consider, for example, the FIPA specification (http://www.fipa.org/) for agent systems [4][5]. Agents are organized into platforms, which are aggregations of agents sharing common services. The platform includes a management service, a directory service and a message gateway service. The management service provides agent lifecycle services. The directory service provides a lookup facility to locate agents within this and other platforms. The transport service controls the exchange of messages within the platform and across the platform. A platform may span multiple devices. The platform management, directory, and gateway services run in a main container, typically on a resource-heavy device such as a PC. Resource-constrained devices, such as a small appliance, will typically run just an agent container that interacts with the main container for services and to locate other agents. The grouping

of agents into platforms can be logical, physical, contextual or simply arbitrary. In the store scenario, for example, the businessman might have a single platform for all of the devices on his person, with each device running one container. The main container might run on the cell phone, on a separate personal mobile gateway [6], or on a remote machine, perhaps provided by the cell phone company, with which the cell phone is able to maintain connection. The store could run its own platform, with a separate container on the printer, or the printer could be running its own platform.

3.2 Communication and Ontology

Agents communicate through asynchronous message passing, using a loosely-formatted, text-based communication language. Messages include a header (to, from, etc.), an ontology, a message type, and a variable-length body. FIPA specifies a dozen or so high-level message types, such as Notify, Query, Inform, etc. In addition to requests to perform, communication languages allow agents to exchange information such as commitments, goals, beliefs, and intentions. In our example, the cell phone can communicate its intention to use a ringer, while a store agent can communicate a desire that ringers not be used.

3.3 Representing Goals and Strategies

In our scenario, the goal of printing from the cell phone could be addressed with a goal tree. The top-level goal of maximizing available memory could be or-decomposed into less abstract alternative goals of printing or uploading documents from memory. The printing goal, in turn could be and-decomposed into child goals of finding a printer, converting the document to an acceptable format, negotiating terms, and getting the user's approval. In this form, new goals can be generated for new contexts, such as the arrival of a new document. Decision rules [7] could also be employed to respond to new situations, such as the decision of whether or not to ring, or what to do when contacted by a store.

3.4 Matching and Planning

Opportunistic behavior occurs when combinations of components find opportunities to mutually satisfy their goals. This behavior requires each agent to support some form of problem-solving/planning task to match its goals against the abilities and goals of others [7]. In particular, one device may need to negotiate some aspects with another, and it might need to break a goal into useful sub-goals and see if different devices can satisfy each of the sub-goals, are willing to satisfy them, and then ensure that they commit the necessary resources so that it can schedule the various sub-actions and achieve its higher level goal.

At the simplest level, my goal needs to exactly match your capabilities. More generally, some form of pattern matching is needed to see if my goal matches your abilities. Depending on how we describe goals and abilities (e.g., if we use facts and

rules in a forward chaining style like that of CLIPS [9] or JessTM [10], there will be some parameters that are bound during a "unification" or pattern matching process[7]. If more than one match is possible, we would use some form of utility function to evaluate which one is most advantageous. In the scenarios we describe above, we could ask the users if they are willing to pay a certain amount for the use of the other device or service, or could be asked to select between several choices. In other cases, some built in evaluation and filtering can either make the decision to proceed or significantly prune the choices before asking the users.

If my goal is not "directly" matched by any of the abilities offered by the devices in my neighborhood, I could break my goal into sub-goals in one or more ways, and try to satisfy all of the sub-goals in of these sets; again, if more than one solution is possible, we would evaluate the alternatives using a utility function. At the simplest level, a goal would be decomposed into a set of sub-goals using one or more stored planning templates - for example to print a word document, if the printer can not handle Word directly, we would first seek a service that will convert the document to PDF and then ask the printer to print the PDF document. A more complex planner would use several problem-solving and planning strategies.

Some agent toolkits provide many or all of these capabilities in a useful form. For example, the Zeus agent toolkit [11], has a built in problem-solver, planner and rule-based goal/fact system, as well as a protocol-based negotiation system. These capabilities could allow agents in each device to come to some agreement, reserve and schedule resources, and then perform the actions.

The basic idea of seeking and discovering the ability of neighboring devices or services is similar to that of SUN's peer-to-peer Jini [12] or UUDI/XML-based W3C web services. In Jini the advertising of abilities uses an object's API of public method interfaces, while in web-services, a UDDI registry contains a higher-level WSDL description in XML[13]. But using rule-based goals and abilities enables the matching and negotiation to occur at a much richer and more semantically meaningful way.

5 Discussion and Related Work

Future cell phones will do much more than make calls and keep phone books. Perhaps the greatest potential that they hold will require sensing the environment, interacting with nearby devices, and exhibiting opportunistic behavior specific to current user goals and context. In this paper we present a design and implementation paradigm based on agent-like behavior and goal directed design.

Object oriented design assumes a fixed architecture with known interfaces, but allows the types of objects within the relationships to vary. In the scenario described in this paper, we do not know what devices will be encountered, what types of functions they will be able to perform, or what goals they will seek to attain. A scenario that would otherwise be captured in a use-case, in our example might need to be generated on the fly at runtime. Some agent development methodologies treat agent-oriented development as an extension of object oriented development [14]. Goals are discussed in requirements and analysis, but the implementation is likely to have much the same rigidity as normal object oriented design.

Tropos is an agent-oriented software development methodology that preserves goals as first class concepts from early requirements all the way through implementation and deployment [15]. The resulting flexibility and adaptability (hence the name Tropos) makes it well suited for the kinds of applications we describe here. One of the MIT Oxygen projects proposes to formalize goals as a language construct – an interesting idea, but little has yet been published [16]. Some of our current ideas on agent-oriented design and agent systems on small devices grew out of earlier experience building personal assistants on a variety agent platforms [17]. Newell first proposed a goal-driven approach in his still inspiring 1982 paper, The Knowledge Level [18].

References

1. Odell, J.:Designing Agents: Using Life as a Metaphor, Distributed Computing, July, 1998, pp.51-56.
2. Durfee, E.H.: Scaling Up Agent Coordination Strategies, IEEE Computer, July, 2001, pp.39-46.
3. Griss, M.: My Agent Will Call Your Agent, But Will It Respond, Software Development Magazine, Feb, 2000.
4. O'Brien, P.D., Nicol, R.C.: FIPA – Towards a Standard for Software Agents, BT Technical Journal, 16(3) (1998) 51-53.
5. Bellifemine, F., Poggi, A., Rimassi, G.: JADE: A FIPA-Compliant agent framework, Proc. Practical Applications of Intelligent Agents and Multi-Agents, April (1999) 97-108; also http://sharon.cselt.it/projects/jade .
6. PMG World Magazine, http://www.pmgmag.com
7. Logan, B.: Classifying Agent Systems. In Proc. of the AAAI-98 Workshop on Software Tools for Developing Agents, Wisconsin, USA (1998)
8. Russell S., Norvig, P.: Artificial Intelligence: A Modern Approach, Prentice-Hall (1995)
9. CLIPS, a forward chaining rule system, written in C. http://www.ghg.net/clips/CLIPS.html.
10. A forward and backward chaining rule system, written in JAVA. http://herzberg.ca.sandia.gov/jess/
11. Nwana, H.S., Ndumu, D.T., Lee, L.C., Collis, J.C.: Zeus: A Toolkit for Building Distributed Multi-Agent Systems. Proceedings of the Third International Conference on Autonomous Agents (1999) 360-361
12. Jini Network technology - http://wwws.sun.com/software/jini/
13. For UDDI and WSDL see http://www.w3.org/TR/wsdl and http://www.uddi.org/
14. Wooldridge, M., Jennings, N.R., Kinny, D. (2000), " The GAIA Methodology for Agent-Oriented Analysis and Design", Autonomous Agents and Multi-Agent Systems, 3(3), (2000) 285-312
15. Perini, A., Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J: Towards an Agent Oriented Approach to Software Engineering, In Omicini, A., Viroli, M. (eds): WOA 2001 – Dagli Ogetti agli Agenti: Tendenze Evolutive dei Sistemi Software, Modena
16. Ward, A., Terman, C., Saif, U.: Goal Oriented System Semantics, March 2002 http://cag.www.lcs.mit.edu/~umar/publications/Goals_abstract.pdf
17. Griss, M., Letsinger, R., Cowan, D. Sayers, C., VanHilst, M., Kessler, R.: CoolAgent: Intelligent Digital Assistants for Mobile Professionals – Phase 1 Retrospecitve. HP Laboratories technical report HPL-2002-55(R), July 2002
18. Newell, A.: The Knowledge Level. AI Magazine, 2(2) (1981) 1-20