Technical Report

UCAM-CL-TR-647
ISSN 1476-2986

Number 647

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

An agent architecture for simulation of end-users in programming-like tasks

Sam Staton

October 2005

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

ISSN 1476-2986

Preface

In the Summer of 2001 I was fortunate to be able to work with Alan Blackwell on a prototype of a simulation tool suggested in Section 2.2 of his article

> Blackwell, A.F. (2001). *See What You Need: Helping end users to build abstractions.* Journal of Visual Languages and Computing, **12**(5), 475-499.

The tool used an agent/agenda model to simulate an end-user with regard to some simple programming-like tasks. The present rather informal notes document my work on the system. They were distributed at the CHI 2002 workshop on *Cognitive Models of Programming-Like Processes.*

The results of the 'See What You Need' project are studied in the following paper.

> Blackwell, A.F. (2002). *First steps in programming: A rationale for Attention Investment models.* In *Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments*, pp. 2-10.

I am grateful to Alan for his enthusiasm and encouragement.

Sam Staton
Cambridge, 2005.

Chapter 1

The model

1.1 A basis

We use the following as a basis for our model of the the mind.

- That the mind can be simulated as a society of agents (see Minsky). Each agent exists to accomplish a particular goal. A goal is said to be of a higher level if and only if there exists at least one set of goals (called lower level goals) such that the achievement of all of the goals in this set amounts to the accomplishment of the higher level goal. A higher level goal is said to 'decompose' into the lower level goals. Note that there might be several decomposition strategies for a higher level goal, that is, there may be several sets of lower level goals.

- That the above agents use a medium here called a 'factset' to communicate. The society uses the factset to store probabilistic facts about the state both of the outside world, and of the internal behaviour of the society. For example, the society may wish to store a fact about the number of words in a document, the distance to a nearby town, or the decomposition strategies for a particular class of goal. Further more, the factset is arranged in as a memory hierachy, whereby, for example, a few recently accessed facts are stored where they can be quickly retrieved.

- That, for some definition of *effort*, we strive to minimise the effort expended (see Zipf) [1] This does not lead to the conclusion that we should be in bed for our entire lives, since we are aware to some extent of the relationship between future effort and our present behaviour. Furthermore effort must be defined in appreciation of the full structure of the mind, such that, for example it is effortful not to conform to society.

  Of course, our architecture cannot hope to take into account the full complexity of this definition of effort and the infinite society of agents; in our model we use a simpler definition of effort and add structure to the model to compensate to some extent for the simplifications.

[1]This is of course a recursive assertion: we strive to minismise the effort expended in striving to minimise the effort expended!

1.2 An implementable architecture

Our model is concerned in particular with the achievement of a goal. Thus we consider one particular 'goal agent' whose goal *must* be accomplished. It is as if failing to achieve the goal results in an infinite amount of effort, and we build this notion into the architecture.

The system then proceeds to achieve this goal, with minimum effort. There is no 'accuracy' component: the goal is necessarily entirely achieved.[2]

We also use an imposed structure and some heuristics to explain the way that humans try to avoid future effort. Perhaps this could more purely be explained according to the above principles by saying that it is actually an effort to ignore a future effort, due to the structure of the brain and/or things we have learnt or regretted. Anyhow, in our architecture, future effort is coded directly into the model, with the oversimplified assumption that future effort is avoided just as much as current effort (of course, this is the most 'rational' approach), although in practice details of the effort involved for a future task are known less well.

Furthermore, the inclusion of decomposition specifications complicates the architecture of the fact set. When we decompose a task, we perform some sort of pattern matching to find previous similar tasks that have been decomposed. Thus, in our architecture, we decompose some of the agents[3] in to the design, imposing a design-time structure. However, the society in the mind must contain an infinite number of members, and an infinite data structure cannot exist statically, so it may be necessary to allow for dynamic instatiation and restructuring. Indeed, the decomposition of agents does take time, and a society of fewer agents would operate faster. The society should take this into account, and decide whether decomposition is tactical. Furthermore, in our architecture, we encode the decomposition specifications into the agents themselves: the factset is used entirely for information about the outside world.

Dependencies between the agents are also specified at design time. An agent A is defined as waiting for another agent B iff agent A cannot execute in the current system and the execution of agent B would result in a system in which A could execute. To scan through all the agents to find one (such as B) would take a long time, although this would in some cases yield better results than design time specification.

A final simplication is the use of a flat structure (and its behaviour as a flat structure) in the factset. There is no reason for this other than that the scenarios we have experimented with have been small, and, with the decomposition not relying on the factset to as great an extent, a hierachical structure would make less difference.

[2]An accuracy component can be introduced by defining effort to incorporate it, such that the system exists to accomplish a goal while minimising a function of effort and accuracy. In such a case, it could be argued that the goal should be defined differently. For example, consider the scenario of a secretary in an office, who has been asked to replace misspelt words in a very long document. If the goal was 'to replace misspelt words' the secretary might well spend a very long time achieving this if he/she were only minimising effort, since it is hard to guarantee a document to be entirely mistake free. Thus one might introduce an accuracy component, such that the goal should be achieved with a low effort and high accuracy. To define a function of effort and accuracy is difficult (they have different units, for example). We would propose that in fact the goal should be 'to act as a secretary', and the secretary would then be well aware of the extra effort involved (in terms of discipline, redundancy, embarrasment, etc) if a sensible degree of accuracy was not attained.

[3]In the simplest and present case, we decompose as many as is possible.

1.3 Extensions

Many improvements could be made to the implementation.

- We mention above that a memory hierachy structure could be used for the model; a more precise model should use this. Further, studies of automaticity show that there is a higher level (particularly for knowledge of decomposition) in the hierachy than the 'cache' that we discuss. With this, concurrent behaviour seems to becomes possible. We have not accounted for this in the above model.
- Better implementation of the tactical decomposition (probably involving the factset to some extent) would provide more accurate performance.
- Above we discussed that dependencies have been specified at design time. The increased complexity of the system if they were to be investigated at run time would lead to the system thinking forward more, since the system would be permanently aware of the changes in the effort required to achieve the goal.
- In our architecture, we have used the mean as an estimator for comparison; perhaps the architecture could be made more accurate (in terms of results, as opposed to representation) quickly by adopting a less rational estimator, such as proposed by Tversky and Kahnemann.
- It might be preferable to more openly incorporate accuracy into the model somehow; in some scenarios it is almost as important as the future effort component that we *have* included.
- The relationship between future effort and present effort could be made more explicit; we could expose a function mapping a future-effort/time-until pair to the present effort that it incurs.

A complete implementation of this model may then even serve as a complete computation system, with the lowest level agents representing the atomic instructions.

Chapter 2

A formal specification

We now present a formalised specification which was written alongside the program code to ensure consistency across the ideas. It is therefore partly in note form but could be of interest to a future worker.

2.1 A system

A system exists to achieve an ultimate goal with minimum attentional effort. For example, a system might exist to fix spelling in a word processor document. The system contains

- A society of agents,
- A factset.

Agents may be created and destroyed; we can hypothesise about different *factsets*; we discuss execution in terms of an *agenda*. Thus every system can be described as a member of Sys, which is the set of all society-factset tuples.

2.2 A society of agents

A society contains a set of agents $A \subseteq A_{\mathcal{E}}$ (where $A_{\mathcal{E}}$ is the set of all agents), each existing to accomplish a goal. Each agent in A has a (possibly empty) set of strategies, itself a subset of the set of strategies in the society, $S \subseteq S_{\mathcal{E}}$, (where $S_{\mathcal{E}}$ is the set of all strategies) such that there exists an injection

$$\textit{child-strategies} : A \times \textit{Soc} \to \mathcal{P}(S),$$

and every strategy has exactly one corresponding parent agent, i.e. there exists a function

$$\textit{parent-agent} : S \times \textit{Soc} \to A_{\mathcal{E}}$$

such that

$$a = \textit{parent-agent}(s, \textit{soc}) \iff s \in \textit{child-strategies}(a, \textit{soc}).$$

Thus *child-strategies* can be said to partition S. In addition, each strategy has a set of child agents, such that there exists a function

$$\textit{child-agents} : S_{\mathcal{E}} \times \textit{Soc} \to \mathcal{P}(A),$$

and a corresponding function

$$\textit{parent-strategies} : A \times \textit{Soc} \to \mathcal{P}(S)$$

such that

$$s \in \textit{parent-strategies}(a, soc) \iff a \in \textit{child-agents}(s, soc).$$

We also define

$$\textit{siblings}(s, soc) = \textit{childStrategies}(\textit{parentAgent}(s, soc), soc),$$

for all $s \in S$, and

$$\textit{siblings}(a, soc) = \{a' \in \textit{child-agents}(s, soc) \mid \exists s \in \textit{parent-strategies}(a, soc)\},$$

for all $a \in A$.

A non-action agent a accomplishes the corresponding goal when all the child agents of at least one strategy have executed. Let Executed(a') be the proposition that an agent a' has executed, then

$$\text{Executed}(a) = \exists s \in \textit{child-strategies}(A) \,.\, \forall a' \in \textit{child-agents}(s) \,.\, \text{Executed}(a').$$

We mention the term non-action agent above; an action agent $a \in AA$ ($AA \subset A$) is one such that $\textit{child-strategies}(a) = \emptyset$. Clearly the proposition Executed(a) must be defined differently from above, in particular, action agents terminate the above recursive definition of Executed].

Let $TA = \{a \in A \mid \textit{parent-strategies}(a) = \emptyset\}$ be the set of top level agents. The goal agent $ga \in TA$ of a system is the agent which represents the goal which the system exists to achieve.

To summarise: a society $soc \in Soc$ (Soc being the set of all societies) is a tuple consisting of

- a set of agents $A \subseteq A_{\mathcal{E}}$,
- a set of strategies $S \subseteq S_{\mathcal{E}}$,
- a goal agent $ga \in A$.

2.3 Random variables

We deal frequently in this architecture with random variables, generally over the reals (all of the domains that we deal with can be mapped to a sub set of the reals). Let RV be the set of random variables, and define a probability density function p_{rv} for $rv \in RV$

$$p_{rv} : \mathbb{R} \to [0, 1].$$

Another notation is

$$P(rv = x) = p_{rv}(x),$$

and we can discuss

$$P(rv < x) = \int_{-\infty}^{x} p_{rv}(x)dx,$$

and so on. We define addition, subtraction, multiplication and division operations, such that

$$p_{rv_1 \oplus rv_2}(x) = \int_{-\infty}^{+\infty} p_{rv_1}(x \bar{\oplus} i) \cdot p_{rv_2}(i)\, di$$

where $\bar{\oplus}$ is the inverse of some operation $\oplus \in \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ such that $x \bar{\oplus} y = z$ if and only if $z \oplus y = x$[1]. We will need to refer to a real estimator $\widehat{rv} \in \mathbb{R}$ of $rv \in RV$. This must be such that $\widehat{rv_1} < \widehat{rv_2}$ if and only if the attribute described by rv_1 has less perceived value than the attribute described by rv_2. An approximate estimator is the expected value, i.e. the arithmetic mean across the probability distribution, although a more accurate estimator in terms of human perception might be found by applying principles described by Tversky and Kahneman in their prospect theory, discussed later.

We then define $\mathrm{argmax}_{x \in X}\, f(x)$ (for a function $f : X \to \mathbb{R}$) to be the value of x such that

$$\neg \exists x' \in X \,.\, f(x') > f(x).$$

We define argmin_x similarly.

2.4 Factsets

There is an entity called a factset, which is a collection of named random variables describing the outside world, and some internal characteristics, as perceived by the system of agents. We define the set of factsets $FS \subseteq (String \times RV)$, where *String* is the set of all strings over (say) the roman alphabet.

2.5 Execution

When an action agent executes, the factset is changed: both the environment and the perceived environment are changed. In addition the society may also be changed: a new agent may be introduced[2] or the structure may be changed[3], although in general the goal agent will remain the same.

We formalise this by defining an execution relation

$$\rightsquigarrow \subseteq Sys \times AA_{\mathcal{E}} \times Sys,$$

so that

$$sys \overset{a}{\rightsquigarrow} sys'$$

indicates that the execution of agent a in system sys generates a system sys'.

[1]The inverse of add is subtract, the inverse of multiply is divide, etc.. Observe that for addition, the integral is the convolution; this can quickly be computed via the fast fourier transform. However, we could not find a quick technique for evaluating for the other operations, and the obvious technique (via numerical integration, from the given equation) is unfortunately very time consuming.

[2]The set of agents in the society (A, which is generally finite) is a subset of the set of all agents ($A_{\mathcal{E}}$, which is infinite), and the knowledge gained by the execution of an agent may indicate that an agent not previously considered in the society, should be, possibly to the extent that the goal of the system cannot be acheived without it. It may also be possible to remove an agent, for example, if it has read a particular word and it is not conceivable that it should ever be beneficial to read it again.

[3]For example, knowledge gained or the effect of execution may cause dependencies to change.

2.6 Sequencing

It may be that a particular agent *cannot* execute in some environments; for example, one cannot complete a 'search' dialog if it is not visible. Thus we define a relation $\mathit{available\text{-}in} \subseteq A_{\mathcal{E}} \times \mathit{Sys}$ such that

$$a \ \mathit{available\text{-}in} \ \mathit{sys} \iff \exists \mathit{sys}' \in \mathit{Sys} \ . \ \mathit{sys} \overset{a}{\rightsquigarrow} \mathit{sys}';$$

the negation, $\neg(a \ \mathit{available\text{-}in} \ \mathit{sys})$ can be denoted $\mathit{sys} \overset{a}{\not\rightsquigarrow}$.

We define a relation

$$\mathit{waiting\text{-}for} \subseteq A_{\mathcal{E}} \times \mathit{Sys} \times A_{\mathcal{E}}$$

such that $a \, \mathit{waiting\text{-}for}_{(A,S),\mathit{fs}} \, a'$ if and only if $a \in A$ cannot execute in the system $((A,S),\mathit{fs})$ until $a' \in A$ has executed, that is,

$$a \ \mathit{waiting\text{-}for}_{\mathit{sys}} \ a' \iff \neg a \ \mathit{available\text{-}in} \ \mathit{sys} \wedge a \ \mathit{available\text{-}in} \ \mathit{sys}' \wedge \mathit{sys} \overset{a'}{\rightsquigarrow} \mathit{sys}.$$

Then we define a relation

$$\mathit{depends\text{-}on} \subseteq A_{\mathcal{E}} \times \mathit{Sys} \times A_{\mathcal{E}}$$

such that $a' \ \mathit{depends\text{-}on}_{\mathit{sys}} \ a \iff \exists \mathit{sys} \in \mathit{Sys} \ . \ a' \ \mathit{waiting\text{-}for}_{\mathit{sys}} \ a$, and

$$\mathit{prerequisites} : A_{\mathcal{E}} \to \mathcal{P}(A_{\mathcal{E}})$$

such that $a' \in \mathit{prerequisites}(a) \iff \exists \mathit{sys} \in \mathit{Sys} \ . \ a \ \mathit{waiting\text{-}for}_{\mathit{sys}} \ a'$. Thus something may depend on something if it would be waiting for it in any conceivable environment.[4]

2.7 Properties of agents

Associated with each agent is:

- a cost function $\mathit{cost} : A \times \mathit{Sys} \to \mathit{RV}$, returning the effort consumed in carrying out the task described by the given agent, within the given system[5];
- a payoff function $\mathit{payoff} : A \times \mathit{Sys} \to \mathit{RV}$, returning the reduction in the required amount of effort to acheive the goal of the system as a consequence of the knowledge gained;
- a benefit function $\mathit{benefit} : A \times \mathit{Sys} \to \mathit{RV}$, returning the reduction in the effort required to acheive the goal of the given system, not as a consequence of the knowledge gained but as a consequence of the effect on the external system; and

[4] In our architecture, we generally define only the *prerequisites*, and define the other relations approximately from that. Clearly, a search through the results of all agents would take an infeasible amount of time

[5] In our implementation we only take into account the factset aspect of the system; to take account of changes in structure would obviously yield more accurate results but at the expense of much longer execution times.

- a balance function *balance* : $A \times Sys \to RV$ such that for all $a \in A$ and $sys \in Sys$

$$balance(a, sys) = benefit(a, sys) + payoff(a, sys) - cost(a, sys).$$

We can define *chosen-strategy* : $A \times Sys \to S$ as

$$chosen\text{-}strategy(a, sys) = \underset{s \in child\text{-}strategies(a)}{\operatorname{argmin}} \left\{ \widehat{balance(s, sys)} \right\},$$

and define a predicate *Chosen-Strategy* such that for a strategy $s \in S$ and $sys \in Sys$,

$$\begin{aligned} &Chosen\text{-}Strategy_{sys}(s) \iff \\ &\qquad (\ s = chosen\text{-}strategy(parent\text{-}agent(s), sys)\ ) \\ &\qquad \wedge\ (\ \exists s' \in parent\text{-}strategies(parent\text{-}agent(s))\ . \\ &\qquad\qquad Chosen\text{-}Strategy_{sys}(s') \\ &\qquad\qquad \vee\ parent\text{-}agent(s) = ga\ ) \\ &\qquad \vee\ \exists a \in A\ .\ a\ waiting\text{-}for_{sys}\ parent\text{-}agents \wedge Chosen\text{-}Strategy_{sys}(a) \end{aligned}$$

and similarly for an agent $a \in A$ and $sys \in Sys$,

$$\begin{aligned} &Chosen\text{-}Strategy_{sys}(a) \iff \\ &\qquad (a = ga) \vee \big(\exists s \in parent\text{-}strategies(a)\ .\ Chosen\text{-}Strategy_{sys}(s)\big)\ . \end{aligned}$$

In practice we cache the evaluated properties and only re-evaluate for different factsets.

2.7.1 Definitions for action agents

For an action agent $a \in AA$, $cost(a, fs)$ returns the perceived attentional effort[6] required to carry out the task described the agent a. It will most probably depend on the perceived environment. The benefit of an action agent $a \in AA$ is the reduction in cost of the goal function due to the execution of a, i.e.

$$benefit(a, sys) = \begin{cases} cost(a, sys) & \text{if } Chosen\text{-}Strategy_{sys}(a) \\ 0 & \text{otherwise} \end{cases}$$

The payoff of an agent is defined as the reduction in wasted effort due to the execution of that agent. Wasted effort is a random variable, defined such that

$$\begin{aligned} p_{wasted\text{-}effort(a,sys)}(x) = P\ (\ &\exists s \in child\text{-}strategies(a)\ . \\ &balance(chosen\text{-}strategy(a), sys) - balance(s, sys) = x\ )\,, \end{aligned}$$

i.e. the probability that another strategy would take precisely x less effort. Then

$$payoff(a, sys) = wasted\text{-}effort(a, sys) - wasted\text{-}effort(a, execute(a, sys)).$$

A positive payoff is, then, a measure of the more informed decision being made due to knowledge learned about the world as a result of an action.

[6]Here we describe attentional effort in terms of a real number, prinicipally because values of attentional efforts are well ordered. Attentional effort can be described as a function of a number of properties, including time; a simple approximation is to say that the real number representing the attentional effort consumed is the time taken.

2.7.2 Definitions for non-action agents and strategies

We can define these properties for strategies:

$$prop(s, sys) = \sum_{\forall a \in child\text{-}agents(s)} prop(a, sys)$$

for some strategy $s \in S$ and where *prop* is one of *cost*, *payoff*, *benefit* and *balance*. We can then define these properties for a non-action agent $a \in (A \setminus AA)$:

$$prop(a, sys) = prop(chosen\text{-}strategy(a, sys), sys).$$

2.8 An agenda

The agenda controls the behaviour of the system. In paricular, an agent a can will transition a system *sys* to a system *sys*′ (notated $sys \xrightarrow{a} sys'$) if and only if requested to by the agenda. The agenda is responsible for organising the computation of the various properties of agents, and enforces the rule

$$sys \xrightarrow{a} sys' \iff \left(sys \overset{a}{\leadsto} sys'\right) \;\wedge\; a = \operatorname*{argmin}_{a' \in A} \left\{ \widehat{benefit(a'}, sys) \right\}$$

where $sys = ((A, S), FS)$.

Note that if two agents have very similar estimators of their balances, the behaviour becomes less ‘deterministic’; small changes in the perceived view of the world have marked effects on the future behaviour.