Reacting to Unexpected Events and Communicating in spite of Mixed Ontologies

Adolfo Guzman, Carmen Dominguez and Jesus Olivares

Centro de Investigación en Computación, Instituto Politécnico Nacional, Av. J. Bátiz esq. Miguel Othón, C. P. 07738, Mexico City.
aguzman@cic.ipn.mx, http://www.cic.ipn.mx

Abstract. We describe our experiences in building agents (and their environment) that could solve important problems in agent to agent communication, in manners that are not pre-programmed or reactive: *(1) Two agents may have different ontologies.* We do not assume that agents share the same classification of concepts: they have each one its own ontology or concept categorization. Agents can not exchange concepts: they have to exchange symbols (words of natural language), which the receiving agent has to map to the intended concept, for interaction to be meaningful. *(2) The model for interaction.* Our agents interact through scripts or frames having roles, resources and prerequisites. Agents select which roles in what scripts to play, in order to satisfy their purposes. *(3) Unexpected events happen* and throw the agents out of their current plan or execution. *(4) Planning* is needed in this changing world.

Key words: Ontology, unexpected events, concept tree, script, multiple threads.

1 Introduction

The project seeks to produce a theory, model and environment populated with agents that follow useful behavior (such as "go and sell these apricots") derived from purposes that take into account resources. These agents interact with agents written elsewhere, thus differing in their intentions, models, vocabulary, and messages.

Agents engage in "orchestrated interactions" or scripts. An agent must obey not only its own rules (behaviors, algorithms): it must also adopt the rules imposed to it by the scripts on which it takes part. For instance, if it goes into a restaurant in the role `customer` (to eat something), it must obey (most of) the rules that the script `at_a_restaurant` impose on the `customer`. In this way, an agent acquires additional obligations (additional threads to execute) on top of those initially owned.

Different ontologies do exist. Most agents are not built by us, but by somebody else. Thus, our agents will interact mostly with "unknown and never seen before"

agents from all over the planet. These will express their wishes or purposes uttering different messages coming from their different knowledge structures. Hence, mechanisms to exchange knowledge among heterogeneous systems are needed.

The need for multiple threads. For versatility, an agent may have several behaviors: algorithms or threads that pursue some purpose (`how_to_be_rich`, `how_to_cross_the_street`, `how_to_bargain...`) and may be simultaneously executing several of them, and thus pursue several purposes concurrently.

Unanticipated events will happen. An agent was `selling_a_car`, when the `buyer` ran away. What to do? An agent can not have a specific program to handle every possible exception (the "frame problem" of John McCarthy); it must have more general rules. Unexpected events are words in the second tape of an Interaction Machine [16].

2 Related Works

Languages for agent execution. KQML [2], a declarative language, is a message format and a message handling protocol to enable agents to interoperate and share knowledge in run time without any concern about the contents of the messages. TCL [5] is a high-level scripting language that enables a user to define mobile agents. Tele-Script is also an important recent development in the area. Java is multithreaded. Creating LIA (Language for Interacting Agents) facilitates easier coding and study of desired features. It differs from KQML in the sense that KQML is not concerned with the content of the message and therefore makes no changes nor equivalence among ontologies. LIA differs from TCL in that LIA agents communicate even if they have different ontologies. TCL and KQML are not concurrent.

Interaction Machines. They are defined [16] as extensions to the Turing Machine model to enable it to deal with models that are incompletely specified and that can be completed interactively; they are concerned with the incompleteness of Godel's theorem. An Interaction Machine is a Turing Machine extended with an additional tape that contains an infinite number of (infinite types of) strings written with letters of an (infinite) alphabet. Interaction Machines enable us to model open systems: those exposed to external events, such as unplanned events. Our work on these events is inspired by this model, which postulates an infinite number of *types* of external events, with an *infinite alphabet.* It is thus impossible to write a program to handle even each *type* of event (there is an infinite number of them). To overcome this difficulty, we organize the (infinite) collection of "unanticipated" strings into a tree of unforeseen events (Figure 1), following the lines of CYC [6] and Clasitex [9].

Ontologies. The first author worked in the CYC Project [13], which tried to construct the common knowledge tree in order to solve big problems in Artificial Intelligence. CYC shows that it is possible to form taxonomies of specialized knowledge areas (which is what we intend to do here, see Table 1), in addition to classifying the common knowledge (goal that, due to its extension –between one and ten million concepts– was not achieved by that project). Trees of specialized knowledge, where the tree takes the form of a data dictionary, were used by the first author [7-9].

Similar scenarios. [11, 12] describe a scenario similar to the one we propose, but with single-threaded code and a common ontology, outlining how a set of autonomous agents cooperate to coherent management of information in environments where there are diverse information sources.

Our current and previous related work. This paper is based on our theses [1] and [15]. [10] describes earlier work. [7] uses a common ontology to map (manually) the data dictionaries of an otherwise strange data base, thus making its tables, fields and values understandable to the casual user. Written in db manager Progress. [9] relates words to concepts; it finds the main topics in an article written in Spanish. It does not work on key words, but on concepts. It uses a concept tree. For this reason, it can find an article talking about shoes, even if the article does not contain such word, but it contains instead boot, moccasin, sandals..., even if these words may refer to other contexts or concepts: moccasin is also a tribe of American Indians... [3, 4] extend [9] by using weights for selecting concepts.

2.1 Status

A *theory* has not been developed, except a simple one for unforeseen events. A *model and a working environment* have been developed. An *imperative language*, LIA, and its programming environment has been constructed (in C and Java, for a PC), and used for simple examples. We wrote a *compiler* from LIA to p-code, which is then interpreted. This provides for easy change of LIA syntax and semantics. Once LIA is frozen, we will probably build a compiler from it to Java. Now, the environment assumes that all agents work inside the same computer; later, the agents should be set free to run on different nodes of the Web. For this, we plan to make it FIPA-complaint. MEI, the *Machine of Unexpected Events***,** which contains a simple micro planner (Section 5.1), as well as COM, the *Ontology Comparator*, are working. There is a *parallel planner* [15]. COM now works with fixed relations (verbs); we will relax this later, through the use of nodes in the ontologies representing *the relations.* We propose work on automatic handling and *recovery of e-errors in e-commerce*, based on MEI. *Applications* in LIA are scanty, due to its youth.

3 Model for Agent Interaction in Our Work

This is an overview; the next sections provide more detail.

3.1 Agents Are Multi-Threaded, Have Resources and Purposes

Scripts (`renting_a_house, at_vacation, at_an_auction`) describe the intercourse between several *roles* (`cooker, student, owner...`), following R. Schank and Marvin Minsky [14]. Each role can be instantiated later by an agent. Each role is a program (a thread) in LIA, having requirements (prerequisites for its

instantiator agent), resources consumed, purposes achieved and resources produced at the end of the interaction. Scripts are not active, they do not run until its roles are instantiated by agents. A role interacts (exchanges information) with another role via LIA commands *accept* and *out* [15].

Agents are autonomous, proactive units (individuals; software packages) that initially possess several threads (for instance, `how_to_swim`, `how_to_be_honest`), purposes (Ex: to sell these apricots, to buy a piano...)[1] and resources (apricots, $10,000, a VW car, knows how to cook...). Each agent decides which of its threads to activate, in view of purposes and resources. Often, to achieve a purpose, it must engage in interactions via scripts with other agents. It does so by voluntarily acquiring (obeying, following, instantiating) some *role* in some *script* (those that best match[2] its abilities and resource limitations). An agent may engage in several *scripts* simultaneously.

An agent may replan its purposes in view of achievements, closeness to purposes, and resource status.[3] A fortuitous event (Section 5) may alter the plan of the agent and reaction threads may be used to deal with it. This forces them to micro planning, and macro planning [15].[4]

3.2 There Is an Environment, and a Language for Enacting these Agents

The environment provides: (a) an editor in which to write LIA threads, define agents and scripts; (b) a P-compiler for LIA; (c) an execution machine, that executes the P-code; (d) MEI; (e) the ontology comparator; (f) global and regional variables for the agents; (g) a matcher of agent purposes to resources produced by each role of a script[2]; (h) a (macro) planner.[3, 4] The LIA world contains resources, global variables (time, temperature...), regional variables,[5] agents, scripts, and unexpected events. Agents and scripts are constructed using LIA [15].

Differentiating features of our work: (a) it handles unforeseen events; (b) agents can communicate even if they use different knowledge organizations/structures.

3.3 They Communicate Using Each One its Own Ontology

Unless agents are written by the same person, they can not be sure that they exchange *words* that are universally understood by everybody.[6] Thus, they have to exchange

---

[1] A *purpose* is a proposition that, when it becomes True, it is considered fulfilled.

[2] This matching is not described here, but it also uses COM and the planner.

[3] At the moment, this replanning is "automatic," instead of being agent-requested.

[4] Planning occurs not only because unforeseen happenings, but also due to resource depletion, failure or forfeit in a given interaction (the `buyer` ran away; I could not sell my car).

[5] Regional variables are used ("seen") only by agents and roles which declare them.

[6] There are some words or symbols that are unambiguous (map to a unique meaning or concept): 7, π, London, Ludwig van Beethoven, Fourier Transform... Most words (mole, star...) are ambiguous. A concept is usually represented by more than one word: the concept that I

ambiguous words or descriptions of what they want or mean. Since we want our agents to communicate with *your, his* and *her* agents, we do not assume that all agents use the same interpretation for a natural language word, or that they share the same concepts or the same concept organization. Instead, agents have to face the problem of how to make sure that what I hear is what you mean; i. e., to be reasonably sure that my mapping of *your* words to my concept is probably what *you* had in mind. If in doubt, our agents ask clarifying questions or queries to the other agent, until an acceptable meaning (concept) is transmitted.

Table 1. Two similar but not identical ontologies or trees of concepts. Concepts appear in bold: **Grain,** meaning a seed of a plant used for eating. Words appear in italics: *grain,* which may mean **Grain** (seed of plant), **Small particle** (*bit, pellet, grain, speck, fragment*), or **Texture** (*texture, grain, nap, striation*).

Ontology of Agent A	Ontology of Agent B
Fruit *fruit, fruits*	**Seed** *seed, grain*
Grain *grain, seed, seeds*	**Sorghum** *sorghum*
Bean *bean, frijol*	**Oat** *oats, oat, grits*
Soya bean *soya bean*	**Bean** *bean, kidney bean*
Black bean *black beans*	**Black bean** *black bean, frijol negro*
Cereal *cereal*	**Soya bean** *soya bean, soybean*
Wheat *trigo, wheat*	**Corn** *maize, maiz, maíz*
Corn *corn*	**Wheat** *trigo*
Sorghum *sorghum*	**Peanut** *peanut, maní, cacahuate*
Citric *citric, citrics*	**Fruit** *fruit*
Orange *naranja, orange*	**Tangerine** *tangerine*
Lemon *lemon, limón,*	**Lemon** *lemon*
Apricot *apricot*	**Avocado** *avocado*
Pineapple *piña, pineapple*	**Pineapple** *pineapple*
Avocado *avocado, avocados*	**Orange** *orange*
	Prune *prune*

4 Communication between Agents that use Different Ontologies

This section presents our approach (others exist) on how agents using different dialects or concept hierarchies (trees of concepts) communicate meaningfully.

Matching words arising from concepts in different ontologies. We describe here COM which, when two agents interact, has to map words to concepts. When an agent

---

have in my mind of a certain cereal, written *maiz* in Spanish, is also mapped into by words such as elote, pozol (Spanish), maize, corn (English),...

(A, say) utters a word (*corn*, in "I want to sell *corn*") to B (the listening agent), several cases arise:

(1) B knows word *corn* (not the example in Table 1) and maps into the same concept which has the same father in B. Thus, *corn* in A maps to **Corn** son of **Seed**; in B also into **Corn** son of **Seed.** We can say that A has transmitted **Corn** through *corn* to B.

(2) B has no knowledge of word *corn* (Refer to Table 1). In this case, B guesses and asks A (all this intercourse is done by COM, automatically, without explicit calls from A or B, so that –if COM is successful– they appear to be transmitting concepts among themselves, when in fact they are exchanging words): *"Is it a kind of a fruit?"*. A answers with **Cereal**, the father of **Corn**, and transmits *"it is a cereal"*, which makes a recursive call to COM. In our example, word *cereal* is also not understood by B. Then A tries **Grain,** the father of **Cereal,** and transmits *"it is a grain or seed or seeds"*. B has these words and thus knows that A is talking about **Seed**. [We went up in the ontology trees, looking for some common concept. If one or two steps upwards do not produce a match, perhaps their ontologies are incomparable, and further communication is impossible.] Now, we want to go down; A must convey **Corn** to B**,** not just **Grain.** Thus A sends B all the sons and grandsons of **Grain,** together with their properties: (*bean size 1cm, skin smooth...), (trigo size 1mm, skin smooth, shape elliptical...)...*, which B must match against the sons and grandsons of its **Seed.** This matching is a recursive call to COM, since what is *skin* for one is *peel* or *epidermis* for the other, what is *pale orange* for A is just *orange* for B, and one expresses sizes in centimeters, but the other in inches... [7]

(3) B knows *corn* as **Corn** but its father is **Seed,** while in A, the father of **Corn** is **Cereal,** and the father of **Cereal** is indeed **Seed** (not the example in Table 1)**,** thus:

A: **Seed – Cereal – Corn** B: **Seed – Corn**

In this case, a match is obtained, although **Corn** has in A a grandfather that matches just the father (not the grandfather) of **Corn** in B.

(4) when *corn* arrives to B, it may have many possible matches. (Think of you wanting to buy a screwdriver, and talking to a hardware store clerk that sells 25 classes of screwdrivers). A: **Screwdriver** *screwdriver* B: **Screwdriver** – **Phillips screwdriver, Flat screwdriver**, **Z-shaped screwdriver...**

In this case, a match has been found by COM, and further disambiguation is not possible by using the ontology tree of A (its **Screwdriver** has no leaf nodes), but by resorting to use, price... intended ["For what do you want the screwdriver?" "How large?". This extension is beyond current COM].

(5) More cases exist [15], but the reader gets the idea.

How big can a given tree of concepts be? CYC [13] assumes that there are between one and ten million common concepts (common sense concepts, that everybody shares). From this tree, an agent is interested only in a few hundred: seeds and their

[7] The concepts in an ontology have properties (attributes) and values, not shown in Table 1.

properties, say. A common tree for concept disambiguation is not needed, but it may help.

5 Handling Unexpected Events

Autonomous agents, as well as human beings, must face the fact that the world is unpredictable, due to incomplete information, uncertain environment, unknown processes, acts of other agents, or just Murphy's Law. An unexpected event, when sensed by an agent, modifies its participation in *scripts*, forcing it to execute contingency or emergency roles (called reaction threads), to postpone or cancel some current scripts in which it is engaged, and later, to do replanning.[3] Non perceived events are ignored, although that may lead to catastrophe.

Unexpected events are handled by MEI, a machine placed outside the interaction environment, which (1) produces unexpected events at random times; (2) locates each agent capable of perceiving an event,[8] when it occurs; (3) for each of these agents, MEI interrupts its threads, (4) selects and starts some reaction thread in response to the event; (5) activates some of the interrupted threads; (6) detects the end of the event; (7) activates some more of the interrupted threads, and (8) stops (usually) the reaction thread. In this manner our agents react to unexpected events. We will later incorporate into each agent calls to functions (2) to (8), to make them more autonomous.

5.1 How to React to an Infinite Number of Unexpected Events

There is an infinite number of fortuitous events, but an agent can only have a small number of predefined reaction threads: it can know how to react to "winning the lottery" (reaction depends if it has no savings, owns its home...), but not how to react to "finding some money". How can it survive? With the help of the tree of unexpected events. This tree (Fig. 1) is infinite in principle, and each node contains an event and the names of the possible reaction threads for that event; see also Fig. 2. More general events appear near the root. The branches denote the relation "subset". Reaction threads often have preconditions for they to be useful: "must have umbrella", "must have raincoat", "must be near a shade"... are some preconditions for certain reaction threads for event "rain".

When one of the infinite number of unexpected events occurs, a perceiving agent uses the tree of unexpected events to select, from the reaction threads it owns, the most specific one pertaining to the event. In this manner (4) above gets executed.

Every agent owns at least one reaction thread: the most general one.

---

[8] An agent perceives (detects its beginning and end times, as well as its other features: intensity...) an event if it has a reaction behavior for that event or for an specialization of it.

To know which of the normal threads stopped in (3) are continued in (5), in spite of the unexpected event, the agent uses an *incompatibility* algorithm (not described here) to detect which threads can not run simultaneously with the reaction thread (4).

The reaction thread started in (4) is ended (8) when its *purpose* has been achieved, usually because the unexpected event stops. At this moment, the incompatibility algorithm restarts (7) some more of the threads that were not restarted in (5), due to incompatibility with the unexpected event. Finally, some of the threads stopped in (3) are never restarted, due to lack of resources. Replanning[4] (not discussed here) may be needed. Execution of (2) to (8) is called microplanning. [15] shows applications.

unexpected_event: `still; astonished.`
- **natural_event**: `cry; run.`
  - rain: `open_umbrella; wear_raincoat; run; get_wet.`
  - earthquake: `faint; freeze; help_people; hide.`
  - fire: `call_fireman; run; tell_others.`
  - ...
- **life_threatening_event**: `cry; pray.`
  - assault: `cry; call_police; call_family.`
  - accident: `call_ambulance; call_police; get_scared.`
  - sickness: `call_doctor; tell_others.`
    - sick_pilot: `replace_pilot.`
  - ...
- **lucky_event**: `be_happy.`
  - won_lottery: `save_money; buy_house; buy_car.`
  - bumped_into_old_friend: `go_to_the_restaurant; greetings.`
  - ...
- **offers_event**: `accept_proposal; be_happy.`
  - offer_job: `begin_work.`
  - offer_gift: `accept_gift.`
  - ...
- **lack_event**: `ask_help, stay_suspended.`
  - things_lack: `replace_for_newOne.`
    - airplane_lack: `cancel_fligh; delay_flight.`
    - money_lack: `request_to_the_Bank.`
  - defect: `repair_defect.`
    - fail_airplane: `move_passengers.`
    - product_defect: `replace_for_newOne.`
  - ...
- **... (many_other_events):**...

Fig. 1. Tree of unexpected events. In **bold** are the events; in `Courier font` are the possible reaction threads to that event. Every agent is born with a finite number of reaction threads. The tree may be infinite, since it contains *all* possible reaction threads.

```
Role open_umbrella()


print(" Open the umbrella");  //no further
        ctions
```

Fig 2. Reaction thread, written in LIA. This role sits in some agents, those that are able to perceive rain and know that one way to react to rain is `open_umbrella`. An agent may have other reaction threads for rain, for instance `run, wear_raincoat`. These are the reactions known to the agent for unexpected event rain. Which one to execute depends on the resources available to the agent at the time of the rain.

6 Conclusions and acknowledgements

Communication of concepts among unfamiliar agents must be through symbols or words of a natural language. The receiving agent must map the symbols or words it receives, into the right concept in its *own* ontology; hence, the paper gives a useful solution to the problem of mapping a concept in one ontology to the closest concept in another.

Unforeseen events are handled using the background of Interaction Machines.

Acknowledgements. We are grateful to IPN authorities for their wisdom in founding (1997) CIC and its Agents Laboratory. We thank Prof. Michael N. Huhns and his group at University of South Carolina for advice and help during fruitful conversations. Work was partially supported by grants Conacyt-28026, NSF-Conacyt-32973, and CGEPI-980744. The second author acknowledges CONACYT and IPN-COFAA research assistantships.

References

(•) In Spanish. Authors' papers can be read and copied, freely, full text, from CIC's Digital Library, through http: //www.cic.ipn.mx/~aguzman

1. Dominguez, C.: Handling Infinite Unexpected Events in Agents Interactions. M. Sc. thesis in Computer Science, CIC-IPN Mexico City (2002) •
2. Finnin, T.; Weber, J.; Widerhold, G., et al. Specification of the KQML Agent Communication Language (draft). The DARPA Knowledge Sharing Initiative External Interfaces Working Group (1993)
3. Gelbukh, A, Sidorov, G., and Guzman, A.: A Method Describing Document Contents through Topic Selection. Lecture notes in Workshop on String Processing and Information Retrieval, (IEEE Computer Society, Los Alamitos, CA) Cancun, Mexico (1999) 73-80
4. Gelbukh, A, Sidorov, G., and Guzman, A.: Document Comparison with a Weighted Topic Hierarchy. DEXA-99. Lecture Notes in 10-th International Conference on Database and

Expert System Applications, Workshop on Document Analysis and Understanding for Document Databases, (IEEE Computer Society, Los Alamitos, CA) Florence, Italy (1999) 566-570
5. Gray, Robert S.: Agent Tcl. In Dr. Dobb's Journal, March (1997)
6. Guha, R.V. and Lenat, D.: Enabling Agents to Work Together *CACM*, No. 37 (1994) 7
7. Guzman, A.: Project "Access to Unfamiliar Data Bases." Final Report, IDASA, Mexico City. (1994) •
8. Guzman, A.: ANASIN, Intelligent Analyser and Synthesiser of Information. Technical Report: User Manual. IDASA. Mexico City (1994b) •
9. Guzman, A.: Finding the Main Themes in a Spanish Document. Journal Expert Systems with Applications, Vol. 14, No. 1, 2, (1998) 139-148
10. Guzman, A., Olivares, J., Demetrio, A., and Dominguez, C.: Interaction of Purposeful Agents that Use Different Ontologies. In: Osvaldo Cairo, Enrique Sucar, Francisco J. Cantu (eds.): Lecture Notes in Artificial Intelligence 1793, MICAI 2000*:* Advances in A. I. Springer Verlag, Heidelberg (2000) 557-573.
11. Huhns, M. N.; Singh, M. P. and Ksiezyk T.: Global Information Management Via Local Autonomous Agents. In: M. N. Huhns, Munindar P. Singh, (eds.): Readings in Agents, Morgan Kauffmann Publishers, Inc. San Francisco, CA (1997)
12. Huhns, M. and Singh, M.: Managing Heterogeneous Transaction Workflows with Cooperating Agents. In: Nicholas R. Jennings and Michael J. Wooldridge, (eds.): Agent Technology: Foundations, Applications and Markets, Springer-Verlag, Heidelberg (1998) 219-240
13. Lenat, D. and Guha, R.: Building Large Knowledge-Based Systems. Reading, MA: Addison Wesley., Reading, MA. (1989)
14. Minsky, Marvin: The Society of Mind**.** Simon & Schuster Inc. (1985)
15. Olivares, Jesus: An Interaction Model between Purposeful Agents with E-Commerce Examples. In Ph. D. Thesis. Defended in November 2001. CIC-IPN, Mexico City. (2002) •
16. Wegner, Peter: In Tutorial Notes: Models and Paradigms of Interaction**,** Department of Computer Science, Brown University, USA, September (1995)