Simplifying the Development of Intelligent Agents

Michael Winikoff, Lin Padgham, and James Harland

RMIT University, Melbourne, AUSTRALIA
{winikoff,linpa,jah}@cs.rmit.edu.au,
http://www.cs.rmit.edu.au/~{winikoff,linpa,jah}

Abstract. Intelligent agents is a powerful Artificial Intelligence technology which shows considerable promise as a new paradigm for mainstream software development. However, despite their promise, intelligent agents are still scarce in the market place. A key reason for this is that developing intelligent agent software requires significant training and skill: a typical developer or undergraduate struggles to develop good agent systems using the BDI model (or similar models). This paper identifies the concept set which we have found to be important in developing intelligent agent systems and the relationships between these concepts. This concept set was developed with the intention of being clearer, simpler, and easier to use than current approaches. We also describe briefly a (very simplified) example from one of the projects we have worked on (RoboRescue), illustrating the way in which these concepts are important in designing and developing intelligent software agents.
Keywords: AI Architectures, distributed AI, multiagent systems, reactive control, software agents.

1 Introduction

Intelligent agents is a powerful Artificial Intelligence technology which shows considerable promise as a new paradigm for mainstream software development. Agents offer new ways of abstraction, decomposition, and organisation that fit well with our natural view of the world and agent oriented programming is often considered a natural successor to object oriented programming [12]. It has the potential to change the way we design, visualise, and build software in that agents can naturally model "actors" – real world entities that can show autonomy and proactiveness. Additionally, social agents naturally model (human) organisations ranging from business structure & processes to military command structures. A number of significant applications utilising agent technology [11] have already been developed, many of which are decidedly non-trivial. For example, some of the military simulation work involves thousands of dMars plans [23].

In this paper we consider *intelligent software agents* based on the *Belief Desire Intention* (BDI) model of Rao and Georgeff [8, 17]. Our work in general is focussed on multi-agent systems although this paper focuses on the concepts required for the internals of each individual intelligent agent - a necessary pre-requisite for multi-agent systems with teams or societies containing such agents.

An intelligent agent is one which is able to make rational decisions, i.e., blending proactiveness and reactiveness, showing rational commitment to decisions made, and exhibiting flexibility in the face of an uncertain and changing environment.

Despite their promise, intelligent agents are still scarce in the market place[1]. There is a real technical reason for this: developing intelligent agent software currently requires significant training and skill. Our experience (and the experience of others) is that a typical

[1] Although abuse of buzzwords is, alas, all too common.

developer or final-year undergraduate student struggles to develop good agent systems using the BDI model (or similar models).

Decker [5] discusses problems that undergraduate students have in approaching agent oriented development. These include a lack of suitable background in AI (planning and goal oriented programs), poor software engineering skills, and a lack of experience at dealing with concurrent and distributed programming/debugging, and with communication protocols.

We have found the key problem to be that students cannot clearly identify the necessary pieces to break the program into and thus tend to build monolithic plans which try to handle all contingencies internally, rather than create an appropriate collection of plans which can be applied in different contexts. They also have significant difficulty with interfacing the agent to its environment.

Other reasons why developing intelligent agent systems is difficult include:

- Immature tool support: Although high quality agent development tools are emerging, there is a lack of good debugging tools and a lack of tools which integrate an internal agent architecture with suitable middleware & infrastructure. Additionally, many tools are research prototypes and are lacking in efficiency, portability, documentation, and/or support.
- The need for processes and methodologies: Programmers are familiar with designing object oriented systems. However, the design of agent oriented systems differs in a number of ways. For example, identifying roles, goals, and interaction patterns.
- Design guidelines and examples: Designing a collection of plans to achieve a goal is different to designing a single procedure to perform a function. This difference is fundamental – developing intelligent agents is a different programming paradigm and needs to be learnt and taught as such.
- Complex concepts such as intentions are difficult to explain; this isn't helped by a lack of agreement on concepts and inconsistent terminology.
- Lack of a suitable[2] text book: much of the work on intelligent agents is scattered across many research papers (sometimes collected into volumes). We are not aware of any single suitable textbook for design and programming of intelligent agents.

In the process of working on a number of agent programs, teaching students and assisting them to build agent programs, and developing and running workshops for academia and industry[3], we have developed an initial process of agent design and development for BDI systems.[4] This process is explained more fully in the technical report [15] and is still being refined and developed.

This paper identifies the concept set which we have found to be important in developing BDI systems and the relationships between these concepts. These of course rely heavily on the standard BDI concepts though we have found it necessary to clarify some of the differences between just what these concepts are in the initial philosophical work of Bratman [2], the logical theories of Rao and Georgeff [17] and others (e.g. [4, 21]) and the implementations such as PRS [9, 10], dMars [1, 6] and JACK [3]. We have also found it important to place some emphasis on the concepts of percepts and actions which appear in many generic models of agents (e.g. [20]) and which are very important in the interfacing of the agent deliberation to the external environment. We have found a need to separate

[2] Suitable: Aimed at undergraduates or professional developer and contains enough detail to answer the question "How would I actually go about building an intelligent agent?".

[3] Workshops have been developed and delivered in association with Agent Oriented Software

[4] Using primarily dMars from the Australian Artificial Intelligence Institute, and more recently JACK from Agent Oriented Software.

more clearly between events and goals than is done in dMars or JACK and to provide greater support within the execution engine for reasoning about goals than is usually done in BDI agent systems [22].

We describe briefly the key concepts found in the BDI literature, then describe the concepts we have found to be important and why. We follow this with a brief (and very simplified) example from one of the projects we have worked on (RoboRescue), illustrating the way in which these concepts are important in designing and developing a BDI agent. Further work includes developing a clear intuitive formal semantics for these concepts, refining our existing design methodology and developing support tools including support for design, development and debugging.

2 Background: The BDI Model

The BDI model [8, 17, 18] is a popular model for intelligent agents. It has its basis in philosophy [2] and offers a *logical theory* which defines the mental attitudes of Belief, Desire, and Intention using a modal logic; a *system architecture*; a *number of implementations of this architecture* (e.g. PRS, JAM, dMars, JACK); and *applications* demonstrating the viability of the model. The central concepts in the BDI model are [8, page 144]:

Beliefs: Information about the environment; *informative*.
Desires: Objectives to be accomplished, possibly with each objective's associated priority/payoff; *motivational*.
Intentions: The currently chosen course of action; *deliberative*.
Plans: Means of achieving certain future world states. Intuitively, plans are an abstract specification of both the means for achieving certain desires and the options available to the agent. Each plan has (i) a body describing the primitive actions or sub-goals that have to be achieved for plan execution to be successful; (ii) an invocation condition which specifies the triggering event[5], and (iii) a context condition which specifies the situation in which the plan is applicable.

The BDI model has developed over about 15 years and there are certainly strong relationships between the theoretical work and implemented systems. The paper [18] describes an abstract architecture which is instantiated in systems such as dMars and JACK and shows how that is related to the BDI logic. However, the concepts we have found useful for development within these systems do not necessarily match the concepts most developed in the theoretical work. Neither are they necessarily exactly the concepts which have arisen within particular implemented systems such as JACK. An additional complication is confusion and small differences between similar concepts, such as Desires and Goals, which receive differing emphasis in different work at different times.

Desires are understood to be things the agent wants to achieve. They play an important role in the philosophical foundation, but the logical theory deals primarily with Goals. Goals in the logic are assumed to be a consistent set of Desires. Philosophically there is no requirement for desires to be internally consistent. At the implementation level the motivational concept is reduced to events – goals are implicit and the creation of a new goal is treated as an event which can trigger plans. In the theoretical model plans are simply beliefs or intentions. However in the implementations plans are a central concept. Events are ignored in the theoretical framework but play a key role in implementations, although they are not well distinguished from goals. Some key differences between the philosophy, theory, and implementation viewpoints of BDI are shown in the table below.

[5] Some events are considered as goal-events.

Philosophy:	Belief	Desire	Intention
Theory:	Belief	Goal	Intention
Implementation:	Relational DB (or arbitrary object)	Event	Running Plan

3 Concepts for Intelligent Agents

As indicated, there are confusions and subtle distinctions in the concepts used when discussing intelligent agents at various levels. Terminology used varies between systems (e.g. what one system calls a "goal" another will call an "objective"); and there are subtle distinctions between terms/concepts such as desires and goals. Also the same term may be overloaded, for example the intentions of Rao & Georgeff's BDI logic are mental attitudes whereas in an implemented BDI system such as dMars or JACK an intention is effectively a running plan.

We describe the set of concepts which we have come to use in developing intelligent agent applications. We believe that these are necessary and sufficient for building the sort of applications appropriately approached using BDI agents, and we hope that they are simple, and clearly explained. The work to develop a formal semantic framework for these concepts, thus developing closer links between a theoretical framework and an implemented development platform, is work in progress.

To illustrate these concepts concretely we provide examples from a very simplified version of RoboRescue, an application we have worked on recently.

3.1 Agent Models and Concepts

We build up our description of an intelligent agent by beginning with a basic, and universally agreed upon (See for example [20]), property of agents: they are situated (see figure 1). Thus, we have **actions** and **percepts**. Internally, the agent is making a **decision**: from the set of possible actions As it is selecting an action (or actions) to perform ($a \in As$). Loosely speaking, where the description of the agent's internal workings contains a statement of the form "[select] $X \in Y$" then we have a decision being made. Thus the type of decisions being made depend on the *internal agent architecture*.

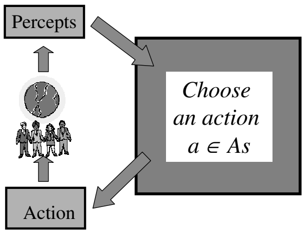

Fig. 1. Agents are Situated

An **action** is something which an agent does, such as *move_north* or *squirt*. Agents are situated, and an action is basically an agent's ability to effect its environment. In their simplest form actions are atomic and instantaneous and either fail or succeed. In the more general case actions can be durational (encompassing behaviours over time) and can produce partial effects; for example a failed *move_to* action may well have changed the agent's location. In addition to actions which directly affect the agent's environment, we also want to consider "internal actions". These correspond to an ability which the agent has which isn't structured in terms of plans and goals. Typically, the ability is a piece of code which either already exists or would not benefit from being written using agent concepts, for example image processing in a vision sub-system.

A **percept** is an input from the environment, such as the location of a fire and an indication of its intensity. The agent may also obtain information about the environment through sensing actions.

A **decision**: The essence of intelligent agents is rational decision making. There are a number of generic, non-application-specific questions which intelligent agents must answer, such as: *Which* action shall I perform now? *Which* goal do I work on now? *How* shall I attempt to realise this goal? *Where* shall I go now (for mobile agents)? And *who* shall I interact with (for social agents)?

Mechanisms to answer these kinds of questions are core intelligent agent processes. They result in decisions which must fulfil rationality conditions, in that we expect that decisions be persistent and only be revisited when there is a good reason for doing so. It is also important that the answer to the questions not be trivial: if an agent only has a single goal at a time and a single means of realising this goal then we have reduced the agent to the special case of a conventional program and there is no scope for decision making or for flexible, intelligent behaviour.

Note that although the concept of a decision is fundamental to intelligent agents, it is not always necessary to represent the decisions explicitly. For example, the decision regarding choice of goal could be represented using a "current goal" variable which is updated when a decision is made.

We now consider the internal workings of the agent (see figure 2). We want our intelligent agents to be both *proactive* and *reactive*[6]. A proactive agent is one which pursues an agenda over time. The agent's proactiveness implies the use of **goals** and modifies the agent's internal execution cycle: rather than select an action one at a time, we select a goal which is persistent and constrains our selection of actions. A *reactive* agent is one which will change its behaviour in response to changes in the environment. An important aspect in decision making is balancing proactive and reactive aspects. On the one hand we want the agent to stick with its goals by default, on the other hand we want it to take changes in the environment into account. The key to reconciling these aspects, thus making agents suitably reactive, is to identify *significant* changes in the environment. These are **events**. We distinguish between percepts and events: an event is an interpreted percept which has significance to the agent. For example, seeing a fire is a percept. This percept could give rise to a *new fire* event or a *fire under control* event depending on history and possibly other factors.

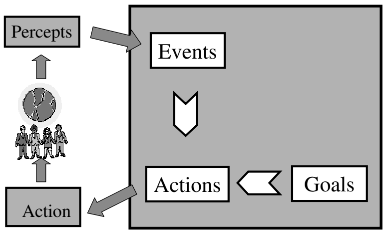

Fig. 2. Proactive Agents have Goals, Reactive Agents have Events

A **goal** (variously called "task", "objective", "aim", or "desire") is something the agent is working on or towards, for example *extinguish_fire*, or *rescue_civilian*. Often goals are defined as states of the world which the agent wants to bring about; however, this definition rules out maintenance goals (e.g. "maintain cruising altitude") and avoidance goals, or safety constraints (e.g. "never move the table while the robot is drilling"). Goals give the agent its autonomy and proactiveness. An important aspect of proactiveness is the persistence of goals: if a plan for achieving a goal fails then the agent will consider alternative plans for achieving the goal in question. We have found that goals require greater emphasis than is typically found in existing systems. It is important for the developer to identify the top level goals of the agent as well as subsidiary goals which are used in achieving main goals. Our modified execution engine does significantly more reasoning about goals

[6] We also want them to be *social*. However, as noted earlier, this paper focuses on the concepts required for the internals of each individual agent - a necessary pre-requisite for multi-agent systems.

than is usual in BDI implementations [22], including reasoning about interference between goals and how to select goals when it is not consistent to pursue them simultaneously. We differentiate between top level goals and subsidiary goals in that subsidiary goals are not important in their own right and may therefore be treated differently in the reasoning process than top-level goals.

An **event** is a significant occurrence. Events are often extracted from percepts, although they may be generated internally by the agent, for example on the basis of a clock. An event can trigger new goals, cause changes in information about the environment, and/or cause actions to be performed immediately. Actions generated directly by events correspond to "reflexive" actions, executed without deliberation. Events are important in creating reactive agents in that they identify important changes which the agent needs to react to.

Agents in realistic applications usually have limited computational resources and limited ability to sense their environment. Thus the auxiliary concepts of **plan** and **belief** are needed. Beliefs are effectively a cache for perceived information about the environment, and plans are effectively a cache for ways of pursuing goals (see figure 3). Although both of these concepts are "merely" aids in efficiency, they are not optional. Beliefs are essential since an agent has limited sensory ability and also it needs to build up its knowledge of the world over time. Plans are essential for two reasons. The first is pure computational efficiency: although planning technology and computational speed are improving, planning from action descriptions is still incompatible with real time decision making. The second reason is that by providing a library of plans we avoid the need to specify each action's preconditions and effects: all we need to provide for an action is the means to perform it. This is significant in that capturing and encoding sufficient knowledge to be able to represent the effects of continuous actions operating over time and space in a continuous and uncertain world in a way that would support first principles planning, is unrealistic for large applications.

A **plan** is a way of realising a goal, for example a plan for achieving the goal *extinguish_fire* might specify the three steps: plan a route to the fire, follow the route to the fire, and squirt the fire until it has been put out. Although the concept of a plan is common there is no agreement on the details. From our point of view it is not necessary to adopt a specific notion of a plan, rather we can specify abstractly what a plan should provide: a plan for achieving a goal provides a function which returns the next action to be performed. This function takes into account the current state of the world (beliefs), takes into account what actions have already been performed, and might involve sub-goals and further plans. For computational reasons it is desirable for this to at least include a "library of recipes" approach, rather than requiring construction of plans at runtime from action descriptions.

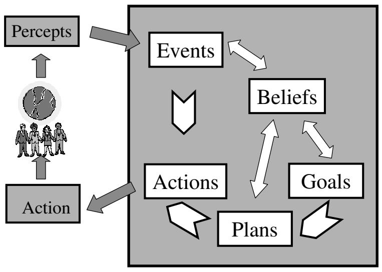

Fig. 3. Adding Plans and Beliefs

A **belief** is some aspect of the agent's knowledge or information about the environment, self or other agents. For example an agent might believe there is a fire at X because she saw it recently, even if she cannot see it now.

3.2 Execution Cycle

These concepts (actions, percepts, decisions, goals, events, plans, and beliefs) are related to each other via the execution cycle of the agent. An agent's execution cycle follows a sense-think-act cycle, since the agent is situated. The think part of the cycle involves rational decision making, consisting of the following steps: (depicted in figure 4)

1. Percepts are interpreted (using beliefs) to give events
2. Beliefs are updated with new information from percepts
3. Events yield reflexive actions and/or new goals
4. Goals are updated, including current, new and completed goals.
5. If there is no selected plan for the current goal, or if the plan has failed, or if reconsideration of the plan is required (due to an event) then a plan is chosen.
6. The chosen plan is expanded to yield an action
7. Action(s) are scheduled and performed

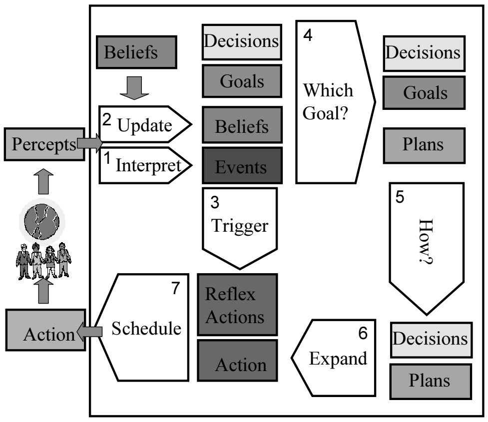

Fig. 4. Agent Execution Cycle

By comparison, the BDI abstract execution cycle [16, 18] consists of the following steps: (1) use events to trigger matching plans (options), (2) select a subset of the options, (3) update the intentions and execute them, (4) get external events, and (5) drop successful and impossible attitudes. The execution cycle presented here differs from the BDI execution cycle in a number of ways including the use of reflexive actions, the derivation of events by interpreting percepts, the process of going from goals to plans to actions, and increased reasoning about goals. However, the two primary contributions are the role of top level goals (which are distinguished from events and from sub-goals, and are persistent) in achieving proactiveness, and the role of events (as *significant* occurrences) in creating suitably reactive agents.

4 A Case Study: RoboRescue

One of the applications we have worked on recently is RoboRescue. We describe a greatly simplified version of a part of this application in order to illustrate concretely the concepts we have identified.

RoboRescue [19] is a long-term (50 years!) project which has the goal of creating robotic squads which could be deployed in the aftermath of a disaster such as an earthquake. Tasks to be carried out include rescuing & evacuating people and controlling fires. Challenges faced include the lack of information and an environment which contains obstacles (including collapsed buildings, obstructed roads, fires, etc.) and potentially limited communication.

The RoboRescue simulator has a number of components which simulate different aspects of a disaster scenario. The intention is that these aspects combine synergistically to provide a realistically complex and challenging environment. There are a number of different types of agents in the system including fire engines, ambulances, civilians, and police agents. At each simulator cycle agents receive visual information and possibly heard information.

Due to space limitations in this paper we focus on a single agent type (fire engine) and a simple set of its behaviours focussed on fire extinguishing.

4.1 A Design Process and its Application

To design the fire engine agent using the concepts described earlier we look at each concept and identify instances of it in the agent system. This process is, in general, iterative: when designing a plan we may realise that the agent needs to know a certain piece of information which implies the addition of a belief which might imply the addition (or modification) of goals and plans. A comprehensive methodology for the detailed design of agent systems is described in [15]. Here we concentrate on illustrating the way in which instances of the relevant concepts are identified and defined.

Decisions are not considered here since questions which an agent needs to answer as it runs aren't specific to a given application domain; rather, they are specific to a given agent architecture. Detailed design regarding how agents will work together, exactly what plans will be used and what sub-goals will be needed is a design task well beyond the scope of this paper and requiring the more extensive methodology of [15]. Rather, we focus on the initial aspects of the process and the identification of some of the relevant concepts in each class. More concept instances will inevitably be identified as the design progresses.

Percepts: Percepts represent an interface to the environment, so are often, as in this case, at least partially predefined. There are two types of percepts - visual and auditory. Auditory information may be broadcast information which in the case of a fire-engine agent may be a message from another fire-engine agent, the fire-engine center, or a civilian either crying for help or stating that they have heard a cry for help. It may also be a message directed specifically to that agent from another fire-engine agent. The content of messages from fire-engine agents and the fire-engine center is an aspect of perceptual information which is under the control of the designer and must be decided. Visual information contains a current view of the environment including such things as roads, buildings, fires and their intensity, etc.

Percepts must be processed to build up knowledge of the environment (beliefs) and to extract events. In the case of the visual information in RoboRescue it is first processed in order to add any information to the map of the world that is built incrementally. It is then processed to extract the position of each fire which is assessed to see whether there is a fire-related event which should be generated and to further update the knowledge of the environment.

Actions: The actions which an agent can perform are also part of its interface to the environment. As indicated in describing actions, there may be additional actions defined beyond those that affect the environment, but these are a starting point. In this case the agent can perform the external actions of *squirting* which reduces the fire at the current location, the action of *moving* which moves the agent an unspecified number of steps along a given route, and the actions of *telling* (broadcasting) and *saying* (sending) a message. The move and squirt actions may need to be applied repeatedly to achieve the desired goal, for example a fire may need to be squirted multiple times before it is extinguished and the agent may need to move several times before it reaches its destination.

We also identified early on the action of planning a route between two points as an internal action. A route is a necessary parameter for the move action, code existed for planning a route given the map, a current position and a destination, and there seemed to be no clear advantage in using goals and plans to achieve this task, particularly given the existing code.

Thus we had an initial set of five actions, four that were part of the external interface to the simulator and one which was internal.

Goals: An obvious major goal of the fire engine agent is to put out any fire. We also identified a goal of discovering fires. Additional thought yielded for us the further goals of assisting a team-mate to put out a fire and coordinating a team effort to extinguish a fire. The goals of the agent obviously have to do with the motivation for the system, but are not externally defined in the way that at least some of the percepts and actions are. Choosing the appropriate set of top-level goals is one of the early design decisions that need to be made.

It is tempting to treat goals as being implied by the beliefs: any belief in the existence of a fire implies a goal extinguish the fire. However, this approach has a number of issues. Firstly, it is hard to indicate that certain fires should not be pursued (e.g. because another agent is dealing with them). Secondly, it is difficult to add goals which are not directly prompted by environmental cues. For example, the goal to find fires exists independent of cues in the environment, and should result in exploratory behaviour if no other goals are inhibiting this goal.

In addition to identifying the goals of an agent, we need to specify when to adopt and drop goals (including how to recognise when they are achieved), plans for achieving these goals, sub-goals that may be part of achieving each goal and relative priorities and interactions between the goals.

As indicated in section 3, events, i.e. significant things which happen in the environment, often result in the adoption of goals. Thus the event of receiving a message from a team-mate requesting help in extinguishing a fire is likely to result in a goal to assist that team mate to put out his fire. The event of noticing a fire is likely to result in a goal to extinguish that fire. Thus we move onto events.

Events: Here we are identifying significant occurrences that are likely to make the agent add or delete goals, change goal priorities, or change how the agent is pursuing a goal. We also look for significant occurrences which affect our beliefs. Many events will be the result of processing percepts - e.g. extracting information about fires and their locations, checking these against a list of fires we already know about, and obtaining any "new-fire" events. Some events will also be generated internally as a result of the agent's own behaviour. For example after sending a request for assistance with a fire, the agent may generate a "help-requested" event. As events are used to update beliefs, trigger plans and generate reflexive actions there are likely to be a large set of events which are developed iteratively in the process of developing the full design. The detailed design methodology we report in our technical report allows for a layered identification of events in conjunction with incremental refinement of sets of plans.[7]

The initial events that we identify typically have to do with the significant occurrences that will alert the agent to the need to instantiate one of its top level goals, recognise one of its top level goals as achieved, or indicate a need for reconsideration.

In this example such events include

- "new-fire" which causes instantiation of an "extinguish" goal and also causes a reflexive action to broadcast the existence of the fire to other fire-engine agents;
- "fire-extinguished" event which causes the agent to recognise that a goal has been achieved;
- "fire-urgent" event which indicates fire is growing and is larger than a given threshold. This indicates a need for reconsideration and in our design may lead to a goal to coordinate a team effort to extinguish the fire;
- "help-requested" which leads to a reconsideration of current priorities and may lead to a goal to assist team-mates;

Identification of events determines what interpretation or processing we need to do with the percepts received in order to be able to recognise events. This in turn also often affects what information about the environment we need to represent - e.g. to be able to generate a new fire event from a visual percept we need to explicitly represent which fires we already know about.

[7] These sets of plans and related events are actually JACK capabilities.

Beliefs: Beliefs are really any knowledge the agent maintains. Some of this information may be kept in the form of a special purpose knowledge database, other information may be kept in arbitrary suitable data structures. For this application the primary information needed is an internal map of the environment including the location of fires, roads, buildings, etc. Updating of this data structure is part of the important processing of percepts.

Beliefs tend to be used primarily in two ways. The first is in extracting events from percepts: to recognise a new fire we have to have knowledge about existing fires. The second is in determining which plan should be used to achieve a goal in a particular situation. Information that allows us to choose between two alternative ways of achieving a goal - or even whether there is any way of achieving a goal, depends on some representation of beliefs.

Some of the beliefs we have identified as important here (in addition to the map) are: which fires are being attended to and by who; current priority of fires; and whether a route is available to a particular location (it may not be, either due to blockages, or insufficient information about the environment).

Plans: Plans describe various ways for us to achieve our goals. To get the full power of the BDI approach it is advantageous to define simple plans, with use of sub-goals wherever possible. Initially there may only be one straightforward way to achieve each goal, but new variations can be added in a modular fashion. This allows development of a simple agent that manages straightforward cases first, with addition of variations for more complex cases afterwards.

Plan sets need to be checked for coverage and overlap regarding the situations which can arise.

A very simple pair of plans for achieving the goal to extinguish a fire at X would be one plan which simply squirts until the fire is extinguished (suitable if the agent is already at location X), and another plan which obtains a route to X, moves to X, then squirts until the fire is extinguished (suitable if the agent is not already at X). Alternatively we could have a plan for putting out a fire and a plan for extinguishing a fire as shown below:

Goal: put-out_fire(position)	**Goal:** extinguish_fire(position)
plan_route(position)	**Condition:** location = position
move(route)	squirt
extinguish_fire(position)	extinguish_fire(position) **unless** nofire(position)

In fact this plan is simple enough that it can be implemented reactively using a set of trigger response rules, all that needs to be stored for this approach is the target's coordinates. However this approach is unable to handle sequences of actions where the triggers aren't in the environment and is unable to manage commitment. For these reasons plans need internal state to track what has been done, and need to be able to specify a sequence of actions to be done.

5 Discussion

We have presented the concepts we have found important in building BDI agent systems and the relationships between these concepts. We have found these concepts to be clearer and easier to teach and use than the BDI model.

The concepts presented:

– Distinguish between goals and sub-goals, between percepts and events, and between events and goals. BDI implementations, by comparison, do not distinguish these and

merge them all into an "event" type. We feel that this distinction is important since goals and events play roles in achieving reactivity and proactiveness and have rather different properties: for example, goals persist until they are achieved.
- Explicitly represent goals. This is vital in order to enable selection between competing goals, dealing with conflicting goals, and correctly handling goals which cannot be pursued at the time they are created and must be delayed [22].
- Highlight goal selection as an important issue. By contrast, BDI systems, such as AgentSpeak(L) [16] simply assume the existence of a selection function.
- Emphasise the importance of percepts and actions.
- Highlight the role of events in creating reactive agents.

We also introduced reflex actions, generalised the concept of a plan, decomposed the concept of intentions (which we have found to be difficult to explain and teach) into the simpler notion of a decision, and provided a hierarchical, staged presentation of the concepts.

Our design process (which is still being refined and developed, and which is explained more fully in [15]) is focussed on the detailed design and we view it as complementary to methodologies such as GAIA [24] and Tropos [14] which focus more on the higher level design and analysis and on the requirements aspects of agent systems.

The choice of concepts was driven by three sources. Primarily, our experience working on a number of agent programs, teaching students and assisting them to build agent programs, and developing and running workshops for academia and industry. Secondly, a survey of a range of agent systems (see below); and finally, "bottom up" derivation of concepts from first principles. We are also in the process of interviewing students and programmers to identify what concepts they see as natural and easy to grasp for developing agent systems.

Although there is some degree of consensus in the deliberative agent research community that the BDI model is a reasonable common foundation for intelligent agents there are also known shortcomings of the model [7]. Thus, in order to avoid being "BDI-biased" we surveyed a range of agent systems which address the internals of an intelligent software agent. Many agent systems do not address the internals of an agent, choosing to focus on other issues such as middleware & infrastructure, and agent mobility. The systems we surveyed were Hierarchical Agent Control, TÆMS, Shoham's Agent Oriented Programming, RETSINA, DECAF, Sensible Agents, AgentSpeak(L), CASA, JIAC, BOID, PaRADE, SIM-AGENT, IMPACT, and Hayes-Roth's Guardian system. Space does not permit a detailed discussion of these systems; for more details see *http://www.cs.rmit.edu.au/˜winikoff/SAC*.

In addition to identifying common concepts which we believe are fundamental, our survey of agent systems also identified concepts which were *not* common and which highlight scope for variation between agent systems. These include resources, capabilities, and intentions. Also, the amount of planning done varies as does the representation of plans, for example some systems represent plans as a graph structure with goals, decomposition, and actions. Other systems treat plans as pieces of Java code. There is also little agreement on how to structure teams of agents and how they interact.

5.1 Further Work

There is much further work to be done including continuing to apply the concepts and design process to various applications. We are also starting to survey students and professionals regarding the concepts they regard as natural for developing agent systems. Developing a formal semantics for the concepts we have identified, as well as developing support tools (design, development, and debugging) are also high priorities. Finally, the concepts identifed need to be extended (and revised) to support the creation of social intelligent agents.

References

1. AAII. dMARS Technical Overview. *The dMARS V1.6.11 System Overview*, 1996.
2. M. E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.
3. P. Busetta, R. Ronnquist, A. Hodgson, and A. Lucas. JACK Intelligent Agents - Components for Intelligent Agents in Java. Technical report, Agent Oriented Software Pty. Ltd, Melbourne, Australia, 1998.
4. P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
5. K. Decker. DECAF update: Agents for undergrads & planning for a smart scheduler. Presentation at the Workshop on Infrastructure for Agents, MAS, and scalable MAS at Autonomous Agents 2001, May 2001.
6. M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dMARS. In M. Singh, A. Rao, and M. Wooldridge, editors, *Intelligent Agents IV: Proceedings of the Fourth International Workshop on Agent Theories, Architectures, and Languages*, pages 155–176. Springer-Verlag LNAI 1365, 1998.
7. M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge. The belief-desire-intention model of agency. In *Proceedings of Agents, Theories, Architectures and Languages (ATAL)*, 1999.
8. M. Georgeff and A. Rao. Rational software agents: From theory to practice. In Jennings and Wooldridge [13], chapter 8, pages 139–160.
9. M. P. Georgeff and A. L. Lansky. Procedural knowledge. *Proceedings of the IEEE Special Issue on Knowledge Representation*, 74:1383–1398, 1986.
10. F. F. Ingrand, M. P. Georgeff, and A. S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6), 1992.
11. N. Jennings and M. Wooldridge. Applications of intelligent agents. In Jennings and Wooldridge [13], chapter 1, pages 3–28.
12. N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.
13. N. R. Jennings and M. J. Wooldridge, editors. *Agent Technology: Foundations, Applications, and Markets*. Springer, 1998.
14. J. Mylopoulos, J. Castro, and M. Kolp. Tropos: Toward agent-oriented information systems engineering. In *Second International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS2000)*, June 2000.
15. L. Padgham and M. Winikoff. A methodology for agent oriented software design. Technical Report TR-01-2, School of Computer Science and Information Technology, RMIT University, 2001.
16. A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. Technical Note 64, Australian Artificial Intelligence Institute, Feb. 1996. Also appeared in Agents Breaking Away (LNAI 1038, p42-55).
17. A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-Architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning, Proceedings of the Second International Conference*, pages 473–484, Apr. 1991.
18. A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In C. Rich, W. Swartout, and B. Nebel, editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 439–449, San Mateo, CA, 1992. Morgan Kaufmann Publishers.
19. RoboCup-Rescue home page. *http://robomec.cs.kobe-u.ac.jp/robocup-rescue/*, 2001.
20. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
21. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.
22. J. Thangarajah. Representation of goals in the belief-desire-intention model, 2000. Honours thesis, supervised by Lin Padgham.
23. G. Tidhar, C. Heinze, and M. Selvestrel. Flying together: Modelling air mission teams. *Applied Intelligence*, 8(3):195–218, May 1998.
24. M. Wooldridge, N. Jennings, and D. Kinny. The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3), 2000.