Appl Intell
DOI 10.1007/s10489-011-0331-y

Critical reasoning: AI for emergency response

Stephen Potter

© Springer Science+Business Media, LLC 2011

Abstract Effective response to emergencies depends upon the availability of accurate and focused information. The goal of the FireGrid project is to provide an architecture by which the results of computer models of physical phenomena can be made available to decision-makers leading the response to fire emergencies in the built environment. In this paper we discuss the application of a number of AI techniques in the development of FireGrid systems, and include algorithms developed for reasoning about dynamic situations. It is intended that this paper will be of technical interest to those who have to construct agents that are able to reason about the complexities of the real world, and of more general appeal to those interested in the ontological and representational commitments and compromises that underlie this reasoning.

Keywords Emergency response · Decision support · Belief revision

1 Introduction

To minimize losses to life and property during emergencies, responders must make decisions in a timely manner; and, if the right decisions are to be made, the availability of relevant information is critical. Technological developments have ensured that computer systems have become an important source of information in even the most difficult and demanding situations, with access to mapping tools, database records and other compiled information being widely accepted and assimilated into responders' systems of work. The successes of these systems have served to inspire investigations into more ambitious uses of computers in this area. The vision behind the FireGrid project [5, 11] is of a generic software architecture that provides decision-makers with access to information based on interpretations (of varying sophistication and complexity) of real-time sensor data polled from within and around the environment of an emergency. In the first instance, the project has focused on supporting the response by emergency services to fires in the built environment.

For obvious reasons, fire-fighters will rarely be aware of the exact conditions that hold within a building during a fire incident and, consequently, they will be compelled to make intervention decisions based on the information provided by their own senses and the reports of others, and drawing heavily on their training and on their past experiences of fires. However, given the complex nature of fire, for even the most experienced of fire-fighters the accurate interpretation and prognosis of conditions is a difficult task. Advances in several technologies when taken together suggest one way in which this difficulty might be eased:

- Developments in sensor technology and a reduction in unit cost offer the prospect of deploying large-scale, robust and cost-effective sensor networks within buildings;
- Advances in the understanding of fire and related phenomena have resulted in sophisticated computer models which might be used to interpret sensor data;
- The availability of distributed High Performance Computing (HPC) and data processing over a grid suggests a platform on which these models, calibrated against and steered by the live sensor data, could be run quickly enough to make their use during emergencies a practical proposition.

S. Potter (✉)
School of Informatics, The University of Edinburgh, Edinburgh, UK
e-mail: s.potter@ed.ac.uk

Published online: 15 December 2011

The FireGrid approach aims to improve—both in range and quality—the information available to fire-fighters. The emphasis of the project lies firmly on the integration of existing technologies in the areas mentioned above rather than on the development of new ones. In practical terms, this involves the coupling of diverse computational models, seeded and steered as appropriate by real-time sensor data, and processed using HPC resources accessed across a grid, and all placed in the context of a system interface tailored to the target user working in his or her operational context. While there have been numerous computer systems developed in recent years to assist emergency response, the motivation driving FireGrid, the combination of technologies it draws upon, and the particular challenges that it presents have no real precedents.

Artificial Intelligence represents the fourth core technological field in the FireGrid project. AI techniques and approaches to software development play an important role in FireGrid, both in the development of software and of the underlying concepts that serve to link together the various components of the architecture. The following section provides some background information to help understand the use of AI in FireGrid. This is followed by a detailed description of some of the AI technologies used in FireGrid, specifically the system ontology and its uses, and belief revision and hazard interpretation mechanisms, with a discussion of the issues raised, and concludes with a brief discussion of the interface developed to display the system results to the end-user. A final section offers some conclusions.

2 FireGrid: background and requirements

In this section we provide more detailed information about the background against which AI is to be used within the FireGrid project. Specifically, we discuss the requirements and constraints that emerge from two stakeholder communities, namely fire-fighters (as the target user group, that is, as 'consumers' of the information provided by a prospective FireGrid system) and fire modellers (as the core technology—and hence, information—providers), as well as describing the nature of the sensor data that is to underpin this information.

2.1 FireGrid and the fire-fighter

Before any FireGrid system could be designed, it was necessary to understand in detail the nature of the task that such a system will support. This has been done by study of the available literature (which, in this case, takes the form of best-practice field manuals) and a series of interviews with serving fire officers, which eventually led to the development of prototype interfaces to elicit feedback.

During emergency response the *tactical* level of activity generally encompasses those directional and command activities that have the express aim of bringing about the overall objectives of the response. This is in contrast to the lower *operational* level, which involves the actual physical activities that individuals and teams of individuals employ to achieve the tactical ends, and the higher *strategic* level, which involves establishing the objectives of the response and, often, of the responding organization as a whole. It is at the tactical level that computer-based decision-support systems are often aimed, since it is there we find a degree of mental reflection and engagement with a specific incident (sense-making and planning), occurring above the (often instinctive or reactive) physical interactions with the incident found at the operational level and below the wider contextual and cultural considerations of the strategic level. FireGrid too aims to provide information of use at this tactical decision-making level; accordingly the most obvious target user is (using UK fire service terminology) the Incident Commander (IC) (or, more realistically, a senior support officer stationed in an on-site command room and detailed to monitor the FireGrid system and report directly to the IC). The IC is:

> ... responsible for the overall management of the incident and will focus on command and control, deployment of resources, tactical planning, the coordination of sector operations ... and the health and safety of crews. [15], pp. 15–16.

Rather than being determined in advance, the range of possible incidents and contributing factors means that the response to any given incident is left to the experience and expertise of the IC in question, except when very specific or rare hazards are involved (such as incidents involving chemicals or aircraft). However, one decision is effectively universal when dealing with building fires, regardless of specifics: the decision of whether or not to send fire-fighters into the building. Fire-fighters may be sent into a building (and the IC is then said to have adopted an *offensive tactical mode*) if and only if the IC considers that in doing so the chances of saving people (especially) or property outweighs the additional risk to fire-fighters. Otherwise a *defensive tactical mode*—the default—is adopted, whereby the fire-fighters stay outside the building until such time as either the fire is extinguished, all occupants accounted for and the incident closed or else the arrival of additional information causes the IC's interpretation of the conditions to alter so that an offensive mode is now considered appropriate.

Whether offensive or defensive tactics are adopted, this decision is subject to continuous review by the IC through a process known as *dynamic risk assessment*. This process, which of necessity is often done rapidly and with incomplete or uncertain information, represents an attempt to rationalize

the factors contributing to the tactical mode decision. This gives us most the appropriate target for the information provided by a FireGrid system: this information (and its modes of presentation) should be such as to contribute to the IC's dynamic risk assessment. And the need to rapidly assess this information led to the early suggestion by fire-fighters of a 'traffic light'-like display of the risks within the building, and idea that, as will be seen, would be adopted for FireGrid.

2.2 FireGrid and fire data

The acquisition and use of data describing the incident is integral to the FireGrid concept. We envisage a continual process of high-quantity data collection from sensors deployed around buildings. Any sensor that records data potentially useful for the response (either directly, or as input to a model) could be deployed; typical examples include smoke detectors, thermocouples (temperature sensors), and CO and CO_2 detectors. Usually these sensors will be polled in batch mode periodically by one or more 'data loggers', physical devices with which groups of sensors have some direct communications link, and which also convert the signals received from the sensors into their corresponding quantities (so, for instance, voltage levels reported by thermocouples are converted into the corresponding temperature readings). In modern systems, these steps are automatic, and at this point these data values can be stored in a centralized database (to which all models have access). Since sensors (or their lines of communication) can be noisy or can fail because of manufacturing flaws or the extremes of the fire incident itself, the data first needs to be verified and filtered to provide some measure of the accuracy and quality of data received from the building. To address this issue, we have developed a constraint-based filtering algorithm for validating sensor readings [26]. The data values must also be tagged with meta-data describing their origin and sampling time, before being transferred in real-time into some data storage facility. While a higher sampling rate produces more data and increases the potential for downstream data processing, it also means more data must be processed, transferred and stored in the same amount of time. Hence, some practical compromise must be found.

In addition to the dynamic real-time data that is being continuously fed from the sensor network, the database also holds data that is (reasonably) time-invariant such as the geometry/layout of the building, the types of sensors and their locations, the types and material of furniture and the location of fire suppression systems. This 'static' data is vital if the sensor readings are to be interpreted appropriately by the models. The content of this information is driven to a large extent by the needs of the various available models.

2.3 FireGrid and the fire modeller

Another key FireGrid concept is the use of computer models of physical phenomena to provide information to emergency responders. Given the initial focus of FireGrid on fires in the built environment, our interest lies in as far as possible adapting and using existing models that are designed to interpret and predict the behaviour of the fire, the movement of smoke, the reaction of the building and its occupants during the incident, and so on—anything that might be relevant to decision-making during an emergency response. Hence, the other major area of expertise influencing the command-and-control elements of the architecture surrounds the modelling of fire and its related phenomena. Experts in this field are usually academics and, as a consequence, their objectives are not necessarily the same as those of the FireGrid project.

These computational models, then, transform the collected data into descriptions of the current status and predictions of the development of the incident. This may be done by models employing interpretative and simulation methods of varying sophistication and complexity. A simple model might involve, for instance, little more than a simple calculation over the latest sensor readings to provide the current maximum temperature in a room, whereas richer approaches that exploit Computational Fluid Dynamics or Finite Element methods can model complex aspects of fire development and associated phenomena such as structural integrity, smoke movement and human egress.

The use of these models raises a number of issues [20]. Since these models have not been developed for emergency response purposes, their outputs will not necessarily contain that information most relevant to the IC and the risk assessment task. And where the results do provide this information it may not be expressed in the most appropriate manner, and may contain some degree of uncertainty, which again is potentially problematic for the IC who, it seems fair to say, would prefer to deal in certainties.

For the purposes of FireGrid ideally any model would operate independent of any particular context (such as a specific building) or incident, and would rely for its initial conditions on information acquired from the system database, with updates based on later sensor data whenever appropriate. In this context, the computational cost of running a simulation arises from the interplay of a number of factors: the complexity of the underlying model; its spatial and temporal scope; the amount of data to be processed; and the desired accuracy and precision of the results. At their most costly—say, models that attempt to extrapolate from the data and produce accurate, precise and far-reaching predictions of the course of the incident—these simulations are computationally intensive, and require a proportional amount of computational power if results are to be produced in a timely

fashion. Furthermore, strategies for effective and efficient data communication are required to support this processing. These considerations have led to the adoption of High Performance Computing (HPC), with computationally hungry simulation models deployed on specific HPC resources (and optimised to run most effectively on those resources), and accessed using a grid computing infrastructure.

2.4 FireGrid: system requirements

Notwithstanding the difficulties sketched above, there is a clear overlap between the interests of both modellers and fire-fighters at the level of the physical properties of fires and their environments: modellers have a scientific interest in these properties, and they are of concern to fire-fighters insofar as they determine the hazards that may be encountered during an incident. In practical terms, it becomes necessary to define more formally this intersection through the definition of a FireGrid system *ontology*. With this ontology established, it can then be used for the development of knowledge-based 'wrappers' for existing models so that the information they produce is more directly relevant to the response task. This ontology also allows users to interact with a FireGrid system in ways that will be described later in this paper.

In addition, situating the FireGrid system in (from the perspective of the user) an agent-based framework through which models can be invoked and their results collected would allow the system to be constructed and deployed in a modular and distributed fashion. This has benefits from a software engineering perspective—a system can be developed in the form of self-contained and more readily manageable components—and also from a deployment perspective, since different components could be integrated flexibly to form different systems to respond to incidents of different types, while the need to use HPC resources for deployment of models means that some form of distributed computational architecture becomes necessary.

Hence, we view a FireGrid system as a multi-agent system, and more specifically as some federation of interacting humans and computer agents; the actual constitution of the system will be dictated by the circumstances (and may change over the lifetime of a given system or incident). As such, agents are either autonomous software processes (an example might be some fire model, appropriately wrapped to allow it to interact with other agents) or some combination of human and intelligent interface (such as the principal IC interface, which will be discussed in more detail later). Agents interact in this system by passing messages between one another; hence, the only demands we place on agents in the system are that their messages conform to predefined format (of which more below), and that they all use the same message-passing medium (in the case of FireGrid, the I-X system [23, 24] has been used to implement the agents and provide this message-passing framework).

The AI in such a system resides in the knowledge-level structures and algorithms that allow communication of commands and advice between the agents, and that allow automated reasoning about the content of these communications. These knowledge-level structures and algorithms are the subject of the following section. (More details about the system architecture from a fire modeller's perspective can be found in [25], while Han et al. [14] provides a description from HPC/grid computation point of view.)

3 AI and FireGrid: technologies

In this section we describe the application of AI tools and technologies in the context of FireGrid. In particular, we discuss the ontology that was developed to underpin a FireGrid system and the uses of this ontology; the belief revision mechanism used to maintain a consistent view of the information provided by the models in a FireGrid system; and of the use of rules to interpret the set of beliefs in terms of the hazards it implies.

3.1 The FireGrid ontology

A FireGrid system is intended to allow its fire-fighter user to relate the output from simulation and interpretation models to the risks faced in the current incident. In order to do this, some common form must be identified within (or else imposed upon), on the one hand, the information emerging from the computer models, and on the other, the information that is understood by fire-fighters and can be related to the risks they face. In other words in AI terms it is necessary to establish an ontology for use within the system. An ontology sets out in explicit terms the key concepts in the domain, along with the relationships that hold among them, and in so doing it defines the terminology to be used when referring to these concepts. Based on discussions with both fire-fighters and modellers, we (the knowledge engineers on the project) were able to identify a number of common concepts understood by both modellers and fire-fighters. In its underlying approach this ontology draws on other ontological work, in particular the categories defined in DOLCE [13], but it was primarily envisaged as an engineering tool, able to meet the requirements of the FireGrid system. The ontology was developed, revised and extended as our grasp of the scope and aims of the project developed. In this process, the ontology had already begun to fulfil one of its principal (and often observed) roles, that of allowing the various human participants in the project to establish some basis for mutual understanding.

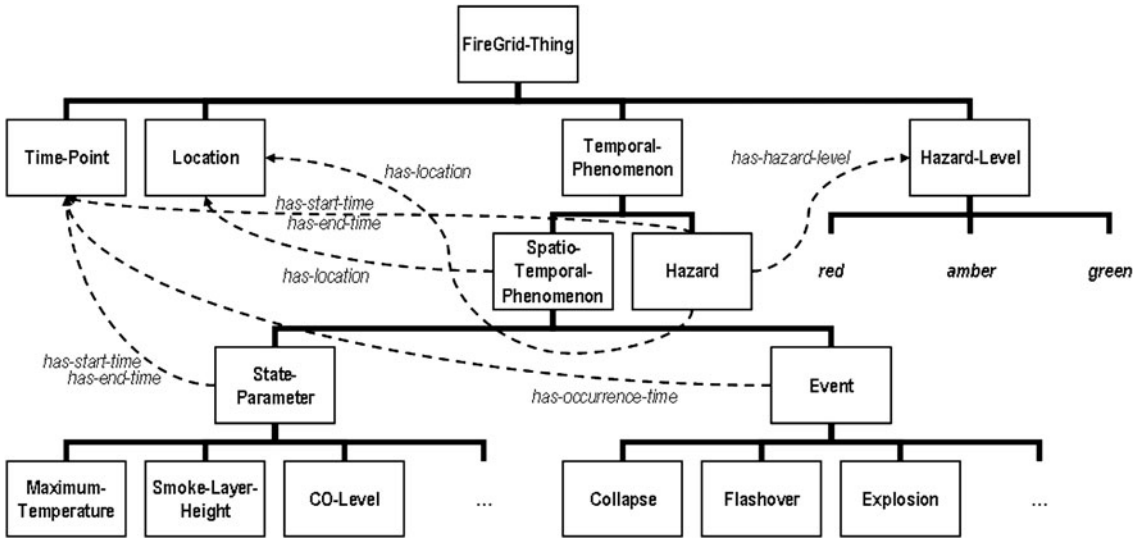


Fig. 1 A high-level view of part of the FireGrid ontology

Below we discuss some of the key ontological definitions that emerged, specifically in the area of understanding the relationship between the interpretations of the outputs of fire models that are possible and the implications these interpretations can have for fire-fighters. These considerations led to the introduction of and distinction between *State-Parameters* and *Events*; the notion of *Hazards*; and a number of ontological assumptions made about space and time. A high-level view of part of the FireGrid ontology is shown in Fig. 1.

In Fig. 1 the concepts (*Time-Point*, *Location*, etc.) shown in boxes are arranged in a sub-concept hierarchy below the most generic *FireGrid-Thing* concept. Dashed lines denote relationships among concepts. So, for instance, the concept of *Event* is a sub-concept of *Spatio-Temporal-Phenomenon*; it in turn has sub-concepts corresponding to different types of event; and it has specific relationships with *Time-Point* and, via its parent concept, with *Location*. Terms in the hierarchy shown without boxes (here, red, amber and green) correspond to instances of the parent concept (*Hazard-Level*).

3.1.1 State parameters and events

The FireGrid ontology makes a distinction between *State-Parameters* and *Events*. State parameters are quantities that are considered to be continuously measurable for some place over some duration of time. Illustrative sub-concepts include *Maximum-Temperature* and *Smoke-Layer-Height*. As a vital constituent of the definition of these parameters it is also necessary to establish (and enforce) a consistent unit of measurement for expressing values (which might also include some frame of reference)—for instance, a *Maximum-Temperature* value is expressed in terms of degrees Celsius, whereas a *Smoke-Layer-Height* value is expressed in terms of metres above the floor of some location. Although apparently quite straightforward, care needs to be taken when defining *State-Parameters*: we choose *Maximum-Temperature* rather than simply *Temperature* since, while at any given location a wide range of temperatures might be measureable (and not inconsistently so), this would be more difficult to reason about (we would have to maintain a richer representation of temperature) and moreover (and perhaps more pertinently) is the maximum value that effectively defines the danger to individuals in that space. (It can also be seen that the definition of these parameters and the concept of location are not wholly independent.)

Events, in contrast to *State-Parameters*, are considered to be instantaneous occurrences at some location; examples are *Collapse* and *Explosion*. Furthermore, it is asserted that a certain sub-concept of *Event* can only occur once (if at all) at a particular location during the incident—although in reality multiple occurrences of an *Event* sub-concept are certainly possible, this constraint was imposed to ease the conceptual difficulties of knowing which occurrence of a particular type of *Event* a model is referring to, when, for example, it provides a revised set of predictions. One further complexity that accompanies the concept of *Event* is that since an *Event* need never occur at any particular location (this in contrast to *State-Parameters*, which can be considered to have some value at any point in time and space, even if this is not known or is a zero or negligible value), there must be some mechanism for retracting a previous assertion of the occurrence of an *Event* if subsequent model results now indicate that it will not occur. In other words, the ability to talk about the occurrence of *Events* must bring with it the ability (and an accompanying obligation) to talk about the *non*-occurrence of *Events*.

It is intended that both *State-Parameters* and *Events* can be derived mechanistically (albeit perhaps only through additional knowledge-based interpretation) from the output of models; and while no model need refer to all possible *State-Parameters* and *Events*, if any given model does not produce information about at least one recognized *State-Parameter* or *Event* then to all intents and purposes it is useless and irrelevant to FireGrid.

3.1.2 Hazards

From the fire-fighter's perspective, the values of *State-Parameters* and the occurrence of *Events* can be related (again, through knowledge-based interpretation) to the concept of a *Hazard*: a *Hazard* is defined as something that can impinge upon the operational safety of fire-fighters at a particular place for some particular duration. It is not necessary to further elaborate *Hazard* with sub-types, since these will effectively be determined by the *State-Parameter* or *Event* which brings about the *Hazard* in the first place. For the purposes of relating *Hazards* to a simplified traffic light paradigm for information presentation, we can define the concept of *Hazard-Level* as being a relative measure of the severity of a *Hazard* that pertains at some time at some location; and, more specifically, we can identify three specific instances of *Hazard-Level* and define these in terms directly related to fire-fighting operations:

- A *green Hazard-Level* should be interpreted as "the system is unaware of any specific hazard to fire-fighters operating under normal safe systems of work at this location at this time";
- An *amber Hazard-Level* as "additional control measures may need to be deployed to manage hazards at this location at this time";
- A *red Hazard-Level* as "this location may be dangerous for fire-fighters at this time".

3.1.3 Space and time

As defined above, each *Hazard*, as well as each *State-Parameter* and *Event*, extends to a particular physical location, which raises the question of the definition and extent of *Location* within the system ontology. This is not as straightforward as may first appear; in models, differentiated spaces (usually) correspond to roughly homogenous volumes of gas bounded by physical partitions (and hence often correspond to rooms), but this may vary if the model is of either a large space or a high resolution. For fire-fighters, on the other hand, the notion of *Location* is situation-dependent and dynamic, depending on (among other things) the nature and scale of the building including its vital access and exit routes, the position of the fire incident and any occupants, the tactical operations that are currently underway and so on. Moreover the fire and its related phenomena can often alter the topology of a building. A further complexity lies in the relationship between *Location* and *State-Parameters*/*Events*. Consider the parameter *Maximum-Temperature*: if a particular *Location* is too large—say an entire floor of a multi-storey building—then the value of this parameter cannot be used to infer an accurate appraisal of the real risk to individuals within this space, and hence this information may have an undesirable effect on decisions regarding fire-fighting operations.

To reconcile these views we have adopted a pragmatic approach, defining contiguous, unchanging locations each of which corresponds to a room in the building in question, since this is one notion that seems to be mutually understood.

Similarly *Hazards*, *State-Parameters* and *Events* all occur in time, and the fire-fighters' decisions relate to both their understanding of what is currently happening and what is predicted to happen in the future. And, as for *Location*, the handling of time within a FireGrid system is not a simple matter. It is necessary that all information in the system is tagged with absolute timestamps, rather than referring to relative times (and it follows that the clocks of system components that generate or present information must be synchronized). We shall return to the representation of time in a FireGrid system in the context of the discussion of belief revision given below.

In summary, it is insufficient—and from the perspective of the system, meaningless—to talk of *State-Parameters, Events* or *Hazards* without also referring to some location and some time.

3.1.4 Applying the FireGrid ontology

As mentioned above, the ontology proved to be a useful tool for communication among the system-builders and with other interested human parties; it is also used to express and communicate messages within the agent-based system, and more specifically in this case, the information that is generated by the models. This usually requires the model to be wrapped by appropriate interpretation code, developed with the cooperation of the modeller, that is able to interpret the native output of the model (typically this output will consist of 'raw' numerical data representing the values of key parameters) in terms of *State-Parameters* or *Events*.

With the models appropriately wrapped, communication occurs in the context of a 'query-answering' approach that enables the system user—through an appropriate user interface agent—to construct and pose to a FireGrid system requests for specific information. This involves the introduction of different types of query, with each type intended to elicit particular instances of *State-Parameter* or *Event*. For example, a confirm-query is used to request the current (or

most recent) value of a designated *State-Parameter* at some location; an extension of this, a monitor-query indicates that the requester wishes to be kept informed of the value of the designated *State-Parameter* with values updated at some stipulated frequency; and a predict-when-query is used to request the forecast time of the occurrence (if any) of some specified type of *Event* at a location:

confirm-query state-parameter-type: maximum-temperature
location: room-1
predict-when-query event-type: collapse location: room-2

Hence, the query types effectively provide a set of agent message *performatives*, with each indicating both how the rest of the message contents should be interpreted and what constitutes an appropriate response(s) (which we denote with an inform performative).

All requests are handled by an intermediary agent in the form of an autonomous *query manager* agent. This agent effectively acts as a centralized repository of the details of all model agents in the system. Each model agent advertises its availability to the query manager; along with its name, the model agent provides some description of the information-providing capabilities of its model, and an indication of the parameters required for invocation of the model. We assume that the model agent wrapper encapsulates the actual mechanism for invoking the models; the diversity of models and of the platforms they run on—which in this case include HPC resources accessed over a grid, which requires authorization and appropriate interactions with middleware—mean that this must be engineered in collaboration with the model developer, and often with resource managers, since the objectives of producing real-time information can entail running models on platforms other than those on which they have been developed and run in the past.

The description of an agent's information-providing capabilities provides the basis for deciding which models to invoke: an (agent's) model is considered appropriate for answering a query if its information-providing capabilities 'satisfy' the query. These capabilities are also expressed using the terms of the ontology; consider the following rule that expresses the capabilities of a 'current maximum temperature' agent:

If *query is of type* confirm-query $\wedge$ *query is about state-parameter* Maximum-Temperature then
this model is capable of answering the query
Else
this model is not capable of answering the query

Given expressions of model capabilities in this form, the query manager should be able automatically to match queries to model agents. In the event of more than one agent being capable of satisfying the query, rather than making an arbitrary choice, the decision of which to use might call upon other information surrounding the agent and its model—one could envisage weighing model certainty, speed, quality, and even factors such as service costs. Alternatively, multiple agents might be invoked and the results given by each somehow combined or else used to confirm or corroborate those of others. In our experiments, however, the issue did not arise since the models used had no such overlapping capabilities. See [20] for more details of this query-answering approach.

Its model having been invoked and results generated, the agent then sends these as messages, relayed via the query manager, to the user interface agent where the query originated; here, the message contents are processed and reasoned about from the perspective of the end-user—that is, for the purposes of fire-fighting decision support. This brings us to the second broad use of an ontology: in addition to enabling effective communications, an ontology can help define what constitutes useful reasoning, and, where this can be automated, to define appropriate algorithms. Before discussing this reasoning, however, we shall consider in greater detail the content of inform messages that are generated by the model agents, since it is to this content that reasoning is applied.

The discussion of the ontology in previous sections is restricted to the conceptualization of physical phenomena surrounding fires; however the ontology also includes other elements, such as the types of FireGrid agents, definitions of valid queries and answers to pass back and forth between these agents and the content of these messages. In the form of an illustrative fragment of a context-free grammar, we present a description of the definition of the content of inform messages, with reference to the ontological terms introduced previously.

MessageContents = StateOrEventExpression
LocationExpression
StateOrEventExpression = StateExpression |
EventExpression
LocationExpression = "has-location" Location
StateExpression = (MaxTemperatureExpression |
SmokeLayerHeightExpression |...)
StateTimeExpression
MaxTemperatureExpression = "Maximum-Temperature"
Operator MaxTemperatureValue
...
Operator = "=" | ">" | "≥" | "<" | "≤"
MaxTemperatureValue = *defined numerical convention for temperature*
StateTimeExpression = "has-start-time" Time-Point [
"has-end-time" Time-Point]
...
EventExpression = ("flashover" | "collapse" | ...) (
EventTimeExpression | "will-not-occur")
EventTimeExpression = ("has-occurrence-time" Time-Point)
...

Location = “room 1” | “room 2” | ...
Time-Point = *defined numerical convention for time*

This (part of the) grammar is used to form constatives that refer to *State-Parameters* and *Events*, that is, its use determines the range of statements that model agents can make and, correspondingly, the range of statements about the world that interface agents can expect to receive. Note that here we make further pragmatic decisions about what can be said. For instance, while values of *State-Parameters* are defined to persist over some duration, in practice (especially when interpreting their current values) the extent of this duration will be unknown; hence, by convention we allow that an expression of such a value may have a start time only, with its end time understood as being (as yet) undetermined. We also introduce a means of stating that some *Event* will not occur at some location, along with application-specific terms for referring to these locations.

In practice, this ontological grammar is translated into the form of an XML Schema (with the necessary modifications entailed by the syntax and conventions of schemata), which provides a convenient way of sharing the grammar and compliance-checking of well-formed messages within code. In the context of the schema, locations, which are application specific, are defined through the use of a unique URI for each (which all agents in the system are then constrained to use when referring to that location); and times are expressed in “Unix time” as the number of milliseconds since midnight on 1st January 1970, a choice made for the sake of convenience, since this value can be expressed as a simple integer and is expressive enough for the precision required here.

```
<xs:complexType name=''messageContents''>
    <xs:sequence>
        <xs:element name=''state-or-event-expression''
            type=''tns:stateOrEventExpressionType''></xs:element>
        <xs:element name=''location'' type=''xs:anyURI''></xs:element>
        <xs:element name=''time-point'' type=''xs:long''></xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name=''stateOrEventExpressionType''>
    <xs:choice minOccurs=''1'' maxOccurs=''1''>
        <xs:element name=''state-expression'' type=''tns:stateExpressionType''>
        </xs:element>
        <xs:element name=''event-expression'' type=''tns:eventType''></xs:element>
    </xs:choice>
</xs:complexType>

<xs:complexType name=''stateExpressionType''>
    <xs:choice minOccurs=''1'' maxOccurs=''1''>
        <xs:element name=''max-temperature-expression''
            type=''tns:maxTemperatureExpressionType''>
        </xs:element>
        <xs:element name=''smoke-layer-height-expression''
            type=''tns:smokeLayerHeightExpressionType''>
        </xs:element>
        ...
    </xs:choice>
</xs:complexType>

<xs:complexType name=''maxTemperatureExpressionType''>
    <xs:sequence>
        <xs:element name=''operator'' type=''tns:operatorType''></xs:element>
        <xs:element name=''max-temperature-value''
type=''tns:maxTemperatureValueType''></xs:element>
```

```
    </xs:sequence>
</xs:complexType>

...

<xs:simpleType name=''operatorType''>
    <xs:restriction base=''xs:string''>
        <xs:enumeration value=''equal-to''></xs:enumeration>
        <xs:enumeration value=''greater-than''></xs:enumeration>
        <xs:enumeration value=''less-than''></xs:enumeration>
        <xs:enumeration value=''greater-than-or-equal-to''></xs:enumeration>
        <xs:enumeration value=''less-than-or-equal-to''></xs:enumeration>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name=''maxTemperatureValueType''>
    <xs:restriction base=''xs:double''>
        <xs:minInclusive value=''0''></xs:minInclusive>
    </xs:restriction>
</xs:simpleType>
...
<xs:simpleType name=''eventType''>
    <xs:restriction base=''xs:string''>
        <xs:enumeration value=''flashover''></xs:enumeration>
        <xs:enumeration value=''collapse''></xs:enumeration>
        ...
    </xs:restriction>
</xs:simpleType>
...
```

3.2 Automated reasoning for FireGrid

In order to provide useful decision-making support for firefighters, a FireGrid user-interface agent must interpret the (potentially very many) messages that it receives. This interpretation is achieved through cycles of *belief revision* and *hazard interpretation*, with the results presented to the user in a manner designed to allow a quick and accurate assessment of the incident to be made.

3.2.1 FireGrid belief revision

At any time, the information that is presented to the IC is based on the current set of beliefs about the environment that is maintained by the user-interface agent. For the purposes of this discussion, and regardless of any other beliefs that the agent may be said to hold, we define a *belief* to be some proposition about a *State-Parameter*, *Event* or *Hazard* that is held to be true for some location and at some time (in practice, to have a common representation we assert that all beliefs are held to be true over some duration of time, with (instantaneous) *Events* held to last over an interval defined by identical start and end time-points). In addition, every belief must have one or more justifications, indicating the rationale for believing it. The justification will usually be one or more messages, sent via the query manager, from the models in the system: the content of these messages provides the basis for beliefs (initially, the agent has an empty set of beliefs). It is also possible that the justification might consist of some subset of existing beliefs along with some rule that allows the inference of the belief in question from that subset. While these rules could be used to express physical or temporal relationships between beliefs (although in this case particular care must be taken to not introduce circular relationships, with beliefs directly or indirectly being used to justify the beliefs on which they themselves are based), here they are used exclusively to express the relationship between beliefs about *State-Parameters*/*Events* and beliefs about *Hazards* (of which more below).

As more information arrives from the models, a process of belief revision is required to maintain the consistency of the set of beliefs. Katsuno and Mendelzon [16, 17] distinguish *belief update* from *belief revision*: whereas the latter is the process of revising an agent's beliefs in the light of new

information about a fundamentally unchanging world, belief update applies in situations where the facts of the world are changing and the beliefs of the agent must be updated as a result. According to this definition, then, what is required for FireGrid is belief update; but for the purposes of this paper we shall use the term belief revision, since it is the more widely recognized throughout the field of AI and beyond. That said, the mechanism required for FireGrid differs in certain important aspects from conventional AI approaches to belief revision/update which tend to conform to the postulates set out in [1], the so-called *AGM postulates*. Specifically, whereas common approaches to belief revision operate at an abstract logical and application-independent level, and usually over simple atomic propositions, for FireGrid the revision must specifically take into account application-dependent ontological notions and the needs and capabilities of the end-user, and we have more complex propositions about the world which can be manipulated by the revision mechanism in more complex ways.

Hence, we have had to develop our own belief revision algorithm, which will be described below. While a number of implementations of belief revision systems are available in various formats [7, 18, 19], due to the age of the available source codes, the complexity of the propositions used here and the domain-specific nature of the algorithm, none of these was felt to provide a suitable basis for development, and so the algorithm was implemented from scratch.

Algorithm: Revise-Beliefs (m, S)
  Input: m : *message*, S : *set of beliefs*
  n ← *new belief created from (and justified by) the contents of* m
  ! *b.location returns the location the belief is about.*
  B ← {b | b ∈ S ∧(b.location = n.location)}
  now ← *the current time*
  If (now—n.end-time) > *tolerance* then
    ! *message is about the past, so ignore*
    Return
  Else If n *is about some event of type* e then
    ! *only one event of each type per location allowed:*
    If ∃b ∈ B *such that* b *is also about* e then
      Retract-Belief(b, B)
    If n *is not a notification of the non-occurrence of* e then
      ! *add new belief:*
      B ← B ∪{n}
  Else
    C ←{b | b ∈ B ∧ b *is about the same type of state-parameter as* n}
    For all b ∈ C then
      ! *note that b.start-time* and *b.end-time refer to the start and end times of the content of some*
      ! *belief* b *(ie the duration of a state-parameter or the occurrence of an event) and not of the*
      ! *belief itself!*
      ! *check whether n and b start at effectively the same time:*
      If (|n.start-time—b.start-time| ≤ *tolerance*) then
        If b.start-time < n.start-time then
          n.start-time ← b.start-time
        Retract-Belief(b, B)
      Else
        ! *check whether temporal extensions of b and n overlap:*
        If (b.start-time < n.start-time) ∧(b.end-time ≥ n.start-time) then
          ! overlap with b starting earlier than n: adjust b.end-time:
          b.end-time ← n.start-time—δ
        Else (b.start-time > n.start-time) ∧(b.start-time ≤ n.end-time) then
          ! *overlap with n starting earlier than b: adjust n.end-time:*
          n.end-time ← b.start-time—δ
    ! *add new belief:*
    B ← B ∪{n}

Algorithm: Retract-Belief(b, B)
  Input: b : *belief*, B : *set of beliefs*
  ! *first check whether b is used to justify any other beliefs:*
  C = {c | c ∈ B ∧ b *is used to justify* c}
  For all c ∈ C then
    If *without* b *there is no justification for* c then
      Retract-Belief(c, B)
  B ← B / {b}

When a new message arrives from a model, it will correspond to a proposition about either the value of a *State-Parameter* or the occurrence (or non-occurrence) of an *Event* at some time at some location (and it is tacitly accepted that this proposition is 'wholly believed' by the model at the time it was sent; in this treatment, we disregard the possibility that the messages can express 'degrees of belief' derived from probabilistic models. We also assume that all messages—and hence, all models—are believed equally by the agent). The contents of this message have to be considered in the context of the existing beliefs—in the terminology of belief revision, the current theory of the state of the incident must be revised in the light of the message contents in such a way as to incorporate the new information while maintaining the consistency of the theory as a whole.

For each of the locations we choose to maintain an independent set of beliefs. In other words, we have an independent, deductively closed theory for each location, and the overall task now comprises a number of independent sub-tasks, the aim of each of which is to maintain the consistency of one of these sub-theories. Each sub-theory consists of an (initially empty) set of formulae, statements about the environment in the corresponding location that are believed to hold over some specified time interval. Consistency within the sub-theory is determined by the ontological constraints that are placed on *State-Parameters* and *Events* as described above. For instance the appearance in the same sub-theory of two or more formulae that assert different values of *Maximum-Temperature* at the same time would represent an inconsistency, as would two or more appearances of formulae describing a *Collapse Event*.

The decision to divide the problem into the maintenance of independent sub-theories, one per location, requires some further comment. In reality, locations are unlikely to be independent in this way, and the conditions in one location will directly or indirectly affect conditions in another (indeed, this is how fire spreads); and from the beliefs held in one sub-theory one might be able to infer much about the conditions in the neighbouring locations. However, since these 'spatial' interactions would be reflected in the sensor data and in the interpretations of this data by the models, it was decided that explicit spatial reasoning about beliefs represented an unnecessary complication in this case. For example, it would not be necessary to infer from a rise in *Maximum-Temperature* in one location that there would be a commensurate increase in this parameter in a neighbouring location, since—assuming both locations to be fitted with appropriate sensors—this increase would be recorded by sensors and relayed to the system. On the other hand, as we shall see, temporal reasoning is a crucial to the belief revision process. In fact, the algorithm can be seen as an extension of belief revision with something akin to Allen's interval-based temporal logic [2, 3] and application-specific heuristics for reasoning about the temporal extension for which formulae hold. Interesting future extensions of the work reported here would be to attempt to enrich the model of change with spatial interpretations of the sensor data (dos Santos et al. [9]) or using multi-dimensional modal logics incorporating both temporal and spatial reasoning [4].

The workings of the algorithm are best illustrated through the use of an example. Consider a message from some model agent received by a user-interface agent containing the following statement:

maximum-temperature = 230°C has-start-time 12:54:32
  has-location room-1

If, when it received this message, the interface agent currently believed nothing about the *Maximum-Temperature* in *room-1* (one of the instances of *Location* in this case), the message alone would provide sufficient justification for the agent now believing the contents of the message. That is, if the corresponding sub-theory contains no formula referring to *Maximum-Temperature* a new formula can be created from the message contents and added to the sub-theory—a case of *theory expansion*: a new formula can be added to the set representing the sub-theory for *room-1* with no resulting inconsistency. Moreover, since nothing else is known about the values of this *State-Parameter* in this *Location*, the reasoning mechanism would assign a duration to this statement stretching from the indicated time to some indefinite point in the future. That is, since it is not believed otherwise, an assumption of the belief revision mechanism is that the values of *State-Parameters* persist, and hence in this case the *Maximum-Temperature* in *room-1* would now be believed to remain at 230°C indefinitely—or at least until such time as some other message causes this belief to be revised with a definitive end-point.

If, on the other hand, something is already believed either about the current or the future values of the *Maximum-*

Temperature at this location, then a more complex train of reasoning begins, which attempts to reconcile this message with the existing beliefs. This may involve adjusting durations of beliefs *(theory revision)* or, where there seems to be a direct contradiction—which would arise if some formula is already believed about the value of *Maximum-Temperature* at the same time indicated in the message—choosing to adopt one or other of the possibilities and disregarding the other (the simultaneous *expansion* and *contraction* of the sub-theory). While this might be done on the basis of, say, the relative trustworthiness of the originating models, in practice (and in this discussion) we tend to trust models equally, and rely instead on a single general principle encoded in the revision mechanism: the more recent the message the more likely it is that its contents are true (and hence should be believed). This principle is more subtle than might first appear. Firstly, since *interpretations* of the current state are—by definition—going to arrive later than *predictions* about the current state that were made at some time in the past, this principle effectively favours interpretations over predictions. This can justified by the general notion that, all other things being equal, any prediction based on some set of data will be less certain than an interpretation of the same set of data, and moreover, the interpretation will likely be based on more (and more recent) information than the earlier prediction. Secondly, the principle also effectively favours predictions made later over those made earlier; again, all other things being equal, the further into the future the time to which the prediction refers, the less certain the prediction. A later prediction will, on this basis, be more trustworthy than an earlier prediction about the same future time.

Contradictions, then, become apparent when trying to reconcile inconsistent state descriptions about the same location at the same time. Since we have effectively compartmentalized the incident into discrete locations, determining whether the message contents refer to the same location is straightforward: we maintain independent sub-theories for each location, and every well-formed message is constrained to refer to one specific location. However, determining if the contents refer to the 'same' time is more problematic; since absolute timestamps are used to represent time points, the contents of a message and an existing belief about a state parameter may have widely differing values at times that differ by perhaps only milliseconds. Of course, such a transition in values is possible (often coinciding with some event), but in practice is more likely to result from the divergence between new information, derived from later sensor readings or more complete simulations, and a belief based on obsolete information. Moreover, maintaining beliefs at too high a resolution of time can have undesirable repercussions for the user interface since the presented information can fluctuate too rapidly to be grasped and acted upon by the user: even for real-time applications there must be some temporal stability in the user interface in order for it to be useful (we shall return to this point later).

Accordingly, we choose to try to reduce the number of these transitions by defining that if the difference between the start time of a belief and the start time indicated in the contents of a message is within some tolerance (we shall have more to say about the value of this tolerance below) then they refer to the 'same' time. Furthermore, this tolerance helps to overcome the problem that interpretations of 'current' state based on sensor data will always refer to the past due to the inevitable lags and delays in the system; with this tolerance, these interpretations can be assumed to be about 'now'.

A further complexity arises when the content of a message is a prediction—that is, it purports to describe the value of a *State-Parameter* or the occurrence of some *Event* at some *Location* at some future time. While this might be adopted as a belief with a duration as before, the inexorable flow of time will mean that, assuming this belief has not been retracted or modified in the meantime, at some time the prediction will come to refer to the current time, and in the absence of other information a choice must be made about whether or not to accept the predicted value as an actual current value. Here we choose to accept the value as being current, justified on the basis of *some* information, the product of a more or less trusted process, being better than no information.

Note that the algorithm assumes that messages arrive singly and sequentially, and are handled in the order they arrive, and hence, that this order defines the relative recency of messages (although delays in the system might actually mean that 'more recent' messages take longer to arrive).

The algorithm presented above is primarily concerned with reasoning with information about things happening in time; and it should be obvious that time is intrinsic to the notion of change implied by the task of belief revision/update. While many previous approaches to belief revision have been content to treat time implicitly, there is no shortage of works that explicitly consider the temporal aspects of belief revision (see, for example, the approaches of Rao and Foo [21], Sripada [22] and Bonanno [6]; a typical strategy is the use of a modal logic to model temporal modalities; such an approach is not applicable here since we choose to resolve the set of beliefs into categorical proposition about current and future times). These approaches, however, are concerned with the temporal validity and extension of the beliefs themselves—that is, those times during some sequence of revisions for which a particular belief holds—rather than beliefs that are themselves about things that happen within time. While the former aspect is undoubtedly important in the FireGrid application domain (for instance, during post-incident debriefings, it is important to be able to relate decisions made and actions taken to what was believed at the

time), it is not the focus of this paper (as presented above the revision algorithm suggests that a belief is simply discarded when it is retracted; however, in the implemented system the temporal interval over which it was believed is closed, and the belief thereby retained but disregarded during subsequent reasoning). Notwithstanding this difference of emphasis, some of the approaches to temporal belief revision suggested in previous work can be seen to have similarities to the approach specified in the algorithm given above. This is perhaps unsurprising since each of the sub-theories could, in theory at least, be replaced by a sequence of belief sets, one for each interval defined in the set. While this might make some sense for 'discrete' time-intervals (such as days of the week) for which reasoning might be applied independently, here the definition of possible intervals is unlimited. As mentioned above, in its handling of time the belief revision mechanism perhaps has more in common with Allen's interval algebra [2, 3].

3.2.2 Hazard rule-based interpretation

Assuming that the set of beliefs has been revised and is consistent, the next step is to interpret these beliefs by applying a set of *hazard rules* to them. In effect this is simply a second step in the belief revision task, where, having established consistent sub-theories about *State-Parameters* and *Events*, we now must consider the effect of any revisions made on the set of beliefs about *Hazards*. We choose to separate these tasks since Hazards are dependent on *State-Parameters* and *Events* but not vice versa, and so achieving a stable set of beliefs about these allows us to determine a stable set of beliefs about *Hazards*. Moreover, here the reasoning is based not on ontological considerations but on user-defined rules. These rules represent expert knowledge about fire-fighting capabilities and practice; a simple example might be something like:

If maximum-temperature $\geq 100°$C *at some location* l
  from start-time s *until end-time* e then
  there exists a Hazard *with* Hazard-Level = amber
  at location l *from start-time* s *until end-time* e

where l, s and e are variables. A hazard rule consists of the conjunction of one or more conditions and a single conclusion, which corresponds to an interpretation of the conditions in terms of a *Hazard* (with associated *Hazard-Level* value) for the time and place in question. In addition, a hazard rule—especially one that refers to less commonly encountered hazards—may have an associated explanation and recommendations. So, for instance, a rule referring to excessive carbon monoxide levels may offer the explanation that levels in that range can "cause headache, fatigue and nausea" alongside the recommendation to "avoid prolonged exposure or consider the use of breathing apparatus". It is intended that the set of hazard rules is derived with the assistance of fire-fighting experts; and that, moreover, different rules might apply in different contexts (such as when there are hazards specific to a building, or to provide rules appropriate to particular users). As will be seen, this allows users to tailor the way that the interface agent interprets its set of beliefs about *State Parameters* and *Events* without recourse to modification of the underlying reasoning mechanisms. The algorithms for applying the hazard rules are outlined below.

Algorithm: Interpret-Beliefs(R, S)
  Input: R : *set of hazard rules*, S : *set of beliefs*
  For all l ∈ set of locations then
    ! *consider beliefs about each location separately*
    B ← {b | b ∈ S ∧ b *is about location* l}
    For all r ∈ R then
      C ← Satisfies-Conditions(r, B)
      If C ≠ ∅ then
      ! *note that C is the set of sets of beliefs each of which satisfies the rule conditions:*
        For all J ∈ C then
          h ← new hazard belief
          h.hazard-level ← hazard level indicated by conclusion of r
          h.location ← l
          h.start-time ← b.start-time s.t. b ∈ J ∧ b has the latest start-time in J
          h.end-time ← c.end-time s.t. c ∈ J ∧ c has the earliest end-time in J
          if (h.start-time = h.end-time) then
            ! *hazard inferred from rule testing for occurrence of some event; and thus is*
            ! *currently instantaneous; modify to extend duration indefinitely:*
            h.end-time ← ∞

```
! the combination of set of beliefs J and the rule r justify this hazard:
h.justification ← {<J,r>}
If ∃i ∈ B such that (h.hazard-level = i.hazard-level) then
  ! A hazard of the same level already occurs for this location;
  ! check whether its duration overlaps that of the new one.
  If (h.end-time < i.start-time) ∨(h.start-time > i.end-time) then
    ! no overlap—so simply add h as new hazard belief:
    B ← B ∪{h}
  Else
    ! there is some overlap—so merge hazards:
    If (h.start-time < i.start-time) then
      ! overlap with new hazard starting earlier than existing—extend hazard:
      i.start-time ← h.start-time
    If (h.end-time > i.end-time) then
      ! overlap with new hazard ending later than existing—extend hazard:
      i.end-time ← h.end-time
    ! . . . and incorporate justification for new hazard:
    i.justification ← i.justification ∪ h.justification
Else
  B ← B ∪{h}
```

Algorithm: Satisfies-Conditions(r, B)

```
! note: actually returns a set of sets - all possible justification sets
Input: r : hazard rule, B : set of beliefs
C ← ∅
! each set of beliefs. . .
J ← {j | j ∈ B ∧(∀k.(k∈J ∧ k ≠ j) → about-same-time(j,k)) ∧(∃c.c∈r.conditions ∧ satisfies(j,c))}
C ← C ∪ {J}
Return C
```

For each rule, then, a search is made in the set of beliefs for subsets of beliefs (of maximal cardinality) that together satisfy the rule conditions, which includes being all believed about the same *Location* and about the same time (that is, there must be at least one time-point which all these beliefs are about). If such a subset exists, then the conclusion of the rule can be drawn. Not apparent in this high-level description of the algorithm is the complexity of identifying these matching subsets of beliefs; we have implemented our own algorithm for this matching, but it might be the case that more efficient matching is possible (and may already have been proposed elsewhere, although if so we are not aware of this). Standard matching algorithms, such as the Rete algorithm [12], typically rely on simple pattern matching among asserted atomic propositions and so lack the richer notion of finding correspondences in time and space that is required here.

An inferred instance of a *Hazard* results in a new belief (or in the modification of an existing belief about a *Hazard* with an additional justification), with a duration delimited by the latest start time and earliest end time among the subset of beliefs satisfying the conditions. This in itself presents a problem in the case where a *Hazard* is inferred solely on the basis a belief about some *Event*: since an *Event* is considered to occur instantaneously—that is, at a single time-point—so too would any inferred *Hazard*. Since it is not clear what this might mean, nor how it might be interpreted by the user, we have excluded the idea of an instantaneous *Hazard*; instead we modify our definition to assert that any such inferred *Hazard* has a duration extending from that time-point to some indefinite point in time, in other words, effectively until the end of the time window for the application. This reasoning is justified by the fact that the types of *Event*, as defined in the ontology, refer to occurrences that essentially and irrevocably alter the conditions in the location (and hence would also render any subsequent information for that location produced by the FireGrid system questionable at best, since sensors might have been compromised or even destroyed, and assumptions built into models negated), and in such a way that, from that time forward, fire-fighting operations should proceed only with full awareness of this *Hazard*. Note too that the requirement that rule conditions are met by a subset of beliefs each of which refers to the same time effectively means that conditions that refer to the occurrences of two or more *Events* can only be satisfied by belief in their simultaneous occurrence (which is unlikely

but not impossible—and the likelihood is governed to some extent by the chosen granularity of the time-point representation). In practice, however, and again in large part due to the nature of *Events*, rule conditions refer to at most a single type of *Event*.

If a newly inferred *Hazard* is found to overlap in time with an existing *Hazard* of the same level in the same place, we essentially merge these into a single belief by asserting that there are now additional justifications for the existing *Hazard* and extending its temporal duration as appropriate.

Note that changes to the beliefs about *State Parameters* and *Events* due to belief revision can effectively mean that earlier inferences about *Hazards* no longer hold: this is a conventional *truth maintenance* problem. However, rather than implement a full-blown Truth Maintenance System (TMS, [8, 10]), we have implemented a partial solution: by maintaining justifications for all beliefs, we have a basic TMS that enables us to identify when there appears to be some reason to question those beliefs; but rather than reason about the implications of now-untenable beliefs, we have adopted the simpler, but maybe less efficient, expedient of re-computing the *Hazards* following changes to the belief set. Here, since the reasoning is relatively shallow, in terms of the 'chains' of inference that are constructed—typically messages are used to infer beliefs about the environment which in turn are used to infer beliefs about hazards—it was felt that the additional memory and computation overheads associated with full TMS meta-reasoning would not be rewarded with improved performance. While the use of more complex chains of reasoning might mean that some more sophisticated TMS is required, we have yet to analyse in full the complexity and requirements of this reasoning and so this remains an open question.

Another question concerns the dynamics of the hazard interpretation process, and of the underlying belief revision mechanism. In theory, there could be a constant stream of messages arriving at the interface agent and, as we've discussed, these could contain conflicting or contradictory information. While one might hope that the content of messages as a whole form a coherent picture of the incident, belief revision under such conditions could lead to an unstable set of beliefs, possibly with rapid fluctuations between extremes of *State-Parameter* values or even between one belief and its opposite (in the case of *Event* occurrence/non-occurrence). And this instability would likely be reflected in the interpreted *Hazards*, and any attempt to display these would result in a confusing, disorienting and practically useless interface for the user. The presented information *must* have some degree of stability, enough at least to allow the decision-maker to appraise the situation and—when appropriate—make a decision. But since the goal of FireGrid is to make available to responders real-time information, there is clearly a trade-off involved in producing some level of 'useful stability'. In some sense, the range and extent of information that is promised by the FireGrid architecture now becomes the problem: are users (and the automated reasoning processes that we devise to help them) capable of handling this information? Would we be simply replacing one problem of a lack of information with another of its surfeit, and leading to no discernible improvement in the response?

The answers to these questions remain to be seen. Here, this problem is ameliorated in a number of different ways. Firstly, the tolerance introduced above for determining whether two statements refer to the 'same' time effectively has a 'smoothing' effect by reducing—eliminating—temporally local fluctuations, and since this tends to favour interpretations of current state over predictions, and interpretations (especially those that are more immediately derived from sensor readings, such as temperature values) tend to be—but are not always—less contentious, the effect is one of making the current state appear relatively stable, even if later predictions continue to vary. A value of 30 seconds for this tolerance was found experimentally to provide a reasonably stable interface for the sort of applications we have in mind. Secondly, when querying the output of models with a monitor-query, we are careful to disallow frequencies that are too high (once every 30 seconds is the maximum permitted rate), and queries of other types are throttled too, to prevent the user from requesting more information than can be handled.

A final point here concerns when the cyclical application of the belief revision and hazard interpretation mechanisms. It would be possible to apply these two steps following the receipt of each message at the user-interface agent. However, since the number of such messages can be very large, rather than having the agent spend much of its processing time continuously performing one task then the other, instead we choose to periodically invoke the hazard inference task, assuming that changes have been made to the belief sets in the meantime. And once again this choice has the effect of providing a greater degree of stability to the presentation of hazards.

3.2.3 Hazard presentation

Once the set of hazards has been inferred, these must be presented to the emergency responder in a manner that will help the user make a swift and correct response decision. This requires an understanding of the particular capabilities, roles and tasks of the responder, the medium for presenting the information, and the operational context in which the information is delivered, as well as more general theories of situation awareness and human interface design. The content and complexity of the interpretation is also an issue here, since this can encompass both real and projected values and

their variations in space and time and can contain uncertainties and errors.

As described above, our initial focus in FireGrid concentrated on providing support for the Incident Commander and his/her decision of whether to adopt offensive or defensive tactics, a decision based on the process of dynamic risk assessment. This decision is made for specific places at specific times—hence the interface would need to represent both the spatial and temporal dimensions of the incident. With the assistance of experienced fire-fighters factors influencing the risk assessment can be built into the conditions of hazard rules, with the conclusions then indicating their relative severity in terms of the tactical decision. The next question, then, is how best to present these conclusions to the user.

The pressures of performing the risk assessment task on the incident ground are such that seemingly conflicting requirements emerged for information to be presented both in a manner that can be rapidly assimilated into this assessment process, and in a way that provides sufficiently detailed rationale to allow its careful consideration. As mentioned previously, from discussions with fire-fighters emerged the idea that these requirements might be reconciled at the interface level through a 'traffic light' display (already evident in the reasoning about *Hazard-Levels* described in the previous section) to give an at-a-glance overview of the current status at a particular location, with a point-and-click facility for delving into the reasons for the colour of light displayed.

Since the application of the hazard rules may have resulted in the inference of multiple simultaneous *Hazards*, to provide a summary of this information that is more readily assimilated by the responder a further reasoning step collates these into a single cumulative expression of the *Hazard* (and accompanying *Hazard-Level* value) for each location. This is a straightforward matter of determining the 'worst' *Hazard-Level* that is believed to occur at that location at the current time. So, for instance, if the believed state of *room-A* at the current time has allowed the inference of two current *amber Hazards* and one current *red Hazard* then the cumulative current *Hazard* in *room-A* is at a *red Hazard-Level*.

In interface terms, then, the cumulative current *Hazard*(-*Level*) at a particular location is used directly to colour the corresponding traffic light for that location in the graphical user interface. Feedback from potential users suggested that, to supplement this, some direct indication of future *Hazards* would also be useful, and so a second 'light' was added above the first to display the worst *Hazard-Level* predicted to occur in the future. Clicking within a location causes a pop-up window to appear in which are detailed the hazard rules which fired to produce the indicated *Hazard-Level*, along with any explanations and advice which accompany those rules. In addition a time-line indicates when any *Hazards* are predicted to occur within a time-frame projected into the future (pragmatically set to 15 minutes for our relatively short-lived experiments, but different (types of) incidents might demand different time-frames); moving a 'slider' allows the user to explore the nature and basis of these hazards at different points along the time-frame. Screenshots illustrating these features will be presented later in this paper.

4 FireGrid system application: a case study

For reasons that should be obvious, validating the FireGrid architecture, and more specifically, the various reasoning and representation schemes that have been developed, presents a number of practical difficulties. The project has adopted an incremental approach, gradually increasing the number and sophistication of implemented components, with experiments based on simulated, pre-recorded and eventually live data collected from fires. This has been supplemented with interface mock-ups used to elicit feedback from fire-fighters about the presentation of information (and the information itself). This process culminated in the deployment of a 'complete' FireGrid system to provide real-time information during an experiment involving a real fire (under controlled conditions) that was conducted before a select audience of interested parties at the test facilities at the Building Research Establishment (BRE), near London.

This experiment involved a fire initiated in a specially constructed experimental rig representing a small 4-room apartment. The notional scenario for the experiment concerned the possibility of occupants trapped in the apartment: the tactical decision was whether or not to send fire-fighters into the building to conduct a search (although no actual fire-fighting activities or any other intervention in the course of the fire was performed during the experiment). A member of the FireGrid team played the end-user role of support officer to the IC (a senior fire officer was among the audience).

A total of 125 sensors placed throughout the rig measured temperatures, heat flux, gas (O_2, CO, CO_2) concentrations and deformation of structural elements. Values from each of these sensors were polled in batch mode at roughly 3-second intervals, and fed to a database server housed off-site, from where models could request the data they required. This rig and its contents were intended to produce a 'flashed-over' fire in a relatively short time (in the event the whole experiment, from ignition to manual extinguishment lasted around one hour). An *Event* in terms of the FireGrid ontology, flashover typically occurs when the gases produced by a fire in some enclosed space reach temperatures high enough (above 500°C, as a rule of thumb) to ignite simultaneously all combustible matter in the vicinity. From the

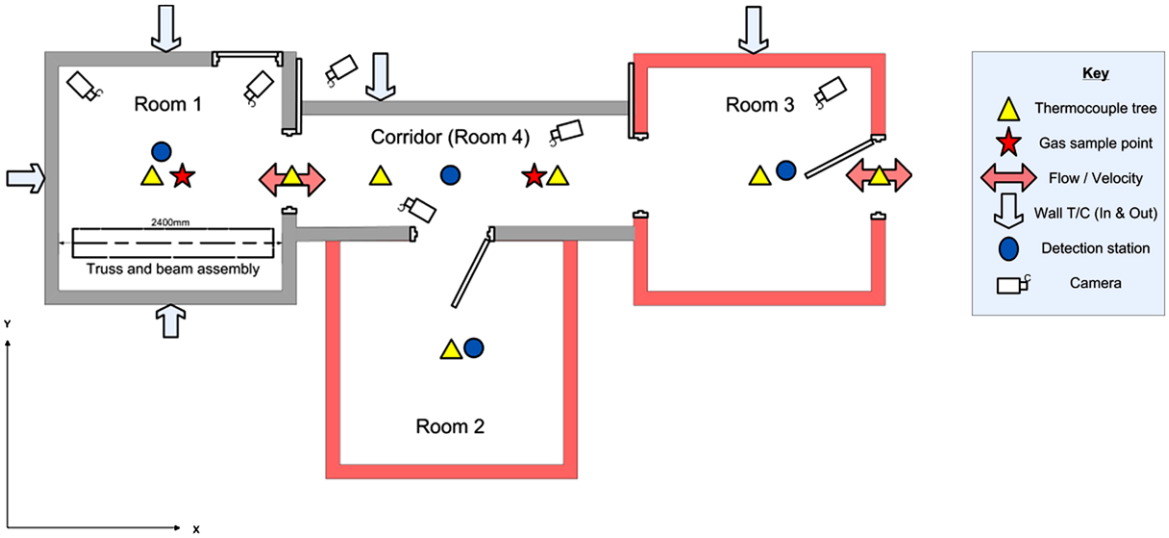


Fig. 2 The experimental rig (based on an original diagram courtesy of P. Clark of the Building Research Establishment)

perspective of responders, flashover represents a potential transition from a contained fire to an uncontrolled fire. In addition, certain structural elements of the rig were expected to deform and fail during the fire; the potential collapse of ceilings and floors is, of course, a major hazard for fire-fighters. A plan of the experiment apartment, showing the placement of sensors, is given in Fig. 2.

A number of different models were employed to interpret sensor data and make predictions based on this data; these were run on various local and remote resources as demanded by their processing requirements. Each was given an agent wrapper so as to be able to respond to any queries for information by translating its model's results into the appropriate messages.

Once the fire had been ignited and the alarm raised (by a state-of-the-art smoke detection device connected to the FireGrid system), the user constructed a number of queries, which were then relayed to the appropriate models, to monitor maximum temperatures, the level of smoke, build-up of gases, etc., along with requests for the predictions about the future values of these *State Parameters* and the occurrences of *Events*. Figure 3 to Fig. 9 show how the results of the reasoning in the previous section were presented to the user as the fire escalated and spread. Technically, the experiment was felt to be a success; the various components worked as envisaged, and the feedback from those present on the nature and quality of the information provided by the interface was unanimously positive. It is quite another thing, however, to demonstrate that the interface, the reasoning and wider system that underlies it, and indeed the entire FireGrid concept, can have a beneficial effect on responder decision-making in real emergency situations.

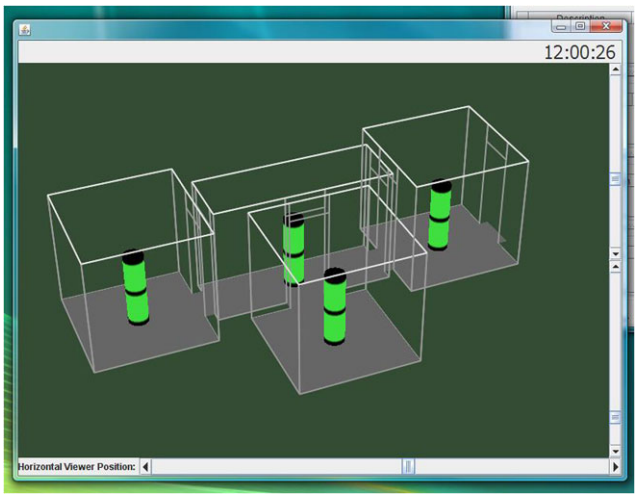


Fig. 3 The user interface at the outset of the experiment; a 3D representation of the 'apartment' used during the demonstration is shown, with the *green* traffic light in each location indicating that conditions are still amenable to fire-fighting

5 Conclusions

This paper has described the FireGrid project, an ambitious attempt to harness the potential of a number of complementary technologies to provide real-time information to emergency responders. Among these technologies, Artificial Intelligence concepts and approaches have effectively been used to integrate the system—and its developers—at the knowledge level. The reality of handling real, dynamic information has involved difficult and complex choices at every turn. While we have been able to draw upon a range of established AI techniques—including multi-agent systems, ontologies, belief revision and truth maintenance, and rule-based reasoning—for a number of reasons, the foremost

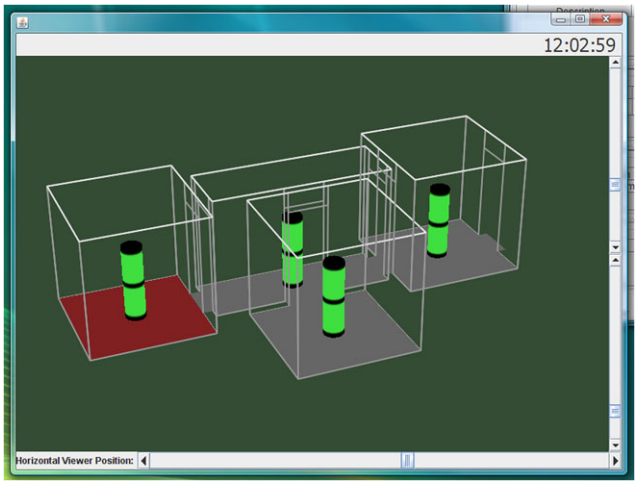


Fig. 4 A fire is detected in one of the rooms by one of the sensors in the system; the floor of the corresponding room in the display is highlighted in *red* to convey this information

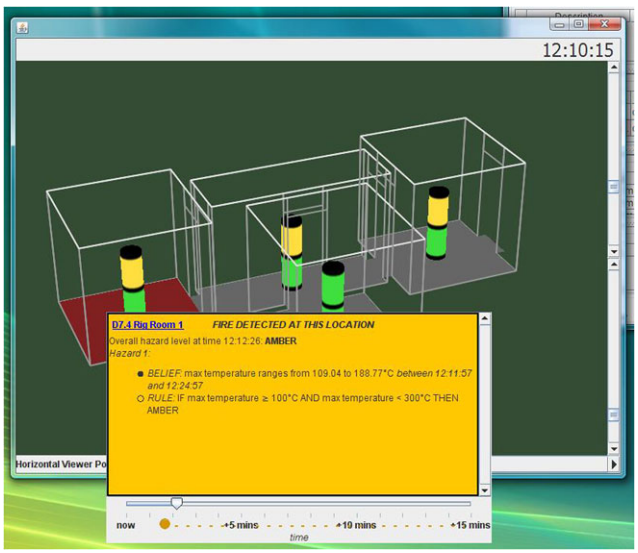


Fig. 5 The first set of predictions about future conditions in the apartment arrives from one of the models in the system. The pop-up window indicates that, based on these projected conditions, and specifically the suggested maximum temperatures, an amber hazard is predicted to begin in just under 2 minutes' time in room-1. Note that the traffic light for this room—and for two of the other rooms—is now showing the future cumulative worst hazard level (the upper light) to be amber. The current cumulative hazard level is *green* throughout the apartment (lower light)

being a discrepancy between AI theory and practice, and specifically when reasoning about time, existing techniques were found to be lacking, and so specific applications, interpretations and algorithms have had to be developed and implemented. While these solutions might well be of use to those working on problems with similar characteristics, it is hoped that the general discussion and issues raised by this paper will be of general interest.

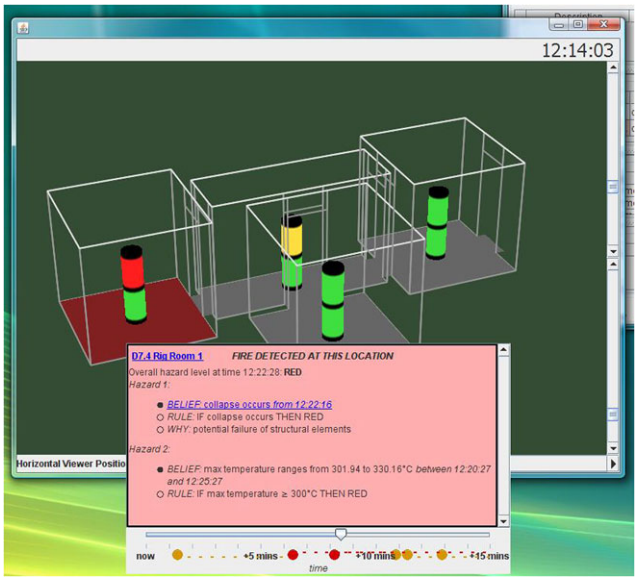


Fig. 6 More predictions arrive; the time-line at the bottom of the pop-up window now shows multiple hazards predicted at various times in the next 15 minutes, including some of hazard level *red*. Accordingly, the cumulative worst predicted hazard level shown on the traffic light is now *red*, and note that the latest predictions for conditions in the far room have caused the worst predicted level there to be 'improved' to *green*

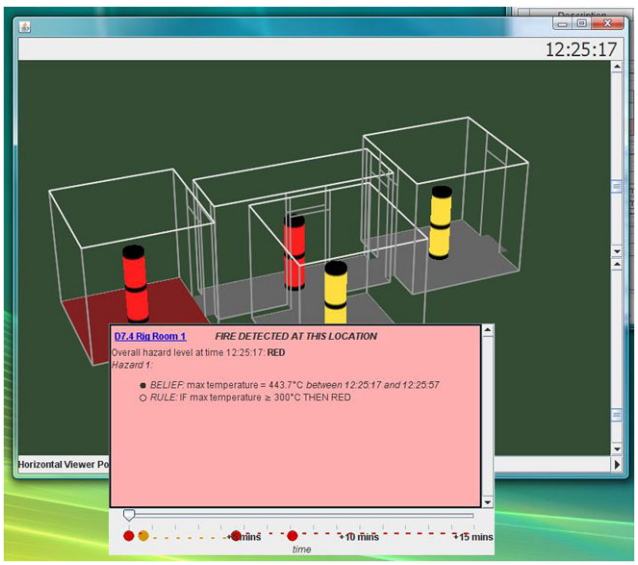


Fig. 7 The current maximum temperature in room 1 is such that the current cumulative worst hazard level is now *red*

At the time of writing, the FireGrid project has reached the end of its initial 3-year funding period. Notwithstanding the success of the experimentation reported above, the range and complexity of fire incidents, and the difficulties of interpreting and especially predicting fire conditions within buildings mean that it is unlikely that FireGrid systems will be deployed as real emergency response aids at any time

in the near future. Moreover, when one considers the possible implications of the decisions that fire-fighters make, the operational validation of the FireGrid approach, of its individual components and any systems comprised of these presents a problem (as is often the case for AI applications which, by their nature, tend to deal with heuristic and approximate methods, rather than certainties and guaranteed results). Here, of course, system validation is compounded by the fact that large-scale emergency incidents can, at best, only be simulated under laboratory conditions, and then only at considerable cost.

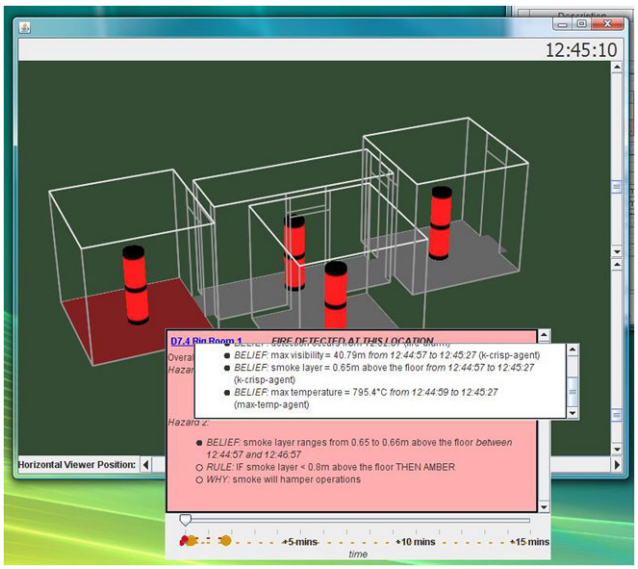


Fig. 8 The state of the apartment just after flashover, with high temperatures (and associated hazards) throughout

Acknowledgements The work reported in this paper has formed part of the FireGrid project. This project is co-funded by the UK Technology Strategy Board's Collaborative Research and Development programme, following an open competition. The University of Edinburgh and project funding partners are authorized to reproduce and distribute reprints and on-line copies for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of other parties. The University of Edinburgh is a charitable body, registered in Scotland, with registration number SC005336. The author would like to thank his colleagues on the FireGrid project who contributed in one way or another to the development of the techniques and results reported here, and who collectively made working on the project such a rewarding experience. He would also like to thank the anonymous reviewers whose thoughtful comments helped to improve the paper.

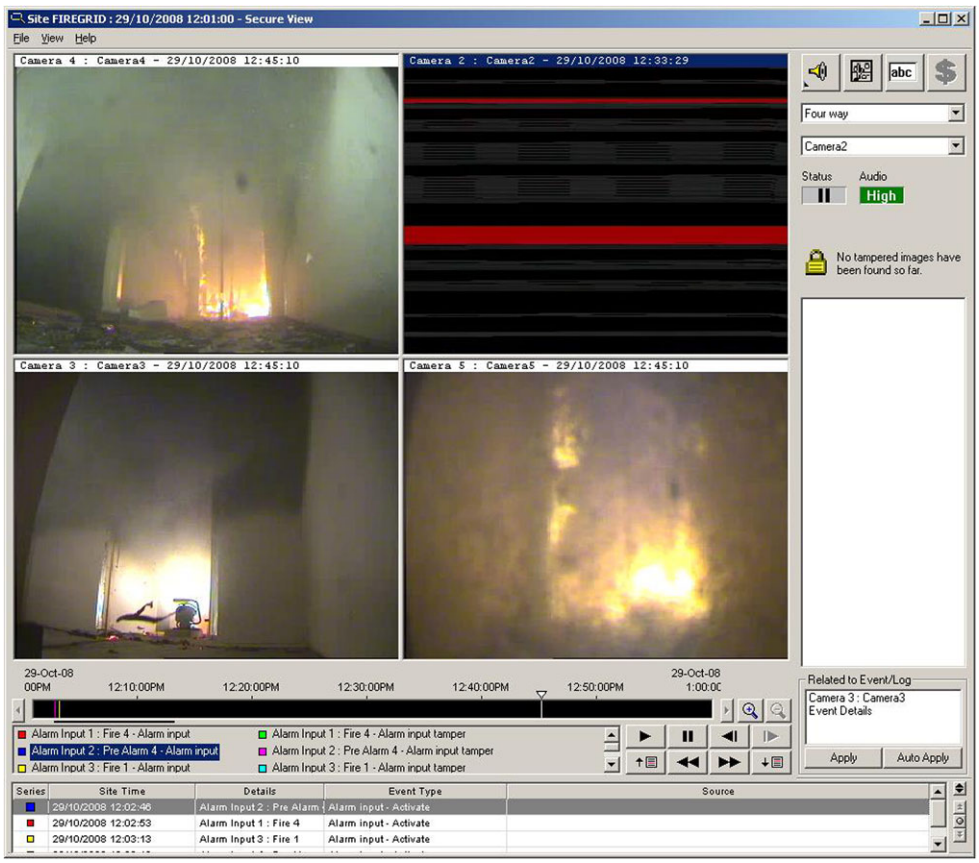
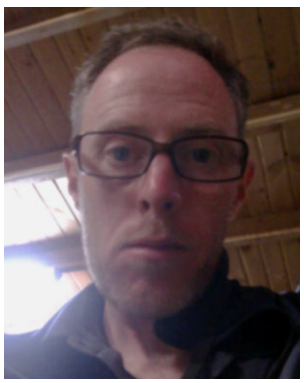


Fig. 9 Simultaneous camera footage from within the rig gives some idea of the actual conditions represented to the fire-fighter by Fig. 8. The fire has flashed-over, with flames in every room (and one camera has been destroyed)

References

1. Alchourròn CE, Gärdenfors P, Makinson D (1985) On the logic of theory change: partial meet contraction and revision functions. J Symb Log 50:510–530
2. Allen JF (1983) Maintaining knowledge about temporal intervals. Commun ACM 26(11):832–843
3. Allen JF (1984) A general model of action and time. Artif Intell 23:2
4. Bennett B, Cohn AG, Wolter F, Zakharyaschev M (2002) Multi-dimensional modal logic as a framework for spatio-temporal reasoning. Appl Intell 17(2):239–251
5. Berry D, Usmani A, Torero J, Tate A, McLaughlin S, Potter S, Trew A, Baxter R, Bull M, Atkinson M (2005) FireGrid: integrated emergency response and fire safety engineering for the future built environment. In: UK e-science programme all hands meeting (AHM-2005)
6. Bonanno G (2007) Axiomatic characterization of the AGM theory of belief revision in a temporal logic. Artif Intell 171(2–3):144–160
7. Chou TSC, Winslett M (1991) The implementation of a model-based belief revision system. ACM SIGART Bull 2(3):28–34. Special issue on implemented knowledge representation and reasoning systems
8. de Kleer J (1986) An assumption-based TMS. Artif Intell 28:127–162
9. dos Santos M, de Brito R, Park H-H, Santos P (2009) Logic-based interpretation of geometrically observable changes occurring in dynamic scenes. Appl Intell 31(2):161–179
10. Doyle J (1979) A truth maintenance system. Artif Intell 12:231–272
11. The FireGrid Consortium, http://www.firegrid.org/, last accessed: 19th September 2011
12. Forgy C (1982) Rete: a fast algorithm for the many pattern/many object pattern match problem. Artif Intell 19:17–37
13. Gangemi A, Guarino N, Masolo C, Oltramari A, Schneider L (2002) Sweetening ontologies with DOLCE. In: Gómez-Pérez A, Benjamins VR (eds) Knowledge engineering and knowledge management. Ontologies and the semantic web, 13th international conference, EKAW 2002. Springer, Berlin, pp 166–181
14. Han L, Potter S, Beckett G, Pringle G, Welch S, Koo S-H, Wickler G, Usmani A, Torero J, Tate A (2010) FireGrid: an e-infrastructure for next-generation emergency response support. J Parallel Distrib Comput 70:1128–1141
15. HM Fire Service Inspectorate (2002) In: Fire service manual, volume 2: fire service operations, incident command. HM fire services inspectorate publications. The Stationary Office, London
16. Katsuno H, Mendelzon AO (1991) On the difference between updating a knowledge base and revising it. In: Proc 2nd int conf on the principles of knowledge representation and reasoning (KR'91). Morgan Kaufmann, San Mateo, pp 387–394
17. Katsuno H, Mendelzon AO (1991) Propositional knowledge base revision and minimal change. Artif Intell 52:263–294
18. Le Berre D (2001) ADS: a unified computational framework for some consistency and abductive-based propsositional reasoning. In: Proc 2nd Australasian workshop on computational logic (AWCL01), Gold Coast, Australia, 31 Jan–1 Feb 2001
19. Liberatore P, Schaerf M (2000) BReLS: A system for the integration of knowledge bases. In: Proc 7th int conf on principles of knowledge representation and reasoning (KR 2000). Morgan Kaufmann, San Mateo, pp 145–152
20. Potter S, Wickler G (2008) Model-based query systems for emergency response. In: Fiedrich F, Van de Walle B (eds) Proc 5th int conf on information systems for crisis response and management (ISCRAM), Washington DC, USA, May 2008
21. Rao AS, Foo NY (1989) Minimal change and maximal coherence: a basis for belief revision and reasoning about actions. In: Proc 11th int joint conf on artificial intelligence (IJCAI-89), pp 966–971
22. Sripada SM (1993) A temporal approach to belief revision in knowledge bases. In: Proc 9th IEEE conf on artificial intelligence for applications (CAIA'93), Orlando, Florida, March 1993
23. Tate A (2000) Intelligible AI planning. In: Proceedings of ES2000, 20th BCS special group on expert systems international conference on knowledge based systems and applied artificial intelligence. Springer, Berlin, pp 3–16
24. Tate A (2003) <I-N-C-A>: an ontology for mixed-initiative synthesis tasks. In: Proceedings of the workshop on mixed-initiative intelligent systems (MIIS) at the international joint conference on artificial intelligence (IJCAI-03), Acapulco, Mexico, August 2003
25. Upadhyay R, Pringle G, Beckett G, Potter S, Han L, Welch S, Usmani A, Torero J (2008) An architecture for an integrated fire emergency response system for the built environment. In: Proc 9th IAFSS international symposium on fire safety science, Karlsruhe, Germany, September 2008
26. Wickler G, Potter S (2009) Information-gathering: from sensor data to decision support in three simple steps. Intell Decis Technol 3:3–17



Stephen Potter is currently a visiting researcher at the School of Informatics, where he previously worked in the Artificial Intelligence Applications Institute, specializing in emergency response applications. He holds degrees in computer science, artificial intelligence and mechanical engineering, and welcomes any serious offers of employment.