Final Technical Report

Using Shared Models of Activity for Coalition Task-Driven Cooperation

Austin Tate, Jeff Dalton, John Levine and Jussi Stader

Artificial Intelligence Applications Institute
The University of Edinburgh
Appleton Tower, Crichton Street, Edinburgh EH8 9LE, UK

Principal Investigator: Prof. Austin Tate. Tel: +44 131 650 2732

Contract No. F-30602-00-1-0024
Contract Value: $800,000
DARPA Order No. J662/02
Effective Date of Contract: 1 February 2000
Contract Expiration Date: 31 December 2002
Period of Work Covered: 1 February 2000 to 31 December 2002
Report Date: 6-Feb-2003

Intentionally Blank

Acknowledgements

The I-X project is sponsored by the Defense Advanced Research Projects Agency (DARPA) and US Air Force Research Laboratory Command and Control Directorate under grant number F30602-99-1-0024.

The broader I-X research program work has also been partially supported by the UK Defence Evaluation Research Agency (DERA) under the I-Con project and under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University.

Intentionally Blank

Contents

Abbreviations

The following abbreviations and acronyms are used within this report. They are collected together here to act as a reminder wherever the context is not clear.

AIAI	Artificial Intelligence Applications Institute
CoABS	Control of Agent-Based Systems DARPA Program
CoAX	Coalition Agents eXperiment
CISA	Centre for Intelligent Systems and their Applications
DAML	DARPA Agent Markup Language DARPA Program
<I-N-C-A>	Issues – Nodes – Constraints – Annotations Ontology
<I-N-CA>	Issues – Nodes – Critical/Auxiliary Constraints Ontology
<I-N-OVA>	Issues – Nodes – Orderings/Variables/Auxiliary Constraints
I-DE	I-X Domain Editor
IHMC	Institute for Human & Machine Cognition
$I\text{-}P^2$	I-X Process Panel
I-Plan	I-X Planning System
I-X	Intelligent Technology Research Program
O-Plan	Open Planning Architecture
UWF	University of West Florida

Table of Figures

Intentionally Blank

1. Summary

I-X is a research programme with a number of different aspects intended to create a well-founded approach to allow humans and computer systems to cooperate in the creation or modification of some product or products such as documents, plans, designs or physical entities – i.e., it supports *synthesis tasks*.

The I-X research draws on earlier work on Nonlin (Tate, 1977), O-Plan (Currrie and Tate, 1991; Tate et.al., 1998; Tate et.al., 2000a; 2000b). <I-N-OVA> (Tate, 1996a; 2000a) and the Enterprise Project (Fraser and Tate, 1995; Uschold, et.al., 1998) but seeks to make the framework generic and to clarify terminology, simplify the approach taken, and increase re-usability and applicability of the core ideas.

The I-X research programme includes the following threads or work areas:

1. **I-Core**, which is the core architecture and the underlying ontology of activity and processes termed <I-N-C-A>[1], and the terminology used to describe applications, systems or agents built in the I-X framework.

2. **I-P^2**, which are I-X Process Panels used to support user tasks and cooperation.

3. **I-Plan**, which is the I-X Planning System. This is also used within I-P^2 and other applications as it provides generic facilities for supporting planning, process refinement, dynamic response to changing needs, etc.

4. **I-DE**, which is the I-X Domain Editor, which is itself an I-X application but is also used to create and maintain the domain models, including especially the process models and activity specifications used throughout the system.

5. **I-Views**, which are viewers for processes and products, and which are employed in other applications of I-X. I-Views can be for a wide range of modalities and types of user.

6. **I-Faces**, which are underlying support utilities to allow for the creation of user interfaces, inter-agent communications and repository access.

7. **I-X Applications** of the above threads in a variety of areas depending on our current collaborations. These currently include:
   - Coalition Operations (CoAX)
   - Emergency and Unusual Procedure Assistance (I-Rescue)
   - Help Desk Support (I-Help)
   - Multi-Perspective Knowledge Modelling and Management (I-AKT)

[1] <I-N-C-A> is the preferred naming convention – standing for Issues – Nodes – Constraints – Annotations. Earlier, more limited usage, variants were <I-N-OVA> and <I-N-CA>.

- Medical Best Practice Procedures or Protocols
- Natural Language Presentations of Procedures and Plans
- Collaborative meeting and task support (I-Room)

8. **I-X Student Projects**, which are deepening and refining a number of aspects of the I-X research programme.

9. **I-X Technology Transfer**, including work on standards committees.

The DARPA-funded component of the I-X work has primarily focussed on the development of the I-X concepts and mixed-initiative human-centric multi-participant task support and collaboration interfaces in the context of a realistic and challenging multi-national coalition military scenario. This involves UN peacekeeping operations in a fictional country named Binni (Rathmell, 1999) set in the year 2012. This scenario was created by The Technical Cooperation Program (TTCP) to foster the development of militarily relevant Knowledge Systems for Command & Control Research. A highly dynamic scenario in which the tasks to be performed are changing and in which the coalition participants are rapidly evolving is used to show the value of the I-X approach. This is done through the creation of I-X Process Panels (I-P^2) and their use to link human and system agents. These systems were demonstrated within the Coalition Agents eXperiment (CoAX) – a series of demonstrations over a 3 year period from 2000-2002 and to be described later in this report and in some detail in attached papers.

2. History and Comparison of I-X to other Approaches

I-X is based on ideas that have emerged in research on O-Plan since 1983 (Tate et.al., 1998; Tate et.al., 2000a; 2000b and the Enterprise project in the mid 1990s (Fraser and Tate, 1995; Uschold, et.al., 1998). At the time of its design in 1983, O-Plan (the Open Planning Architecture) was intended to offer a more flexible and modular way to build planning systems over those being created in the mid 1970s (Nonlin, NOAH, Deviser). In particular it sought to make the plan in a partially developed state be an entity that could exist separately to the planner. Previous planners maintained with in their internal control data structures their lists of flaws, interactions or "criticisms" that were to be fixed. This was made explicit and declarative in O-Plan using an "agenda" associated with each plan being developed.

Over the years the modules and components in O-Plan have been clarified and made more generic, a process accelerated by trying to use the design in other applications:

- for a system, PLANIT (Planner's Interactive Toolkit), involving human planners and automated planning systems working in a human-driven fashion which spanned project planning, process design and job shop scheduling within the UK Alvey research programme (Drummond and Tate, 1992);

- for a job-shop scheduler, TOSCA (The Open Scheduling Architecture), in a system for practical uses in Hitachi (Beck, 1993);

- for a system for planning and dealing with execution failures by repairing plans, OPTIMUM-AIV, for the European Space Agency (Parrod and Valera, 1993; Aarup et.al., 1994; Tate; 1996b);

- for support to Air Campaign Planning (ACP) and Military Operations in Urban Terrain (MOUT) in work with DARPA, the USAF and the US Army (Tate et.al., 1998; Tate et.al. 2000);

- and for the Enterprise Architecture which took O-Plan ideas into business process management and workflow – including some manual process support at Unilever that has had enormous commercial benefits (Fraser and Tate, 1995; Uschold et.al. 1998).

This re-use of the concepts led us to feel the original O-Plan module or component definitions had their limitations, and their terminology was too orientated towards support to planning tasks.

At the time of its design, O-Plan had similarities to the Blackboard Architectures (see for example Hayes-Roth and Hayes-Roth, 1979). O-Plan was specifically designed to avoid aspects of that architecture in the indirect way in which it mapped Knowledge Source Activation Records (KSAR) to the Knowledge Sources (KS) available. O-Plan had an agenda whose entries mapped quite directly to its Knowledge Sources. I-X terminology now refers to these as the List of Issues and Issue Handlers (so not committing to an agenda style of implementation). O-Plan

also differentiated between general and maximally capable Knowledge Sources which could write any type of information into the Blackboard (where the plan or product was being created) and more limited special functionality embedded in "Constraint Managers" which could just check the information in the Blackboard/product and give yes/no/maybe answers on the validity of the Blackboard or product description. This information could then be used by other, later Knowledge Sources to carry on processing. Decision-making in the system was thus located in Knowledge Sources and not the special, limited Constraint Managers. As in Blackboard systems, the O-Plan controller dealt with ordering decisions on which agenda entry to handle next – localising such control information too. These features enabled O-Plan to scale better to larger realistic tasks.

O-Plan also attempted to create a software engineering structure and systems integration framework for building practical planning systems. The design proved to have much in common with the contemporary work in the early 1980s at NASA and the National Bureau of Standards (NBS) on layered reference architectures for space station telerobotics and for flexible manufacturing cells. See for example the NASREM work (Albus et.al. 1987) and its later military version generated by Hayes-Roth (DICAM). This work has continued over the years with many variants of systems that can take architecture "cells" and compose them in various ways, recursively, peer-to-peer and fractally. Recent work on this was done in a working group on architectures by people engaged on many of these earlier architectures (Hayes-Roth, Tate, etc.) in a group set up by DARPA to look at next generation Intelligent Battle Force Management which reported to DARPA in 1999.

O-Plan had a very optimistic and forward looking target for its delivery platforms when first designed in 1983. It was designed for significant distributed systems implementations utilising:

- geographically distributed systems (perhaps far distributed with some elements in deep space and others being ground based with 8 hours message transmission times);

- symmetric local processors on which to mount parallel versions of the principal components of an O-Plan agent or system – allowing for multiple platforms to be running concurrently for dealing with agenda entries on one or more concurrently developed shared product models;

- fine grain parallel systems for constraint management, graph algorithm processes and similar lower level functions. Implementations were even prototyped and fabricated in silicon to add in as specialised co-processors for constraint handling.

Much of the resulting structure and overhead for distributed implementations turned out to be far ahead of the time at which realistic implementation was possible, and this became a burden within the implementation. I-X allows for the same levels of implementation sophistication as did O-Plan, but does not clutter the elegance of the approach by embedding it into the interfaces and component boundaries designed. This encourages simpler implementations where appropriate.

I-X has taken the core ideas in O-Plan and simplified them while making them more generally applicable. It uses terminology more suited to a wider class of problems such as planning, design and configuration – i.e. synthesis tasks.

<I-N-C-A> (Issues – Nodes –Constraints - Annotations) is the basis of the ontology which underpins the I-X approach, and provides the framework for the representation used to describe processes and process products within I-X systems and agents. It is based on <I-N-OVA> (Tate, 1996a), a planning orientated specialization of the more general <I-N-C-A> ontology. <I-N-OVA> did not emerge until the mid 1990s, although the components of its design had been in O-Plan since its inception. A terminology change to call agenda entries "Issues" was suggested by a DARPA program manager (Craig Wier) in the early 1990s, and led to very profitable interaction with the engineering issue-based design community as a result[2]. A joint interest in emerging standards for process modelling in IDEF-0 (and later IDEF-3) between DARPA, AIAI and ISX Corporation also led to clarifications of how a plan was represented in O-Plan and how it could be a basis for rich and enrichable process and activity representation standards and to support process enactment and workflow. Discussions between the theoretical and practical AI planning communities (in particular Subbarao Kambhampati, David Joslin and Austin Tate) led to many clarifications and terminology changes (the notion of separating "critical" and "auxiliary" constraints depending on whether they were shared between reasoning components of the system or used within single modules arose at this time).

Involvement in emerging standards for planning in the military, process modelling, project management, workflow, etc., led to the description of <I-N-OVA> as a wholly constraint based ontology that could underpin a strong and simple representation for describing behaviour and activity of all types. Its power was validated by a study of 26 different representations of plans and processes (Schlenoff et. al., 1999) from a wide range of fields as part of the background studies for the creation of the NIST process Specification Language (PSL)[3]. <I-N-OVA> came out as having maximum coverage of the detailed requirements list for PSL of all the representations studied. Indeed those elements it did not cover were related to items intended to be handled by plug-ins to <I-N-OVA> and deliberately not committed to within <I-N-OVA> itself. A description of the relationships between these various standards efforts and the role <I-N-OVA> played in this is in Tate (1998).

I-X was conceived of in the late 1990s as a way to draw on the best of this earlier work, while making it much more generic and suitable to many kinds of product synthesis or modification tasks. <I-N-C-A> is an even more flexible and general-purpose constraint-based representation (compared to <I-N-OVA which is a specialisation) of any synthesised artifact.

[2] This cooperation continues to this day with projects such as CoAKTinG – http://www.aktors.org/coakting/
[3] See http://www.nist.gov/psl/

3. I-X Approach

The I-X approach involves the use of shared models for task-directed cooperation between human and computer agents who are jointly exploring (via some predefined or dynamically created processes) a range of alternative options for the synthesis of one or more artifacts such as a design or a plan (termed a product).

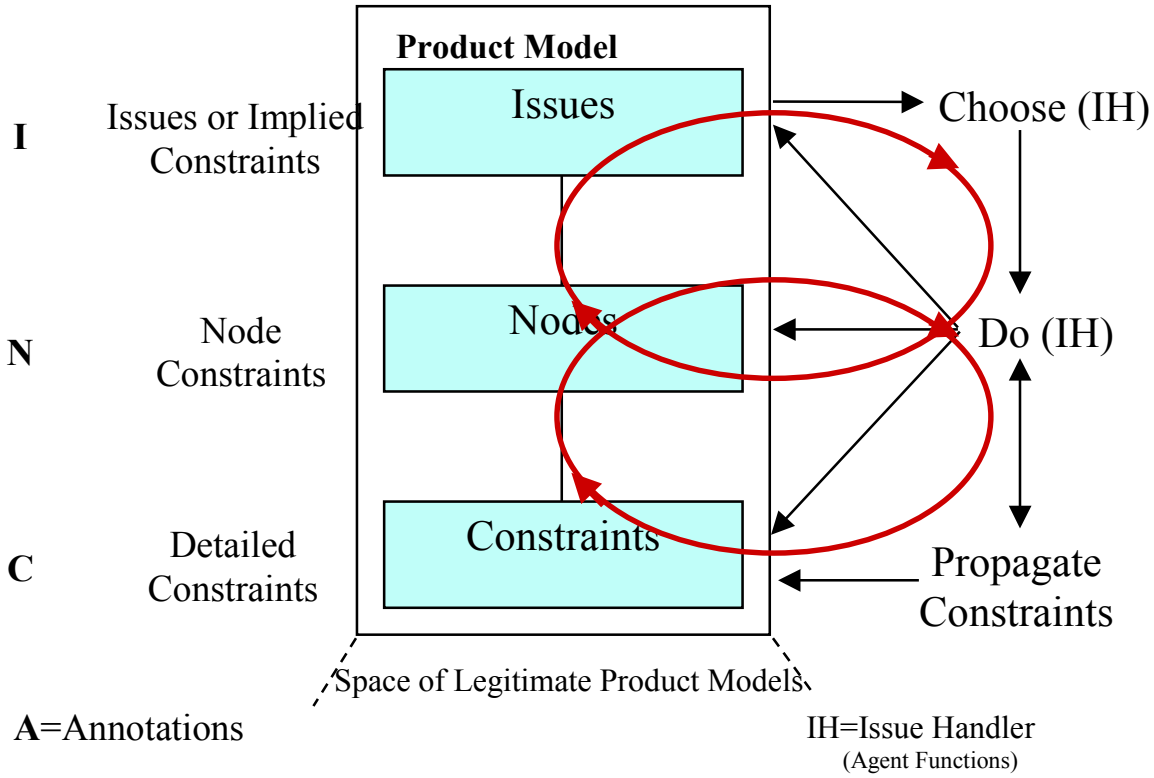


Figure 1: 2 Cycles of I-X Processing

An I-X system or agent has two cycles (as shown in the figure above which shows an abstract or algorithmic view):

- Handle Issues
- Respect Domain Constraints

An I-X system or agent carries out a (perhaps dynamically determined) process that leads to the production of (one or more alternative options for) a synthesised artifact.

An I-X system or agent views the synthesised artifact as being represented by a set of constraints on the space of all possible artifacts in the domain.

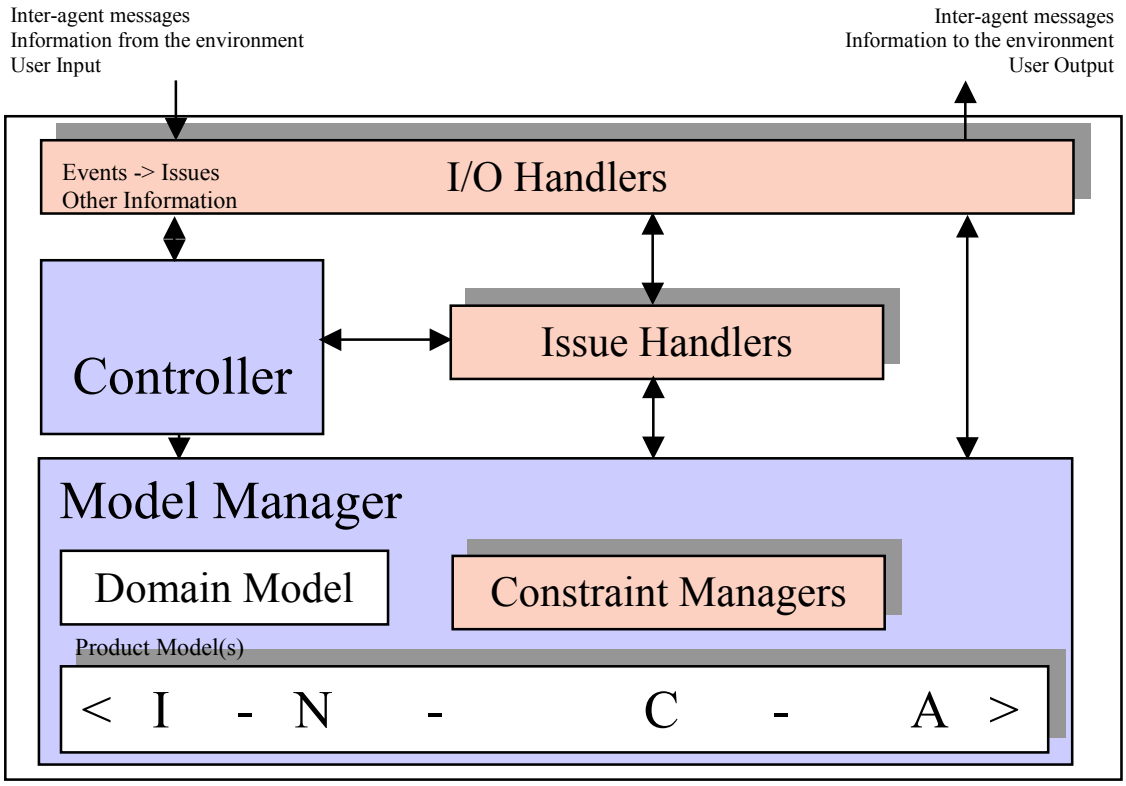


Figure 2: I-X System Architecture

I-X also involves a modular systems integration architecture (as shown in the figure above) that strongly parallels and supports the abstract view described. This is basically a Model – Viewer – Controller style of architecture. Plug-in components for Issue handlers, Constraint manager, I/O Handlers and Viewers allow for specific I-X systems to be created using this abstract architecture.

4. <I-N-C-A> Ontology

The I-X Intelligent Systems Technology approach uses the core notion of the representation of a process or product as a set of nodes making up the components of the process or product model, along with constraints on the relationship between those nodes, a set of outstanding issues and related annotations- <I-N-C-A> - Issues, Nodes, Constraints and Annotations[4].

The work involves the investigation of the use of shared models for task-directed communication between human and computer agents who are jointly exploring a range of alternative options for activity.

Six concepts are being used as the basis for exploring task-orientated multi-agent and mixed-initiative work involving users and systems. Together these provide for a shared model of what each agent can and is authorised to do and what those agents can act upon. The concepts are:

1. Shared Object/Product Model -- a structured representation of the object being modelled or produced using a common constraint model of the object or product (<I-N-C-A>).
2. Shared Planning and Activity Model -- a rich plan representation using a common constraint model of activity (<I-N-OVA>, itself a specialisation of <I-N-C-A>).
3. Shared Task Model -- Mixed initiative model of "mutually constraining the space of objects/products".
4. Shared Space of Options -- explicit option management.
5. Shared Model of Processing Capabilities -- handlers for issues (functional capabilities described in <I-N-OVA>) and constraint managers.
6. Shared Understanding of Authority -- management of the authority to do work (to handle issues) and which may take into account options and levels of abstraction of the model of the object or product.

In particular, this work carries forward the development of a strong systematic ontology to underpin the models of processes and activity - including continuing to engage in and promote its use as a basis for standards. The work draws on the initial efforts to create an ontology suitable for the conceptual description of all aspects of an organisation - the Enterprise Ontology and on the development of the <I-N-OVA> constraint model of activity.

[4] At various stages of the development of the I-X research the typography for rendering <I-N-C-A> has varied as the components have received clarification. <I-N-CA> originally stood for Issues, Node, Critical and Auxiliary Constraints. The aspect of separating critical (shared communications) constraints from auxiliary (separately managed) constraints is still important within the I-X architecture, but is now considered a part of managing the "C" (constraints) component. The annotations were always present in the ontology and can be attached to all components, but the top level annotations capturing the rationale behind the synthesised product or the process/plan being described has required more prominence as the work has continued and as mixed-initiative and human communications aspects have become more important. Hence, the rendering <I-N-C-A> with the extra hyphen now stands for Issues, Nodes, Constraints and Annotations.

The **issues** in the specifications state the outstanding items to be handled and can represent unsatisfied objectives, problems which analysis has shown need to be addressed, etc. The I constraints can be thought of as implying further constraints which may have to be added into the design in future in order to address the outstanding issues.

The **nodes** in the specifications describe components that are to be included in the design. Nodes can themselves be artifacts that can have their own structure with sub-nodes and other <I-N-C-A> described refinements associated with them.

The **constraints** restrict the relationships between the nodes to describe only those artifacts within the design space that meet the requirements. The constraints are split into "critical constraints" and "auxiliary constraints" depending on whether some constraint managers (solvers) can return them as "maybe" answers to indicate that the constraint being added to the model is okay so long as other critical constraints are imposed by other constraint managers. The maybe answer is returned as a disjunction of conjunctions of such critical or shared constraints.

The **annotations** add additional human-centric information or design and decision rationale to the information describing the artifact.

The choice of which constraints are considered **critical** and which are considered as **auxiliary** is itself a decision for an application of I-X and specific decisions on how to split the management of constraints within such and application. It is not pre-determined for all applications. A temporal activity-based planner would normally have objects/variable constraints (equality and inequality of objects) and some temporal constraints (maybe just the simple before{time-point1, time-point-2} constraint) as the critical constraints. But, in a 3D design or a configuration application object/variable and some other critical constraints (possibly spatial constraints) might be chosen. It depends on the nature of what is communicated between constraint managers in the application of the architecture.

5. I-X Process Panels and Related Tools

The aim of an I-X Process Panel (I-P^2) is to act as an intelligent workflow support, reporting and messaging "catch all" for its user. It can act in conjunction with other panels for other users if desired.

- Can take requests to:
  - Handle an issue
  - Perform an activity
  - Add a constraint
  - Support an annotation

- Deals with these via:
  - Manual (user) activity
  - Internal capabilities
  - External capabilities (invoke or query/answer)
  - Reroute or delegate to other panels or agents (pass)
  - Plan and execute a composite of these capabilities (expand)

- Receives reports and messages and, where possible, interprets them to:
  - Understand current status of issues, activities, constraints and annotations
  - Understand current world state, especially status of process products
  - Help control the situation
  - Improve annotations

- Copes with partial knowledge and can operate even where little or no pre-built knowledge of the domain is available.

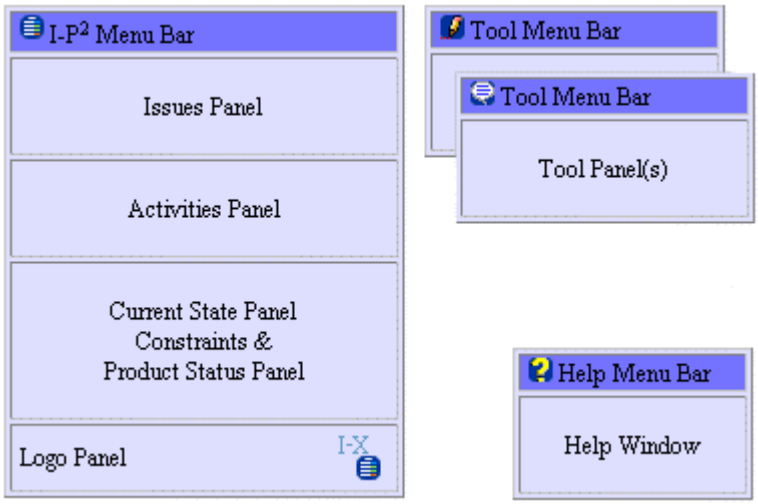


Figure 3: Anatomy of I-X Process Panels

An I-X Process Panel supports a user or collaborative users in selecting and carrying out "processes" and creating or modifying "process products". Both processes and process products are abstractly considered to be made up of a set of **"Issues"** which are associated with the processes or process products to represent unsatisfied requirements, problems raised as a result of analysis or critiquing, etc. They are mainly composed of **"Nodes"** (activities in a process, or parts of a process product) which may have parts called sub-nodes making up a hierarchical description of the process or product. The nodes are related by a set of detailed **"Constraints"** of various kinds. Finally there can be **"Annotations"** related to the processes or products which provide rationale, information and other useful descriptions. Processes and process products in I-X are represented in the <I-N-C-A> (Issues - Nodes – Constraints - Annotations) Constraints Model of Synthesised Artifacts.

Three example process panels are shown in the figure below. These panels are from a demonstration of agent systems within a military Coalition context – part of the Coalition Agents eXperiment – to be described in a later section.

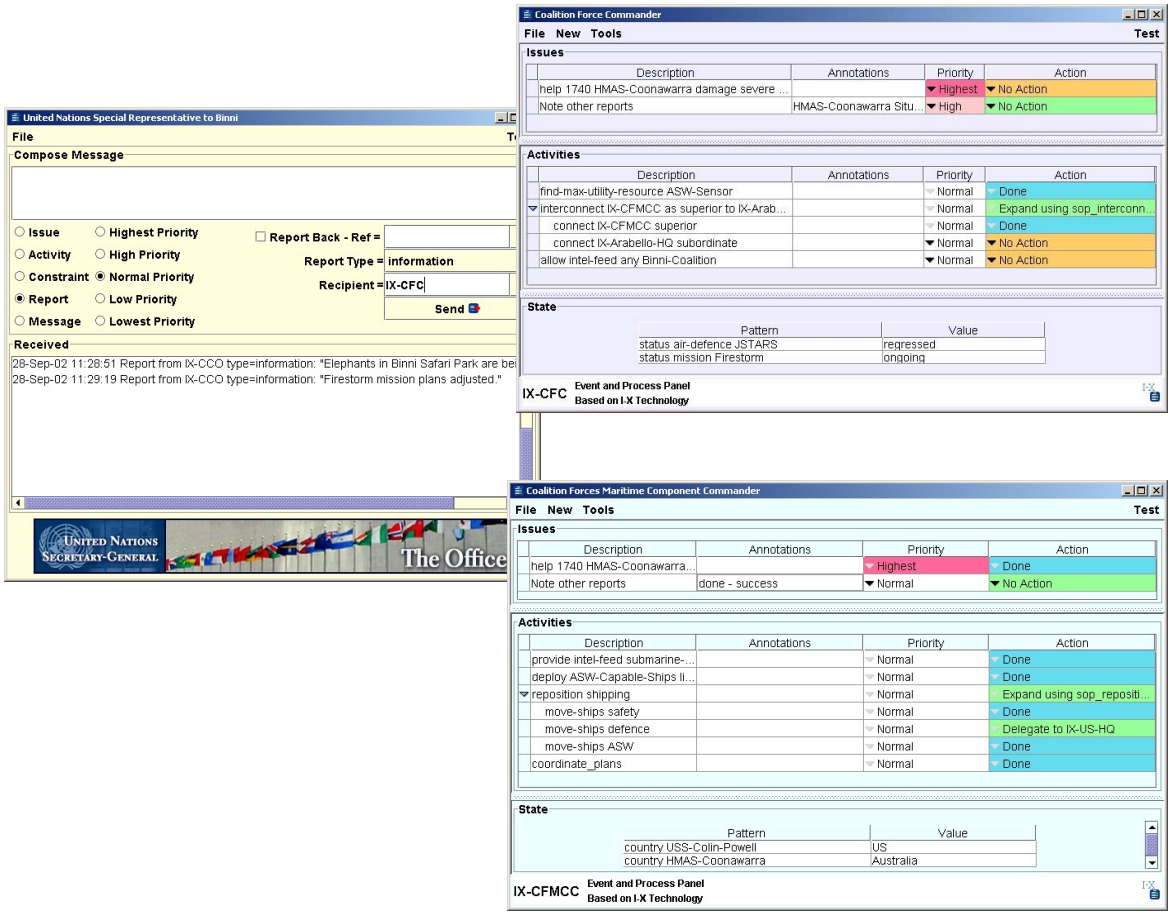


Figure 4: I-X Process Panels

An I-X Process Panel contains a number of sub-panels that describe:

- A set of "issues" to be "handled".
- A set of "activities" to be "performed".
- Current state information reflecting the current set of "constraints" to be "respected". This includes the status of "process products" being manipulated by the processes.
- Related "annotations".

The panel supports it's user in handling issues, deciding on a course of action and performing activities, and maintaining awareness of the current state, constraints, process products, etc. Entries on panels can be expanded using information provided in the domain model or process library used by a panel, or the entries can be passed between panels.

Various operations can be performed on I-P^2 panel entries. These include, where relevant, the ability to pop-up a window with more details and annotations for the entry (say an activity or an issue), or to expand or contract the display of some levels of hierarchically specified activities, to send information about the entry to the Messenger tool for sending on to others (perhaps in a modified form), etc.

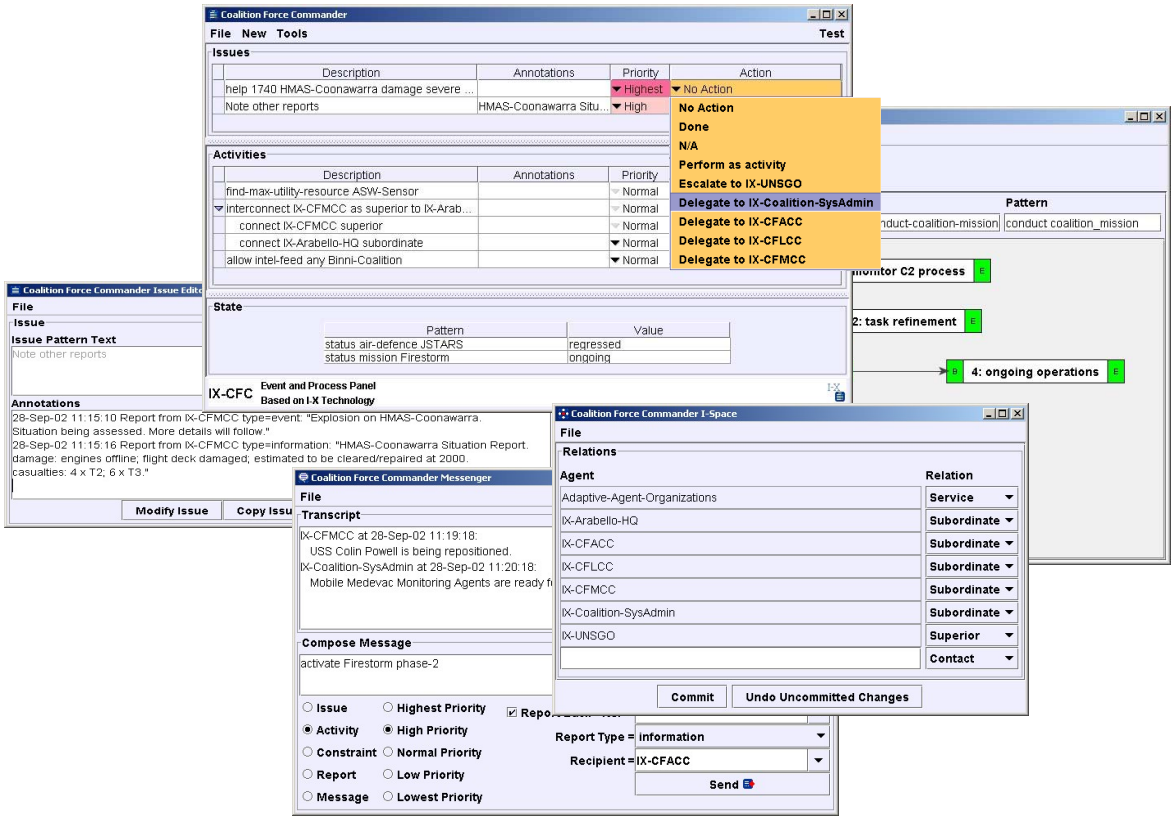


Figure 5: I-P^2 Main Window and Tools

A “tools” menu is available to make accessible the following:

- The I-DE domain or process library editor to view, edit or add to the list of process descriptions which may be used to “expand” entries on the process panel.
- A tool to view and change the relationships of the current panel to others (“I-Space”).
- An instant messaging or “chat” tool to communicate in free format or via the encouraged <I-N-C-A> structured forms with other I-X Process Panels and other systems (for “intelligent messaging” or “semantically augmented messaging”).
- An HTML web page viewer, also used to provide help to the user of a panel.

The process descriptions used by I-X Process Panels are kept in a domain library. This can be loaded when a panel is started, and can be added to dynamically by a user of a panel.

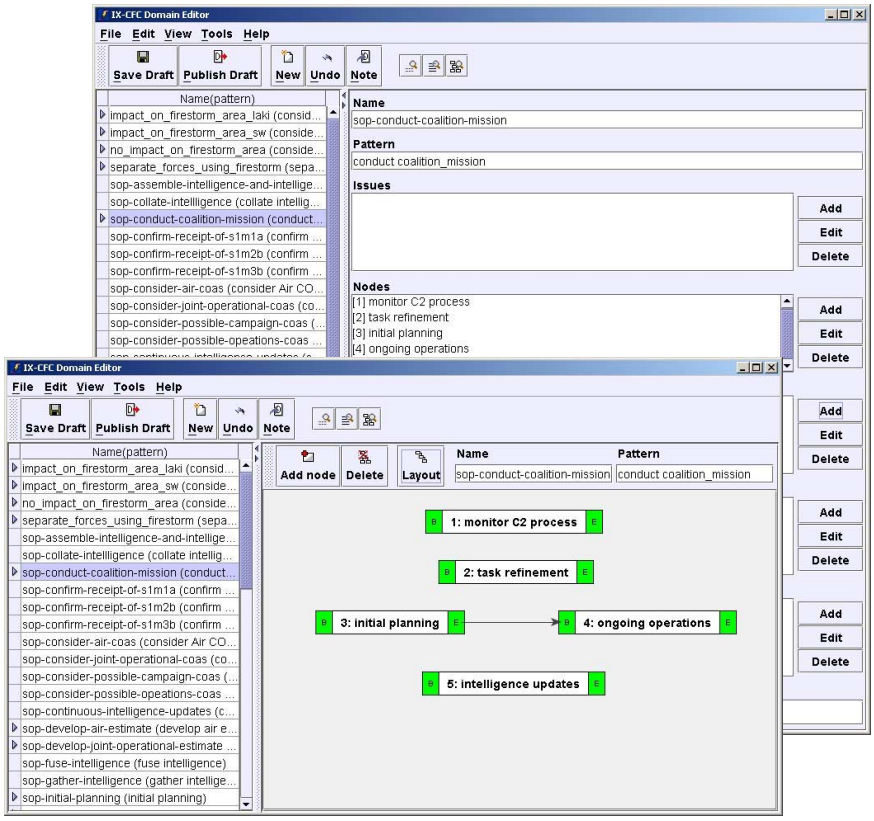


Figure 6: I-DE Domain Editor

The domain and process editor allows for multiple perspectives and views to be used to create rich process models. The domain editor provides both simple “minimal” views suitable for normal users of process panels and a “comprehensive” view that allows an advanced user to specify more complex temporal and world-state (condition/effect) constraints. Other constraints, like spatial ones or constraints on resources, can also be specified using the advanced view. A

graphical view provides an alternative view to the form-based views. The graphical view illustrates precedence relationships between the sub-steps of a process. This view can also be used to specify task breakdown structures via the expansion of nodes in the graph. In the advanced view, a tabbed option is available to allow access to related information about a domain model. The minimal, comprehensive and graphical views are all available via the Activities tab. Also available is a "Grammar" tab to view further details of the patterns used in describing issues, activities and constraints.

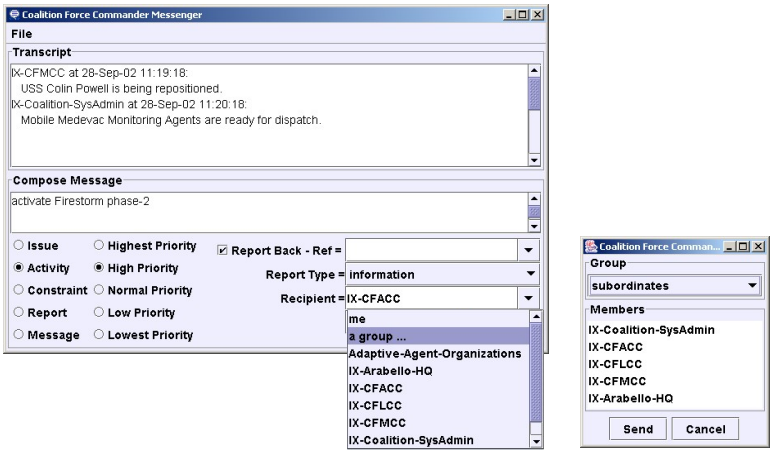


Figure 7: I-X Intelligent Messaging Tool

The I-X Messenger tool is used to compose and send structured task-related messages to other panels and agents. It also shows any "chat" or free-format messages received form other agents (in the Transcript window). You can send messages to your own panel ("me") and there is a simple group sending facility (which will be expanded in future releases).

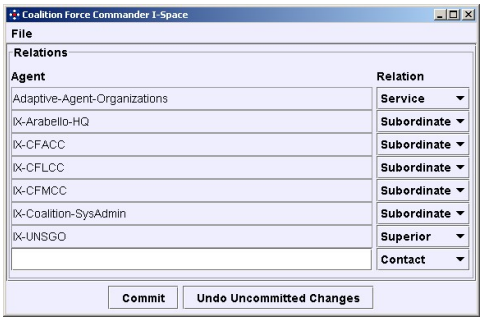


Figure 8: I-Space Panel/Agent Relationship Management Tool

The I-Space tool allows for the management of the organisational relationships of the current panel (referred to as "me") to other panels, agents and external services. New agent names can be added. Existing agents or panels can have their relationship altered.

6. CoAX and Binni

6.1 Binni Scenario

I-X has been tested and demonstrated alongside other agent and intelligent systems technology in collaborative work within the Coalition Agents eXperiment (CoAX)[5]. A suitably realistic scenario for CoAX and I-X experiments was required. We adopted and expanded the fictional "Binni" scenario (Rathmell, 1999) developed for The Technology Co-operation Programme (TTCP) which involves Australia, Canada, New Zealand, UK and the USA.

In this scenario the year is 2012 and global warming has altered the political balance of the world. The action is set in an area that is currently the Sudanese Plain to the West of the Red Sea (see below). Previously uninhabited land in the plain is now arable and the area has received large amounts of foreign investment. It is now called "The Golden Bowl of Africa".

Figure 9. Map of Binni Region of the Red Sea

A conflict has developed between two countries in the area. To the north is Gao, which has expansionist aspirations but which is only moderately developed, with old equipment and with a mostly agrarian society. To the south is Agadez, a relatively well developed and fundamentalist country. Gao has managed to annex an area of land, called it Binni, and has put in its own puppet

[5] See http://www.aiai.ed.ac.uk/project/coax/

government. This action has come under fierce attack from Agadez. Gao has played the 'threat of weapons of mass destruction' card and has enlisted support from the UN, which has deployed a force, the UN War Avoidance Force for Binni (UNWAFB), to stabilize the region.

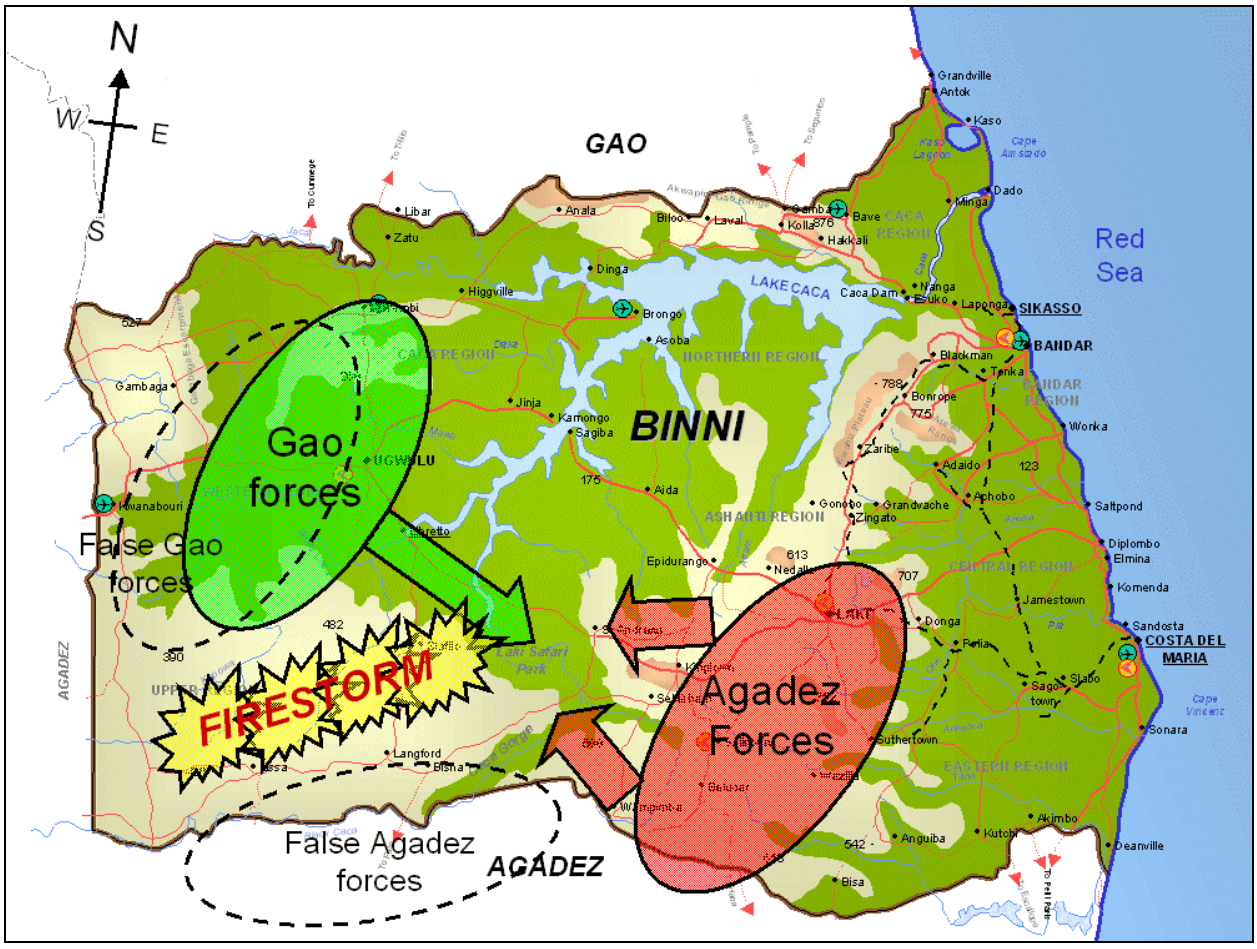


Figure 10. Map of Binni showing planned Firestorm Region and Deception

This basic scenario was adapted for a number of CoAX demonstrations, beginning with the initial planning phase, then moving onto shorter timescales and more dynamic, uncertain events for the execution phase.

6.2 CoAX Binni 2000 Scenario

The events of the CoAX Binni 2000 (shown in September 2000) demonstration focused on the initial planning phase.

After exploring a number of options to separate the opposing forces and restore the peace in the region, the deployment of a large ground observation and peace enforcement force and other courses of action have been rejected, and a "Firestorm" mission code-named "Operation Flash"

has been decided upon. This will clear land to enable simpler remote and ground observations with less risk to the Coalition peacekeepers. The events of the 2000 demonstration showed the Coalition doing initial information gathering and planning. The Coalition participants are the USA, UK, Australia and Gao. Gao has host nation status but its intentions are unclear and it is distrusted. Special steps are taken to monitor the information passed to and from Gao within the Coalition.

During the demonstration, misinformation feeds by Gao (intended to displace the firestorm to allow Gao to take an advantage and move forward) are detected and thwarted. Gao becomes belligerent and launches a denial of service attack against the Coalition's C3I infrastructure. This is automatically detected and thwarted using the advanced agent capabilities available to the Coalition.

6.3 CoAX Binni 2001 Scenario

The events of the CoAX Binni 2001 (shown in October 2001) demonstration move on from this initial planning and information gathering phase to a specific day and time in the execution phase, involving the monitoring, battle management and short-notice replanning associated with Coalition operations.

The firestorm mission has been planned and aircraft are being prepared for their missions. However the news media breaks a story that wildlife in an important safari park in Binni may be at danger as the park overlaps the firestorm area. With only an hour to go, the UN Secretary General's Special Representative to Binni asks the Coalition Force Commander to consider the wildlife risk aspects of the planned approach. Dynamic information gathering and information feeds using agent technology are employed to create a real time feed of the position of some at risk large mammals. After consideration it is decided to continue with the firestorm mission, but to replan as necessary to avoid risk to wildlife. Firestorm targets are adjusted in time or secondary targets selected as necessary for the first wave of firestorm bombing. The impacts of these changes on the coalition's medical and humanitarian operations are automatically detected, and unintended conflicts between disjoint coalition operations are avoided.

Agadez seeks to use this complication to seize the initiative and launches fighter attacks against Coalition airborne high value assets that are monitoring the operation. This is detected and important monitoring agents are moved to alternative computational platforms as the monitoring aircraft regress.

6.4 CoAX Binni 2002 Scenario

The CoAX Binni 2002 demonstration (shown in October 2002) follows the regression of the high-value airborne assets when Agadez attacks the coalition and includes a highly dynamic aspect of the situation in which a new country joins the Coalition and provides new capabilities.

Figure 11: Attack on Australian Ship

Agadez, seeing that its fighter attack cannot be successful, activates two submarines in the Red Sea that attack an Australian monitoring ship causing damage and casualties. Arabello, a country on the Eastern seaboard of the Red Sea, has sophisticated Anti-Submarine Warfare (ASW) capability due to its reliance on the free flow of shipping in the region. It has been unwilling to join in the Coalition operation in Binni to date due to regional concerns. Now seeing the escalating situation, wishing to support a trading partner and friendly nation under direct attack (Australia) and seeing the risk to shipping posed by the Agadez submarine activity, they offer their services to the Coalition.

This is quickly agreed, and in-place monitoring facilities from Arabello are linked rapidly and appropriately into the Coalitions C3I agent framework. Coalition ASW activity forces Agadez to back down. Seeing the resolve of the Coalition forces and the strengthening international support for its operations, Gao and Agadez agree to return to the peace talks conference at the UN.

The aim of the CoAX Binni 2002 demonstration as set by the DARPA CoABS Program Management was “To show, with conviction, that an agent-enabled infrastructure significantly aids the construction of a Coalition 'command support system' - and improves its effectiveness”. The objective was to address the unique aspects of achieving coherent Coalition operations from diverse 'come-as-you-are' elements. The operational and technical objectives of CoAX were to:

- show how flexible, timely interaction between different types of (sometimes incompatible) systems and information 'objects' is effectively mediated by agents - leading to agile C2 and improved interoperability;
- show how ease of composition, dynamic reconfiguration and proactive coordination of Coalition entities leads to adaptive responses to unexpected events at 'run-time' - providing robustness in the face of uncertainty;

- show how loosely-coupled agent architectures - where behaviors and information are 'exposed' to the community - are more efficient and effective than monolithic programs.
- show how agent policies and domain management help facilitate:
  - selective sharing of information between Coalition partners - leading to coherent operations;
  - control of appropriate agent behavior - leading to an assured and secure Cyberspace environment;

The CoAX objectives were met by developing and evaluating agent-based computing, including services for managing agent domains and tasks through:

- technical demonstrations of increasing complexity involving the integration of legacy, heterogeneous and dispersed information systems in an agent-enabled environment;
- the provision of Coalition-oriented generic grid services enabling 'come-as-you-are' elements to participate effectively at short notice;
- the integration and use of existing / legacy military applications from different countries;
- the use of a varied set of both domain-management-aware and 'non-aware' agents;
- publishing technical reports and research papers.

6.5 CoAX Binni 2002 Technologies

There were 30 participants in total in the CoAX demonstrations, which took place between February 2000 and October 2002. Austin Tate acted as coordinator for this series of demonstrators; with several individuals and organizations taking other management and systems integration lead roles in each demonstration. Further details of all participants and their contributions are available at the CoAX web site – http://www.aiai.ed.ac,uk/project/coax/

In the CoAX Binni 2002 demonstration, the following technical contributions were included:

AIAI, University of Edinburgh, UK http://www.aiai.ed.ac.uk/project/ix/
I-X - Intelligent Task Support and Messaging - Process and Event Panel Technology - Augmented messaging between agents concerning issues, activities, constraints and annotations / reports (<I-N-C-A>) and intelligent planning aids to assist with task support for individuals and teams.

AFRL http://www.rl.af.mil/
The Consolidated Air-Mobility Planning System (CAMPS) has featured in earlier demonstrations in the role of an agent-wrapped legacy system successfully interoperating with foreign Coalition systems.

BBN Technologies http://www.bbn.com/
Mixed Initiative Agents and Dynamic Information Flow technologies. Simworld event driven built on OpenMap and the CAMPS tool mentioned above.

CMU http://www.cs.cmu.edu/~softagents/
Anaconda Grid Agent Communications Visualization of inter-agent messaging for building trust in a software agent environment and DAML-S Matchmaker.

DSTO, Australia http://www.dsto.defence.gov.au/
ATTITUDE is an agent system that will contribute to improved situation awareness through visualization, information fusion, automation of routine aspects of Joint and Coalition operational planning, INT/OPS execution, operational logistics, and COA analysis – as part of DSTO's FOCAL (Future Operations Centre and Analysis Laboratory) program.

Dartmouth College http://agent.cs.dartmouth.edu/
Mobile Agents for Medical Monitoring allow medical personnel to monitor remote casualties *without* transmitting large volumes of data across the network. Only the monitoring code and highly filtered medical data ever cross the network. Medics can monitor large number of casualties (even Coalition partner casualties) without affecting other operations.

GITI / ISX http://coabs.globalinfotek.com/
CoABS Grid - a framework for federating heterogeneous agent systems. It enables rapid integration of heterogeneous applications and it is designed to meet the challenges of the military environment. Also, Verona - a powerful Knowledge Management tool that empowers users to easily capture, organize, manage and share knowledge-rich content.

Lockheed Martin ATL http://www.atl.external.lmco.com/
With the Interoperable Intelligent Agent Toolkit (I2AT), military IT staff can rapidly implement modified or entirely new agent behaviors. I2AT agents excel at integrating disparate stovepipe system and at assisting decision makers in the exploitation of their information and services.

University of Michigan http://ai.eecs.umich.edu/people/durfee/COABS/
Multilevel Coordination Agent (MCA) works top-down with plans developed independently by different coalition teams. MCA predicts unintended interactions, identifies candidate resolutions (such as merging steps that duplicate effort or inserting timing constraints to deconflict plans), and presents the commander with ranked candidates for selection.

MIT
http://ccs.mit.edu/roma/
The MIT Robust Open Multi-Agent Systems (ROMA) develops multi-agent systems for open contexts where the constituent agents must run in the dynamic and potentially failure-prone environments at hand.

OBJS http://www.objs.com/agility/tech-reports/
eGents are agents which communicate over email using open communication languages and exploiting the robust (and pervasive) email infrastructure.

QinetiQ, UK http://www.qinetiq.com/
Decision Desktop (DD) augments decision-makers' cognition by enabling them to acquire, visualize and manipulate diverse and dynamic information - however they wish and whenever they need it. DD uses software agents to acquire and translate data using Semantic Web technologies to reason effectively with the information - and behave more intelligently.

Stanford University http://www.stanford.edu/
Market Mechanisms Technology which investigates ways in which 'market forces' can be used to influence software-agent behavior.

University of Maryland http://www.cs.umd.edu/projects/impact/
IMPACT agents for reasoning with probabilistic temporal information - asset movement prediction agent.

University of Texas at Austin http://www.lips.utexas.edu/
Sensible Agents "Trustworthiness Evaluation" manages the inherent uncertainty associated with assessing the situational picture by asserting the perceived trustworthiness of information sources and their data. In a situation, "Adaptive Agent Organizations" works to form the best organization for each agent goal to efficiently respond to environmental dynamics.

University of Southern California / ISI http://www.isi.edu/ariadne/
Ariadne provides the ability to query on-line sources as if they were databases using machine learning technology to build "wrappers" for querying online sources and then ensuring that they continue to provide the expected data. In a Coalition environment, this capability is important for rapidly accessing data from both Coalition partners and open sources

UWF/IHMC and Boeing http://www.coginst.uwf.edu/kaos/
KAoS Technology provides agent domain management services and agent policy administration tools which enable warfighters to define suitable agent domains and ensure policy uniformity (and hence agent behaviors) across multiple platforms and hosts within a domain.

University of West Florida / IHMC http://www.coginst.uwf.edu/nomads/
NOMADS Technology provides two key enhancements: the ability to capture the execution state of agents and control the resources they consume and to provide support for strong or transparent mobility for agents. These control mechanisms can also be used to protect against denial of service attacks by malicious agents.

6.6 CoAX Binni 2002 Storyboard

Two mythical countries, Gao (in the north) and Agadez (in the south), are in conflict over an area of land that was annexed by Gao and set up as a new state called Binni. Gao has called on the UN to protect Binni and a Coalition Force has been deployed to stabilize the region. In the previous CoAX Binni 2000 and 2001 demonstrations we showed the information-gathering phase at the start of Campaign Planning and the execution of a so-called 'firestorm' mission.

UN forces have been deployed for some time and we will focus on the challenging Execution Phase of conflict, on 29th Sep 2012, for which the plans have been produced and the orders issued. The execution of the Firestorm mission has completed and, earlier in the day, media concerns about the location of wildlife in the Laki Safari Park (near the Firestorm) caused short-notice replanning to take place with only minutes to spare. New orders were disseminated, plan elements deconflicted and events tracked and handled. Also, the opponents, Agadez, flew hostile air-to-air missions against UN 'high-value assets' (the AWACS and JSTARS) and both humans and agents responded. It is now early evening and a number of military events are about to unfold in the *CoAX Binni 2002 Demonstration*:

- Part 1 shows how a submarine attack on an Australian ship is reported to the Coalition C4ISR through the software agent network distributed across the Coalition;
- Part 2 shows the collection and distribution of casualty information by agents, triggering automated tools for planning and coordinating Medevacs with ongoing logistics operations - enabling timely rescue and treatment;
- In Part 3, a new country (Arabello), joins the coalition "on-the-fly" (using agents and the CoABS Grid) and agents provide interoperability - enabling the Coalition to use an information feed from an underwater sensor grid;
- Part 4 shows how agents assist with the fusion, sharing and employment of the information from Arabello's underwater array information with that from other Coalition ASW forces and sensors;
- Part 5 shows how agents help the Coalition disseminate the Arabello information, along with existing Coalition information and tools, to enable countermeasures to be deployed - resulting in a successful end to the conflict.

The demonstration shows how an agent-enabled infrastructure significantly aids the construction and employment of a Coalition 'command support system'. It demonstrates:

- How flexible, timely interaction between different types of (sometimes incompatible) systems and information 'objects' is effectively mediated by agents - leading to interoperability;
- How policies and domain management help facilitate selective sharing of information between Coalition partners and control of appropriate agent behavior - leading to an assured and secure environment;

- How the ease of composition, dynamic reconfiguration and proactive coordination of Coalition entities lead to adaptive responses to changes and unexpected events;
- How the loosely-coupled agent architecture - where behaviors and information are 'exposed' to the community - can be more efficient and effective than the monolithic programs usually procured.

CoAX was a successful collaboration, which attracted the interest of a number of military organizations and programs to look at intelligent agent technology for future multi-national and joint forces operations.

7. Conclusions

An I-X Process Panel (I-P^2) supports a user or collaborative users in selecting and carrying out "processes" and creating or modifying "process products". Both processes and process products are abstractly considered to be made up of a set of **"Issues"** which are associated with the processes or process products to represent unsatisfied requirements, problems raised as a result of analysis or critiquing, etc. They also contain **"Nodes"** (activities in a process, or parts of a process product) which may have parts called sub-nodes making up a hierarchical description of the process or product. The nodes are related by a set of detailed **"Constraints"** of various kinds. Finally there can be **"Annotations"** related to the processes or products, which provide rationale, information and other useful descriptions. Thus I-X is based on the <I-N-C-A> Model of Synthesised Artifacts (Issues - Nodes - Constraints - Annotations), a simple abstraction which provides an extremely flexible and extendable representation of the processes and process products in I-X. <I-N-C-A> represents a product as a set of constraints on the space of all possible products within the model of the domain, which the I-X system has. This ontology relates well to emerging standards for process representation and interchange (e.g. in PIF, NIST PSL, DARPA SPAR) and has been used to provide inputs to these standardisation efforts.

I-X draws on the best aspects of work over a 20 year period on O-Plan which provided a flexible component architecture, I-X simplifies and makes more generic the component boundaries and naming conventions used to make the concepts more re-usable and applicable to a wider range of synthesis tasks.

I-X can be considered to make contributions at 3 levels. These are:

1. An outer level approach of handling issues and respecting constraints in the domain model gives an intelligible approach to what an I-X system or agent does.

2. A middle level provides a fractally composable cell, which employs a model/viewer/controller systems integration approach.

3. A detailed level provides an approach that represents and reasons with constraints on the space of all possible artifacts, and allows for the provision of specialised solvers for some or all of these detailed constraints.

I-X makes novel contributions at level 1 and 3, and is compatible with other approaches at level 2.

I-X clarifies the components used in such earlier systems and makes their construction simpler and more easily added to and maintained. But it goes further and is designed to integrate better with the user, and it incorporates the need for openness and intelligibility of the processes carried out and the state of the products being synthesised as a strong requirement. I-X is thus better suited to the design of systems where humans and machines work in a mixed-initiative or cooperative manner. Composability and the ability to link I-X agents or systems together permits

great flexibility – but of course this can only be achieved with intelligent use of the design concepts and well engineered implementations.

I-X and I-X Process Panels (I-P^2) have been demonstrated in a realistic multi-national coalition military scenario. They are being considered for use in a number of future joint and multi-national forces experiments and demonstrations.

References

Aarup, M., Arentoft, M.M., Parrod, Y., Stokes, I., Vadon, H. and Stader, J. (1994) Optimum-AIV: A Knowledge-Based Planning and Scheduling System for Spacecraft AIV, in Intelligent Scheduling (eds. Zweben, M. and Fox, M.S.), pp. 451-469, Morgan Kaufmann.

Albus, J.S., McCain, H.G., and Lumia, R. (1987) "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)", NBS Technical Note 1235, National Bureau of Standards, Gaithersburg, MD, USA.

Allsopp, D.N., Beautement, P., Bradshaw, J.M., Carson, J., Kirton, M., Suri, N. and Tate, A. (2001) "Software Agents as Facilitators of Coherent Coalition Operations", Sixth International Command and Control Research and Technology Symposium, US Naval Academy, Annapolis, Maryland, USA, 19-21 June 2001.

Allsopp, D.N., Beautement, P., Bradshaw, J.M., Durfee, E.H., Kirton, M., Knoblock, C.A., Suri, N. and Tate, A. and Thompson, C.W. (2002) "Coalition Agents Experiment: Multi-Agent Co-operation in an International Coalition Setting", Proceedings of the Second International Conference on Knowledge Systems for Coalition Operations (KSCO-2002), Toulouse, France.

Beck, H. (1993) TOSCA: A Novel Approach to the Management of Job-shop Scheduling Constraints, Realising CIM's Industrial Potential: Proceedings of the Ninth CIM-Europe Annual Conference, pages 138-149, (eds. Kooij, C., MacConaill, P.A., and Bastos, J.).

Currie, K.W. and Tate, A. (1991) O-Plan: the Open Planning Architecture, Artificial Intelligence 52(1), Autumn 1991, North-Holland.

Drummond, M.E., and Tate, A. (1992) PLANIT Interactive Planners' Assistant -- Rationale and Future Directions, reprints of working papers to the Alvey Programme PLANIT Community Club distributed in 1986-7. Available as AIAI-TR-108, AIAI, University of Edinburgh.

Fraser, J. and Tate, A. (1995) "The Enterprise Tool Set -- An Open Enterprise Architecture", Proceedings of the Workshop on Intelligent Manufacturing Systems, International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada, August 1995.
Hayes-Roth, B. and Hayes-Roth, F. (1979) "A Cognitive Model of Planning", Cognitive Science, 1979, pp. 275-310.

Khambhampati, S. and Srivastava, B. (1996) Unifying Classical Planning Approaches, Arizona State University ASU CSE TR 96-006, July 1996.

Parrod, Y., Valera, S. (1993) Optimum-AIV, A Planning Tool for Spacecraft AIV, in Preparing for the Future, Vol. 3, No. 3, pp. 7-9, European Space Agency.

Rathmell, R.A. (1999) “A Coalition Force Scenario ‘Binni — Gateway to the Golden Bowl of Africa’”, Proceedings of the International Workshop on Knowledge-Based Planning for Coalition Forces, (ed. Tate, A.) pp. 115-125, Edinburgh, Scotland, 10th-11th May 1999.

Sacerdoti, E. (1977) A structure for plans and behaviours. Artificial Intelligence series, publ. North Holland.

Schlenoff, C., Gruninger, M., Tissot, F., Valois, L., and Lubell, J. (1999) “The Process Specification Language (PSL): Overview and Version 1.0 Specification", NIST Internal Report (NISTIR) 6459, National Institute of Standards and Technology, Gaithersburg, MD, USA

Smith, S. (1994) OPIS: A Methodology and Architecture for Reactive Scheduling, in Intelligent Scheduling, (eds, Zweben, M. and Fox, M.S.), Morgan Kaufmann, Palo Alto, CA., USA,

Tate, A. (1977) Generating Project Networks, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-77), pp. 888-893, Cambridge, MA, USA, Morgan Kaufmann.

Tate, A. (1996a) "The <I-N-OVA> Constraint Model of Plans", Proceedings of the Third International Conference on Artificial Intelligence Planning Systems, (ed. Drabble, B.), pp. 221-228, Edinburgh, UK, May 1996, AAAI Press.

Tate, A. (1996b) Responsive Planning and Scheduling Using AI Planning Techniques - Optimum-AIV - in "Trends & Controversies - AI Planning Systems in the Real World", IEEE Expert: Intelligent Systems & their Applications, Vol. 11 No. 6, pp. 4-12, December 1996.

Tate, A. (1998) “Roots of SPAR”, in "Special Issue on Ontologies", Knowledge Engineering Review, Vol.13(1), March, 1998, Cambridge University Press.

Tate, A. (2000a) “<I-N-OVA> and <I-N-CA> - Representing Plans and other Synthesized Artifacts as a Set of Constraints”, AAAI-2000 Workshop on Representational Issues for Real-World Planning Systems, at the National Conference of the American Association of Artificial Intelligence (AAAI-2000), Austin, Texas, USA, August 2000.

Tate, A. (2000b) “Intelligible AI Planning”, in Proceedings of the Twentieth British Computer Society Special Group on Expert Systems International Conference on Knowledge Based Systems and Applied Artificial Intelligence, Cambridge, UK, December 2000.

Tate, A., Dalton, J. and Levine, J. (1998) "Generation of Multiple Qualitatively Different Plan Options", Fourth International Conference on AI Planning Systems (AIPS-98), Pittsburgh, PA, USA, June 1998.

Tate, A., Dalton, J. and Levine, J. (2000a) “O-Plan: a Web-based AI Planning Agent”, AAAI-2000 Intelligent Systems Demonstrator, in Proceedings of the National Conference of the

American Association of Artificial Intelligence (AAAI-2000), Austin, Texas, USA, August 2000.

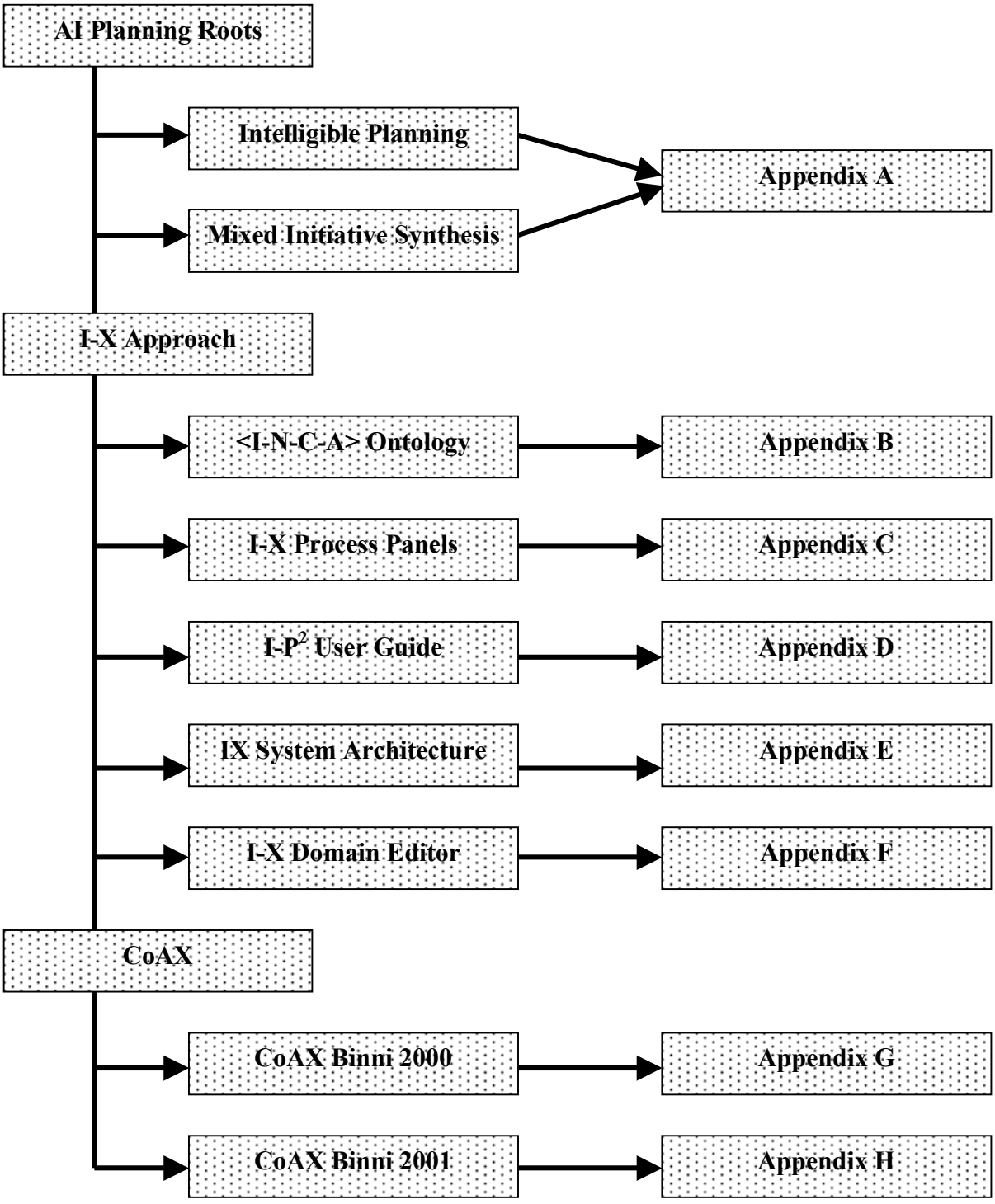
Tate, A., Dalton, J., Jarvis, P. and Levine, J. (2000b) "Using AI Planning Technology for Army Small Unit Operations", Poster Paper in the Proceedings of the Artificial Intelligence Planning and Scheduling Systems Conference (AIPS-2000), Breckenridge, Colorado, USA, April 2000.

Tate, A., Levine, J., Dalton, J. and Nixon, A. (2001) “Task Achieving Agents on the World Wide Web”, in “Creating the Semantic Web”, Fensel, D., Hendler, J., Liebermann, H. and Wahlster, W. (eds.), MIT Press, 2001.

Uschold, M., King, M., Moralee, S. and Zorgios, Y. (1998) "The Enterprise Ontology", in "Special Issue on Ontologies", Knowledge Engineering Review, Vol.13(1), March, 1998, Cambridge University Press.

Roadmap to Attached Papers

The attached appendices contain papers describing a number of aspects of the I-X work in greater detail. They have mostly been published in the peer-reviewed literature. This roadmap highlights the aspects of the research that the more detailed papers describe.



Intentionally Blank

Appendix A: Intelligible AI Planning - Generating Plans Represented as a Set of Constraints

Austin Tate

Abstract: Realistic planning systems must allow users and computer systems to cooperate and work together using a "mixed initiative" style. Black box or fully automated solutions are not acceptable in many situations. Studies of expert human problem solvers in stressful or critical situations show that they share many of the problem solving methods employed by hierarchical planning methods studied in Artificial Intelligence. But powerful solvers and constraint reasoners can also be of great help in parts of the planning process. A new more intelligible approach to using AI planning is needed which can use the best "open" styles of planning based on shared plan representations and hierarchical task networks (HTN) and which still allow the use of powerful constraint representations and solvers.

I-Plan is a design for a new planning system based on these principles. It is part of the I-X suite of intelligent tools. I-Plan is modular and can be extended via plug-ins of various types. It is intended to be a "lightweight" planning system which can be embedded in other applications. In its simplest form it can provide a small personal planning aid that can be deployed in portable devices and other user-orientated systems to add planning facilities into them. In its more developed forms it will approach the power of generative AI planners such as O-Plan. It provides a framework for including powerful constraint solvers in a framework that is intelligible to the users.

I-Plan is grounded in the <I-N-OVA> (*Issues – Nodes - Orderings/Variables/Auxiliary*) constraints model used to represent plans and processes. <I-N-OVA> is intended to support a number of different uses:

- for automatic and mixed-initiative generation and manipulation of plans and to act as an ontology to underpin such use;
- as a common basis for human and system communication about plans;
- as a target for principled and reliable acquisition of plans, process models and process product information;
- to support formal reasoning about plans.

The I-Plan design and the <I-N-OVA> ontology provide an extensible framework for adding detailed constraint representations and reasoners into planners. These can be based on powerful automated methods. But this can be done in a context which provides overall human intelligibility.

Citation: Tate, A. (2000) "Intelligible AI Planning", in Research and Development in Intelligent Systems XVII – the Proceedings of ES2000, The Twentieth British Computer Society Special Group on Expert Systems International Conference on Knowledge Based Systems and Applied Artificial Intelligence, pp. 3-16, Cambridge, UK, December 2000, Springer.

Introduction

Planning is about much more than solving specifically stated problems as efficiently as possible. It is also about modelling domains in which planning takes place, understanding the roles of the various human and system agents involved in the planning process and in the domain in which plans are executed, and it is about communicating tasks, plans, intentions and effects between those agents. Realistic planning systems must allow users and computer systems to cooperate and work together using a "mixed initiative" style. Black box or fully automated solutions are not acceptable in many situations. Studies of expert human problem solvers in stressful or critical situations (Klein, 1998) show that they share many of the problem solving methods employed by some of the methods studied in AI planning to address these issues.

This paper argues that a Hierarchical Task Network (HTN) least commitment planning approach - as used for many years in practical planning systems such as NOAH (Sacerdoti, 1975), Nonlin (Tate, 1977), SIPE (Wilkins, 1988) and O-Plan (Currie and Tate, 1991) - provides an intelligible framework for mixed-initiative multi-agent human/system planning environments. When joined with a strong underlying constraint-based ontology of plans it can provide a framework in which powerful problem solvers based on search and constraint reasoning methods can be employed and still retain human intelligibility of the overall planning process and the plan products that are created.

I-Plan is a design for a new "lightweight" planning system based on these principles. It is part of the I-X[6] suite of intelligent tools and is being designed to be embedded in other applications. I-Plan is modular and can be extended via plug-ins of various types. In its simplest form it can provide a small planning aid that can be deployed in portable devices and other user-orientated systems to add planning facilities into them. In its more developed forms it will approach the power of major generative AI planners such as O-Plan (Tate et. al, 1994; Tate et. al., 2000).

I-X

Work in Intelligent Planning and Activity Management at the University of Edinburgh[7] has led to a number of planning systems and approaches that are re-used on a number of projects. New work will drawn on this work, generalise it, and significantly extend the application of the core concepts and assets, leading to new re-usable components, and create opportunities for applications and further research.

This new programme is called I-X and the core components are a shared model representation called <I-N-CA> and a systems integration architecture. A variety of re-usable components and systems will be built on the new architecture and these will be collectively referred to as I-Technology and I-Tools.

---

[6] I-X is the successor project to O-Plan – see http://www.aiai.ed.ac.uk/project/ix/

[7] See http://www.aiai.ed.ac.uk/project/plan/

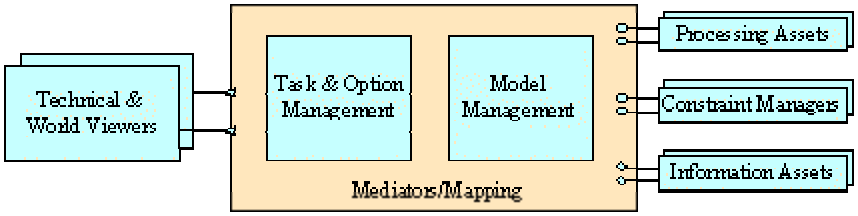


Figure 1: I-X Components

I-X provides a systems integration architecture. Its design is based on the O-Plan agent architecture. I-X incorporates components and interface specifications which account for simplifications, abstractions and clarifications in the O-Plan work. I-X provides an issue-handling workflow style of architecture, with reasoning and functional capabilities provided as plug-ins. Also via plug-ins it allows for sophisticated management and use of the internal model representations to reflect the application domain of the system being built in I-X. I-X agents may be recursively or fractally composed, and may interwork with other processing cells or architectures. This is a systems integration approach now being advocated by a number of groups concerned with large scale, long-lived, evolving and diverse systems integration issues.

The I-X approach has 5 aspects:

1. Systems Integration - A broad vision of an open architecture for the creation of intelligent systems for the synthesis of a result or "product" which is based on a "two cycle" approach which uses plug-in components to "handle issues" and to "manage and respect the domain model".

2. Representation - a core notion of the representation of a process or plan as a set of nodes making up the components of the process or plan model, along with constraints on the relationship between those nodes and a set of outstanding issues. This representation is termed <I-N-CA> - Issues, Nodes, Critical Constraints and Auxiliary Constraints.

3. Reasoning - the provision of reusable reasoning capabilities.

4. Viewers and User Interfaces - to understand user roles in performing activities and to provide generic modules which present the state of the process they are engaged in, their relationships to others and the status of the artifacts/products they are working with.

5. Applications - work in various application sectors which will seek to create generic approaches (I-Tools) for the various types of task in which users may engage. One important application is I-Plan for planning tasks.

We propose to bring together a number of threads of previous research and development, and use state-of-the-art understanding of the conceptual basis for flexible, incremental, mixed-initiative planning and activity management systems. We will incorporate these into an open, flexible,

lightweight and embeddable system. This will be written in Java for portability and to maximise reuse potential. The core of the system will be an agenda-based issue handling system based on workflow principles. It will be specialised to any particular task by incorporating suitable issue-handling capabilities which could be supplied by human or system components. It will be designed to allow for very significant extension via an open capability plug-in interface and via an interface to allow for the use of constraint management methods, feasibility estimators, simulators, etc. The system will be able to inter-work with other workflow and cooperative working support systems, and will not make assumptions about the internal architecture of those other systems.

The components of the I-X systems integration architecture are shown diagrammatically in figure 1 and are as follows:

- Task and Option Management -- The capability to support user tasks via appropriate use of the processing and information assets and to assist the user in managing options being used within the model.

- Model Management -- coordination of the capabilities/assets to represent, store, retrieve, merge, translate, compare, correct, analyse, synthesise and modify models.

- Issue Handlers -- Functional components (distinguished into those which can add to the model (synthesis) and those which analyse the model (to add information only).

- Constraint Managers -- Components which assist in the maintenance of the consistency of the model.

- Information Assets -- Information storage and retrieval components.

- Viewers -- User interface, visualisation and presentation viewers for the model - sometimes differentiated into technical model views (charts, structure diagrams, etc.) and world model views (simulations, animations, etc.)

- Mediators -- Intermediaries or converters between the features of the model and the interfaces of active components of the framework (such as viewers, processing assets, constraint managers and information assets).

A number of different types of “sockets” are available within the framework to reflect the protocols or interfaces into which the various components can fit. The necessity for specific sockets and the types of components vary across projects to some extent, but the separation into viewers, processing assets, constraint managers and information assets has been found to be useful in a number of AIAI projects. This also puts the I-X work on a convergent path with other Model/Viewer/Controller styles of systems framework.

<I-N-OVA> and <I-N-CA>

I-Plan is grounded in the <I-N-OVA> (*Issues – Nodes - Auxiliary*) constraints model which is used to represent plans and processes. The more general <I-N-CA> (*Issues – Nodes - Critical/Auxiliary*) constraints model can be used for wider applications in design, configuration and other tasks which can be characterised as the synthesis and maintenance of an artifact or product.

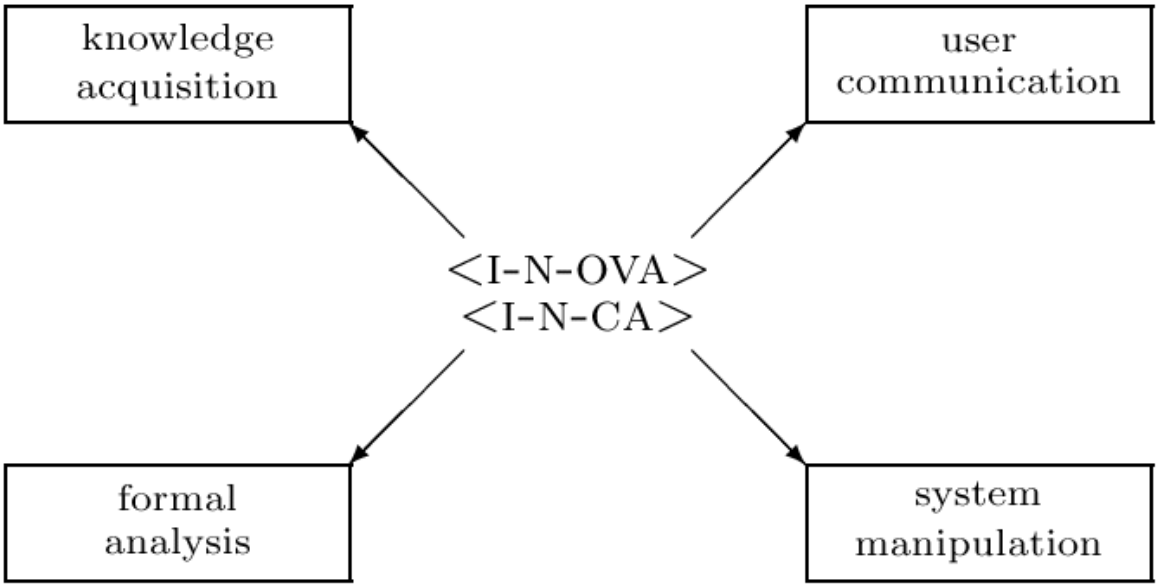


Figure 2: <I-N-OVA> and <I-N-CA> Support Various Requirements

As shown in figure 2, the <I-N-OVA> and <I-N-CA> constraint models are intended to support a number of different uses:

- for automatic and mixed-initiative generation and manipulation of plans and other synthesised artifacts and to act as an ontology to underpin such use;
- as a common basis for human and system communication about plans and other synthesised artifacts;
- as a target for principled and reliable acquisition of plans, process models and process product information;
- to support formal reasoning about plans and other synthesised artifacts.

These cover both formal and practical requirements and encompass the requirements for use by both human and computer-based planning and design systems.

The <I-N-OVA> (*Issues – Nodes - Orderings/Variables /Auxiliary}*) Model is a means to represent plans and activity as a set of constraints. By having a clear description of the different components within a plan, the model allows for plans to be manipulated and used separately from the environments in which they are generated. The underlying thesis is that plans can be represented by a set of constraints on the behaviours possible in the domain being modelled and that plan communication can take place through the interchange of such constraint information.

<I-N-OVA>, when first designed (Tate, 1996), was intended to act as a bridge to improve dialogue between a number of communities working on formal planning theories, practical planning systems and systems engineering process management methodologies. It was intended to support new work then emerging on automatic manipulation of plans, human communication about plans, principled and reliable acquisition of plan information, and formal reasoning about plans. It has since been utilised as the basis for a number of research efforts, practical applications and emerging international standards for plan and process representations. For some of the history and relationships between earlier work in AI on plan representations, work from the process and design communities and the standards bodies, and the part that <I-N-OVA> played in this see Tate (1998).

In Tate (1996), the <I-N-OVA> model is used to characterise the plan representation used within O-Plan and is related to the plan refinement planning method used in O-Plan. The <I-N-OVA> work is related to emerging formal analyses of plans and planning. This synergy of practical and formal approaches can stretch the formal methods to cover realistic plan representations as needed for real problem solving, and can improve the analysis that is possible for practical planning systems.

We have generalised the <I-N-OVA> approach to design and configuration tasks with I, N, CA components - where C represents the "critical constraints" in any particular domain - much as certain O and V constraints do in a planning domain. We believe the approach is valid in design and synthesis tasks more generally - we consider planning to be a limited type of design activity. <I-N-CA> is used as an underlying ontology for the I-X project.

The <I-N-OVA> and <I-N-CA> work is intended to utilise a synergy of practical and formal approaches which are stretching the formal methods to cover realistic representations, as needed for real problem solving, and can improve the analysis that is possible for practical planning systems.

< I-N-OVA> - Representing Plans as a Set of Constraints on Behaviour

A plan is represented as a set of constraints which together limit the behaviour that is desired when the plan is executed. The set of constraints are of three principal types with a number of sub-types reflecting practical experience in a number of planning systems.

Plan Constraints
- I - Issues (Implied Constraints)
- N - Node Constraints (on Activities)
- OVA - Detailed Constraints
  - O - Ordering Constraints
  - V - Variable Constraints
  - A - Auxiliary Constraints
    - Authority Constraints
    - Condition Constraints
    - Resource Constraints
    - Spatial Constraints
    - Miscellaneous Constraints

Figure 3: <I-N-OVA> Constraint Model of Activity

The node constraints (these are often of the form "include activity") in the <I-N-OVA> model set the space within which a plan may be further constrained. The I (issues) and OVA constraints restrict the plans within that space which are valid.

Planning is the taking of planning decisions (I) which select the activities to perform (N) which creates, modifies or uses the plan objects or products (V) at the correct time (O) within the authority, resources and other constraints specified (A). The node constraints in the <I-N-OVA> model set the space within which a plan may be further constrained. The I (issues) and OVA constraints restrict the plans within that space which are valid. The Issues are the items on which selection of Plan Modification Operators is made in agenda based planners.

Others have recognised the special nature of the inclusion of activities into a plan compared to all the other constraints that may be described. Khambhampati and Srivastava (1996) differentiate Plan Modification operators into "progressive refinements" which can introduce new actions into the plan, and "non-progressive refinements" which just partitions the search space with existing sets of actions in the plan. They call the former genuine planning refinement operators, and think of the latter as providing the scheduling component.

If we consider the process of planning as a large constraint satisfaction task, we may try to model this as a Constraint Satisfaction Problem (CSP) represented by a set of variables to which we have to give a consistent assignment of values. In this case we can note that the addition of new nodes ("include activity" constraints in <I-N-OVA>) is the only constraint which can add variables dynamically to the CSP. The Issue (I) constraints may be separated into two kinds: those which may (directly or indirectly) add nodes to the plan and those which cannot. The I constraints which can lead to the inclusion of new nodes are of a different nature in the planning process to those which cannot.

Some ordering (temporal) and variable constraints are distinguished from all other constraints since these act as "critical" constraints, usually being involved in describing the others -- such as in a resource constraint which will often refer to plan objects/variables and to relationships between time points or intervals.

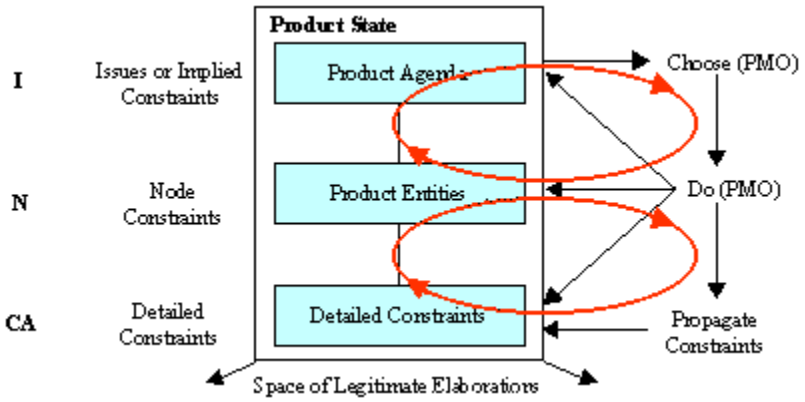


Figure 4: I-X and I-Plan Abstract Architecture: Two Cycles of Processing - Handle Issues, Respect Constraints. PMO=Product Modification Operator

I-Plan Abstract Design

The I-Plan design is based on two cycles of processing. The first addresses one or more "issues" from a task agenda, and the second ensures that constraints in the domain in which processing takes place is respected. So the processing cycles can be characterised as "handle issues, respect constraints". The emerging partial plan or schedule is analysed to produce a further list of issues or agenda entries. A choice of the issues to address is used to drive a workflow-style processing cycle of choosing "Plan Modification Operators" and then executing them to modify the emerging plan state. Figure \ref{two-cycles} shows this graphically for the more general case of designing or synthesising any product - where the issue handlers are labelled "PMO" - which then stands for the "Product Modification Operator".

This approach is taken in systems like O-Plan, OPIS (Smith, 1994), DIPART (Pollack, 1994), TOSCA (Beck, 1994), etc. The approach fits well with the concept of treating plans as a set of constraints which can be refined as planning progresses. Some such systems can also act in a non-monotonic fashion by relaxing constraints in certain ways.

Having the implied constraints or "agenda" as a formal part of the plan provides an ability to separate the plan that is being generated or manipulated from the planning system and process itself and this is used as a core part of the I-Plan design.

Mixed Initiative Planning approaches, for example in O-Plan (Tate, 1994), improve the coordination of planning with user interaction by employing a clearer shared model of the plan

as a set of constraints at various levels that can be jointly and explicitly discussed between and manipulated by user or system in a cooperative fashion. I-Plan will adopt this approach.

Summary

The overall architecture of I-Plan has been described along with the <I-N-OVA> Constraint Model of Activity and the more general <I-N-CA> Constraint Model for Synthesised Artifacts. These are designed to draw on strengths from a number of different communities: the AI planning community with both its theoretical and practical system building interests; the issue-based design community, those interested in formal ontologies for processes and products; the standards community; those concerned with new opportunities in task achieving agents on the world wide web; etc.

<I-N-OVA> is intended to act as a bridge to improve dialogue between the communities working in these areas and potentially to support work on automatic manipulation of plans, human communication about plans, principled and reliable acquisition of plan information, and formal reasoning about plans. <I-N-CA> is designed as a more general underlying ontology which can be at the heart of a flexible and extensible systems integration architecture involving human and system agents.

The I-Plan planner and <I-N-OVA> ontology together provide an extensible framework for adding detailed constraint representations and reasoners which themselves can be based on powerful automated methods. But this can be done in a context which provides human intelligibility of the overall planning process[8].

Acknowledgements

The O-Plan and I-X projects are sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Command and Control Directorate under grant number F30602-99-1-0024 and the UK Defence Evaluation Research Agency (DERA). The U.S. Government, DERA and the University of Edinburgh are authorised to reproduce and distribute reprints 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 official policies or endorsements, either express or implied, of DARPA, the Air Force Research Laboratory, the U.S. Government, DERA or the University of Edinburgh.

[8] The similarity of the AI planning techniques which can be employed within this framework to those observed in expert human problem solving in crisis situations (Klein, 1998) is described in the appendix.

References

Beck, H. (1993) TOSCA: A Novel Approach to the Management of Job-shop Scheduling Constraints, Realising CIM's Industrial Potential: Proceedings of the Ninth CIM-Europe Annual Conference, pages 138-149, (eds. Kooij, C., MacConaill, P.A., and Bastos, J.).

Currie, K.W. and Tate, A. (1991) O-Plan: the Open Planning Architecture, *Artificial Intelligence* 52(1), Autumn 1991, North-Holland.

Khambhampati, S. and Srivastava, B. (1996) Unifying Classical Planning Approaches, Arizona State University ASU CSE TR 96-006, July 1996.

Klein, G. (1998) Sources of Power - How People Make Decisions, MIT Press, 1998.

Pollack, M. (1994) DIPART Architecture, Technical Report, Department of Computer Science, University of Pittsburgh, PA 15213, USA.

Polyak, S. and Tate, A. (2000) A Common Process Ontology for Process-Centred Organisations, Knowledge Based Systems. Earlier version published as University of Edinburgh Department of Artificial Intelligence Research paper 930, 1998.

Sacerdoti, E. (1977) A structure for plans and behaviours. Artificial Intelligence series, publ. North Holland.

Smith, S. (1994) OPIS: A Methodology and Architecture for Reactive Scheduling, in Intelligent Scheduling, (eds, Zweben, M. and Fox, M.S.), Morgan Kaufmann, Palo Alto, CA., USA,

Tate, A. (1994) Mixed Initiative Planning in O-Plan2, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (ed. Burstein, M.), Tucson, Arizona, USA, Morgan Kaufmann, Palo Alto.

Tate, A. (ed.) (1996a) Advanced Planning Technology – Technological Achievements of the ARPA/Rome Laboratory Planning Initiative (ARPI), AAAI Press.

Tate, A. (1996b) Representing Plans as a Set of Constraints -- the <I-N-OVA> Model, Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96), pp. 221-228, (Drabble, B., ed.) Edinburgh, Scotland, AAAI Press.

Tate, A. (1998) Roots of SPAR - Shared Planning and Activity Representation, Knowledge Engineering Review, Vol. 13, No. 1, March 1998.
See also http://www.aiai.ed.ac.uk/project/spar/

Tate, A. (2000) <I-N-OVA> and <I-N-CA> - Representing Plans and other Synthesized Artifacts as a Set of Constraints, AAAI-2000 Workshop on Representational Issues for Real-World Planning Systems, at the National Conference of the American Association of Artificial Intelligence (AAAI-2000), Austin, Texas, USA, August 2000.

Tate, A., Drabble, B. and Kirby, R. (1994) O-Plan2: an Open Architecture for Command, Planning and Control, in Intelligent Scheduling, (eds, Zweben, M. and Fox, M.S.), Morgan Kaufmann, Palo Alto, CA., USA.

Tate, A., Drabble, B. and Dalton, J. (1994) Reasoning with Constraints within O-Plan2, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (ed. Burstein, M.), Tucson, Arizona, USA, Morgan Kaufmann, Palo Alto, CA, USA.

Tate, A., Levine, J., Jarvis, P. and Dalton, J. (2000) Using AI Planning techniques for Army Small Unit Operations, Poster Paper in the Proceedings of the Fifth International Conference on AI Planning and Scheduling Systems (AIPS-2000), Breckenridge, CO, USA, April 2000.

Wilkins, D. (1988) {\em Practical Planning}, Morgan Kaufmann, Palo Alto, 1988.

Annex: Comparing the Intelligible Planning Approach to Studies of Expert Human Planners

This appendix describes some of the features of the O-Plan and I-Plan approaches and shows the similarity of these approaches with those observed in expert human problem solvers performing in stressful or unusual situations. These observations were made in studies over many years by Klein (1998) and he contrasts these with some automated "black box" AI and algorithmic techniques.

The following note was produced on the DARPA O-Plan Project for an US Army Small Unit Operations Application (Tate et. al., 2000) in June 1999 by Austin Tate.

But I Don't Plan, I Just Know What to Do

There are different types of planning technology available from the AI community. This is not restricted to a simple kind of search from some known initial state to some final desired state seeking the best solution according to some predefined criteria. Gary Klein's book (Klein, 1998) on how people make decisions in situations such as military operations, fire fighting, or other life threatening environments provides a rich set of case studies to show that in relatively few situations were deliberative planning techniques in obvious use. People just seemed to be making the "right" choices – or a choice that worked which was all that was required. They attributed their rapid selection of a suitable course of action to training, experience, or even ESP! Where options were deliberated over and evaluated, the situation for those involved was novel or unusual to their previous experience.

Klein's studies show how people in stressful environments select a course of action and adapt it as circumstances alter. Many of the decisions made by the subjects relate to issues which AI planning researchers are addressing. However, they are far removed from the traditional search style of deliberative plan generation. So we need to establish for the outset that the techniques we are calling upon to address potential planning requirements also are much wider than these simple fully-automated search methods. We are seeking to use rich plan representations in a variety of ways. These are listed below, along with cross references to Klein's book, to show how we can address a variety of decision methods which he is advocating, and which are in use by real problem solvers and commanders . The hope is that the planning requirements we are identifying can be mapped to some of the AI concepts we are bringing to bear on practical planning problems.

- Overall management of the command, planning and control process steps to improve coordination.

- Expansion of a high level abstract plan into greater detail where necessary.

- High level "chunks" of procedural knowledge (Standard Operating Procedures, Best Practice Processes, Tactics Techniques and Procedures, etc.) at a human scale - typically 5-8 actions - can be manipulated within the system [Klein, p. 52 and p. 58].

- Ability to establish that a feasible plan exists, perhaps for a range of assumptions about the situation, while retaining a high level overview. [Klein, p.227, "Include only the detail necessary to establish a plan is possible - do not fall into the trap of choreographing each of their movements"].

- Analysis of potential interactions as plans are expanded or developed [Klein, p 53].

- Identification of problems, flaws and issues with the plan [Klein p. 63 and p. 71].

- Deliberative establishment of a space of alternative options perhaps based on different assumptions about the situation involved of especial use ahead of time, in training and rehearsal, and to those unfamiliar with the situation or utilising novel equipment [Klein p. 23].

- Monitoring of the execution of events as they are expected to happen within the plan, watching for deviations that indicate a necessity to re-plan (often ahead of this becoming a serious problem) [Klein p. 32-33].

- AI planning techniques represent the dynamic state of the world at points in the plan and can be used for "mental simulation" of the execution of the plan [Klein, p. 45].

- Pruning of choices according to given requirements or constraints [Klein, p. 94 "singular strategy"].

- Situation dependent option filtering (sometime reducing the choices normally open to one "obvious" one [Klein p.17-18].

- Satisficing search to find the first suitable plan that meets the essential criteria [Klein p. 20].

- Anytime algorithms which seek to improve on the best previous solution if time permits.

- Heuristic evaluation and prioritisation of multiple possible choices within the constraint search space [Klein, p. 94].

- Repair of plans while respecting plan structure and intentions.

- Uniform use of a common plan representation with embedded rationale to improve plan quality, shared understanding, etc. [Klein, p. 275 7 types of information in a plan].

Gary Klein was asked to comment upon this review of AI techniques as compared to his observations of natural problem solving and decision making in humans. He observed the following in this edited Personal Communication to Austin Tate on 24-Jun-1999 (quoted with permission)

1. I felt a strong kinship with what you are attempting. The effort to use satisficing criteria, the use of anytime algorithms to permit continual improvement, the shift from abstract to detailed plan when necessary, the analysis of interactions in a plan, the identification of flaws in a plan, the monitoring of execution, the use of mental simulation, the representation of a singular strategy, heuristic evaluation, plan repair, and so forth are all consistent with what I think needs to be done.

2. My primary concern is how you are going to do these things....The discipline of AI can provide constraints that will help you understand any of these strategies in richer detail. But those constraints may also prevent you from harnessing these sources of power.

3. Your slogan "Search and you're dead" seems right. Unconstrained search is a mark of intellectual cowardice. And it is also not a useful strategy.

Edited version of Personal Communication from Austin Tate to Gary Klein on 25-Jun-1999

I want to clarify my use of the slogan "Search and you're dead" over the last 20 years. This is the headline, but I then clarify what I mean as "(Unconstrained) search and you're dead".

I have found this to be a useful slogan to express my general approach, and it makes for good knock about fun on panels at conferences. The idea should be to richly describe the constraints known using whatever knowledge is available about the problem, and then to seek solutions in that constrained space. We seek to use knowledge of the domain to constrain the use of blind search or "black box" automated methods in ways which are intelligent and intelligible (to humans).

In reality all planning systems we build have sophisticated search and constraint management components, and it is an aim of our research to be able to utilise the best available in an appropriate context. Search can be a useful tactic in situations where you are underconstrained and stuck. AI has made enormous advances in constraint management using search and other methods over the last 5 years - so much so that some of its proponents argue that we do not need to bother with domain expertise or being knowledge-based about many of the problems we are addressing. It's this latter overenthusiasm for one approach which I seek to counter. Even very powerful search can be made more useful if put into a sensible knowledge-based context. This is, of course, more relevant when humans are involved in the decisions as then a more naturalistic style of mutually progressing towards a solution become a key to successful use of the technology.

Appendix B: <I-N-OVA> and <I-N-CA> - Representing Plans and other Synthesised Artifacts as a Set of Constraints

Austin Tate

Abstract: This paper presents an approach to representing and manipulating plans and other synthesised artifacts in the form of a set of constraints. The <I-N-OVA> (*Issues – Nodes - Orderings/Variables/Auxiliary*) constraints model is used to characterise plans and processes. The more general <I-N-CA> (*Issues – Nodes - Critical/Auxiliary)* constraints model can be used for wider applications in design, configuration and other tasks which can be characterised as the synthesis and maintenance of
an artifact or product.

The <I-N-OVA> and <I-N-CA> constraint models are intended to support a number of different uses:

- for automatic manipulation of plans and other synthesised artefacts and to act as an ontology to underpin such use;
- as a common basis for human communication about plans and other synthesised artifacts;
- as a target for principled and reliable acquisition of plans, models and product information;
- to support formal reasoning about plans and other synthesised artifacts.

Citation: Tate, A. (2000) "<I-N-OVA> and <I-N-CA> - Representing Plans and other Synthesised Artifacts as a Set of Constraints", AAAI-2000 Workshop on Representational Issues for Real-World Planning Systems, at the National Conference of the American Association of Artificial Intelligence (AAAI-2002), Austin, Texas, USA.

Motivation

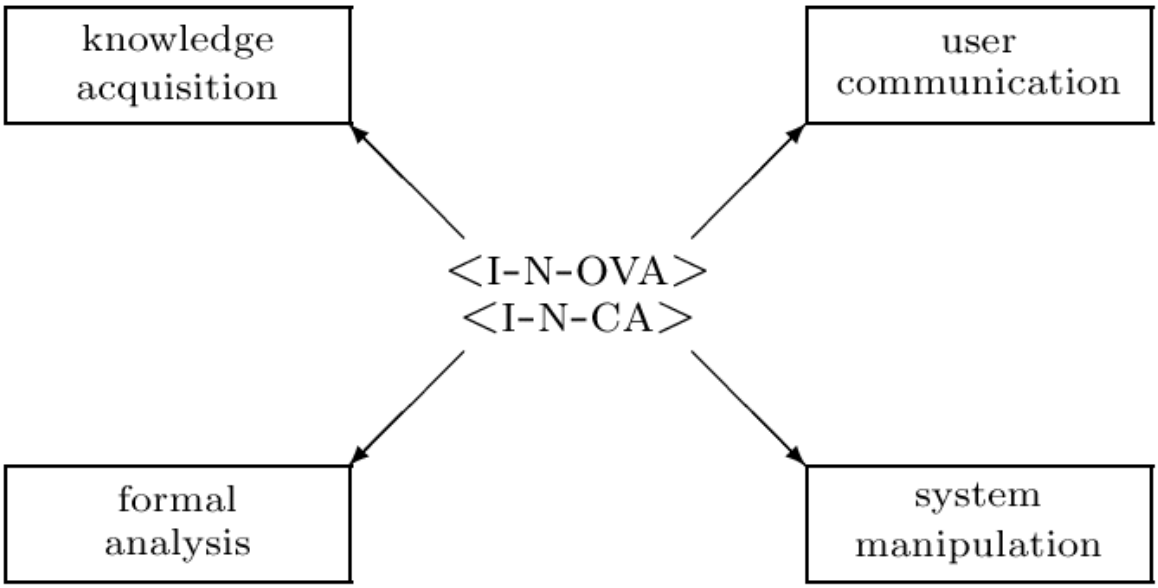


Figure 1: <I-N-OVA> and <I-N-CA> Support Various Requirements

As shown in figure 1, the <I-N-OVA> and <I-N-CA> constraint models are intended to support a number of different uses:

- for automatic manipulation of plans and other synthesised artefacts and to act as an ontology to underpin such use;
- as a common basis for human communication about plans and other synthesised artifacts;
- as a target for principled and reliable acquisition of plans, models and product information;
- to support formal reasoning about plans and other synthesised artifacts.

These cover both formal and practical requirements and encompass the requirements for both human and computer-based planning and design systems.

The <I-N-OVA> *(Issues – Nodes - Orderings/Variables/Auxiliary)* Model is a means to represent plans and activity as a set of constraints. By having a clear description of the different components within a plan, the model allows for plans to be manipulated and used separately from the environments in which they are generated. The underlying thesis is that plans can be

represented by a set of constraints on the behaviours possible in the domain being modelled and that plan communication can take place through the interchange of such constraint information.

<I-N-OVA>, when first designed (Tate, 1996), was intended to act as a bridge to improve dialogue between a number of communities working on formal planning theories, practical planning systems and systems engineering process management methodologies. It was intended to support new work then emerging on automatic manipulation of plans, human communication about plans, principled and reliable acquisition of plan information, and formal reasoning about plans. It has since been utilised as the basis for a number of research efforts, practical applications and emerging international standards for plan and process representations. For some of the history and relationships between earlier work in AI on plan representations, work from the process and design communities and the standards bodies, and the part that <I-N-OVA> played in this see Tate (1998).

In Tate (1996), the <I-N-OVA> model is used to characterise the plan representation used within O-Plan (Currie and Tate, 1991; Tate et.al, 1994) and is related to the plan refinement planning method used in O-Plan. The <I-N-OVA> work is related to emerging formal analyses of plans and planning. This synergy of practical and formal approaches can stretch the formal methods to cover realistic plan representations as needed for real problem solving, and can improve the analysis that is possible for practical planning systems.

We have generalised the <I-N-OVA> approach to design and configuration tasks with I, N, CA components - where C represents the "critical constraints" in any particular domain - much as certain O and V constraints do in a planning domain. We believe the approach is valid in design and synthesis tasks more generally - we consider planning to be a limited type of design activity. <I-N-CA> is used as an underlying ontology for the I-X project[9].

The <I-N-OVA> and <I-N-CA> work is intended to utilise a synergy of practical and formal approaches which are stretching the formal methods to cover realistic representations, as needed for real problem solving, and can improve the analysis that is possible for practical planning systems.

<I-N-OVA> - Representing Plans as a Set of Constraints on Behaviour

The <I-N-OVA> model is a means to represent and manipulate plans as a set of constraints. By having a clear description of the different components within a plan, the model allows for plans to be manipulated and used separately to the environments in which they are generated.

A plan is represented as a set of constraints which together limit the behaviour that is desired when the plan is executed. The set of constraints are of three principal types with a number of sub-types reflecting practical experience in a number of planning systems.

[9] I-X is the successor project to O-Plan – see http://www.aiai.ed.ac.uk/project/ix/

Plan Constraints
- I - Issues (Implied Constraints)
- N - Node Constraints (on Activities)
- OVA - Detailed Constraints
  - O - Ordering Constraints
  - V - Variable Constraints
  - A - Auxiliary Constraints
    - Authority Constraints
    - Condition Constraints
    - Resource Constraints
    - Spatial Constraints
    - Miscellaneous Constraints

Figure 2: <I-N-OVA> Constraint Model of Activity

The node constraints (these are often of the form "include activity") in the <I-N-OVA> model set the space within which a plan may be further constrained. The I (issues) and OVA constraints restrict the plans within that space which are valid. Ordering (temporal) and variable constraints are distinguished from all other auxiliary constraints since these act as *cross-constraints*[10], usually being involved in describing the others -- such as in a resource constraint which will often refer to plan objects/variables and to time points or ranges.

Rationale for the Categories of Constraints within <I-N-OVA>

Planning is the taking of planning decisions (I) which select the activities to perform (N) which creates, modifies or uses the plan objects or products (V) at the correct time (O) within the authority, resources and other constraints specified (A). The node constraints (these are often of the form "include activity") in the <I-N-OVA> model set the space within which a plan may be further constrained. The I (issues) and OVA constraints restrict the plans within that space which are valid. The Issues (I constraints are the items on which selection of Plan Modification Operators is made in agenda based planners.

Others have recognised the special nature of the inclusion of activities into a plan compared to all the other constraints that may be described. Khambhampati and Srivastava (1996) differentiate Plan Modification operators into "progressive refinements" which can introduce new actions into the plan, and "non-progressive refinements" which just partitions the search space with existing sets of actions in the plan. They call the former genuine planning refinement operators, and think of the latter as providing the scheduling component.

[10] Temporal (or spatio-temporal) and object constraints are cross-constraints specific to the planning task. The cross-constraints in some other domain may be some other constraint type.

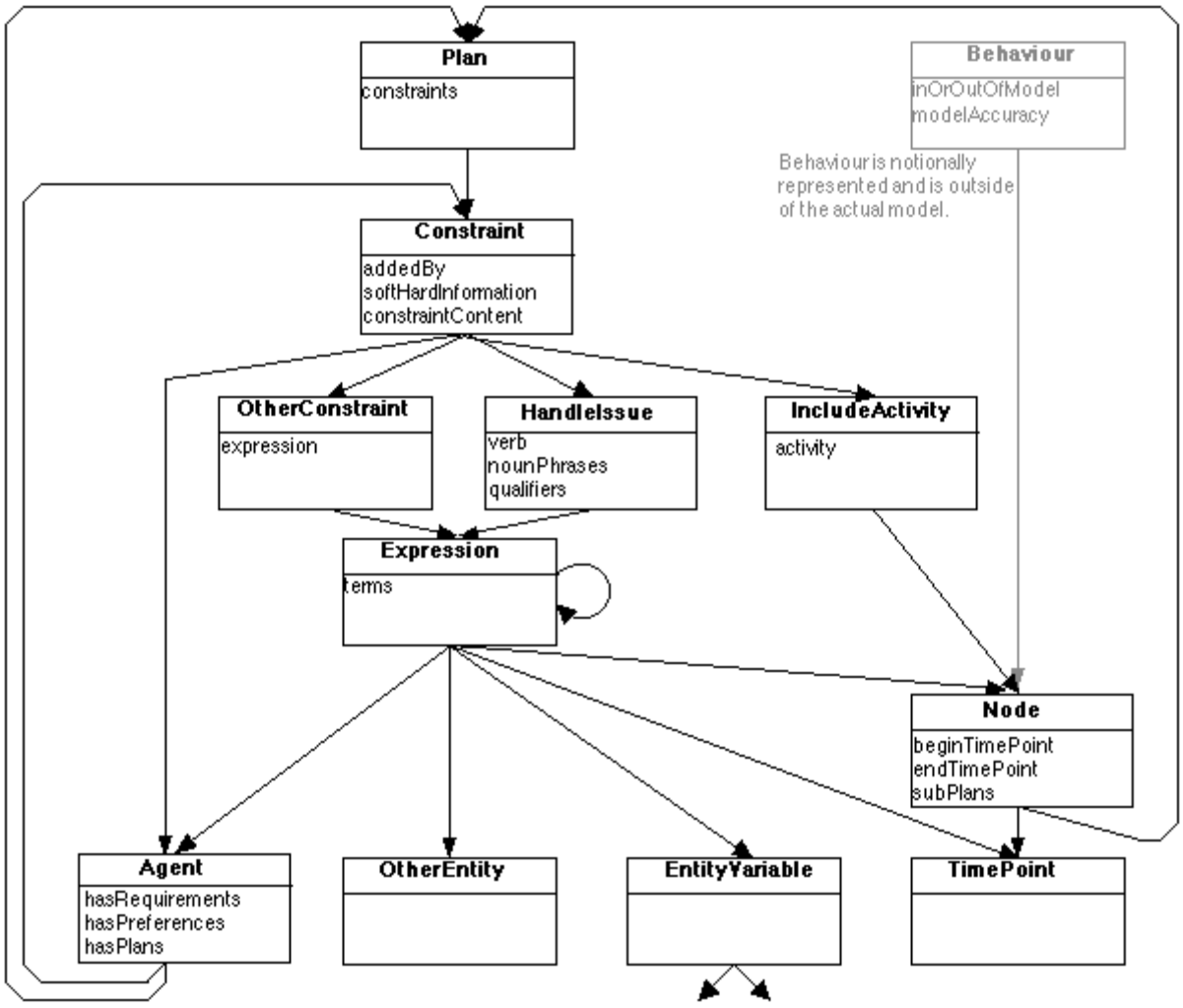


Figure 3: Top Level of <I-N-OVA> Object Model

If we consider the process of planning as a large constraint satisfaction task, we may try to model this as a Constraint Satisfaction Problem (CSP) represented by a set of variables to which we have to give a consistent assignment of values. In this case we can note that the addition of new nodes ("include activity" constraints in <I-N-OVA>) is the only constraint which can add variables dynamically to the CSP. The Issue (I) constraints may be separated into two kinds: those which may (directly or indirectly) add nodes to the plan and those which cannot. The I constraints which can lead to the inclusion of new nodes are of a different nature in the planning process to those which cannot.

Ordering (temporal) and variable constraints are distinguished from all other auxiliary constraints since these act as cross-constraints, usually being involved in describing the others -- such as in a resource constraint which will often refer to plan objects/variables and to time points or intervals.

Mappings to an Object Model

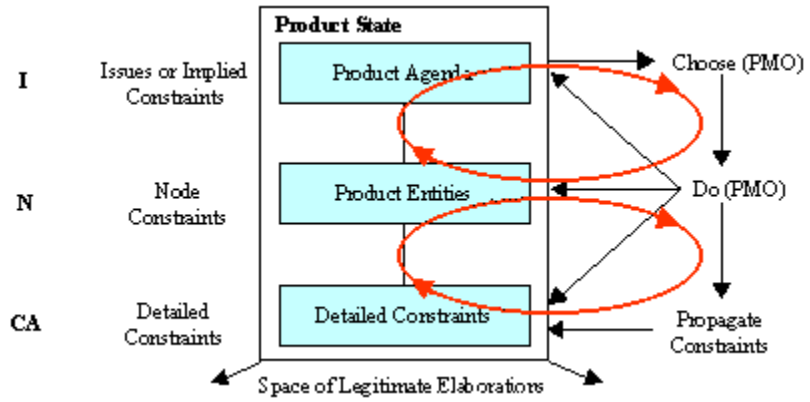


Figure 4: Two Cycles of Processing - Handle Issues, Respect Constraints. PMO=Product Modification Operator

Plans, processes, activity and other synthesised artefacts when described in the form of a set of <I-N-OVA> and <I-N-CA> constraints have a straightforward and easily extended mapping to an object-oriented model, the top level of which is shown in Unified Modelling Language (UML) notation in figure 3[11].

Sorted First Order Logic Base, and XML

<I-N-OVA> and <I-N-CA> are meant as conceptual models which can underly any of a range of languages which can describe activities, plans, processes and other synthesised artifacts. For example, O-Plan is based on <I-N-OVA>, but utilises the Task Formalism domain description language which has a simple keyword introduced syntax.

It is anticipated that any <I-N-OVA> or the more general <I-N-CA> model in whatever language or format it is expressed can be reduced to a conjunctive set of statements in first order logic with strong requirements on the type of the terms involved in each statement - i.e. a Sorted First Order Logic. See Polyak and Tate (2000) for further details, and for a use described in a planning domain modelling support system.

<I-N-OVA> and <I-N-CA> constraint sets lend themselves very well to being used in eXtendible Markup Language (XML) representations of synthesised artifacts, especially when these are still in the process of being designed or synthesised. The processes that are used to do

[11] See http://www.aiai.ed.ac.uk/project/oplan/inova.html for more details and specialisations of the object model related to <I-N-OVA> and <I-N-CA>.

this synthesis and the collaborations and capabilities involved can also be described in <I-N-OVA> and/or <I-N-CA>.

Relationship to Planning Architectures

A general approach to designing AI-based planning and scheduling systems based on partial plan or partial schedule representations is to have an architecture in which a plan or schedule is critiqued to produce a list of issues or agenda entries which is then used to drive a workflow-style processing cycle of choosing a "Plan Modification Operator" and then executing it to modify the plan state. Figure 4 shows this graphically for the more general case of designing or synthesising any product - where the issue handlers are labelled "PMO" - which then stands for the more general "Product Modification Operator".

This approach is taken in systems like O-Plan, OPIS (Smith, 1994), DIPART (Pollack, 1994), TOSCA (Beck, 1994), etc. The approach fits well with the concept of treating plans as a set of constraints which can be refined as planning progresses. Some such systems can also act in a non-monotonic fashion by relaxing constraints in certain ways.

Having the implied constraints or "agenda" as a formal part of the plan provides an ability to separate the plan that is being generated or manipulated from the planning system itself and this is used as a core part of the O-Plan design.

Mixed Initiative Planning approaches, for example in O-Plan (Tate, 1994), improve the coordination of planning with user interaction by employing a clearer shared model of the plan as a set of constraints at various levels that can be jointly and explicitly discussed between and manipulated by user or system in a cooperative fashion.

Summary

The <I-N-OVA> Constraint Model of Activity and the more general <I-N-CA> Constraint Model for Synthesised Artifacts has been described. These are designed to relate strengths from a number of different communities: the AI planning community with both its theoretical and practical system building interests; the issue-based design community, those interested in formal ontologies for processes and products; the standards community; those concerned with new opportunities in task achieving agents on the world wide web; etc.

<I-N-OVA> is intended to act as a bridge to improve dialogue between the communities working in these areas and potentially to support work on automatic manipulation of plans, human communication about plans, principled and reliable acquisition of plan information, and formal reasoning about plans. <I-N-CA> is designed as a more general underlying ontology which can be at the heart of a flexible and extensible systems integration architecture involving human and system agents. This is the aim of new work on the I-X project.

Acknowledgements

The O-Plan and I-X projects are sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Command and Control Directorate under grant number F30602-99-1-0024. The U.S. Government and the University of Edinburgh are authorised to reproduce and distribute reprints 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 official policies or endorsements, either express or implied, of DARPA, the Air Force Research Laboratory, the U.S. Government, or the University of Edinburgh

References

Beck, H. (1993) TOSCA: A Novel Approach to the Management of Job-shop Scheduling Constraints, Realising CIM's Industrial Potential: Proceedings of the Ninth CIM-Europe Annual Conference, pages 138-149, (eds. Kooij, C., MacConaill, P.A., and Bastos, J.).

Currie, K.W. and Tate, A. (1991) O-Plan: the Open Planning Architecture, *Artificial Intelligence* 52(1), Autumn 1991, North-Holland.

Khambhampati, S. and Srivastava, B. (1996) Unifying Classical Planning Approaches, Arizona State University ASU CSE TR 96-006, July 1996.

Pollack, M. (1994) DIPART Architecture, Technical Report, Department of Computer Science, University of Pittsburgh, PA 15213, USA.

Polyak, S. and Tate, A. (2000) A Common Process Ontology for Process-Centred Organisations, Knowledge Based Systems. Earlier version published as University of Edinburgh Department of Artificial Intelligence Research paper 930, 1998.

Smith, S. (1994) OPIS: A Methodology and Architecture for Reactive Scheduling, in Intelligent Scheduling, (eds, Zweben, M. and Fox, M.S.), Morgan Kaufmann, Palo Alto, CA., USA,

Tate, A. (1994) Mixed Initiative Planning in O-Plan2, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (ed. Burstein, M.), Tucson, Arizona, USA, Morgan Kaufmann, Palo Alto.

Tate, A. (ed.) (1996a) Advanced Planning Technology – Technological Achievements of the ARPA/Rome Laboratory Planning Initiative (ARPI), AAAI Press.

Tate, A. (1996b) Representing Plans as a Set of Constraints -- the <I-N-OVA> Model, Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96), pp. 221-228, (Drabble, B., ed.) Edinburgh, Scotland, AAAI Press.

Tate, A. (1998) Roots of SPAR - Shared Planning and Activity Representation, Knowledge Engineering Review, Vol. 13, No. 1, March 1998.
See also http://www.aiai.ed.ac.uk/project/spar/

Tate, A., Drabble, B. and Kirby, R. (1994) O-Plan2: an Open Architecture for Command, Planning and Control, in Intelligent Scheduling, (eds, Zweben, M. and Fox, M.S.), Morgan Kaufmann, Palo Alto, CA., USA.

Tate, A., Drabble, B. and Dalton, J. (1994) Reasoning with Constraints within O-Plan2, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (ed. Burstein, M.), Tucson, Arizona, USA, Morgan Kaufmann, Palo Alto, CA, USA.

Tate, A., Levine, J., Jarvis, P. and Dalton, J. (2000) Using AI Planning techniques for Army Small Unit Operations, Poster Paper in the Proceedings of the Fifth International Conference on AI Planning and Scheduling Systems (AIPS-2000), Breckenridge, CO, USA, April 2000.

Intentionally Blank

Appendix C: I-P^2 - Intelligent Process Panels to Support Coalition Operations

Austin Tate, Jeff Dalton and Jussi Stader

Abstract: I-X is a research programme with a number of different aspects intended to create a well-founded approach to allow humans and computer systems to cooperate in the creation or modification of some product or products such as documents, plans or designs. I-X may also be used to support more general collaborative activity.

The I-X research draws on earlier work on O-Plan (Tate et.al., 1998; Tate et.al., 2000; Tate et.al., 2002), <I-N-OVA> (Tate, 1996), the Enterprise Project (Fraser and Tate, 1995; Stader, 1996); Uschold, et.al., 1998) and the TBPM project (Stader, 2000) but seeks to make the framework generic and to clarify terminology, simplify the approach taken, and increase re-usability and applicability of the core ideas.

I-X Applications are being studied in a variety of areas. These currently include:

- Coalition Operations (CoAX: I-LEED, I-DEEL)
- Emergency and Unusual Procedure Assistance (I-Rescue)
- Help Desk Support (I-Help)
- Multi-Perspective Knowledge Modelling and Management (I-AKT)
- Contextualised Presentations of Procedures and Plans (I-Tell)
- Collaborative Meeting and Task Support (I-Room, I-Space)

An application of I-X Process Panels within a military Coalition context - part of the Coalition Agents eXperiment - CoAX (Allsopp et.al., 2001; Allsopp et.al., 2002) will be described in this paper.

Citation: Tate, A., Dalton, J. and Stader, J. (2002) "I-P^2 - Intelligent Process Panels to Support Coalition Operations", Proceedings of the Second International Conference on Knowledge Systems for Coalition Operations (KSCO-2002), Toulouse, France.

I-X Research Programme

I-X is a research programme with a number of different aspects intended to create a well-founded approach to allow humans and computer systems to cooperate in the creation or modification of some product such as a plan, design or physical entity – i.e. it supports **synthesis tasks**. I-X may also be used to support more general collaborative activity.

The I-X research draws on earlier work on O-Plan (Tate et.al., 1998; 2000; 2002). <I-N-OVA> (Tate, 1996) and the Enterprise Project (Fraser and Tate, 1995; Uschold, et.al., 1998) but seeks to make the framework generic and to clarify terminology, simplify the approach taken, and increase re-usability and applicability of the core ideas.

The I-X research programme includes the following threads or work areas:

1. **I-Core**, which is the core architecture, the underlying ontology of activity and processes termed <I-N-CA>, and the terminology used to describe applications, systems or agents built in the I-X framework.

2. **I-PE**, which is the I-X Process Editor, which is itself an I-X application but is also used to create and maintain the process models and activity specifications used elsewhere.

3. **I-P^2**, which are I-X Process Panels used to support user tasks and cooperation.

4. **I-Plan**, which is the I-X Planning System. This is also used within I-P^2 and other applications as it provides generic facilities for supporting planning, process refinement, dynamic response to changing needs, etc.

5. **I-Views**, which are viewers for processes and products, and which are employed in other applications of I-X. I-Views can be for a wide range of modalities and types of user.

6. **I-Faces**, which are underlying support utilities to allow for the creation of user interfaces (User I-Faces), inter-agent communications (Communications I-Faces) and repository access (Repository I-Faces).

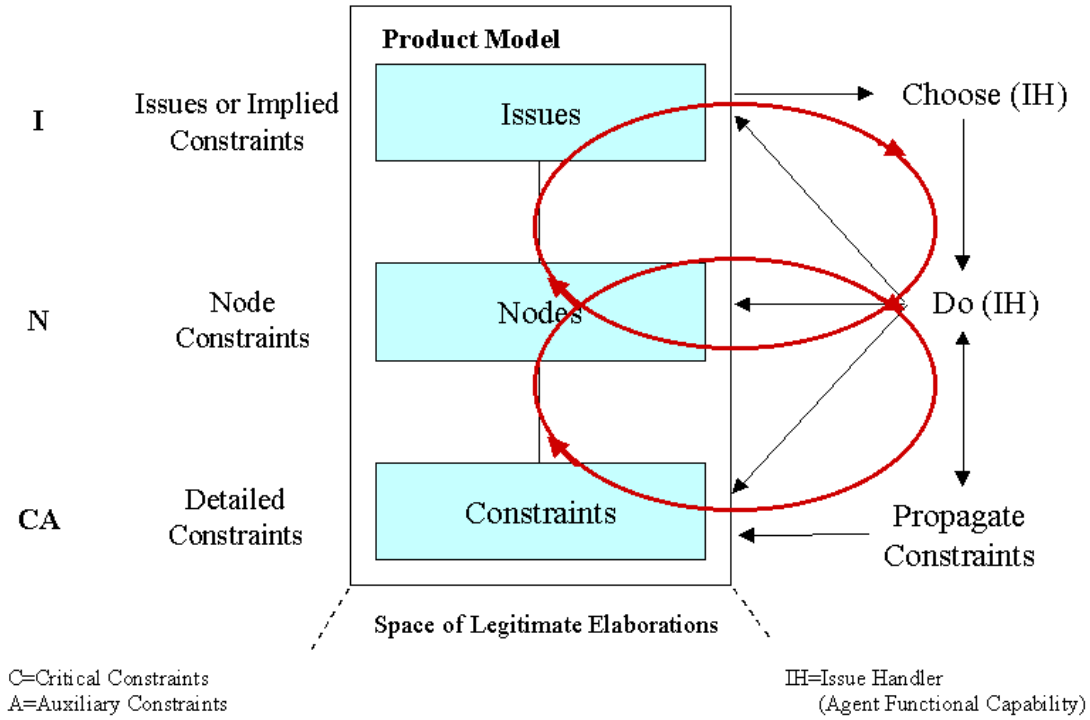
7. **I-X Applications** of the above work areas in a variety of areas. These currently include:
    a. Coalition Operations (CoAX: I-LEED, I-DEEL)
    b. Emergency and Unusual Procedure Assistance (I-Rescue)
    c. Help Desk Support (I-Help)
    d. Multi-Perspective Knowledge Modelling and Management (I-AKT)
    e. Medical Best Practice Procedures or Protocols (I-Medic)
    f. Natural Language Presentations of Procedures and Plans (I-Tell)
    g. Collaborative meeting and task support (I-Me, I-Room and I-Space)

8. **I-X Student Projects**, which are deepening and refining a number of aspects of the I-X research programme.

9. **I-X Technology Transfer**, including work on standards committees, especially for process, plan, activity and capability models.

I-X Approach

The I-X approach involves the use of shared models for task directed cooperation between human and computer agents who are jointly exploring (via some processes) a range of alternative options for the synthesis of an artifact such as a design or a plan (termed a product).



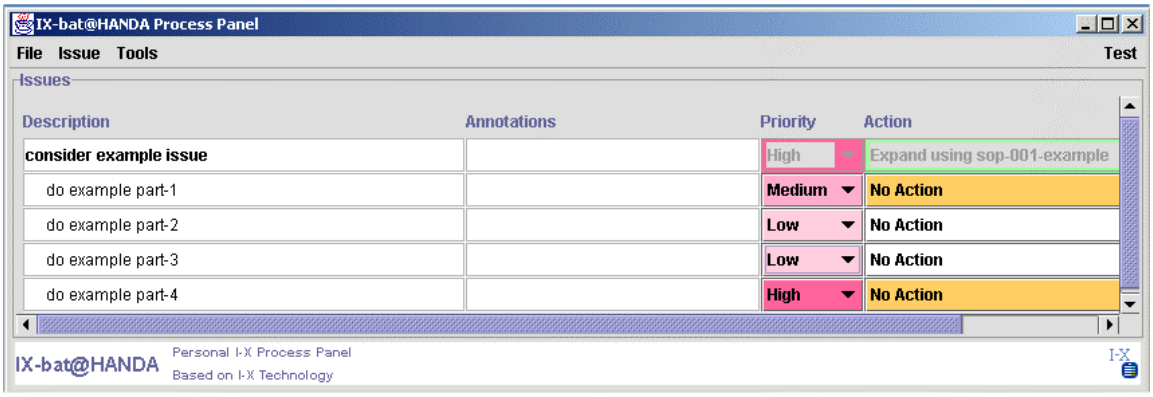
- An I-X system or agent has two cycles:
  - Handle Issues
  - Respect Domain Constraints

- An I-X system or agent carries out a (perhaps dynamically determined) process that leads to the production of (one or more alternative options for) a synthesised artifact.

- An I-X system or agent views the synthesised artifact as being represented by a set of constraints on the space of all possible artifacts in the domain.

I-X also involves a modular systems integration architecture that strongly parallels and supports the abstract view described above.

I-X Process Panels (I-P^2)



The aim of an I-X Process Panel (I-P^2) is to act as a workflow, reporting and messaging "catch all" for its user. It can act in conjunction with other panels for other users if desired.

- Can take ANY requirement to:
  - Handle an issue
  - Perform an activity
  - [later: Add a constraint]

- Deals with these via:
  - Manual (user) activity
  - Internal capabilities
  - External capabilities (invoke or query)
  - Reroute or delegate to other panels or agents (pass)
  - Plan and execute a composite of these capabilities (expand)

- Receives reports and messages and, where possible, interprets them to:
  - Understand current status of issues, activities and constraints
  - Understand current world state, especially status of process products
  - Help control the situation

- Copes with partial knowledge

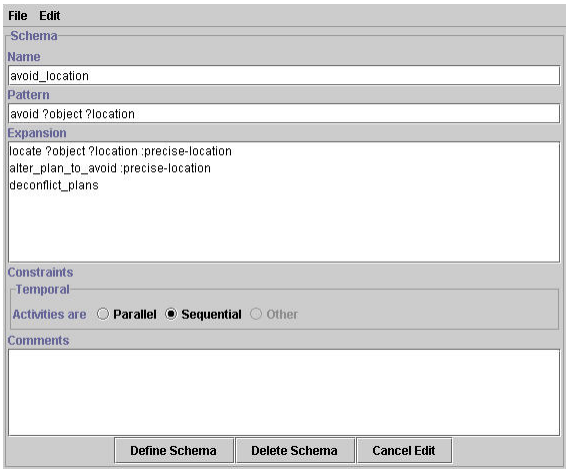
An I-X Process Panel supports a user or collaborative users in selecting and carrying out "processes" and creating or modifying "process products". Both processes and process products

are abstractly considered to be made up on **"Nodes"** (activities in a process, or parts of a process product) which may have parts called sub-nodes making up a hierarchical description of the process or product. The nodes are related by a set of detailed **"Constraints"** of various kinds. A set of **"Issues"** is associated with the processes or process products to represent unsatisfied requirements, problems raised as a result of analysis or critiquing, etc. Processes and process products in I-X are represented in the <I-N-CA> (Issues - Nodes - Critical/Auxiliary) Constraints Model of Synthesised Artifacts.

Three example process panels are shown in the figure below. These panels are from a demonstration of agent systems within a military Coalition context – part of the Coalition Agents eXperiment – CoAX (Allsopp et.al., 2001; Allsopp et.al., 2002).

I-X Process Editor (I-PE)

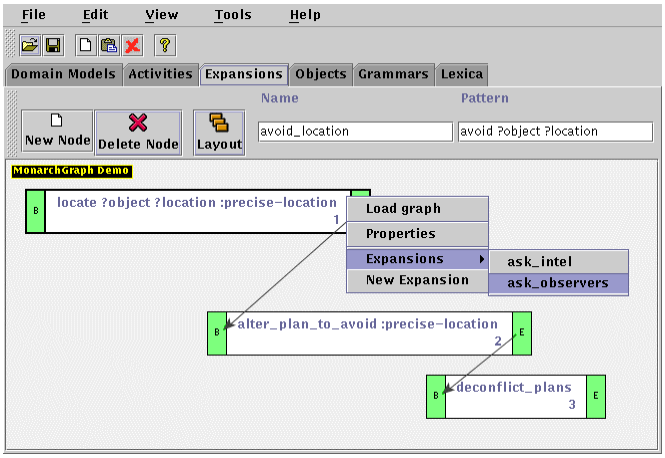
The process descriptions used by I-X Process Panels are kept in a domain library. This can be loaded when a panel is started, and can be added to dynamically by a user of a panel.



Simple View - the process panels contain a simple, form-based domain and process editor (right). This simple editor allows simple task breakdown structures to be specified along with a temporal constraint that the sub-steps should all be sequentially ordered or all kept in parallel.

Advanced View - a more powerful domain and process editor allows for multiple perspectives and views to be used to create rich process models beyond those that can be created with the simple view editor. This can be reached by selecting advanced view from the simple domain/process editor. It is also available as a stand-alone application to maintain a set of domain and process libraries.

The advanced editor consists of a form-based structure editor (not shown), which looks similar to the simple editor but allows the user to specify more complex temporal constraints. Other constraints, like spatial ones or constraints on resources, can also be specified using the advanced view.

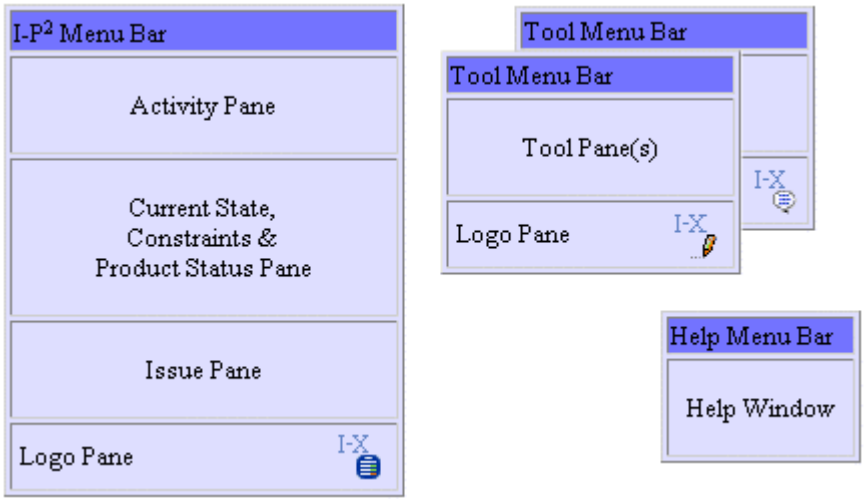


The graphical editor (right) provides an alternative view to the form-based editor. The graphical editor illustrates precedence relationships between the sub-steps of a

process. This editor can also be used to specify task breakdown structures via the expansion of nodes in the graph. Full details of the process and its sub-steps can be accessed via the properties of nodes.

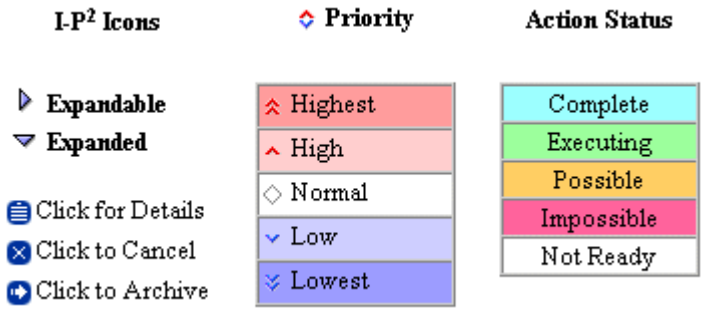
Use of XML and Text Editors - the process and domain models are maintained in XML. You can also modify them using an XML Editing Tool - such as the freely available Microsoft XML Notepad (see http://msdn.microsoft.com/xml/notepad/intro.asp) or a text editor.

I-P^2 Generic Approach



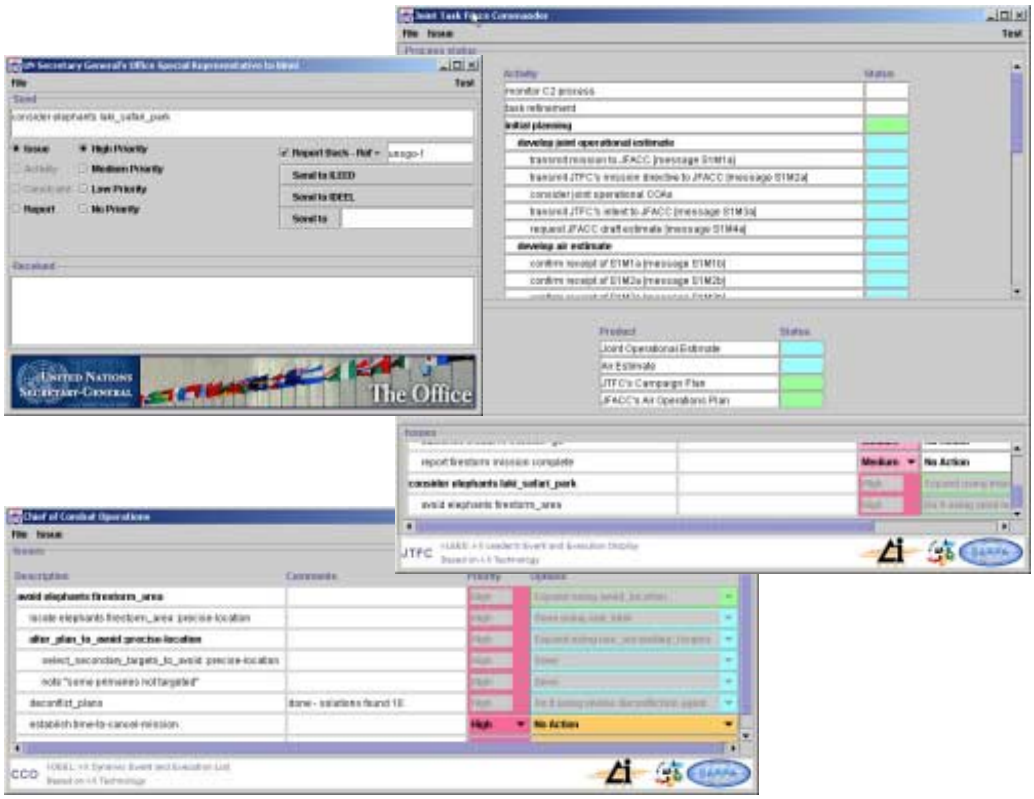
An I-X Process Panel has a menu bar and a number of sub-panels. These can include an activity pane that describes the list of activities to be carried out. Alternative actions to take to perform these activities may be available. A current state pane can be included to describe the current situation, and in particular it may describe the status of the various "process products" being created or modified by the user of the panel. The list of outstanding issues can be included in a pane, and this "to do list" often is at the heart of I-X process panels, and is usually present in all applications. Finally a logo pane can be added to customise the process panel to specific applications.

From the menu bar it is possible to activate a number of "tools" which currently include a free format instant messaging/chat tool, and access to the Process Editor (in Simple or Advanced View varieties). Help is also available from the process panel menu bar.



The process panel includes a number of icons and tabular entries to assist its user to maintain awareness of the current status of activities being executed, process product status and issue status. These are shown in the diagram here.

I-LEED and I-DEEL – A Coalition Application of I-X Process Panels



The CoAX demonstration of the I-X concepts is grounded in a system for supporting event management in a highly dynamic military Coalition environment. Two I-X process Panels are involved in the demonstration. One is called the "I-X Leaders Event and Execution List" – I-LEED – and support the Joint Task Force Commander (JTFC) in the Coalition HQ. The other is called the "I-X Dynamic Execution Event List" – I-DEEL – and support the Chief of Combat Operations (CCO). A third system is provided to act as a source for events and messages to initiate the demonstration. Notionally this acts as the UN Secretary general's Office Special Representative to Binni – the region where the Coalition mission is taking place (Rathmell, 1999).

The process panels support their respective users in mapping events to actions they decide are appropriate to deal with such events. The panels have some (partial) level of process knowledge

in a simple process library, and a way to create / expand task lists / processes on the fly which are dependent on the context or situation that is prevailing at the time. The process panels are designed to be able to be used by any decision-maker operating at different time scales and with appropriate abstraction levels of process description to support people involved in military Command and Control in Coalitions and other operations.

The process panels use the issue-addressing core of I-X to handle issues (derived from externally generated events or user initiated ones) relevant to a Coalition C^2 process within the context of the CoAX Binni scenario. Where these do not match directly to a known capability, the panel seeks (or the user could input) process / task expansions of how to handle these issues and use a very simple expansion engine (a mini-planner termed I-Plan) to match the expanded activities to a range of known capabilities which are performable by the process panel user or by other colleagues or to suitable tasks / solutions which the user could input. Therefore, in a simple way, a process panel can dynamically generate an appropriate response to the issue or event in the current situation - this allows the user to create and interact with a "dynamic event list" to assist with the monitoring of execution outcomes and the resultant actions / changes / new taskings. Links can be created between related tasks (by the user or inferred by the system) and the system can monitor dependencies, etc.

The process panels can identify actions based on known external capabilities to enable the user to "enact" these steps. The process panels can maintain a simple display of the current status of issues and events delegated to the panel and information on how far along in the response process things had proceeded.

Summary

This paper has described the application of I-X Process Panels to military Coalition scenarios. Such process panels can be employed quickly and with partial knowledge to connect together “come-as-you-are” participants and systems together, especially in contexts where physical connectivity of systems is too time consuming, or is not allowed due to security constraints. As process and other knowledge is made available improved interoperability can be supported – allowing for more intelligent task and process management in a loose collaborative setting.

Acknowledgements

The I-X project is partially sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Command and Control Directorate under grant number F30602-99-1. The US Government and the University of Edinburgh are authorised to reproduce and distribute reprints 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 official policies or endorsements, either express or implied, of DARPA, the Air Force Research Laboratory, the US Government or the University of Edinburgh.

References

Allsopp, D., Beautement, P., Bradshaw, J.M., Carson, J., Kirton, M., Suri, N. and Tate, A. (2001) "Software Agents as Facilitators of Coherent Coalition Operations", *6th International Command and Control Research and Technology Symposium*, US Naval Academy, Annapolis, Maryland, USA, 19-21 June 2001.

Allsopp, D., Beautement, P., Bradshaw, J., Durfee, E., Kirton, M., Knoblock, C., Suri, N., Tate, A. and Thompson, C. (2002) "Coalition Agents eXperiment: Multi-agent Co-operation in an International Coalition Setting*", 2nd International Conference on Knowledge Systems for Coalition Operations*, Toulouse, France, 23-24 April 2002.

Fraser, J. and Tate, A. (1995) "The Enterprise Tool Set - An Open Enterprise Architecture", *Proceedings of the Workshop on Intelligent Manufacturing Systems, International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada, August 1995.

Rathmell, R.A. (1999) A Coalition Force Scenario 'Binni - Gateway to the Golden Bowl of Africa', In *Proceedings of the International Workshop on Knowledge-Based Planning for Coalition Forces*, (ed. Tate, A.) pp. 115-125, Edinburgh, Scotland, 10th-11th May 1999.

Stader J., Moore J., Chung P., McBriar I., Ravinranathan M., Macintosh A.. (2000) "Applying Intelligent Workflow Management in the Chemicals Industries"; in *"The Workflow Handbook 2001"*, L. Fisher (ed), Published in association with the Workflow Management Coalition (WfMC), pp 161-181, Oct 2000.

Stader J. (1996) "Results of the Enterprise Project", in *Proceedings of Expert Systems '96, the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK, December 1996.

Tate, A. (1996) "The <I-N-OVA> Constraint Model of Plans", *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, (ed. Drabble, B.), pp. 221-228, Edinburgh, UK, May 1996, AAAI Press.

Tate, A. (1998) "Roots of SPAR", in "Special Issue on Ontologies", *Knowledge Engineering Review*, Vol.13 (1), March 1998, Cambridge University Press.

Tate, A., Dalton, J. and Levine, J. (1998) "Generation of Multiple Qualitatively Different Plan Options", *Fourth International Conference on AI Planning Systems (AIPS-98)*, Pittsburgh, PA, USA, June 1998.

Tate, A., Dalton, J. and Levine, J. (2000) "O-Plan: a Web-based AI Planning Agent", AAAI-2000 Intelligent Systems Demonstrator, in *Proceedings of the National Conference of the American Association of Artificial Intelligence (AAAI-2000)*, Austin, Texas, USA, August 2000.

Tate, A., Levine, J., Dalton, J. and Nixon, A. (2002) “Task Achieving Agents on the World Wide Web”, in “*Creating the Semantic Web*”, Fensel, D., Hendler, J., Liebermann, H. and Wahlster, W. (eds.), MIT Press, 2001.

Uschold, M., King, M., Moralee, S. and Zorgios, Y. (1998) "The Enterprise Ontology", in “Special Issue on Ontologies”, *Knowledge Engineering Review*, Vol.13(1), March, 1998, Cambridge University Press.

Appendix D: I-X Process Panels – User Guide

Austin Tate, Jeff Dalton, Jussi Stader and Stephen Potter

Abstract: A manual for users of the I-X Process Panels. It also contains information on how to create a personalized process panel application, and details of all start-up parameters to the program which allow its behaviour and appearance to be customized.

Citation: Tate, A., Dalton, J., Stader, J. and Potter, S. (2003) "I-X Process Panels – User Guide", Version 2.4, AIAI, University of Edinburgh. Unpublished. Available at http://i-x.info

Introduction to I-X and I-X Process Panels (I-P^2)

I-X Research Programme

I-X is a research programme with a number of different aspects intended to create a well-founded approach to allow humans and computer systems to cooperate in the creation or modification of some product such as a plan, design or physical entity – i.e. it supports **synthesis tasks**. I-X may also be used to support more general collaborative activity.

The I-X research draws on earlier work on O-Plan (Tate et.al. 1998; 2000; 2002), <I-N-OVA> (Tate, 1996), the Enterprise Project (Fraser and Tate, 1995; Stader, 1996; Uschold, et.al., 1998) and the TBPM project (Stader, 2000) but seeks to make the framework generic and to clarify terminology, simplify the approach taken, and increase re-usability and applicability of the core ideas.

The I-X research programme includes the following threads or work areas:

1. **I-Core**, which is the core architecture, and an underlying ontology for activity and processes termed <I-N-C-A> (Issues, Nodes, Constraints and Annotations), and the terminology used to describe systems or applications built in the I-X framework.

2. **I-P^2**, which are I-X Process Panels used to support user tasks and cooperation.

3. **I-DE**, which is the I-X Domain Editor, which is itself an I-X application but is also used to create and maintain the domain description, process models and activity specifications used elsewhere.

4. **I-Plan**, which is the I-X Planning System. This is also used within I-P^2 and other applications as it provides generic facilities for supporting planning, process refinement, dynamic response to changing needs, etc.

5. **I-Views**, which are viewers for processes and products, and which are employed in other applications of I-X. I-Views can be for a wide range of modalities and types of user.

6. **I-Faces**, which are underlying support utilities to allow for the creation of user interfaces (User I-Faces), inter-agent communications (Communications I-Faces) and repository access (Repository I-Faces).

7. **I-X Applications** of the above work areas in a variety of areas. These currently include:
    a. Coalition Operations (CoAX)
    b. Emergency and Unusual Procedure Assistance (I-Aid, I-Help, I-Rescue)
    c. Support Desks (I-Support)

d. Multi-Perspective Knowledge Modelling and Management (I-AKT)
e. Medical Best Practice Procedures or Protocols (I-Medic)
f. Natural Language Presentations of Procedures and Plans (I-Tell)
g. Collaborative meeting and task support (I-Space: I-Room and I-World).
h. Intelligent Messaging (I-Me).

8. **I-X Student Projects**, which are deepening and refining a number of aspects of the I-X research programme.

9. **I-X Technology Transfer**, including work on standards committees, especially for process, plan, activity and capability models.

I-X Process Panels (I-P^2)

The aim of an I-X Process Panel (I-P^2) is to act as a workflow, reporting and messaging "catch all" for its user. It can act in conjunction with other panels for other users if desired.

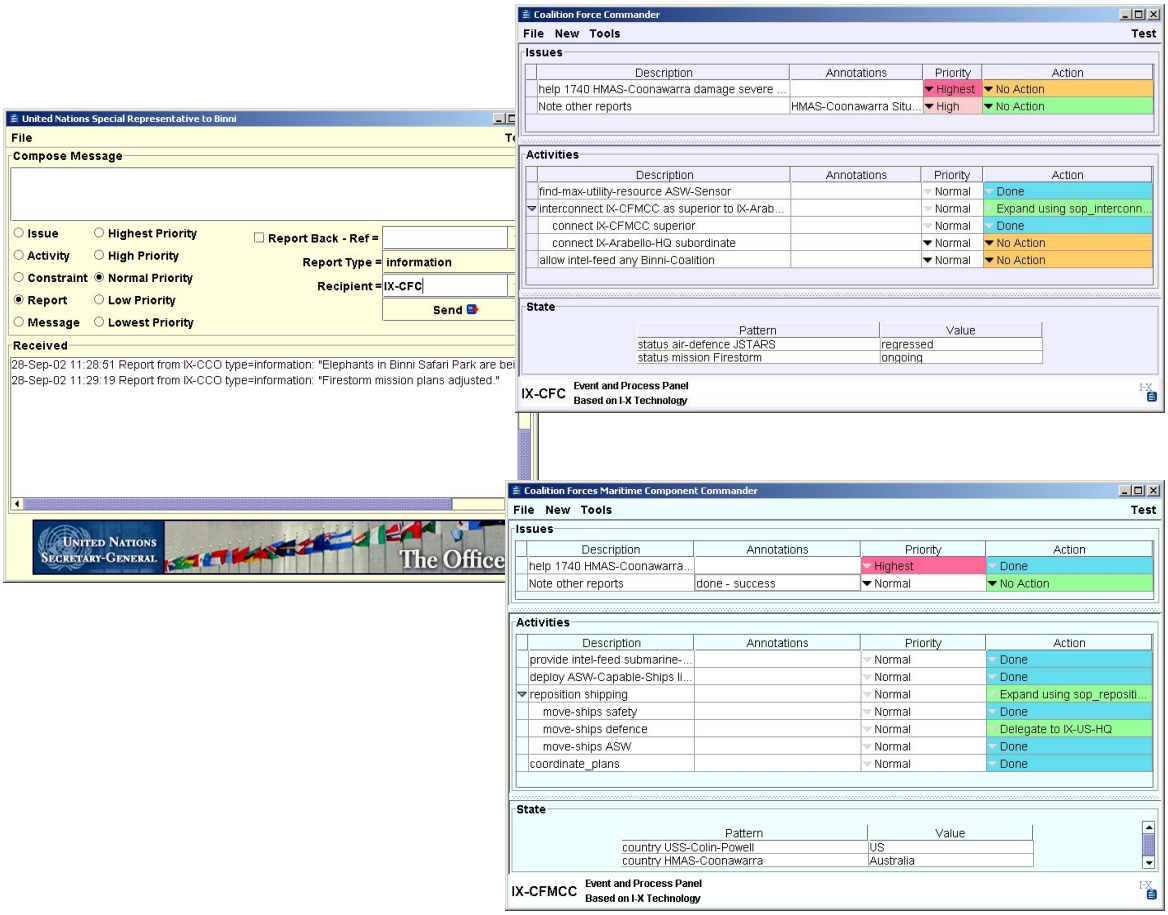
- Can take ANY requirement to:
  - Handle an issue
  - Perform an activity
  - [later: Maintain a constraint]
  - [later: Note an annotation]

- Deals with these via:
  - Manual (user) activity
  - Internal capabilities
  - External capabilities (invoke or query)
  - Reroute or delegate to other panels or agents (escalate, pass or delegate)
  - Plan and execute a composite of these capabilities (expand)

- Receives reports and messages and, where possible, interprets them to:
  - Understand current status of issues, activities, constraints and annotations
  - Understand current world state, especially status of process products
  - Help control the situation

- Copes with partial knowledge

Three example process panels are shown in the figure below. These panels are from a demonstration of agent systems within a military Coalition context – part of the Coalition Agents eXperiment – CoAX (Allsopp et.al. 2001, 2002).

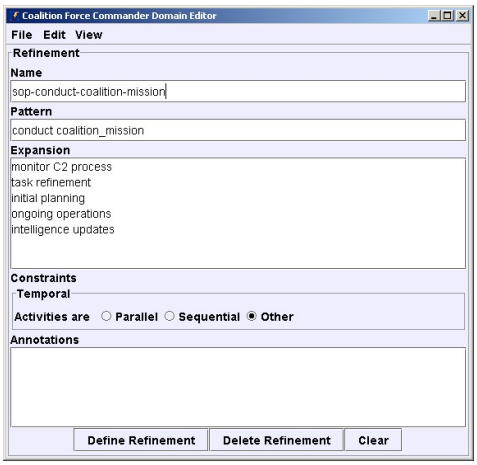


An I-X Process Panel supports a user or collaborative users in selecting and carrying out "processes" and creating or modifying "process products". Both processes and process products are abstractly considered to be made up on **"Nodes"** (activities in a process, or parts of a process product) which may have parts called sub-nodes making up a hierarchical description of the process or product. The nodes are related by a set of detailed **"Constraints"** of various kinds. A set of **"Issues"** is associated with the processes or process products to represent unsatisfied requirements, problems raised as a result of analysis or critiquing, etc.

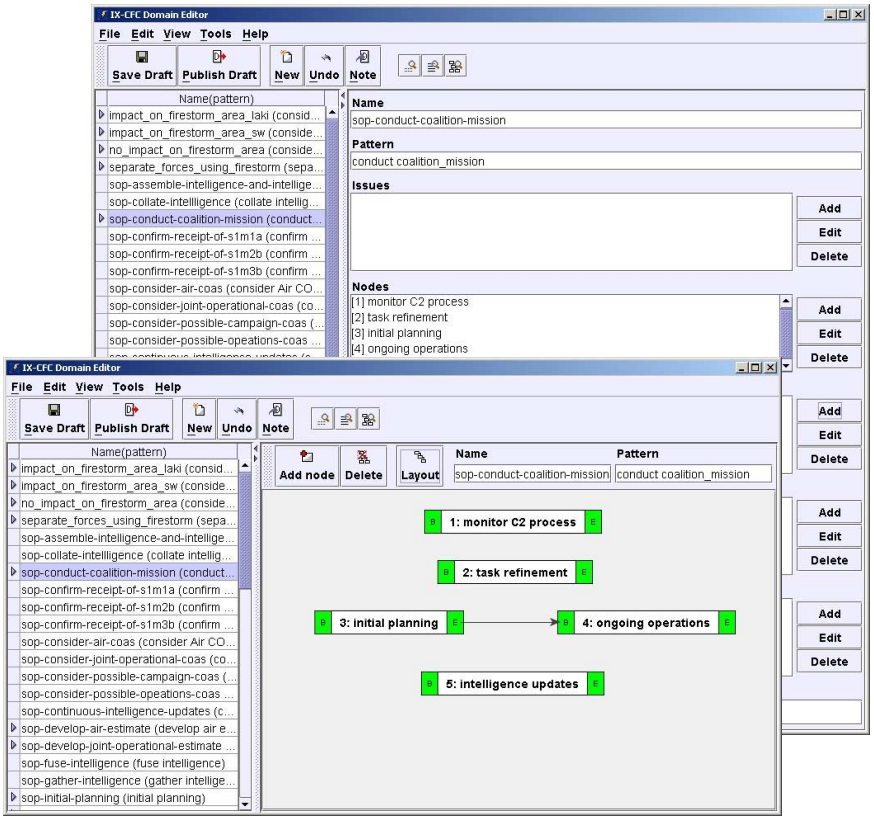
Processes and process products in I-X are represented in the <I-N-C-A> (Issues – Nodes – Constraints – Annotations) model of synthesised artifacts (Tate, 2000).

I-X Domain Editor (I-DE)

The process descriptions used by I-X Process Panels are kept in a domain library. This can be loaded when a panel is started, and can be added to dynamically by a user of a panel.

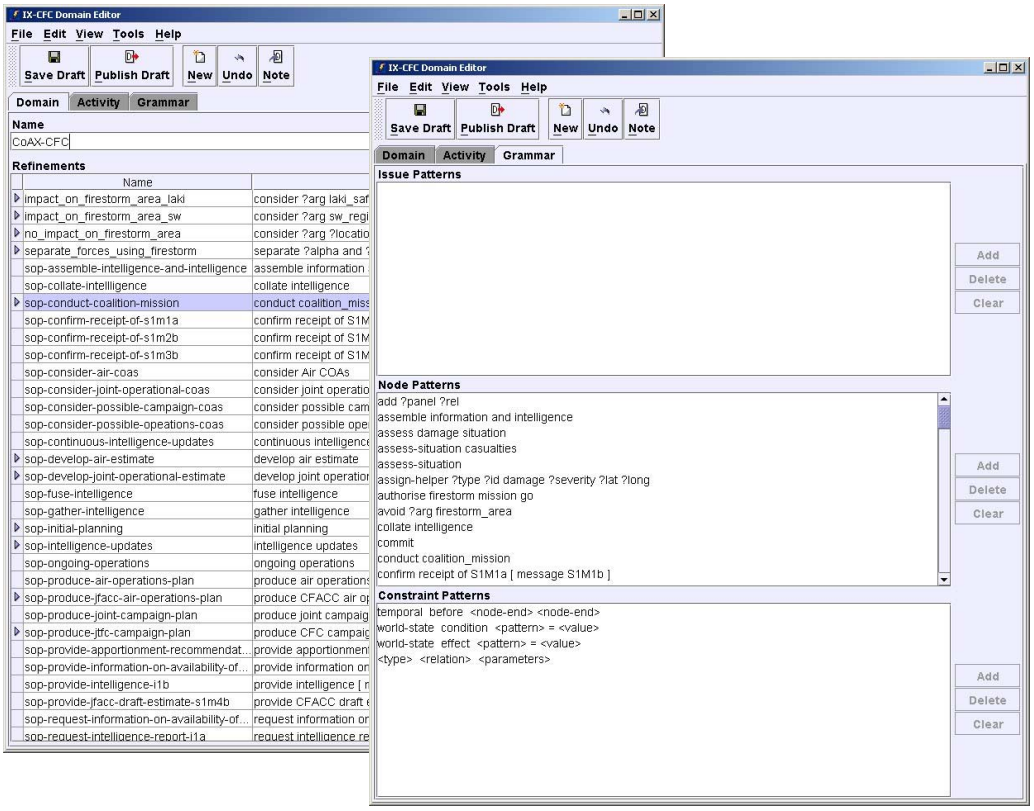


Simple Mode - the process panels contain a simple, form-based domain and process editor (right). This allows simple task breakdown structures to be specified along with a temporal constraint that the sub-steps should all be sequentially ordered or all kept in parallel.

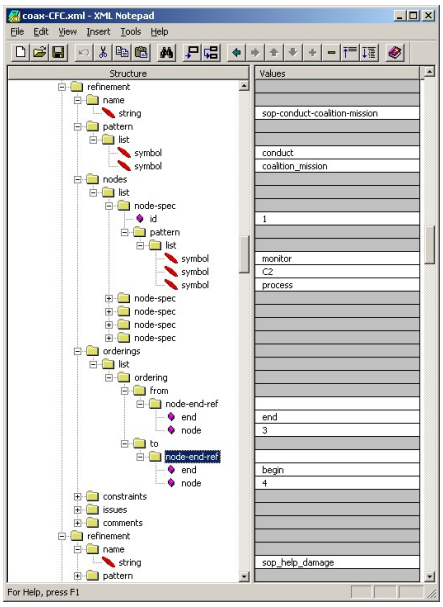


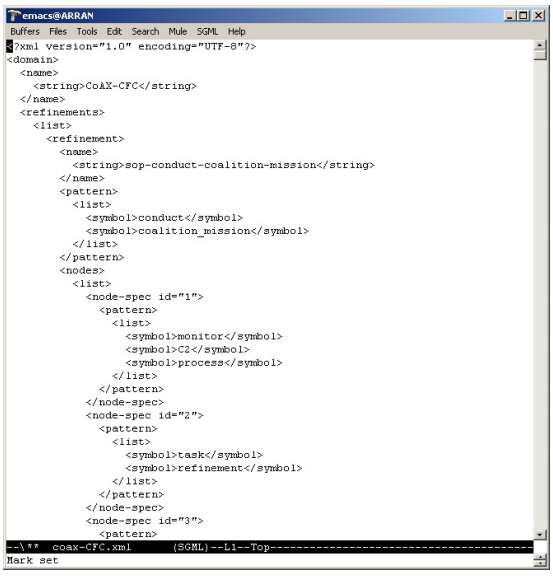
Advanced Mode - a more powerful domain and process editor allows for multiple perspectives and views to be used to create rich process models beyond those that can be created with the simple mode editor. This can be reached by selecting advanced view from the Views pull down menu of the domain editor. I-DE is also available as a stand-alone application to maintain a set of domain and process libraries. The advanced editor provides a "minimal" view which is deliberately similar to the Simple Mode Editor (and could one day replace it). It also provides a "comprehensive" view that allows the user to specify more complex temporal and world-state (condition/effect) constraints. Other constraints, like spatial ones or constraints on resources, can also be specified using the advanced view. A graphical view provides an alternative view to the form-based views. The graphical view illustrates precedence relationships between the sub-steps of a process. This view can also be used to specify task breakdown structures via the expansion of nodes in the graph. In the

advanced view, a tabbed option is available to allow access to related information about a domain model. The minimal, comprehensive and graphical views are all available via the Activities tab. Also available are "Domain" and "Grammar" tabs to view further details.



Use of XML and Text Editors - the process and domain models are maintained in XML. You can also modify them using an XML Editing Tool - such as the freely available Microsoft XML Notepad (see http://msdn.microsoft.com/xml/notepad/intro.asp) or a text editor.





Quick Start Guide

To quickly get started using the I-X Process Panels and run the demonstrations follow the following procedure. This is specifically for Microsoft Windows platforms – but a similar procedure can be used on Unix/Linux also.

1. Obtain and uncompress the I-X system distribution. This will create a single directory with all necessary files. It can be placed anywhere.

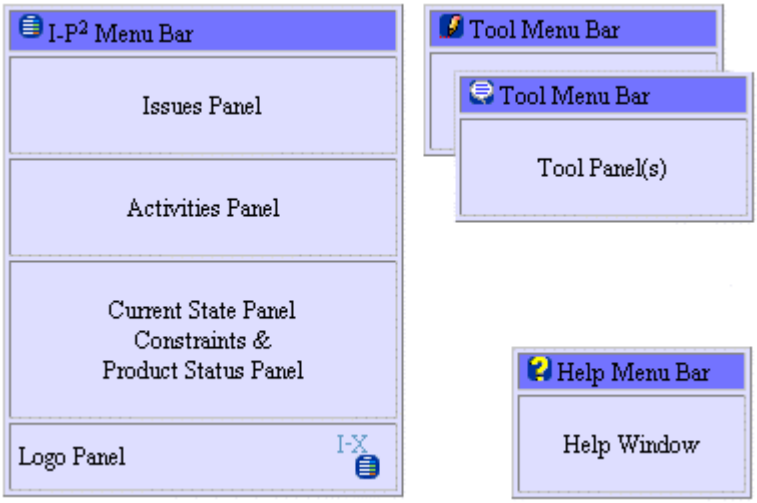
2. Ensure that you have a working Java Development Kit environment with the necessary Java programs on your current path, i.e. you should be able to run a command "java" from any location. You will need to alter the script provided in scripts\win\java-command.bat to set the path explicitly if this is not the case.

3. An example for using the I-X Process Panels is available in apps\isample. You can start up a demonstration by executing (e.g. by double clicking on) 3 of the batch script files in this sub-directory called scripts\win. Run xml-itest-nameserver.bat (done first to provide a simple name/address lookup service to the other process panels), and then xml-supervisor.bat and xml-operator.bat. A process panel customised to have a name based on your system user name and machine domain name is available by running xml-ime.bat. Similar scripts are available for Unix systems in scripts\unix.

4. You can then test the panels by sending sample issues between panels – one way to do that is to use the Test menu on I-Test to send sample issues, activities, constraints, reports or chat-style messages to other panels.

5. To use communications strategies other than simple inbuilt ones (simple and xml - which are available immediately), it is necessary to edit some scripts to set the location of the relevant code on your computer. Edit the script files in comms*\scripts*.

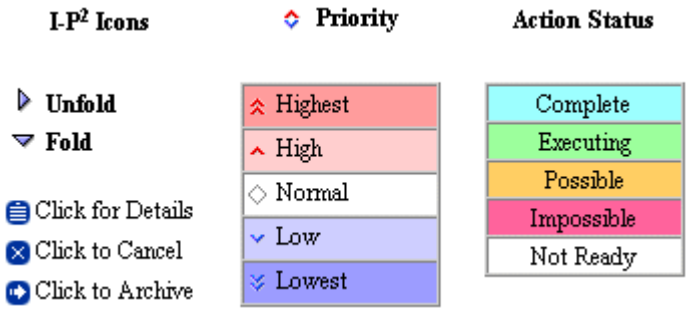
Using an I-X Process Panel



An I-X Process Panel (I-P^2) contains a number of sub-panels that describe:

- A set of "issues" to be "handled".
- A set of "activities" to be "performed".
- Current state information reflecting the current set of "constraints" to be "respected". This includes the status of a range of "process products" being created or manipulated by the processes

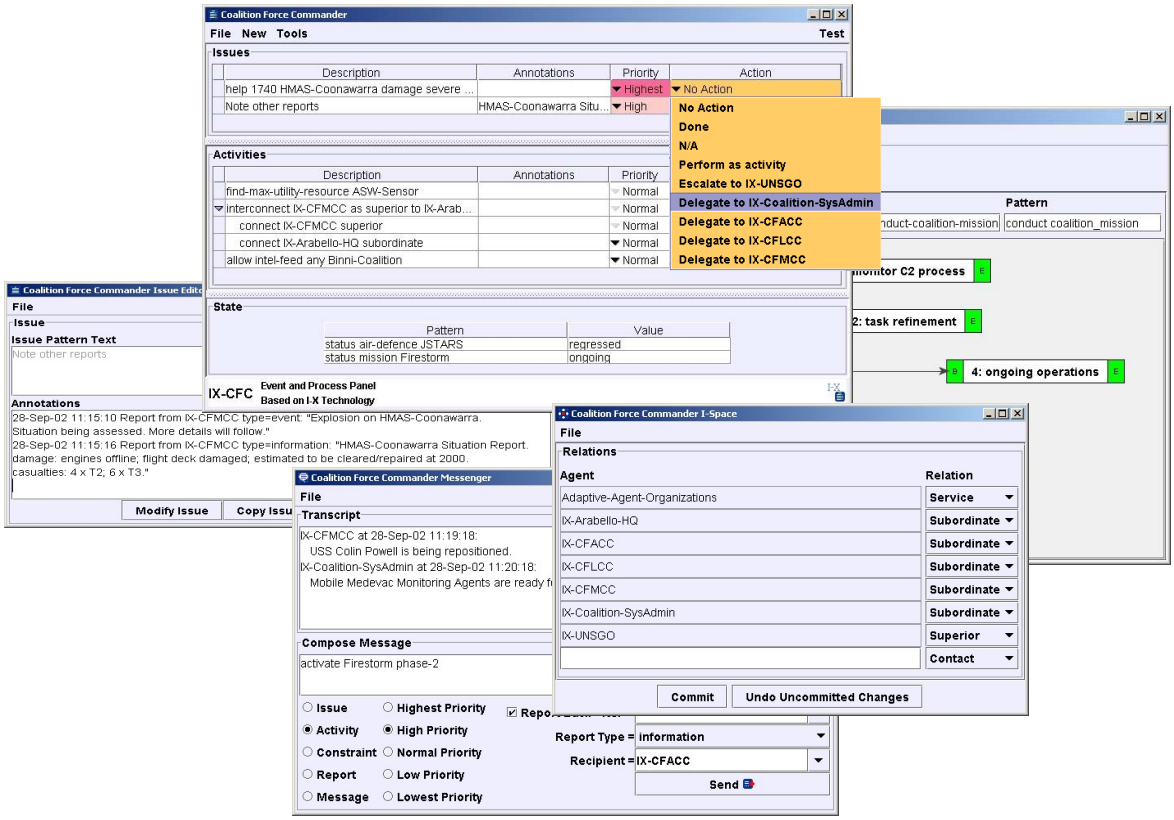
The panel supports it's user in handling issues, deciding on a course of action and performing activities, and maintaining awareness of the current state, constraints, process products, etc.



Entries on panels can be expanded using information provided in the process library used by a panel, or the entries can be passed between panels.

Right click on a line to get a context sensitive menu that describes operations you can perform on the entry. This includes where relevant the ability to pop-up a window with more details of the

entry (say an activity or an issue), or to expand or contract the display of some levels of hierarchically specified activities, to send information about the entry to the Messenger tool for sending on to others (perhaps in a modified form), etc.



A "tools" menu is available to make accessible the following:

- A domain or process library editor to view, edit or add to the list of process descriptions which may be used to "expand" entries on the process panel.
- A tool to view and change the relationships of the current panel to others ("I-Space").
- An instant messaging or "chat" tool to communicate in free format or the encouraged <I-N-C-A> structured forms with other I-X Process Panels and other systems ("intelligent messaging" or "semantically augmented messaging").

Using the I-DE Domain Editor Tool

The main window of the Domain Editor (the frame) contains several editor panels for editing different aspects (or constructs) of the domain. Currently the editors available are

- the Global Domain Editor, which edits information about the domain itself (e.g. the domain name)
- the Activity Editor, which edits information about activities and how they break down into sub-activities (refinement)
- the Grammar Editor, which currently only shows the patterns that are in use in the domain

An editor panel may itself have different "views" that are used to display and edit the panel's constructs. The Activity Editor has three such views:

1. Minimal View: a simplified version of the activity and its refinement. The main simplification is that no constraints are shown
2. Comprehensive View: a view that can display and edit all of an activity's specification
3. Graphical View: a graphical view that uses nodes and arcs to show an activity's sub-activities and the temporal relationships between them.

Domain Editor Window

This window provides access to the most functions of the overall domain editor via its menu bar and access to the most commonly used functions via its tool bar. The window can display in one of three styles: simple, tabbed, and card style. The style can be changed via the Options in the File menu.

The Menu Bar

The menu bar has 5 standard menus:

1. File for closing the Domain Editor and for file access (open/save). All functions here manipulate the domain as a whole, not individual constructs;
2. Edit for manipulating the current construct, i.e. the construct that is currently shown in the Domain Editor's panel;
3. View for changing which panel is shown in the Domain Editor and - if applicable - for changing which view is shown in that panel;
4. Tools for additional support like consistency checks etc.;
5. Help for access to this manual, other help, and information about the application.

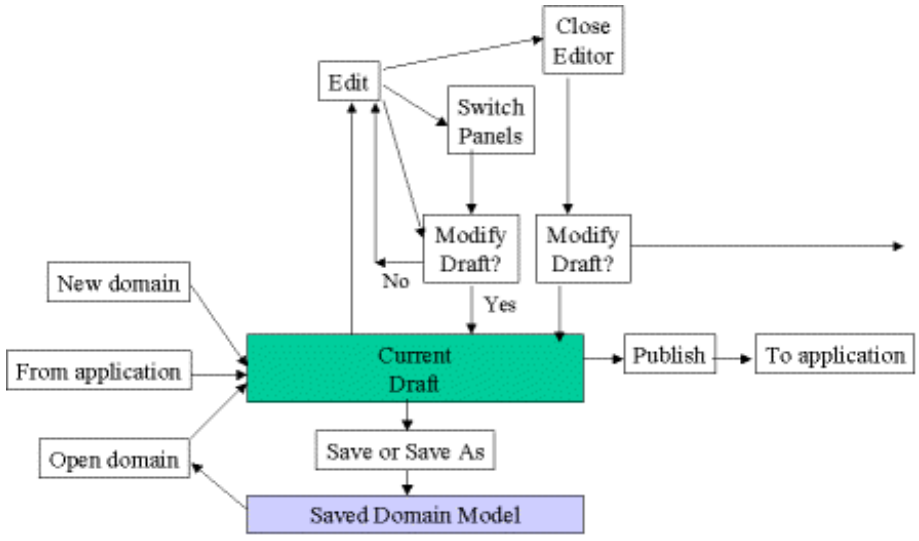
The Tool Bar

The tool bar provides access to the most commonly used functions via buttons. All these functions are also available via the menu bar (in most cases, the image on the toolbar button is shown in the menu next to the corresponding menu item. The toolbar can be switched on and off via Options in the File menu. Moving the mouse over a toolbar button will, after a while, display a "tool tip text" that gives a brief explanation of the button's function.

Working with the Domain Editor

The Domain Editor maintains different levels of updates. The original domain model that the editor is started with is considered a public domain model, which other applications may be using for their own purposes (e.g. within a process panel). This public domain model is kept as it is unless it is explicitly “published” by the Domain Editor's user. (Note that this is true whether the Domain Editor is used in stand-alone mode or as part of another application). There is also a "draft domain model" which is the one that is being edited. The Domain Editor keeps track of any changes that are made to the draft domain model so that updates to the original domain model can be made explicitly.

Saving and Reverting



There are 3 levels of saving:

1. When a construct has been edited in the Domain Editor Panel, initially these changes may be made only in the panel itself, not in the domain construct that is being edited. In order to transfer changes from the panel into the construct in the draft domain, the user has to modify the draft, i.e. note the changes into the draft domain via the toolbar button or the Edit menu. When the user has edited a construct and not modified the draft, the system will prompt the

user to note or discard changes if the user decides to switch constructs, views, or panels, or if the user decides to save or publish the draft domain.

2. Modifying the draft (noting changes) does not save to file, so the next level of saving is to save the draft domain to file. As with all editing applications, it is recommended to do this frequently to ensure that work is not lost. Saving the draft domain to file will write the whole domain with all its constructs into a file in XML format. This can later be loaded into the Domain Editor for further editing, or it can be accessed by other applications.

3. The underlying public domain is not changed by any of the above (simple editing, modifying draft, or saving the draft domain to file). The only way to update the public domain is to publish the draft domain via the toolbar button or the File menu. When this happens, the changes are made to the original domain and these chages will be seen by any applications that have registered as listeners to this domain. Note that publishing is always done for a whole domain, not for individual constructs. Note also that publishing a domain will not save it to file, but the same effect can be achieved by saving the draft domain to file just before or straight after publishing. At that point the draft domain and the public domain can be represented by the same XML structures. It is a good idea to publish from time to time even if the Domain Editor is running stand-alone because it will make the editor more efficient.

It is worth noting that the Global Domain Editor Panel, i.e. the panel that is responsible for editing details about the domain like its name, only considers domain details as part of its editing remit, not the constructs within the domain.

While there is no "undo" function that undoes individual editing steps or editing of individual fields, the following functions are available, corresponding to the 3 levels of saving:

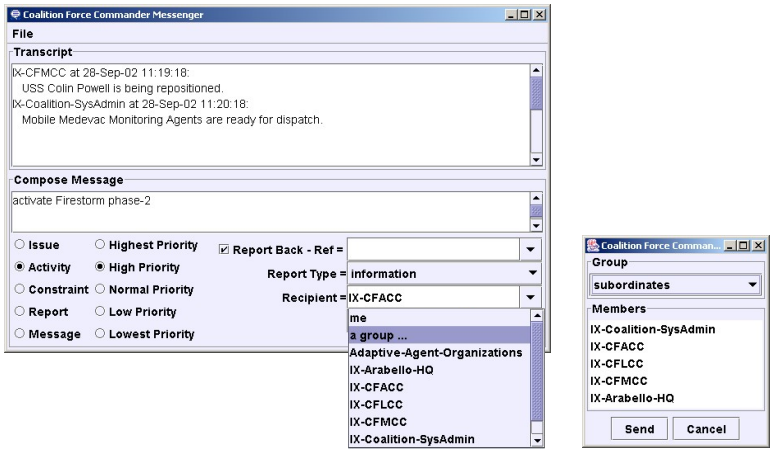
1. Undo: revert a construct to the last time it was saved to the draft domain (via Edit menu), i.e. undo all changes that have only been made in the editing panel;

2. Revert to published: revert a construct to the public version (via Edit menu), i.e. undo all changes to this construct since the domain was last published;

3. Re-load: revert the whole domain to the last time it was saved to file by opening that file via the File menu.

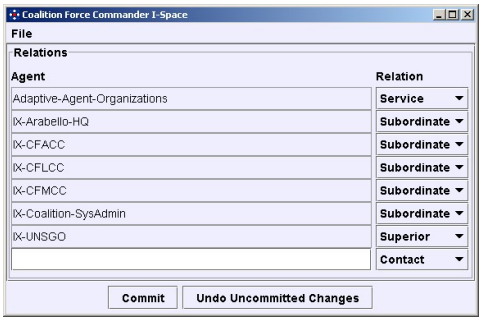
There is a fourth "revert" function for convenience: "discard changes to draft" which reverts the whole domain to the public version, i.e. undo all changes to all constructs since the domain was last published.

Using the I-X Messenger Tool



The I-X Messenger tool is used to compose and send messages to other panels and agents. It also shows any "chat" messages received form other agents (in the Transcript window). You can send messages to your own panel ("me") and there is a simple group sending facility (which will be expanded in future releases).

Using the I-Space Tool



The I-Space tool allows for the management of the organisational relationships of the current panel (referred to as "me") to other panels, agents and external services. New agent names can be added ion the type in box at the bottom. Existing agents or panels can have their relationship altered. The Commit button is used to inform the process panel of any addition or changes to relationships of existing entries. You can undo any changes made to the I-Space table that have not been committed already.

The relationships allow for the setting up appropriate "Action" menu entries for items on the panel. The relationships provided are:

Relationship	Action Menu Item
Superior	Escalate to (with report back)
Peer	Pass to (with report back)
Subordinate	Delegate to (with report back)
Service	Invoke (with report back)
Contact	None
None	None

Providing an external-capabilities description of the verbs associated with any agent can be used to selectively show the agent in the Action Item menu only for the given verbs. If no verb association is provided, it is assumed that all Superiors, peers and Subordinates can take any item (with any verb). It is expected that an external-capabilities description is given for a service or it will not appear on the menu at all.

Creating your own I-X Process Panel

A single process panel or a small cluster of panels in superior, peer or subordinate relationships to one another can be quickly adapted to a new application. We will later support more dynamic and adaptable combinations of multiple panels in more complex organisational structures (which we called I-Spaces), but much of the necessary support for this is not yet sufficiently generic to provide in an easily altered form.

An example I-P^2 application is provided in the apps\isample directory, which can be copied and adapted as follows:

- Copy the whole Isample directory to become a new directory with a name of your choice (e.g. apps\appname).
- In the directory config alter the property file names and contents of those files as you wish to tailor display names and labels used on the process panels.
- You can modify the ways in which the panel "actions" are set up using "I-Space" relationships such as superior, peer and subordinate.
- In the directory images add in any logo or logos for panels as you wish. Replacing the logo images\isample-logo.gif will mean the default logo is amended without further changes.
- Tailor a panel to a new application usually involves providing a suitable "domain model" that describes ways in which activities can be refined into more detailed sub-activities. Domain models in I-P^2 are stored in XML format (although a Lisp-oriented format is also available) and for I-Sample Process Panels are in the domain-library directory. A domain

editor (see below) is provided to create or amend domain models, and domain models can be augmented while a Process Panel is running.

- You can add appropriate “Test Menu” entries (see below) .

A wide range of parameters can be specified to simply customise a range of things about each panel. A property file for a panel can specify most of these and can be set using, e.g.,

ix.ip2.Ip2 -load config/isample-supervisor.props

For example, the file config/isample-supervisor.props contains such things as:

```
symbol-name=Supervisor
display-name=Supervisor I-X Process Panel
logo-line-1=Supervisor I-X Process Panel
logo-line-2=Based on I-X Technology
logo-image=images/isample-logo.gif
domain=isample-supervisor.xml
subordinates=Operator
```

You can also set one property individually when the process panel program is started using a command-line argument, such as

```
"-display-name=App-Name Whatever"
```

The domain model including process descriptions available to the panel can be preloaded from a domain library file (e.g. as in the case above which loads the isample-supervisor.xml file describing sample processes that the panel is made aware of and can use to “expand” entries put onto the panel.

It can be convenient to provide some example issues, activities or other entries that can be added to a panel, which could have come from other systems or panels. It can also be convenient to provide messages that could be sent to other panels and agents. This allows simple demonstrations and testing to occur. The contents of the "Test" menu can be set using an XML file describing the entries, and informing the panel about this file using the

-test-menu=<pathname>

parameter – see the appendix for details of all parameters. An example test menu file follows. “me” for the “to-name” means the message is sent to the current panel rather than externally. The menu text is what actually appears as an entry in the Test menu. The $to item in the menu-text string (if present) is substituted by the “to-name” of the panel or agent to which then message is sent.

```
<?xml version="1.0" encoding="UTF-8"?>
<list>

  <test-item
        menu-text="Send $to a request for transport"
          to-name="Supervisor">
    <contents>
      <activity priority="high"
            report-back="yes">
        <pattern>
          <list>
            <symbol>transport_by_helicopter</symbol>
            <item-var>?wounded</item-var>
            <symbol>field_hospital_a</symbol>
          </list>
        </pattern>
      </activity>
    </contents>
  </test-item>
  ...
</list>
```

More details of setting up a test menu are provided in the appendix.

You can further tailor an I-X Process Panel to a specific application by renaming and amending the I-Sample Process Panel code as follows:

- You can rename the java\isample directory to be java\appname and, in that directory, rename Isample.java to be AppName.java or whatever you wish. Delete the compiled class files included there.
- Edit this renamed file to change the package name from isample to appname.
- Change the class name Isample to AppName whereever it occurs.
- Change any strings that refer to I-Sample to App-Name as you wish.
- You can provide a customised "state" viewer for panels. A description of how to do this is in the I-X Developer Guide.
- Recompile the AppName.java code with the compile script provided.
- In the directory scripts\win (and unix) alter the script names and script contents as necessary to refer to the new name rather than the basic ix.ip2.Ip2 class or the custom isample.Isample class.

The I-X Process Panels can have a number of "issue handlers" which can handle issues in specific ways:

- **Escalate, Pass and Delegate:** One type of handling is to reroute the issue to other users or panels. At present the issue handlers are defined using information supplied in the "I-Space" description for a process panel which is currently done by specifying the superiors, peers, subordinates and contacts parameters (whether via the command line or via the property file provided on panel start up).

- **Invoke:** External agents defined as being a "Service" in the I-Space tool appear on the action menu as capabilities to address items that they are registered as being able to handle (though defining their "external-capabilities" in panel properties.

- **Connect:** Actions of forma (connect panel-a relationship) have a connect handler that sets the appropriate entries in I-Space automatically.

Creating your own I-X Domain/Process Library

Each I-X Process Panel can make use of a domain model or process library which describes ways in which issues can be handled or high level activities can be broken down into more detailed activities which may be performed. A panel can operate without such process descriptions, but becomes more useful and helpful if it has such knowledge. A process library can be loaded when a panel is started up, and additional process descriptions can be provided while it is running, and indeed they can be saved at any stage to amend the stored version for later preloading.

You can create the process descriptions with the I-X Domain and Process Editor provided. It can be run on its own or can be called from within a Process Panel from the Tools menu. Since the process descriptions are actually stored in a simple XML format, it is also possible to use any XML editor to change the descriptions if you wish. The format of the XML is as follows:

```
domain ::=
  <domain>
    <name>string</name>
    <refinements><list>refinement...</list></refinements>
  </domain>

refinement ::=
  <refinement>
    <name>string</name>
```

```
    <variables><list>variable...</list></variables>
    <pattern><list>...</list></pattern>
    <issues><list>issue...</list></issues>
    <nodes><list>node-spec...</list></nodes>
    <constraints><list>constraint...</list></constraints>
    <orderings><list>ordering...</list></orderings>
    <comments>string</comments>
  </refinement>

variable ::=
  <item-var>?name</item-var>

issue ::=
  <issue
      status="status"
      priority="priority"
      sender-id="name"
      ref="name"
      report-back="yes-no">
    <pattern><list>pattern-element...</list></pattern>
  </issue>

node-spec ::=
  <node-spec id="name">
    <pattern><list>pattern-element...</list></pattern>
  </node-spec>

constraint ::=
  <constraint
      type="name">
    <parameters><list>pattern-element...</list></parameters>
  </constraint>

ordering ::=
  <ordering>
    <from>node-end-ref</from>
    <to>node-end-ref</to>
  </ordering>

node-end-ref ::=
  <node-end-ref
      end="end"
      node="name">
  </node-end-ref>
```

end ::= begin | end

status ::= blank | complete | executing | possible | impossible | n/a

priority ::= lowest | low | normal | high | highest

yes-no ::= yes | no

pattern-element ::=
    <symbol>name</symbol> |
    <string>text</string> |
    <item-var>?name</item-var> |
    <integer>digits</integer> |
    <long>digits</long> |
    <double>...</double> |
    <list>pattern-element...</list> |
    other pattern-elements are possible

Example domain models and refinements can be found in the apps\isample\domain-library directory. Variables begin with "?" and can be used anywhere. Unbound variables appear in the process panel and can be bound by the user of the panel (and in later versions by external query capabilities). The representation used is based on the <I-N-OVA> constraint representation of activity (Tate, 1996).

Further Tailoring

An I-X Developer Guide is available with details on more ways to tailor I-X Process Panels and systems. This usually involves Java programming add-ons.

Communications Strategy

I-X Process Panels can also be used with any of a number of "Communications Strategies". Example strategies are provided for the DARPA CoABS Grid ("grid"), the University of West Florida Institute of Human and Machine Cognition (UWF/IHMC) KAoS ("kaos"), the Jabber (www.jabber.org) XML framework ("jabber"), and the UK EPSRC-sponsored Advanced Knowledge Technologies AKT Bus ("akt"). Also provided is an adaptor for a simple direct link between panels possibly supported by a simple name server (referred to as the "simple" or "xml" communications strategy). Writing a suitable Communications Strategy can provide other message transport routes. More details are available in the I-X Developer Guide.

Custom World State Viewer

It is possible to replace the simple table view used for the current world state. Viewers that show the state information clustered into the various objects or process products being handled, and giving their attributes and values in a convenient form can be provided. Graphical images of the process products can be added where required. This could include map-based data to show the position of the objects.

References

Allsopp, D., Beautement, P., Bradshaw, J.M., Carson, J., Kirton, M., Suri, N. and Tate, A. (2001) "Software Agents as Facilitators of Coherent Coalition Operations", 6th International Command and Control Research and Technology Symposium, US Naval Academy, Annapolis, Maryland, USA, 19-21 June 2001.

Allsopp, D., Beautement, P., Bradshaw, J.M., Durfee, E.H., Kirton, M., Knoblock, C.A., Suri, N., Tate, A. and Thompson, C.W. (2002) "Coalition Agents Experiment: Multi-Agent Co-operation in an International Coalition Setting", Special Issue on Knowledge Systems for Coalition Operations (KSCO), IEEE Intelligent Systems, June 2002.

Fraser, J. and Tate, A. (1995) "The Enterprise Tool Set -- An Open Enterprise Architecture", Proceedings of the Workshop on Intelligent Manufacturing Systems, International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada, August 1995.

Stader J., Moore J., Chung P., McBriar I., Ravinranathan M., Macintosh A.. (2000) "Applying Intelligent Workflow Management in the Chemicals Industries"; in "The Workflow Handbook 2001", L. Fisher (ed), Published in association with the Workflow Management Coalition (WfMC), pp 161-181, Oct 2000.

Stader J. (1996) "Results of the Enterprise Project", in Proceedings of Expert Systems '96, the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, UK, December 1996.

Tate, A. (1996) "The <I-N-OVA> Constraint Model of Plans", Proceedings of the Third International Conference on Artificial Intelligence Planning Systems, (ed. Drabble, B.), pp. 221-228, Edinburgh, UK, May 1996, AAAI Press.

Tate, A. (1998) "Roots of SPAR", in "Special Issue on Ontologies", Knowledge Engineering Review, Vol.13 (1), March 1998, Cambridge University Press.

Tate, A. (2000) "<I-N-OVA> and <I-N-CA> - Representing Plans and other Synthesized Artifacts as a Set of Constraints", AAAI-2000 Workshop on Representational Issues for Real-World Planning Systems, at the National Conference of the American Association of Artificial Intelligence (AAAI-2000), Austin, Texas, USA, August 2000.

Tate, A., Dalton, J. and Levine, J. (1998) "Generation of Multiple Qualitatively Different Plan Options", Fourth International Conference on AI Planning Systems (AIPS-98), Pittsburgh, PA, USA, June 1998.

Tate, A., Dalton, J. and Levine, J. (2000) "O-Plan: a Web-based AI Planning Agent", AAAI-2000 Intelligent Systems Demonstrator, in Proceedings of the National Conference of the American Association of Artificial Intelligence (AAAI-2000), Austin, Texas, USA, August 2000.

Tate, A., Levine, J., Dalton, J. and Nixon, A. (2002) "Task Achieving Agents on the World Wide Web", in "Creating the Semantic Web", Fensel, D., Hendler, J., Liebermann, H. and Wahlster, W. (eds.), MIT Press, 2001.

Uschold, M., King, M., Moralee, S. and Zorgios, Y. (1998) "The Enterprise Ontology", in "Special Issue on Ontologies", Knowledge Engineering Review, Vol.13(1), March, 1998, Cambridge University Press.

Annex 1: I-P^2 Parameters

IPC

-symbol-name=symbol

Note that symbol-name, by default, also becomes the ipc-name. If not provided, the default symbol-name is set to
"IX-<user-name>@<machine-name>".

-ipc-name=name

Note that ipc-name is only available for special uses. It is recommended that symbol-name is used generally to name agents, and this will by default be used for the ipc-name. Wherever agent names are required, the ipc-name is used.

-ipc=strategyName
-ipc=class

strategyName can be simple (default) or xml using built in support. With suitable communications strategy add-ons other strategies can be specified such as: grid, kaos, jabber or akt.

See the javadoc for IPC.getCommunicationStrategy(String strategyName)

Default/Simple and XML Communication Strategies

-host=hostname

Used to tell the agent what to call the machine it is running on when the default name will be incorrect. The default is the name returned by
InetAddress.getLocalHost().getHostName()

-port=number

Tells the agent to use a specific port number rather than to ask the underlying operating system to allocate a free one. This is especially useful in environments with a firewall.

-run-name-server

Tells the agent to run a name-server.

-name-server
-name-server=servername:port
-no name-server

Tells the agent whether to use a name-server to look up the addresses of other agents, and if so what host and port to connect to. The name-server servername:port defaults to localhost:5555

Jabber Communication Strategy

-jabber-server=hostname (e.g. jabber.org)
-jabber-port=number (default is 5222)
-jabber-username=username
-jabber-password=password
-jabber-resource=resource (I-X by default)

If the jabber-port number is not given, 5222 is the default as usual for Jabber servers.

Usually leave the jabber-resource at its default value. The assumption is made that all other panels with which this panel will communicate will share the same resource name. Hence, the resource name is used for two purposes:
1. to distinguish I-X panel jabber clients from non-I-X panel jabber clients, and;
2. to distinguish I-X panel clients belonging to a particular I-X 'cluster' from other I-X panels.
Caution should be applied when setting this parameter to anything other than its default value.

-jabber-presence=keyword (e.g. Online)
-jabber-allow-queuing=boolean (default false)

Can set true to allow asynchronous communications between panels that are not on-line at the same time (queuing is provided by Jabber servers). The default mode (false) will indicate a communications failure if the target recipient resource is not on-line or, if no resource is specified; the target user has no on-line resources at all. (In the latter case, if no resource is specified and the target user has an I-X Process Panel on-line, then this Panel will be the preferred destination for the message.)

IXAgent

-debug=boolean

Set to true for more detailed diagnostics in the Java console window.

-classic=boolean

Use alternative (simpler) interface for table views and other user interface elements.

-domain-editor-class=classname
Possible values are currently:
ix.iview.DomainEditor
ix.iview.SimpleDomainEditor

Select domain editor to use.

IP2 Domain Library

-domain-library=pathname (URL syntax)

-domain=filename

Initial Panel Contents (Initial Plan)

-plan=pathname (URL syntax)

An XML file can be provided to specify the initial plan or initial panel contents on start up.

IP2 Visual Appearance

-display-name=text

Note that display-name is set to "<symbol-name> Process Panel" if not explicitly set.

-logo-line-1=text
-logo-line-2=text
-logo-image=pathname (URL syntax)
-metal-theme-secondary-3=colour

-frame-size=WIDTHxHEIGHT

The initial width and height of the process panel in the form WIDTHxHEIGHT (e.g. 800x400)

-font-increment=integer

The relative font size for text, buttons and labels in I-X process Panels. E,g, 2, 4 or –2. Odd numbers (e.g. 1 or 3) may not have bold fonts installed on all systems).

IXAgent I-Space

-superiors=namelist
-subordinates=namelist

-peers=namelist
-contacts=namelist

-external-capabilities=name:verb,...

Note that namelist is a list of names of other process panels or external agent resources. A namelist is comma separated, and the list cannot contain spaces.

external-capabilities specifies the verbs associated with any panel name or external agent name. If the panel or agent is not in a defined relationship, then it is considered to be a "resource" and will only be available for the specific verbs specified. If it is already in a defined relationship (usually for other I-X Process Panels) then it is treated as a restriction specification, so the relevant action menu entries only come up for those specific verbs – rather than for any pattern verb.

Test Menu

-test-menu=pathname (URL syntax)

An XML file can be provided to set the Test menu entries that appear in the top right corner of a process panel, and which can be convenient for testing and demonstrations.

General Notes

1. Filenames and pathnames are relative to the current directory when an application is run. This is usually the root directory for an I-X application using default start up scripts (i.e., <I-X-base-directory>\apps\<app-name>\).

2. When providing command line arguments, the "-" is not part of the parameter name. It is just command line syntax.

3. -load is not a parameter. It is syntax that says to load name=value lines from a file. More than one -load may be specified.

4. -no and -not can negate the following parameter (which is written without the initial "-"). It is equivalent to giving the parameter the value "false" but can be used in cases where it would seem odd to explicitly say "=false".

Annex 2: <I-N-C-A> XML Message Formats

An I-X Process Panel can be sent a number of XML format messages from other agents or systems to give it issues to address, activities to perform and reports to note. A Test agent (called I-Test) is provided to give a simple way to try this out. The format of these messages is as follows:

```
issue ::=
  <issue
       status="status"
       priority="priority"
       sender-id="name"
       ref="name"
       report-back="yes-no">
    <pattern><list>pattern-element...</list></pattern>
  </issue>

activity ::=
  <activity
       status="status"
       priority="priority"
       sender-id="name"
       ref="name"
       report-back="yes-no">
    <pattern>pattern-element...</pattern>
  </activity>

constraint ::=
  <constraint
       type="name"
       relation="name">
    <parameters>
      <list>
        <pattern-assignment>
          <pattern>pattern element...</pattern>
          <value>pattern element...</value>
        </pattern-assignment>
      </list>
    </parameters>
  </constraint>
```

constraint types allowed at present are “world-state” and the relations for this are “condition” and “effect”. The value must be given, but a default “value” can be set to “true”.

```
report ::=
  <report
       report-type="report-type"
       priority="priority"
       sender-id="name"
       ref="name">
    <text>string</text>
  </report>

chat-message ::=
  <chat-message
       sender-id="name">
    <text>string</text>
  </chat-message>

pattern-element ::=
    <symbol>name</symbol> |
    <string>text</string> |
    <item-var>?name</item-var> |
    <integer>digits</integer> |
    <long>digits</double> |
    <double>...</integer> |
    <list>pattern-element...</list> |
    other pattern-elements are possible

report-type ::= success | failure | progress | information | event

priority ::= lowest | low | normal | high | highest

yes-no ::= yes | no

status ::= blank | complete | executing | possible | impossible | n/a
```

Notes

Strings and symbols that contain some special symbols need to have these encoded. Use "&" for ampersands, "<" for less-than, and "&gt" for greater-than.

In attribute values, double quote should be encoded as """.

It is possible to have a variable match all of the remaining elements in a list by using the special symbol "&rest" followed by an ordinary variable. I.e.,
<symbol>&rest</symbol><item-var>?name</item-var>

Annex 3: Test Menu Setup

Test-menu Files

To use a test-menu file, have a command-line arg or .props file entry test-menu=filename, e.g., ip2 -test-menu=somedir/test-sequences.xml

The file contains a list; each element describes a single entry on the test menu. In outline, the file therefore looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<list>
  ...
</list>
```

Three types of entry are allowed:

```
TEST-ITEM ::=
  <test-item
       delay-before="INT">
    <menu-text>STRING</menu-text>
    <to-name>STRING</to-name>
    <contents>SENDABLE</contents>
  </test-item>

TEST-SEQUENCE ::=
  <test-sequence>
    <menu-text>STRING</menu-text>
    <test-items><list>TEST-ITEM...</list></test-items>
  </test-sequence>

TEST-SEQUENCE-GENERATOR ::=
  <test-sequence-generator
       initial-delay="INT"
       delay-between="INT">
    <menu-text>STRING</menu-text>
    <template>TEST-ITEM</template>
    <to-names><list>...</list></to-names>
    <content-strings><list>...</list></content-strings>
  </test-sequence-generator>
```

A SENDABLE is an issue, activity, constraint, report or chat-message.

Note that string-valued fields (such as menu-text) may be written as attributes instead of as elements if the string is sufficiently simple.

Test items, sequences, and sequence-generators do not have to come from files. They are ordinary Java objects that can also be constructed in Java. However, it is often more convenient to specify them in XML.

In each of the above syntaxes, the menu-text is a string that is displayed in the "Test" menu in the top right corner of an I-X Process panel. For a test-item only, any occurrence of "$to" in the menu-text will be replaced by the value of the same test-item's to-name. (That is not done for a sequence, because the messages in a sequence might be to different destinations.)

A to-name is the symbol-name of the agent the test-item's contents should be sent to.

The delay-before in a test-item is the number of milliseconds to wait before sending the contents. Delay-before defaults to 0.

Note that within the SENDABLE test item contents you can specify a sender-id if you wish it to look like an item came from that agent or panel. The syntax for SENDABLE items is included in an earlier section.

Here is an example.

```
<test-item
    menu-text="Give $to a report-back example issue"
    to-name="me">
  <contents>
    <issue priority="high"
          report-back="yes">
      <pattern>
        <list>
          <symbol>note</symbol>
          <string>sample note text</string>
        </list>
      </pattern>
    </issue>
  </contents>
</test-item>
```

That test-item would appear in the "Test" menu as "Give me a report-back example issue" and when selected would send the panel an issue with priority=high, report-back=yes, etc.

Here is a test-item that sends a report after a delay of 4 seconds:

```
<test-item menu-text="Send $to a single report with a delay"
        to-name="me"
        delay-before="4000">
  <contents>
    <report report-type="information"
          text="Here's some information" />
  </contents>
</test-item>
```

A test-sequence contains a list of test-items. The menu-text of those items is ignored (and needn't be specified). The test-sequences' own menu-text appears in the "Test" menu.

When the test-sequence is selected from the "Test" menu, it processes the list of test-items in order. For each item, it waits for the item's delay-before milliseconds and then sends the item's contents to the agent named by the item's to-name. This allows a sequence to send messages to a variety of different destinations. By using delay-before values of zero, it is possible to get several messages to be sent (almost) at once.

Each item in a test-sequence may have a different type of contents. That makes it possible to send an issue to one agent, a report to another, and so forth.

However, in some cases, all of the items in a sequence will have certain things in common; and then it may be possible to use a test-sequence-generator.

A test-sequence-generator contains a single test-item that is used as a template. The menu-text of that item is ignored (and needn't be specified), but all other fields may be significant.

A test-sequence may contain either a list of to-names or a list of content-strings, but not both.

If it contains a list of to-names, a sequence is constructed by making a copy of the template for each of the to-names, replacing the copy's to-name each time. The resulting sequence will send essentially the same message to a series of agents

If there is a list of content-strings instead, the sequence contains one copy for each of the content-strings, with the "main contents" of the copy replaced by the corresponding content-string, suitably interpreted. This sequence will send a series of similar messages of the same type (issue, report, or whatever) to a single agent: the agent specified by the to-name of the template.

The interpretation of a content string depends on the class of the template's contents. If the template contains an issue or activity, the content string is treated as a pattern and parsed in the usual way (with ?names being variables etc); if the template contains a report or chat-message, the content-string is placed in the object's text field. Constraints are not yet supported (although they may appear in ordinary test-sequences.)

The test-items in the generated sequence have delay-before values determined as follows: If the generator specifies an initial-delay, it becomes the delay-before of the first item in the generated sequence. If the generator specifies a delay-between, it becomes the delay-before of all subsequent items in the sequence. Otherwise, the template' delay-before is preserved.

Here is an example that when selected will send a series of chat-messages:

```
  <test-sequence-generator initial-delay="0" delay-between="2000"
                  menu-text=”Send me some example chat messages”>
    <content-strings>
      <list>
        <string>Sample chat text 1</string>
        <string>Sample chat text two</string>
        <string>More chat</string>
        <string>This time there will be
several lines of text
  and maybe some indentation
  just for variety
and a last line</string>
      </list>
    </content-strings>
    <template>
      <test-item>
        <to-name>
          <string>me</string>
        </to-name>
        <contents>
          <chat-message />
        </contents>
      </test-item>
    </template>
  </test-sequence-generator>
```

Menu separators can be specified in the test-menu file. The XML syntax is

```
  <test-separator />
```

Just include it in the list between the test items you want to separate.

Using the "Test" Menu

Entries in the "Test" menu do not have to send messages. They don't even need to involve any of the objects described here. However, this section will describe only the cases that can be specified by a test-menu file.

If a single message is to be sent, without a delay, it is sent as soon as its "Test"-menu entry is selected. That is so regardless of whether it came from a single test-item or from a 1-item sequence.

Otherwise, a new thread is created to supervise the message sending. While that thread is running, the corresponding menu entry is prefixed by "Stop: " and, if selected, stops the thread. When the thread terminates, the entry reverts to its original form.

However, all messages are actually sent by the GUI event thread just as in the normal operation of a panel). The other thread exists only to control the timing.

Messages sent to "me" are given directly to the panel rather than going via the communication strategy.

Appendix E: I-X Systems Architecture

J. Dalton and A. Tate

Abstract: The I-X architecture is a system integration architecture that has close affinities with model-view-controller architectures and with blackboard architectures. It is based on the earlier O-Plan architecture and in its various forms has been used for planners, schedulers, discrete event simulators, and process-monitoring and control agents.

This paper is a high-level technical overview of the architecture, with examples drawn from its use in I-X Process Panels (I-P2s) and, earlier, in O-Plan.

Citation: Unpublished.

Principal Components

The main parts of an I-X agent are shown below, with indentation to show where a component is attached. For example, the controller has issue and activity handlers; the model manager has a constraint associator which, in turn, contains constraint managers.

- I-X agent
  - Controller
    - Issue Handlers
    - Activity Handlers
  - Model Manager
    - Constraint Associator
      - Constraint Managers
  - Domain Model
  - Viewers
  - I-Space Manager
  - Tool Manager
    - Domain Editor
    - I-Space tool
    - Messenger
  - Communication Strategy

Simpler agents, which omit or simplify components, are possible. A model manager might not use a constraint-associator, and the constraint managers might not be separate entities. A very simple agent might not have a model-manager at all. Or an agent might not have a GUI, and so would not contain viewers.

Moreover, I-X is an abstract architecture that does not specify any particular mapping onto programming language constructs. In an object-oriented language, the principal I-X entities, such as controllers, handlers, and constraint-managers will tend to be implemented as objects, but that it not the only possibility. For example, an issue-handler might just be a single method. Although the architecture is thus very flexible, it still embodies organizational principles that have proven useful for a variety of different tasks. An important feature is that the architecture provides "plug-in" points - chiefly in the controller and constraint associator, but also for viewers, tools, etc. This makes it relatively easy to construct a variety of different agents from a toolkit of parts, and it provides a natural way to express the capabilities of an agent: what types of issues, activities, and constraints can it handle?

Here is a brief account of the components' roles; some are later considered in more detail.

An I-X agent is meant to carry out the activities it has been given, address the issues, and respect any constraints. The controller is responsible for how it does this, at the top level. The controller can call on issue and activity handlers as "knowledge sources" that know how to process some, or all, types of issues or activities. Issues, activities, and constraints may be send to the agent from external sources or be created by the issue and activity handlers.

The I-X architecture has been used primarily for synthesis tasks, broadly defined. Then the model manager handles the constraints that define the product or products being created by the agent. The "model" is the set of constraints, plus any issues that relate to the products. To a planner, for instance, the product is a plan. Constraints might require that certain actions be included in the plan, impose a partial order on those actions, state pre- and post-conditions, and so on. Issues might identify parts of the plan that needed more work, for instance that certain actions still needed to be expanded or that some pre-condition had not yet been satisfied.

The model manager may maintain many different versions of its contents, because the agent is performing a search, or so that different possibilities may be compared. Some, at least, of these versions may be referred to in the agent's user interface, or by other agents, as "options".

The constraint associator is a subcomponent of the model manager responsible for dispatching constraints to appropriate managers. Constraint managers, in turn, register with the associator to declare their abilities. A constraint manager might handle a wide range of constraints, or only something very specific such as temporal "before" constraints.

The domain model provides information about the problem domain. For example, in a planner it could contain descriptions of action breakdowns into subactions and of pre- and post-conditions. Viewers are the user interface to the model manager and controller and may also act as controls. In an I-X process panel, for example, the viewers allow the user to select how issues and activities should be handled. Another viewer allows the user to inspect issues and activities, add annotations, and expand activities into subactivities.

The I-Space manager handles information about other agents and their relationships to its own agent - their capabilities, groupings, superior-subordinate status, etc.

"Tools" are another set of components with user interfaces, such as the domain editor (for creating, viewing, and modifying domain models), the messenger (which allows the user to compose and send messages), and an I-Space tool (to inspect and modify information held by the I-Space manager).

The communication strategy is responsible for sending and receiving messages and for the way objects are represented in messages.

Example 1: O-Plan

O-Plan is a Hierarchical Task Network (HTN) planner with an architecture that can be seen as an early version of I-X. Here O-Plan will be described in I-X terms. The I-Plan planner will be very similar at this level, so that this section can also be seen as a preliminary description of I-Plan.

The O-Plan controller manages issue-handling for both the agent and the model. The agent-level issues are used to set up planning problems, request plans, check plans, and so on. (In more

recent systems, these might be activities rather than issues.) In the model, the issues are about steps needed to complete the plan, such as expanding an action into subactions or binding a variable. Issues have patterns that begin with verbs, for example "set_task <task name>", "check_plan", or "expand <node tag>".

The O-Plan controller selects an issue handler in a very simple way: there is exactly one hander for any verb. The controller's algorithm is to repeatedly pick the highest priority issue, regardless of whether it is an agent or model issue, and invoke the corresponding handler, until no issues remain, at which point it waits. New issues may be sent to O-Plan at any time, which will revive this process if it is waiting. (There are some complications, to let O-Plan handle some things more efficiently, and to give the user more control over the process, but they will not be described here.)

O-Plan's model manager maintains a tree of plan states, or partial plans, which consist of a constraints and issues. An agent-level "set_task" issue is used to start the planning process. It adds an "expand_task" issue to the model. The handlers for plan issues can add constraints or issues to the model. Whenever one finds something that can be done in more than one way, it creates an "alternative" (or backtracking "choice point") at the current plan state, picks one of the possibilities it is considering, and records the rest in the alternative. It then creates a child of the current plan state and applies its chosen possibility in that state. Whenever the current state becomes inconsistent, the issue handler that determines this posts a "poison" issue whose handler picks one of the available choice points, returns the model manager to that place in the three, and reposts the issue whose handler created that choice point. The handler will pick up the information it saved about the remaining possibilities and continue as before.

The constraint managers are responsible for checking that the model is consistent and for propagating information derived from the constraints they manage. (For example, if action A1 must complete before action A2 can begin, and A2 must complete before A3 can begin, the handler for temporal "before" constraints can determine that A1 is also before A3.) As with the selection of issue handlers, the manager for a constraint is determined in a simple way: there is exactly one manager for each type of constraint. The constraint associator gives the constraint to the appropriate handler or, if there is not handler for that constraint type, to a special handler that holds "other" constraints.

New issue handlers and constraint managers can be defined in a plug-in fashion, by registering them with the controller or constraint associator respectively.

Plan-viewers may also be plugged in. For example, one was recently added to output plans in the XML format used by I-X agents. In addition, completely different user interfaces have been defined, such as ones that use HTML that can be displayed in a web browser.

Example 2: I-P2

I-X Process Panels are a more complex case than they may appear to be, because the panel is an agent carrying out the process represented by its list of activities, a planner developing that process, and a simulation engine determining when activities are ready to execute by comparing their preconditions against the known state of the world. These aspects all meet in the panel's model manager; however, the planning and simulation aspects are relatively undeveloped and will not be described in any detail.

Since a process panel is for the most part controlled by the user, most handlers to not directly or automatically handle issues or activities. Instead, they produce "handler actions" that are attached such items. These actions provide short descriptions of their function which are displayed in menus by the issue and activity viewers. It's only when the user selects an action that anything actually happens.

Since new actions may become possible (for instance when the panel discovers a new capability of another agent, so that an activity might be delegated to that agent when it couldn't be before, or when the user defines a new way to expand an activity into subactivities), and existing actions may become impossible (as when the user deletes an expansion possibility), handlers must be able to revise the action lists in response to changing circumstances.

In addition, many ways of handing - such as delegating to another agent - may apply to a wide range of issues and activities. It therefore makes no sense to have a simple mapping from verbs to handlers as in O-Plan. Instead, all handlers get a chance to look at items and decide what, if any, handler actions they want to provide.

Nor are handler actions passive entities. They are able to determine whether they are still valid (or should be removed) and whether they are ready to be selected or are waiting for some condition to become true. When an action is selected by the user, it's "handle" method is called. It may perform the action itself, or it may ask the issue or activity handler to do it.

While those aspects of the system are more complex than in O-Plan, the model manager is at present much simpler. There is no constraint associator and no separate constraint managers; the model manager handles all of that itself. Moreover, the manager does not maintain multiple versions of the model and does have to support search and backtracking.

Constraint Managers and Critical Constraints

A constraint manager can return one of three results: yes, no, and maybe. A yes means that the constraint can be added; a no means that it cannot, because it would be inconsistent with one or more constraints already there. The maybe response is more complex. It essentially means that the constraint can be added, but certain other constraints must be added as well. Those additional constraints may not be a simple list; they may instead be a tree, with the planner required to find a path through the tree to a leaf, adding all the constraints it finds along the way.

If an inconsistency is found, it must switch to another path. If it does find a viable path, the other paths that have not yet been tried must be saved in a backtracking "alternative" as described in the O-Plan section above. Since only issue handlers, not constraint managers, are allowed to create alternatives, any tree received from a constraint manager must be placed in an issue for later processing if the handler that received the tree cannot process it itself.)

The operation of a constraint manager and the function of "critical constraints" will be described with the aid of an example.

In O-Plan, one of the constraint managers handles a simple kind of resource constraint that has the syntax "use <item>" where the <item> might be the name of an object or a variable. If, for example, the constraint "use drill" were applied to an action, no other action could use the drill (ie have a "use drill" constraint applied to it) at the same time.

Suppose the constraint manager for "use" constraints were given the constraint

C1: use drill from begin_of action_1 to end_of action_1

If that was its only constraint, it would return "yes".

Now suppose it then received

C2: use bandsaw from begin_of action_2 to end_of action_2

That does not conflict with C1 and so also gets a "yes".

The next constraint might be

C3 use drill from begin_of action_2 to end_of action_2

Now there is a potential conflict. There are two ways to resolve it. Either action_1 must end before action_2 begins, or action_2 must end before action_1 begins. The result is a "maybe" with a two-branch, one-level tree.

Note how it is natural to express this conflict-resolving result in terms of temporal "before" constraints. Constraints that have this role are called "critical constraints". Different types of synthesis tasks may have different types of critical constraints.

Now consider the constraint

C4: use ?tool from begin of action_3 to end_of action_3

This does not specify any particular tool, but in O-Plan variables have enumerated domains, and so the possible values of that variable are known. Presumably they include "drill" and "bandsaw". Thus, there might be a conflict with any of the earlier constraints.

If "before" constraints are the only sort of temporal constraint available, the tree for resolving such conflicts will be rather complex, because there must be a path for each different way of resolving all the conflicts, and for each conflict there two ways to resolve it with "before". If, on the other hand, a "does not overlap" constraint is available, the tree can be simpler.

There is also another way of resolving the conflicts: if ?tool has a value other than "drill", it cannot conflict with C1 or C3; and if it has a value other than "bandsaw", it cannot conflict with C2. This the second type of critical constraint for planning: constraints about the values of variables.

Having more ways to resolve conflicts is good in some ways but bad in another, because it can make the trees in "maybe" responses larger. However, clever constraint managers can take steps to simplify the trees. For instance, some temporal constraints already in the model may eliminate some conflicts, perhaps even allowing a plain "yes" response. Or the constraint manager might notice that C2 and C3 are both about action_2, or that whatever value ?tool is given is bound to be different from one of drill or bandsaw.

Communication Strategies

The communication strategy is responsible for the details of communication with other agents: for the way data is represented and transmitted, and for the way agent names are mapped onto whatever lower-level entities are required. [Footnote: The term "strategy" is perhaps too grand, but it was chosen because "method" is too strongly associated with its meaning in object-oriented programming.]

From the point of view of an I-X agent, it is able to send and receive objects - the same sorts of objects that it normally manipulates, such as issues, activities, constraints, plans, and collections of objects, such as lists, sets, and maps. The agent knows nothing about how objects are encoded and decoded for communication. The external representation is typically some form of XML, but it can also be something else, such as Java's object serialization. Nor does the agent know anything about the technology used to send and receive messages.

By confining this knowledge to the communication strategy, it becomes possible to use the same agents with a variety of different strategies and to avoid becoming tied to any particular approach that may change or become obsolete. This is especially important at a time of rapid and uneven development, such as we're now seeing for XML and agent technologies, and when different projects may be committed to different choices.

Two simple strategies are built-in to make it easy to create and test new agents when more elaborate communication mechanisms are not available or are too much work to use or set up.

Both use a server that maps agent names to host:port addresses. One encodes objects using Java serialization; the other represents objects in XML that is then serialized as a Java String. Other strategies can be plugged-in when an agent is run. At present, there are strategies for the CoABS Grid, KAoS (implemented as a subclass of the Grid strategy), Jabber, and AKTbus.

A strategy must implement only two methods: one that sends an object to a specified destination, and one that tells it to make its agent able to send and receive, by performing whatever registration and setup tasks are necessary. However, it can also affect the agent in certain other ways, such as setting its name (if the name must be established during the agent-registration process) or by adding a "tool" that provides a GUI to data held by the strategy (such as, say, an address book).

Object representation and XML

In order for agents to work with objects, leaving the external representations to communication strategies, it must be reasonably easy to translate between objects and external representations. This is an area of active development, with terms such as "serialization", "marshalling", "unmarshalling", and "data binding" now in common use.

To facilitate such translation, and to avoid being tied to any particular technology of that sort, we have adopted some (JavaBean-like) conventions for the definitions of classes whose instances are meant to me communicable (such as having zero-argument constructors and "get" and "set" methods that correspond to fields) and have defined a straightforward mapping between such objects and XML that can be used when other mappings are not required.

The mapping is in two parts which can be replaced independently. One determines the "syntax" of classes such as which of their fields should be visible; the other uses that information to translate objects to and from XML. Unlike some other approaches, this does not generate class definitions from XML definitions such as XML Schemas. Instead, it uses Java's reflective capabilities to extract information from the class definitions. The "class syntax" used by the XML translator is also used to manipulate objects in other ways, such as deep copying and "walking" structures to extract or change information, and to inform a "tree editor" that can be used view and modify both objects and the corresponding XML. There is also a utility that generates BNF-like syntax definitions, such as the following for Issue and related classes:

```
ISSUE ::= <issue
            status="STATUS"
            priority="PRIORITY"
            sender-id="NAME"
            ref="NAME"
            report-back="YES-NO">
          <annotations><map>...</map></annotations>
          <pattern><list>...</list></pattern>
        </issue>
```

STATUS ::= blank | complete | executing | possible | impossible | n/a
PRIORITY ::= lowest | low | normal | high | highest
YES-NO ::= yes | no

Communication strategies are free to use other mappings between objects and XML if they desire.

Future Directions

Much of the I-X code is still in a relatively early phase of its development in which certain aspects of the overall structure, as well as many of the details, are still being worked out. The aim is to provide a "kit" containing reusable components at the level of controllers, issue-handlers, model-managers, constraint-managers, viewers, and so on; but often they will first be developed in the context of specific I-X systems and then be "promoted" into the kit when their appropriate general form becomes clear. The kit will also evolve as work on new components suggests better implementation techniques and better forms of organization.

It is likely that I-Plan will be the next major agent to be developed. It is intended to be similar to a simpler O-Plan and to be suitable for embedding in other I-X systems. I-Plan, and other systems whose model managers support multiple versions of the model, may use an already implemented context mechanism based on the one in O-Plan. It allows individual fields in objects to contain different values in different contexts, with the contexts forming a tree in which contexts inherit values from their ancestors. (It is possible to extend the same mechanism to allow multiple-inheritance, but we do not anticipate needing that capability.) Since the same data structures can be used in systems that always use only one context, the presence of the context mechanism allows us to define define data structures of greater generality and hence reusability.

I-Space will also see active development, and this will lead to more sophisticated ways to working with groups of agents on shared issues and activities.

Intentionally Blank

Appendix F: Enterprise Modelling, <I-N-C-A>, and the I-DE Domain Editor

J. Stader and A. Tate

Abstract: Enterprise Modelling is the art of capturing and modelling the information about an enterprise that is relevant for supporting the running of that enterprise, covering as many aspects of the enterprise as required. Enterprise Modelling support should ensure that different information capture techniques and notations can be used. It must not be necessary for the models to be complete and it must be possible (and easy) to change and update the models. Models should also be used to their full capacity to support the running of the organisation. I-DE is an Enterprise Modelling tool based on AIAI's I-X Technology. I-DE uses a generic modelling framework, <I-N-C-A>, and specialises it to provide easy to use and effective editing of process (or activity) models suitable for use in a workflow environment.

Citation: Stader, J. and Tate, A. (2003) "Enterprise Modelling, <I-N-C-A> and the I-DE Domain Editor", Version 2.4, AIAI, University of Edinburgh. Unpublished. Available at http://i-x.info

Introduction - Enterprise Modelling and <I-N-C-A>

Enterprise Modelling is the art of capturing and modelling the information about an enterprise that is relevant for supporting the running of that enterprise, covering as many aspects of the enterprise as required.

In the first instance, it has to be decided which information is relevant, and that information has to be captured. The information captured must be both accurate and open to change. Arguably, capturing information is the hardest part of Enterprise Modelling. The modeller needs good capture techniques and suitable notations so that the modeller can easily see what has been modelled and can communicate this to others. There are today many different techniques and notations to support an enterprise modeller in capturing information, many of them informal [5, 8, 7, 9, 15, 2]. There are good reasons for this proliferation of techniques and notations: the modeller needs all the support available, and the better the technique and notations suit the modeller and the aspect of the enterprise that is being modelled, the more effective the modeller can be and the better she can understand and communicate the information. This brings us to the first requirement for enterprise modelling support: any realistic enterprise modelling support will have to be able to provide and cope with different techniques for capturing information, and with different notations (or views) for the information.

Looking seriously at Enterprise Modelling it quickly becomes clear that it is impossible to ever say "this model is finished". We believe that it is impossible to model all aspects of an enterprise accurately and in sufficient detail so that the models truly reflect all aspects of the enterprise [4, 3]. Even if it were possible, the world within and outside an enterprise does not stand still, so changes will always have to be incorporated into the model. Enterprise Modelling efforts can be likened to the painting of the Forth Rail Bridge, a large and intricate metal structure near Edinburgh. This bridge is painted periodically to prevent corrosion of the structure. The painters start at one end of the bridge and work their way to the other end. As soon as the painters reach the other end, it is time to paint the first end again. I.e. it is a continuous job. Our second and third requirements then are that it must not be necessary for the models to be complete (we must be able to cope with incomplete information and we should make use of all the information that we have), and it must be possible (and easy) to change and update the models.

Once enterprise information has been captured, it should be used to support the running of the organisation. How much support the models can provide depends on their quality and their form. If the models are available in paper form (printed documents of diagrams and descriptions), they can be used for documentation and communication ("this is what we do"). This can be useful for stating best practice, for teaching and training purposes, etc. However, paper models are not easy to change and their availability is not great. If the models are available on-line in the form of documents, they are easier to change and (in most organisations) more readily available. However, since Enterprise Modelling is such a difficult job, we should be able to base more support on the models rather than just using them as documentation. The models should be used to support the running of the enterprise much more directly. This usually makes more demands

on the already difficult task of modelling: more information has to be included in the models and the models need to become more formalised. However, the benefits can be significant and are usually well worth the effort. For example, process models and related information can become active in workflow systems and thus directly support the running of business processes [11, 12, 6, 1], other models can be used for skills management [10] and more generally knowledge management. This kind of support puts an organisation in a good position to quickly react to change.

In summary, the high-level requirements relevant to Enterprise Modelling are:

- any realistic enterprise modelling support will have to be able to provide and cope with different techniques for capturing information, and with different notations (or views) for the information;
- it must not be necessary for the models to be complete (we must be able to cope with incomplete information and we should make use of all the information that we have);
- it must be possible (and easy) to change and update the models;
- models should be used to their full capacity to support the running of the organisation.

The work described in this paper starts to cover those requirements using the <I-N-C-A> constraints model.

The Models

In this section we first give a brief overview of <I-N-C-A>, before describing the structures that can be viewed and edited with the help of I-DE.

The <I-N-C-A> constraints model

The <I-N-C-A> (Issues - Nodes – Constraints - Annotations) constraint approach to model specifications [14] can be used to describe any synthesised artefact, e.g. results, models, plans, configurations, designs, etc. An <I-N-C-A> specification defines a set of "nodes" to be included in the design, along with "constraints" on how these nodes can be related to one another and the environment they exist in. It also includes a set of outstanding "issues" and “annotations” related to the artefact(s). By having a clear description of the different components within a synthesised artefact, the model allows for them to be manipulated and used separately from the environments in which they are generated.

At various stages of the development of the I-X research the typography for rendering <I-N-C-A> has varied as the components have received clarification. <I-N-CA> originally stood for Issues, Node, Critical and Auxiliary Constraints. The aspect of separating critical (shared communications) constraints from auxiliary (separately managed) constraints is still important within the I-X architecture, but is now considered all part of managing the "C" (constraints) component of a model. The annotations were always present in the ontology and can be attached

to all components, but the top-level entity annotations capturing the rationale behind the synthesised product or the process/plan being described has required more prominence as the work has continued and as mixed-initiative and human communications aspects have become more important. Hence, the rendering <I-N-C-A> with the extra hyphen now stands for Issues, Nodes, Constraints and Annotations.

The issues in the specifications state the outstanding items to be handled and can represent unsatisfied objectives, problems which analysis has shown need to be addressed, etc. The I constraints can be thought of as implying further constraints which may have to be added into the design in future in order to address the outstanding issues.

The nodes in the specifications describe components that are to be included in the design. Nodes can themselves be artefacts that can have their own design(s) associated with them.

The constraints restrict the relationships between the nodes to describe only those artefacts within the design space which meet the requirements. The constraints are split into "critical constraints" and "auxiliary constraints" depending on whether some constraint managers (solvers) can return them as "maybe" answers to indicate that the constraint being added to the model is okay so long as other critical constraints are imposed. The maybe answer is returned as a disjunction of conjunctions of critical constraints.

Finally, annotations can be added which describe the rationale behind design choices and other useful information.

The choice of which constraints are considered critical is itself a decision for an application of I-X and I-Core. It is not pre-determined for all applications. A temporal activity-based planner would normally have objects/variable constraints (equality and inequality of objects) and some temporal constraints (maybe just the simple before(time-point1, time-point-2) constraint) as the critical constraints. But, in a 3D design or a configuration application object/variable and some other critical constraints (possibly spatial constraints) might be chosen. It depends on the nature of what is communicated between constraint managers in the application of the architecture.

The types of constraints in an <I-N-C-A> model are usually specialised to a great level of detail. For example, in a process model they might be as shown below:

I - Issue constraints
    - which may add an "include node" constraint (i.e. add nodes to the model)
    - which will not add an "include node" constraint
N - Node Constraints
    - "include node" constraints
    - other node constraints
C - Constraints; E.g., if the artefact is an activity plan:
    O - Ordering constraints
    V - Variable and other constraints

X - eXtra constraints (such as):
- Authority Constraints
- World State Constraints
- Resource Constraints
- Spatial Constraints
- Miscellaneous Constraints

A - Annotations; E.g., information about a graphical layout of the nodes in a GUI.

In our generic, conceptual base model, an <I-N-C-A> construct has 5 components:

- name: an identifier for the construct
- nodes: a list of nodes that are part of the construct
- issues: a set of issues related to the use of the construct
- constraints: a set of constraints that apply to the construct and its nodes
- annotations: comments and other useful information about the construct.

I-DE Specifications

All main construct specifications within I-DE conceptually follow the <I-N-C-A> model, except for the grammar and the lexicon. I-DE specifications make use of specialisations of <I-N-C-A> for constraints regularly used in process models and to add human-readable "comments" as part of their annotations. The keywords (and their specialisations) currently used by I-DE specifications are *issue, node, constraint (temporal, before, world-state, condition* and *effect) and annotation (comments).*

Domain

The Domain is a coherent set of specifications. The domain itself has a name and it can have issues, constraints, and annotations. Its nodes consist of the activity specifications and the activity relatable objects that are defined within the domain. A domain can be loaded from files, saved to files, published, and reverted to a previous version.

Activity Specifications

Activity specifications follow the <I-N-C-A> model, but with the following specialisations:

- They have a pattern that describes the activity performed (conventionally starting with a verb).
- Nodes are sub-activities that are specified by giving their activity pattern. Each node has two node-ends: begin and end which are time points that can be referred to within constraints.
- Constraints take the form of type, sub-type, and other parameters such as node-end time point reference(s), "pattern = value", etc. The types and sub-types are keywords that can be shared between applications. There are three specific constraint types that are currently supported further. These are:

1. Orderings, which are precedence relationships between node ends. An Ordering has the following specification: type is "temporal", sub-type is "before", and node references are node-end1 and node-end2. There is no pattern.
2. Conditions, which are descriptions of world state that have to be fulfilled before the activity can be considered for execution. A Condition has the following specification: type is "world state", sub-type is "condition", and node reference must currently be set to "self" but should in future also allow the activity's sub-nodes. Any form of pattern = value is allowed.
3. Effects, which are descriptions of world state that will be true when the activity has been performed. An Effect looks just like a Condition, except for its sub-type, which is "effect".

Activity Relatable Objects

These are currently not implemented and cannot yet be specified in I-DE.

Grammar and Lexicon

Currently this is built automatically to reflect the patterns used in issues, nodes and constraints. It is envisaged that this will eventually allow for managed grammars and lexicons to allow active assistance in modelling and provide active help in maintaining coherence of models.

The Domain Editor (I-DE)

I-DE is based on I-X Technology [13] from AIAI at the University of Edinburgh. The main window of the Domain Editor (the frame) contains several editor panels for editing different aspects (or constructs) of the domain. Currently the editors available are

- the Global Domain Editor, which contain information about the domain itself (e.g. the domain name);
- the Activity Editor, which edits “refinements” which contain information about activities and how they break down into sub-activities;
- the Grammar Editor, which currently only shows the patterns that are in use in the domain.

An editor panel may itself have different "views" that are used to display and edit the panel's constructs. The Activity Editor has three such views:

1. Minimal View: a simplified version of the activity and its refinement. The main simplification is that no constraints are shown

2. Comprehensive View: a view that can display and edit all of an activity's specification

3. Graphical View: a graphical view that uses nodes and arcs to show an activity's sub-activities and the temporal relationships between them.

The Domain Editor Window

This window provides access to most functions of the overall domain editor via its menu bar, and access to the most commonly used functions via its tool bar. The window can display in one of three styles: simple, tabbed, and card style. The style can be changed using the preferences editor via the View menu.
Figure 1 shows the window in simple mode with the activity editor showing.

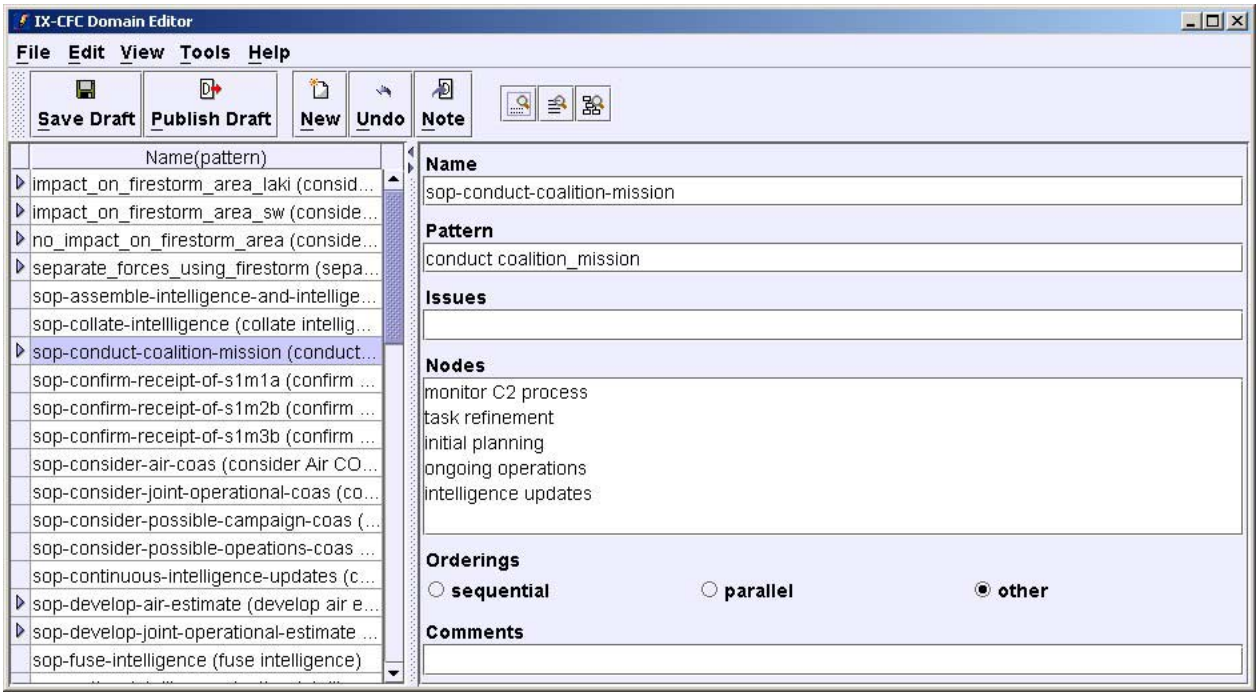


Figure 1: I-DE Window (in simple mode)

The Menu Bar

The menu bar has 5 standard menus:

1. File for closing the Domain Editor, for file access (open/save), publishing, and reverting. All functions here manipulate the domain as a whole, not individual constructs;
2. Edit for manipulating the current construct, i.e. the construct that is currently shown in the Domain Editor's panel. Many of the edit menu options are common to all types of panels (new, modify draft, revert, ...) but there are also more specific options that are only available in particular panels;
3. View for changing the visual set-up of the window, for changing which panel is shown in the window, for changing which view is shown in that panel (if applicable), for changing panel styles, etc.;
4. Tools for additional support like consistency checks etc. This menu also gives access to the preferences editor;

5. Help for access to this manual, other help, and information about the application.

The Tool Bar

The tool bar provides access to the most commonly used functions via buttons. All these functions are also available via the menu bar (in most cases, the image on the toolbar button is shown in the menu next to the corresponding menu item. Moving the mouse over a toolbar button will, after a while, display a "tool tip text" that gives a brief explanation of the button's function. The buttons themselves can either show just an icon or an icon with a short text underneath (determined by user preferences). The toolbar can be switched on and off via the View menu.

Working with the Domain Editor

This section gives an overview of how to work with I-DE. Construct specific editing (e.g. Activity editing) is described separately in more detail. The issues covered here are update levels, workflow, and preferences.

The Domain Editor maintains different levels of updates. The original domain model that the editor is started with is considered a public domain model, which other applications may be using for their own purposes (e.g. within an I-X Process Panel). This public domain model is kept as it is unless an updated model or a replacement model is explicitly "published" by the Domain Editor's user. (Note that this is true whether the Domain Editor is used in stand-alone mode or as part of another application). There is also a "draft domain model" which is the one that is being edited. The Domain Editor keeps track of any changes that are made to the draft domain model so that updates to the original domain model can be made explicitly.

Saving and Reverting

There are 3 levels of saving:

1. When a construct has been edited in the Domain Editor Panel, initially these changes may be made only in the panel itself, not in the domain construct that is being edited. In order to transfer changes from the panel into the construct in the draft domain, the user has to modify the draft, i.e. note the changes into the draft domain via the toolbar button or the Edit menu. When the user has edited a construct and not modified the draft, the system will prompt the user to note or discard changes at suitable points in the editing process: if the user decides to switch constructs, views, or panels, or if the user decides to save or publish the draft domain.
2. Modifying the draft (noting changes) does not save to file, so the next level of saving is to save the draft domain to file. As with all editing applications, it is recommended to do this frequently to ensure that work is not lost. Saving the draft domain to file will write the whole domain with all its constructs into a file in XML format. This can later be loaded into the Domain Editor for further editing, or it can be accessed by other applications.

3. The underlying public domain is not changed by any of the above (simple editing, modifying draft, or saving the draft domain to file). The only way to update the public domain is to publish the draft domain via the toolbar button or the File menu. When this happens, all pending changes are transferred to the original domain and these changes will be seen by any application that has registered as listeners to this domain. Note that publishing is always done for a whole domain, not for individual constructs. Note also that publishing a domain will not save it to file, but the same effect can be achieved by saving the draft domain to file just before or straight after publishing. At that point the draft domain and the public domain can be represented by the same XML structures. It is a good idea to publish from time to time even if the Domain Editor is running stand-alone because it will make the editor more efficient.

While there is no "undo" function that undoes individual editing steps or editing of individual fields, the following undo functions are available, corresponding to the 3 levels of saving:

1. Undo: revert a construct to the last time it was modified in the draft domain (via Edit menu), i.e. undo all changes that have only been made in the editing panel;
2. Revert to published: revert a construct to the public version (via Edit menu), i.e. undo all changes to this construct since the domain was last published;
3. Re-load: revert the whole domain to the last time it was saved to file by opening that file via the File menu.

Note that the first two undo functions apply to an individual construct, while the third applies to the domain as a whole. There is a fourth "revert" function that also applies to the domain as a whole: "discard changes to draft" which reverts the whole domain to the public version, i.e. undo all changes to all constructs since the domain was last published.

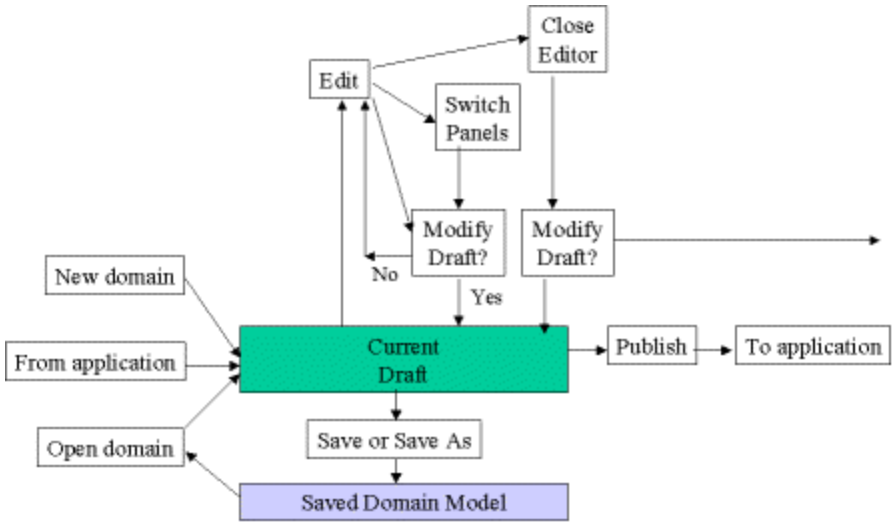


Figure 2: I-DE Workflow

The diagram shows the different states that the system may be in during an I-DE session, and it shows how a user can move between these states. After editing is done, the user may explicitly choose to modify the draft domain model. Also, when a user switches panels (such as moving between the activity and domain panels), the user is given the option to modify the draft domain model or discard any editing that has been done. A modified draft domain model can be saved to file and/or published to be visible to the application (the I-X Process Panel) or both. When the domain editor is closed, the user is given the option to save the current draft to file and/or to publish the current draft to be visible to the application.

Preferences

There are several preferences and two modes of use that the user can choose from. There are default preferences for initial use of I-DE, but the user can change the preferences using a simple preferences editor. The preferences editor allows the user to change preferences, to apply the current preferences to the I-DE they are currently working with, to save preferences so that the editor starts up with the saved preferences, and to revert to previously saved preferences. Below, we describe the two modes of use and the preferences that are under the user's control.

The two modes of use are:

1. simple mode: a cut-down version of I-DE that shows only the essential features needed to quickly put together simple process models. Other user preferences are restricted in this mode, i.e. it can be seen as a quick way to set all preferences to the simplest option. Explicitly changing any of the restricted options will override the simple mode restrictions and result in advanced mode to be set.
2. advanced mode: the full version of I-DE that gives the user full control over preferences and access to all editing facilities.

If the user switches from advanced to simple mode, most preferences will be set and restricted. However their previous settings are kept and when the user switches back to advanced mode all preferences will go back to their previous values.

The preferences that are under the user's control cover the editor in general, sub-editors, and views. These include:

- mode of use: a flag that shows whether simple mode is set;
- button texts: a flag that determines whether text is shown on underneath the icons of toolbar buttons;
- lists as text: whenever there is a list of specifications that can be edited by the user (e.g. activity nodes), the user can choose whether to view these as a list and edit them with special-purpose dialogue-style editors, or to view them as lines of text that can be typed into directly.
- editable: to use the editor as a read-only viewer, its editing facilities can be switched off to ensure that no unintended updates are made. This is not yet implemented.
- panel style: the choice of panel style to use. The panel styles available are minimal (no visual queue for changing panels), tabbed (a tab is shown at the top of the panels that can be used to

switch panel), and card (a choice-box is displayed above the panel that can be used to change panel). Note that it is always possible to change panels by using the Windows option of the View menu.

- view: the view to use. The views available depend on the panel. For example, for the activity panel, the views available are minimal, comprehensive, and graphical.

There are additional choices that affect the display of constraints in the comprehensive view of the activity sub-editor. The three groups of constraints that can be displayed are "orderings", "conditions/effects", and the generic "other constraints". The user can choose to suppress the display of some of these; for example, if only orderings are of interest, all other constraints can be switched off. Note that, whenever the generic "other constraints" are displayed, all available constraints are shown. For example, if only conditions/effects is switched off, conditions and effects will be shown in their generic form under "other constraints". Constraint-related preferences are only available for the comprehensive view - the minimal and graphical view have their own, special purpose way of displaying constraints.

In simple mode, the following preferences are restricted as follows:

- panel style is set to minimal
- lists are displayed as text
- activity view is set to minimal
- only minimal information about orderings are displayed, so all preferences relating to constraints are disabled.

Other preferences are not affected by the choice of mode.

Construct Editing

Each construct editor is responsible only for manipulating the specification of the construct. In the Global Domain Editor Panel this is less intuitive than for the other panels: this editor only considers domain details as part of its editing remit (mainly name and comments), not the constructs within the domain. The only way to manipulate the domain as a whole is via the options in the File menu, and these are available for all panels.

I-DE provides several functions for all construct editors. The implementation of these functions may vary between different construct types, but they should all be available. These common functions are:

- new: create a new construct of this type;
- copy: make a new construct that holds the same details as the current one;
- delete: delete the current construct;
- edit: select a construct to be edited;
- modify draft: save the changes made in the panel into the draft domain;
- revert: revert the construct to the last time it was modified in the draft;
- check: check the consistency within the current construct.

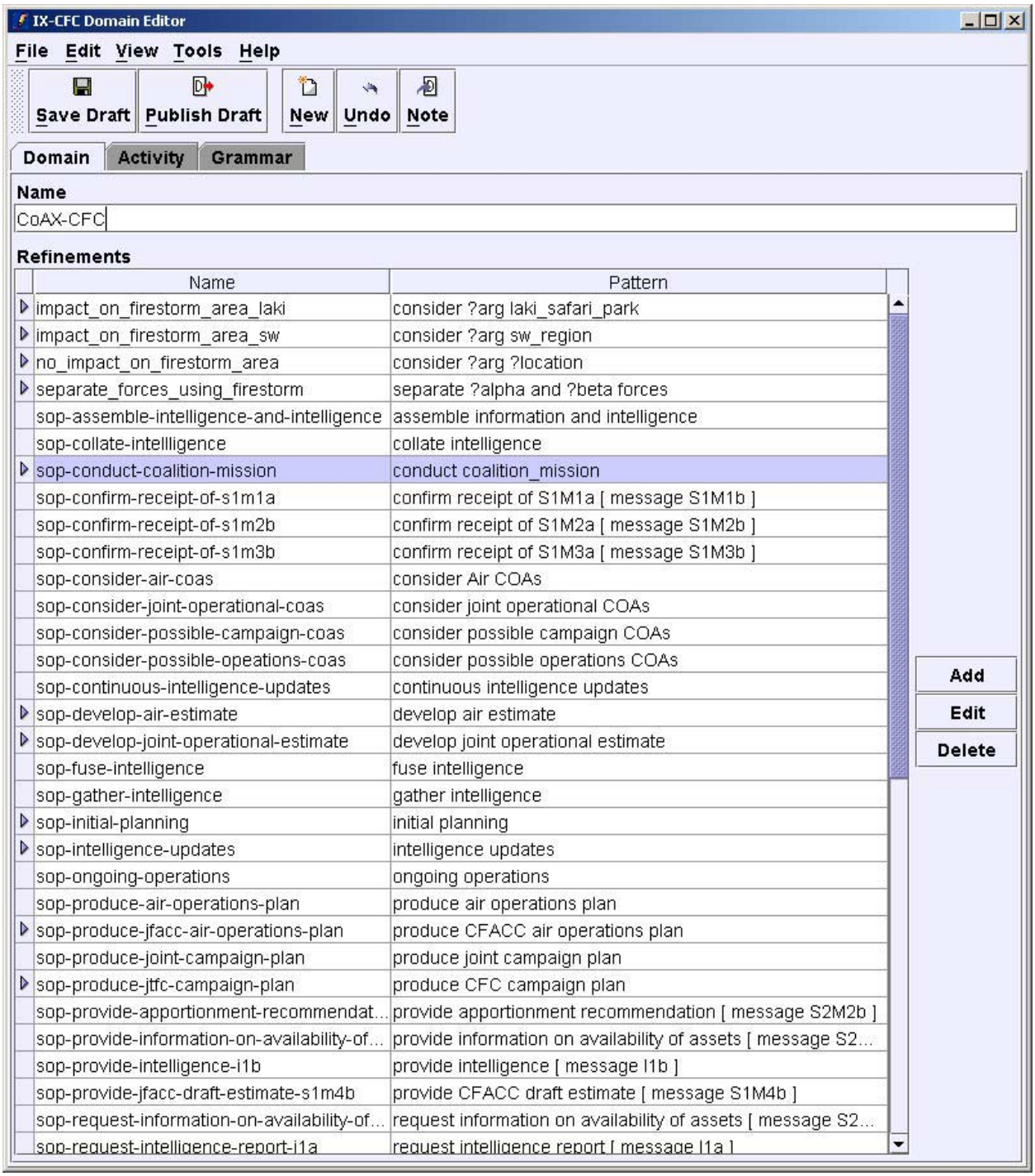


Figure 3: Global Domain Editor

Figure 3 shows the global domain editor and Figure 4 shows the grammar viewer (currently the grammar cannot be edited).

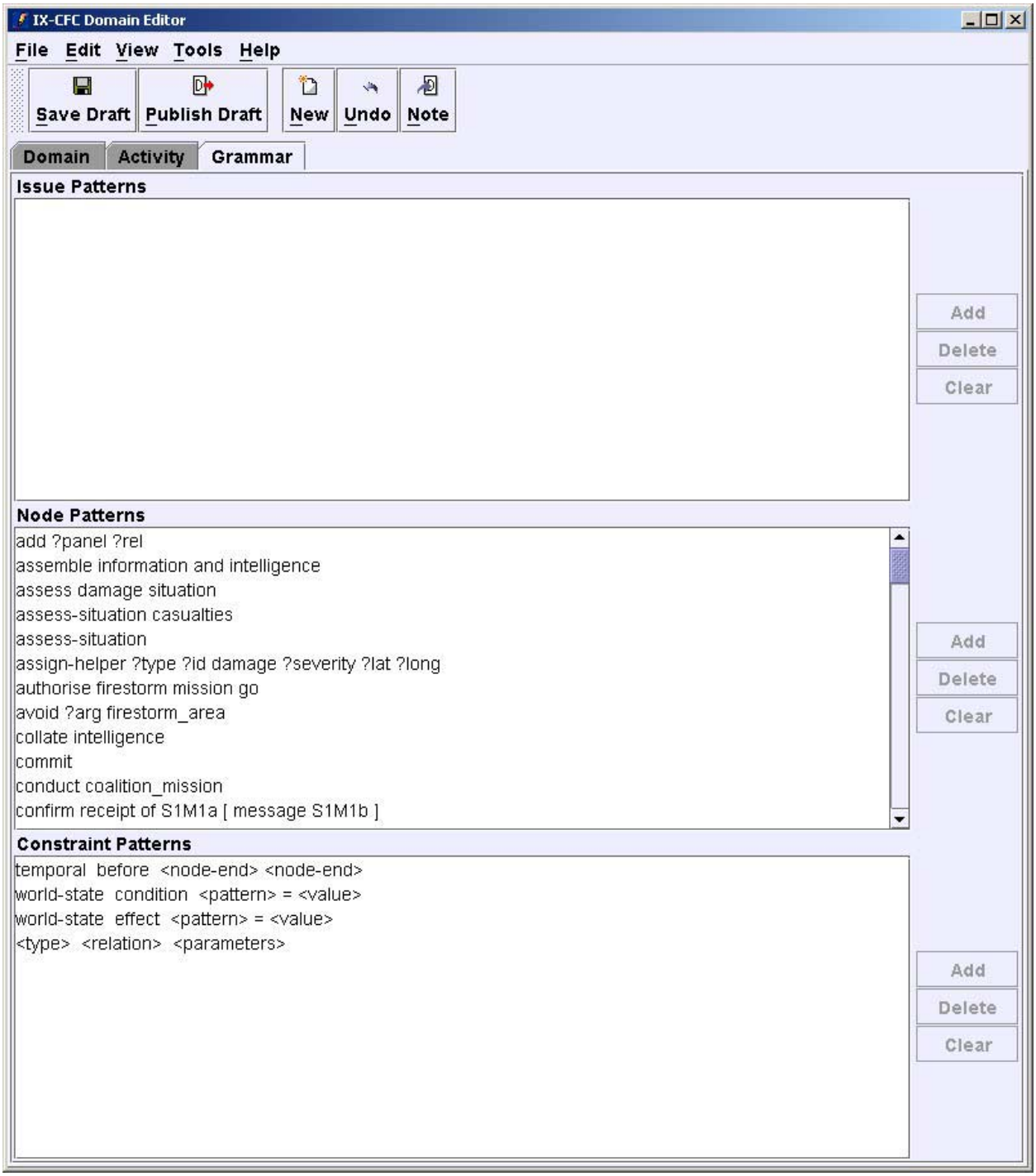


Figure 4: Grammar Editor

Activity Editing

The functions described above are common between editors for different concepts, but there are specific things that are available in I-DE to support activity editing. The Activity Editor has three different views that have different purposes:

1. the minimal view is for quickly defining or editing simple activity specifications.
2. the comprehensive view provides full access to all fields of activity specifications that are known to I-DE.
3. the graphical view for a more visual way to specify or edit sub-activities and ordering constraints between them.

The Minimal View (Figure 5**):** An activity's name and pattern can be specified, along with its list of issues and sub-activities. The sub-activities can be put into sequence or in parallel and the editor will indicate other ordering constraints if they are present, i.e. if they have been specified using a different view or a different editor. Specifications (and edits) can be made by simply typing into the respective fields on the screen.

The Comprehensive View (Figure 6**):** The comprehensive view shares some of the features of the minimal view, but provides additional ones. The name and pattern of the activity are presented and specified or edited in the same way as in the minimal view. For issues and nodes the user can choose whether to type into the fields directly or whether to see fields as lists and use structured dialogue-style editors for specifying and editing. Constraints are split into orderings, conditions/effects, and other constraints. The editor can show and edit these constraint types with specific support, but the user can also decide to suppress the display of specific constraint types, or to view constraints of all types (except orderings) in their generic form.

The Graphical View (Figure 7**):** The graphical view uses nodes and arcs to show sub-activities and ordering constraints between them. There are also text fields for the activity name and pattern and (as in the other views) these can be edited by typing into the fields. Nodes can be added and deleted easily. They each show a node number that is assigned by I-DE whenever a sub-activity is specified (also in the other views), the pattern of the sub-activity, and the sub-activity's two node ends: begin and end. Arcs can be drawn between node ends to specify ordering constraints between the node ends.

In the graphical view, it is easy to move from a sub-activity to its possible expansions (all activity specifications whose patterns match that of the sub-activity) by clicking the right mouse button. This brings up a pop-up menu that lists all matching activity specifications that are currently in the domain. New expansions (activity specifications) can be defined using the same pop-up menu.

The minimal and comprehensive views have a summary panel on the left and an editing panel on the right. The summary panel lists all currently defined activities and their sub-activities as a

tree. Clicking on an activity will put that activity's specification into the editing panel, clicking on a sub-activity will put that node's pattern into the pattern field in the editing panel, ready for the user to provide a new specification with that pattern.

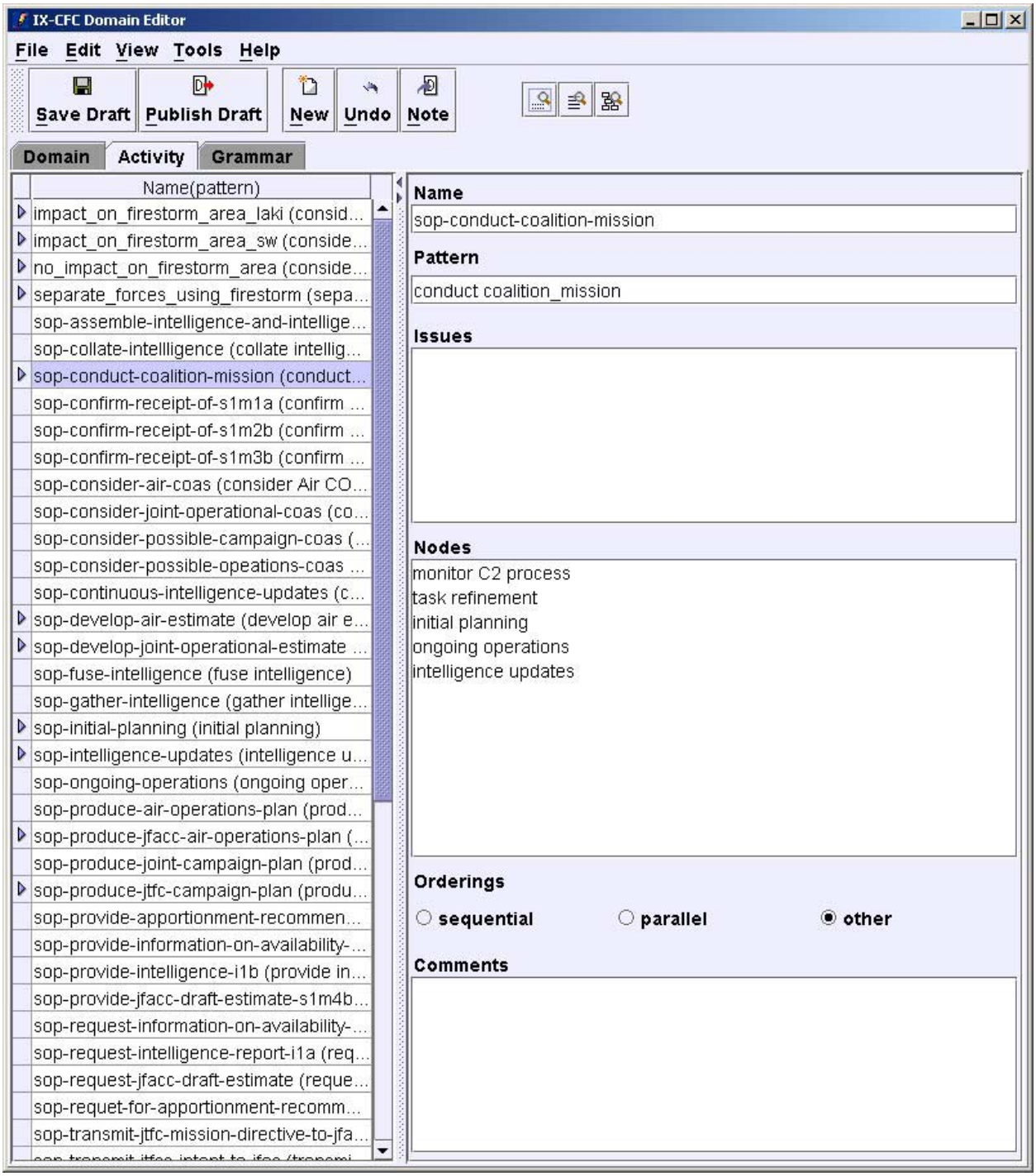


Figure 5: I-DE Activity Editing - minimal view

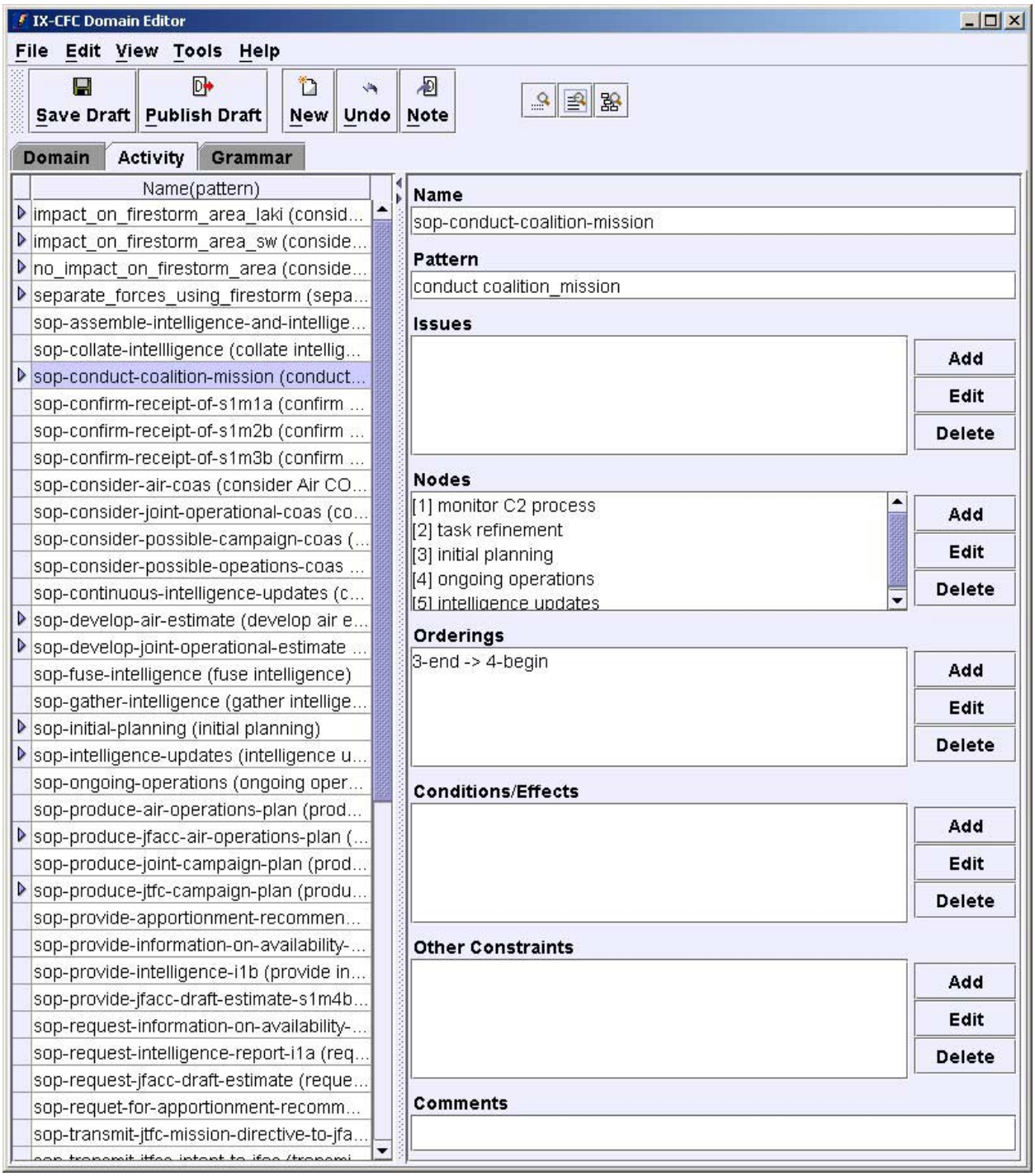


Figure 6: I-DE Activity Editing - comprehensive view

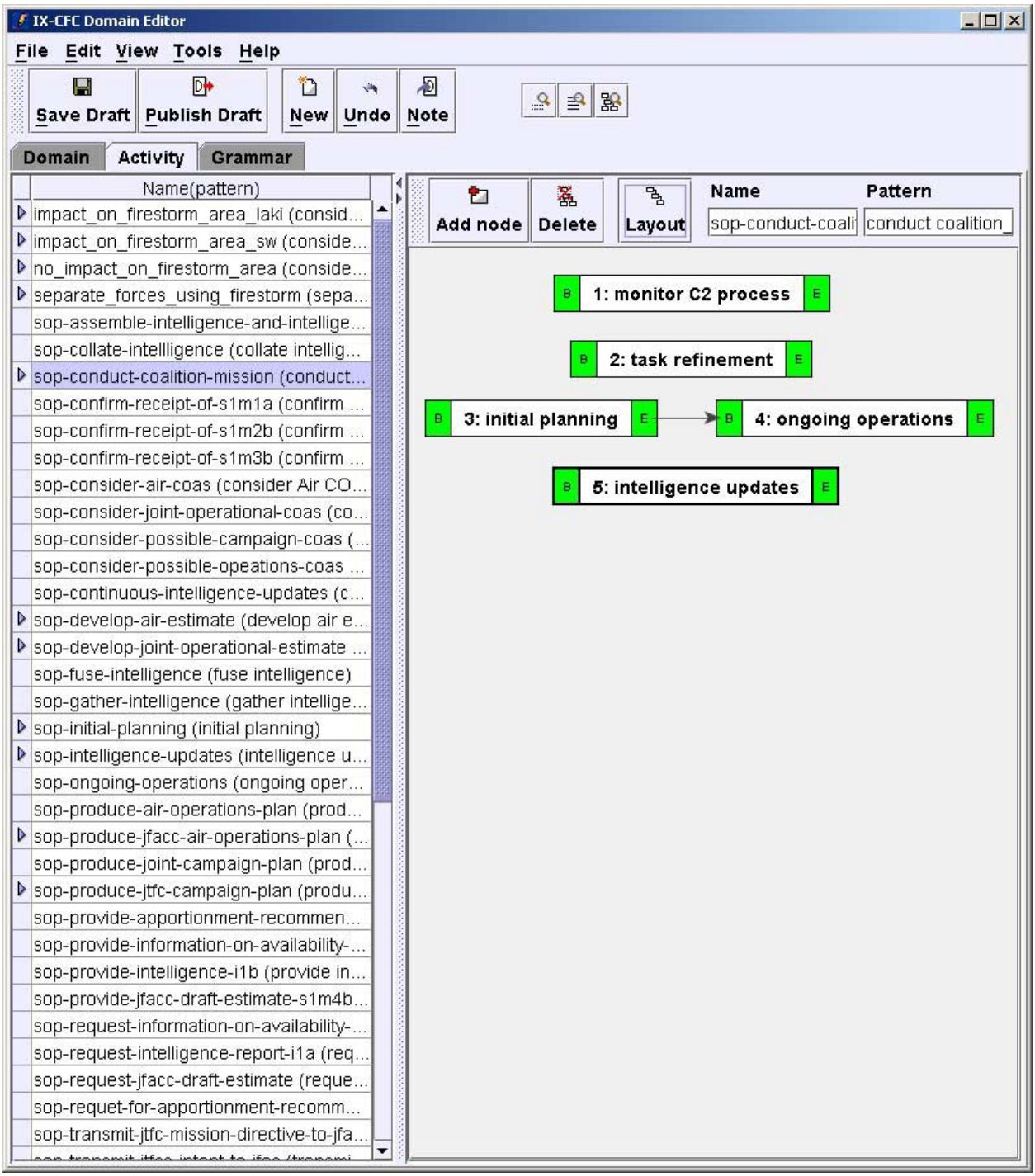


Figure 7: I-DE Activity Editing - graphical view

Variable Declaration

It is possible to use the editor to restrict which variables can be used in an activity specification. This is done with the help of variable declarations which can be made via the "Declare Variables" option in the Edit menu. The user can:

- allow any variable to be used in the specification (default),
- allow no variables to be used in the specification,
- provide a list of all variables that can be used in the specification, or
- "freeze" variable use by declaring all currently used variables.

Once a variable declaration (other than "any") has been provided by the user, the activity editor will attempt to support the user in sticking to the declaration. Whenever the user types a "?" in a field that can contain a variable, the editor checks whether there are any variable declarations are present. If no variables are allowed, the editor complains and the "?" will not appear in the specification. If there is a list of allowed variables, the editor will present this list and allow the user to choose one of the declared variables. This should help the user to adhere to declarations, but it is possible for the user to enter variables that are not allowed.

The editor also lets the user check explicitly whether the activity specification contains any violations via "check consistency" option in the Tools menu. On request, the editor will check whether there are any declarations. If there are, the editor checks whether any un-declared variables are used, and whether any of the declared variables are not used. Either of these two events is considered a violation and will be reported to the user.

Using the Models

A generic modelling framework, such as <I-N-C-A>, allows different kinds of models to be represented in a uniform style. The high-level, generic, multi-purpose structure combined with keywords that indicate the specific semantics of contents makes this framework a powerful representation tool. A shared ontology of keywords can ensure that the keywords are used consistently in the models. Different kinds of problem solvers that also share the ontology of keywords can then determine which specific parts of the models are relevant to them. Ignoring those parts of the models whose keywords are not part of the problem solver's domain can simply be ignored.

If the problem-solving environment is distributed and agent-based and uses a broker to coordinate different problem solvers, the broker may be able to use the ontology of keywords to determine which problem solvers can best contribute to the problem. Comparing the keywords used in the models with the keywords that a problem solver can understand (i.e. are part of the problem solver's domain) should help the broker to match solvers to models and the problem at hand.

It is important to remember that the quality of the models and the amount of information present in the models will significantly affect their usefulness.

Evaluation and Conclusion

At the start of this paper we established a set of requirements for Enterprise Modelling support. We now take these in turn and check how I-DE relates to them:

- any realistic enterprise modelling support will have to be able to provide and cope with different techniques for capturing information, and with different notations (or views) for the information;
    - I-DE illustrates how different viewers editors can be used in conjunction with each other (different sub-panels or sub-editors) and how different views can be used to highlight different types of information (form-based vs. graphical activity editor views).
- it must not be necessary for the models to be complete (we must be able to cope with incomplete information and we should make use of all the information that we have);
    - I-DE looks for those things in models that it can display and update, ignoring the parts that it does not recognise. It also allows for incomplete models to be stored and published. However, generating and saving incomplete models is relatively easy. The main impact of this issue is on systems that use the models as a knowledge base, e.g. a workflow model.
- it must be possible (and easy) to change and update the models;
    - We believe that I-DE is easy to use and that it provides good support for updating models. Its architecture and connection to the I-X framework also makes it easy to set up I-DE so that it can be used together with a workflow system in order to interleave process specification, planning, and enactment.
- models should be used to their full capacity to support the running of the organisation.
    - The kinds of information that I-DE is designed to capture lend themselves well to being used as the basis of workflow.

Overall, I-DE provides good support for Enterprise Modelling. I-DE is implemented in a way that makes it easy to use the editor in different contexts and set-ups. It also makes it relatively easy to provide additional support. In the future, it would be interesting to see how much modelling support can be provided by using I-DE in conjunction with a workflow system running a modelling process model, i.e. to guide the use of I-DE with the help of a model that specifies Enterprise Modelling expertise.

References

1. Chen-Burger Y. and Stader J., “Formal Support for Adaptive Workflow Systems in a Distributed Environment”, to be published in Workflow Handbook 2003, Ed. Layna Fischer, 2003.
2. Dobson J. and Blyth A., Chudge J. and Strens M., “The ORDIT Approach to Organisational Requirements”, Requirements Engineering: Social and Technical Issues, London, ed. Jirotka and J.A.Goguen, Academic Press, 1994.
3. Fox M. and Gruninger M., “Enterprise Modelling”, AI Magazine, AAAI press, Fall 1998, pp.109-121.
4. Fraser J., “Managing Change through Enterprise Models”, Proceedings of Expert Systems '94, the 14th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, UK, 12-14 December 1994.
5. IBM, “Business System Development Method, Business Mapping, Part1: Entities; and Part 2: Processes”, 2nd ed, IBM England, May 1992.
6. Jarvis, P., Stader, J., Macintosh, A., Moore, J., Chung P., “What Right Do You Have To Do That?: Infusing adaptive workflow technology with knowledge about the organizational and authority context of a task”. First International Conference on Enterprise Information Systems (ICEIS-99), Setubal, Portugal,1999.
7. Mayer R, Cullinane T, deWitte P, Knappenberger W, Parakath B, & Wells S, “IICE IDEF3 process description capture method report (al/tr-1992-0057)”. Technical Report, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, 1992. See also http://www.idef.com/
8. NIST, “Integration Definition for Function Modelling (IDEF0)”, Federal Information Processing Standards Publication 183, National Institute of Standards and Technology (NIST), Dec 1993.
9. Ould M, “Business Processes: Modelling and Analysis for Re-engineering and Improvement”, John Wiley and Sons, 1995.
10. Stader J. and Macintosh A., "Capability Modelling and Knowledge Management"; Applications and Innovations in Expert Systems VII, Proceedings of ES 99 the 19th International Conference of the BCS Specialist Group on Knowledge-Based Systems and Applied Artificial Intelligence, Cambridge, December, 1999; Springer-Verlag; ISBN 1-85233-230-1; pp 33 – 50, 2000.
11. Stader J., Moore J., Chung P., McBriar I., Ravinranathan M., and Macintosh A., "Applying Intelligent Workflow Management in the Chemicals Industries", Workflow Handbook 2001, L. Fisher (ed), Published in association with the Workflow Management Coalition (WfMC), 2000, pp.161-181.
12. Stader J, “Results of the Enterprise Project”, Proceedings of the 16th International Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, UK, 1996.
13. Tate A, “I-X: Technology for Intelligent Systems”, www.i-x.info, AIAI, The University of Edinburgh, 2002.
14. Tate, A., “I-X and <I-N-C-A>: an Architecture and Related Ontology for Mixed-initiative Synthesis Tasks**”,** Proceedings of the Workshop on Planning/Scheduling and Configuration/Design at PuK 2001, Vienna, Austria, 2001.
15. Waern A. and Hook K. and Gustavsson R. and Holm P., “The Common-KADS Communication Model”, KADS-II/M3/SICS/TR/006/V2.0, Swedish Institute of Computer Science, Stockholm, Sweden, Dec 1993.

Appendix G: software Agents as facilitators of Coherent Coalition Operations

Dr David Allsopp, DERA
Squadron Leader Patrick Beautement, DERA
Dr Jeffrey M Bradshaw, IHMC / UWF
Dr John Carson, DERA
Dr Michael Kirton, DERA
Dr Niranjan Suri, IHMC / UWF
Prof Austin Tate, AIAI

Abstract: Software agents can be viewed as semi-autonomous entities which help people cope with the complexities of working collaboratively in a distributed information environment. This paper describes the research that DERA is carrying out into Software Agents for use in Command Systems and the collaborative work with the 16 partners of an international Coalition Agents Experiment. Specifically, the paper aims to show that using software agent-based C2 frameworks is a useful way of dealing with the complexity of real-world problems such as supporting agile and robust Coalition operations and enabling interoperability between legacy or previously incompatible systems. In addition, Agent-enabled 'grids' can be used to rapidly integrate a wide variety of agents and infrastructures, with domain management services structuring agent relationships, limiting their behaviours and enforcing Coalition policies.

Citation: Allsopp, D.N., Beautement, P., Bradshaw, J.M., Carson, J., Kirton, M., Suri, N. and Tate, A. (2001) "Software Agents As Facilitators of Coherent Coalition Operations", 6th International Command and Control Research and Technology Symposium, US Naval Academy, Annapolis, Maryland, USA.

Section 1- Introduction and Background

Software agents are currently receiving much attention in the research community. This interest is being driven by the phenomenal growth of the Internet and the World-Wide-Web. Agents can be viewed as semi-autonomous software designed to help people cope with the complexities of working collaboratively in a distributed information environment. This involves the agents communicating between the users and between themselves. The agents are used to find, format, filter and share information, and work with users to make the information available wherever and whenever users need it. This paper describes the research that DERA is carrying out into Software Agents for use in Command Systems (SACIS) and the progress which is being made in the collaborative work with the 16 international partners of the Coalition Agents Experiment (CoAX).

Military Context

Success in military operations involves carrying out high-tempo, coherent, decisive actions (faster than an opponent can react) resulting in decision dominance through the use of command agility. Command agility is about being flexible and adaptable so that fleeting opportunities can be grasped. This is done by the Commander issuing clear intent and then delegating the control authority to subordinates - allowing them the scope to exercise initiative. It also means being innovative, creative and unpredictable in a manner that (even if low-tempo) increases confusion in the mind of an opponent. This process is command led, which means that human decision-making is primary and that the role of technology is secondary. Shared understanding and Information Superiority are key enablers in this process and are fundamental to initiatives such as the UK's Joint Battlespace Digitisation programme[12]. Concepts such as Network-centric Warfare (see http://www.dodccrp.org/ncw.htm) also demand a more decentralised approach such as that provided by software agents.

In addition to the problems of integrating single-service and Joint capabilities into a coherent force, the nature of Coalition (now often just called multi-national) operations implies some need to configure incompatible 'come-as-you-are', or foreign systems, into a cohesive whole rapidly. Many problems in this environment can only be solved by organisational changes and by 'aligning' doctrine, concepts of operations and procedures. Coalition scenarios trigger the need for a rapid on-the-fly response and cannot be predicated on using pre-existing co-ordinated systems - hence the need for a flexible approach which allows capabilities to be assembled at 'run-time'. However, in addressing this requirement for interoperability, it is also crucial to address issues of security of data, control over semi-trusted software from other Coalition partners, and robustness of the resulting system (e.g. the ability to withstand denial-of-service attacks).

The current reality of Coalition Operations is often a picture of data overload and information starvation, labour intensive collection and co-ordination, individual stove-pipe systems,

[12] Which aims to integrate the use of information across the Joint arena by exploiting appropriate doctrine, organisations and procedures, personnel and technology.

incompatible formats, scattered snapshots of the battlespace, and a horrendous technical integration task. This paper aims to show that the agent-based computing paradigm offers a promising new approach to dealing with such issues by embracing the open, heterogeneous, diverse and dispersed nature of the Coalition environment. Indeed, for 'Cyberspace Superiority' to be obtained (as part of the Battlespace) then it is essential that the Coalition Forces are able to act decisively *inside* Cyberspace and that the *only way* that this can be achieved is though a variety of software agents acting on behalf of, or mediating the actions of, human users within Cyberspace - as well as at its margins *and beyond*.

Aims of the "Software Agents in Command Information Systems" and CoAX Projects

In this paper, we report on early progress in DERA's Software Agents in Command Information Systems (SACIS) project. Included in this is a related international collaborative effort whose overall goals are to demonstrate that the agent-based computing paradigm offers a promising new approach to dealing with the technical issues of establishing coherent command and control (C2) in a Coalition organisation. This collaborative effort is being carried out under the auspices of DARPA's Control of Agent Based System (CoABS) programme, which provides a software agent-enabled "Grid" (see http://dtsn.darpa.mil/iso/ or http://coabs.globalinfotek.com/). The collaborative effort is called CoAX (Coalition Agents eXperiment) and it is a CoABS technology integration experiment (TIE) - see http://www.aiai.ed.ac.uk/project/coax/. The overall objectives of all this research are to determine and demonstrate the potential effectiveness of software agent technology to assist with issues of interoperability, etc in the context of military command systems. Specifically, the aims are to show that:

- agents are a useful metaphor for the complexity of real-world systems such as military operations;
- an agent-based C2 framework can support agile and robust Coalition operations;
- software agents can be used to enable interoperability between legacy or previously incompatible systems;
- the CoABS Grid can be used to rapidly integrate a wide variety of agents and systems - i.e., rapid creation of virtual organisation;
- domain policies can structure agent relationships and enforce Coalition policies;
- intelligent task and process management can improve agent collaboration;
- and that semantic interoperability can improve agent collaboration and interoperability between disparate Coalition command systems.

The SACIS and CoAX research has built a software agent testbed based on the DARPA CoABS Grid and this paper will describe the work done, the demonstrations carried out so far, the scenario and storyboard used and some of the initial insights which have been gained.

Structure of the Paper

The paper begins by providing a brief description of the key ideas and technologies underpinning software agents in Section 2. Section 3 describes the Coalition scenario and military command structure used in our demonstration experiments. Section 4 describes the corresponding agent architecture that was developed to reflect the military organisational structure. The events occurring in the storyboard used for the demonstrations so far are given in Section 5. A preliminary assessment of software agent capabilities and a discussion of future research are provided in Section 6. Concluding remarks are given in Section 7.

Section 2 - Software Agent Technology

There are a number of very good sources describing the state-of-art in software agent research and applications: see, for example, the Web site hosted by the University of Maryland [1]; the review article by Jennings et al [2]; the collection of research papers edited by Huhns and Singh [3]; the book edited by Bradshaw [4]; and the papers in the special issue of IEEE Intelligent Systems [5]. Although there is no comprehensive and widely accepted definition of the notion of software agent [6], Jennings et al provide a useful working description:

> "An agent is a computer system situated in some environment that is capable of flexible autonomous action."

Jennings et al point out that "being situated" means that an agent is part of an environment from which it receives sensory input and that the agent can change in some way. Autonomy means that an agent should be able to act without the direct intervention of users. Flexibility implies that the system is responsive, pro-active and social - i.e., agents should, in principle, be able to interact with other agents and humans.

Figure 1 shows four main types of software agents. The 'Housekeeping Agent' (HK) is an entity which is responsible for assisting with the maintenance of Cyberspace, for example, by adjusting resource loading, monitoring for adverse performance, network routing, etc. The 'Information Agent' (IA) facilitates the movement, analysis and formatting of information in and through Cyberspace. A third entity is the 'Mediator Agent' (MA) which is the focal point of interaction between the human user and the underlying information, 'applications' and Cyberspace itself. In addition, there may be hostile versions of these agents. In SACIS and CoAX we use the following general definition for all types of software agents:

"Agents can be viewed as (software) entities acting on behalf of, or mediating the actions of, a human user and having the ability to carry out tasks autonomously to achieve goals or support the activities of the user."

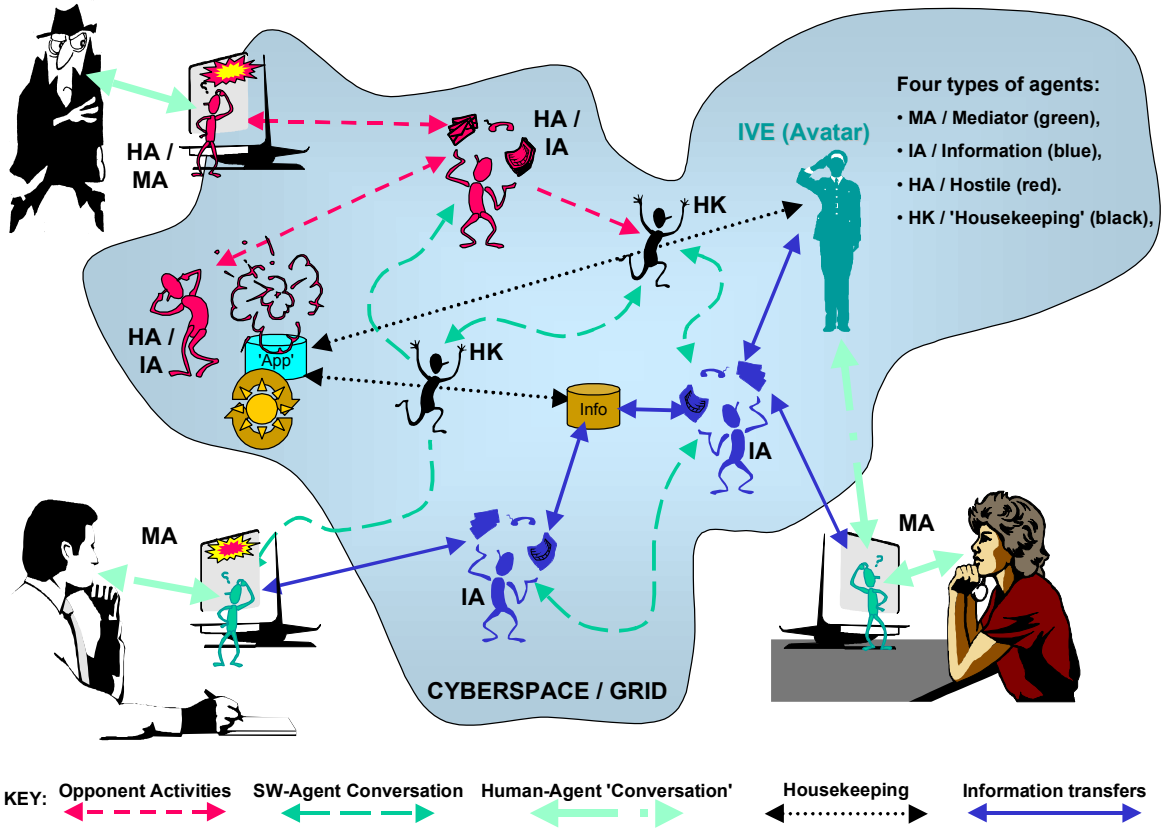


Figure 1 - Types of Software Agent

In order for agents to communicate with one another and to share information, agent communication languages (ACLs) have been developed. There are two main ACLs: the Knowledge Query and Manipulation Language (KQML) and the Foundation for Intelligent Physical Agents (FIPA) ACL [7]. These languages handle propositions, rules and actions and are therefore at a higher level of abstraction than middleware such as CORBA (common object request broker architecture) and RMI (remote method invocation). In addition, there has been much work on the provision of a machine-understandable semantic-web of information such as eXtensible Markup Language (XML) and the Resource Description framework (RDF). Software agents can use RDF to advertise and describe their capabilities and can parse RDF descriptions to ascertain the relevance and utility of information provided by other agents. DARPA has recognised the potential contribution that technologies such as RDF could make to achieve semantic interoperability among software agents and has a new programme underway called DAML (DARPA Agent Mark-up Language [8]). The overall goal of the DAML programme is to develop a language aimed at representing semantic relations in machine-readable ways compatible with current and future Internet technologies. We are currently working on a specification and implementation of a DAML-based policy language (KAoS Policy Language or

KPL), which will be used in CoAX to represent both simple atomic policies and complex constructions.

One way of viewing a community of agents is as a set of distributed, asynchronous processes communicating and sharing information by message passing in some infrastructure. In this regard, an important output from DARPA's CoABS programme is the CoABS Grid - a middleware layer based on Java / Jini technology that provides the computing infrastructure to integrate heterogeneous agent communities and systems rapidly. In a recent article, Jennings [9], argues that the agent paradigm is a good way of building complex software systems in general. Hence the potential benefit of using software agents in the Coalition setting, for example, where legacy command systems could be provided with software agent wrappers that allow them to inter-operate and share information with other systems and agent applications in a loosely connected, heterogeneous architecture that is underpinned by the CoABS Grid. The scenario, used as the basis of the experiments to test this hypothesis, is described in the next section.

Section 3 - A Representative Scenario and Coalition Command Structure

The SACIS and CoAX work needed a suitably realistic scenario for its experiments and so the Team expanded the fictional "Binni" scenario developed by Dr. A.R. Rathmell [10] for The Technology Co-operation Programme[13] (TTCP). In this scenario the year is 2012 and global warming has altered the political balance of the world. The action is set in an area that is currently the Sudanese Plain [see Figure 2 below]. Previously uninhabited land in the Plain is now arable and the area has received large amounts of foreign investment. A conflict has developed between two countries who are fighting for control of Binni. To the north is Gao - which has expansionist aspirations but which is only moderately developed, with old equipment and with a mostly agrarian society. To the south is Agadez a relatively well developed and fundamentalist country. Gao has managed to annex an area of land, called it Binni and has put in its own puppet government. This action has come under fierce attack from Agadez. Gao has played the 'threat of weapons of mass destruction from Agadez' card and has enlisted support from the UN who have deployed a force, the UN War Avoidance Force for Binni (UNWAFB), to stabilise the region.

The UNWAFB has arrived in theatre and is not being opposed by Agadez. Gao is providing 'host nation' support in Binni at the ports and airports in the east and the Coalition Forces are working through an initial planning phase. One of the options under consideration is to lay down a 'firestorm' between the Gao and Agadez forces in this region. This will prove to be contentious as Gao will try to provide false information to displace the Firestorm. Also, the international media will hear of the operation and will object to the bombing taking place near a wildlife refuge area (the Laki Safari Park).

[13] The C3I Group, Technical Panel 9.

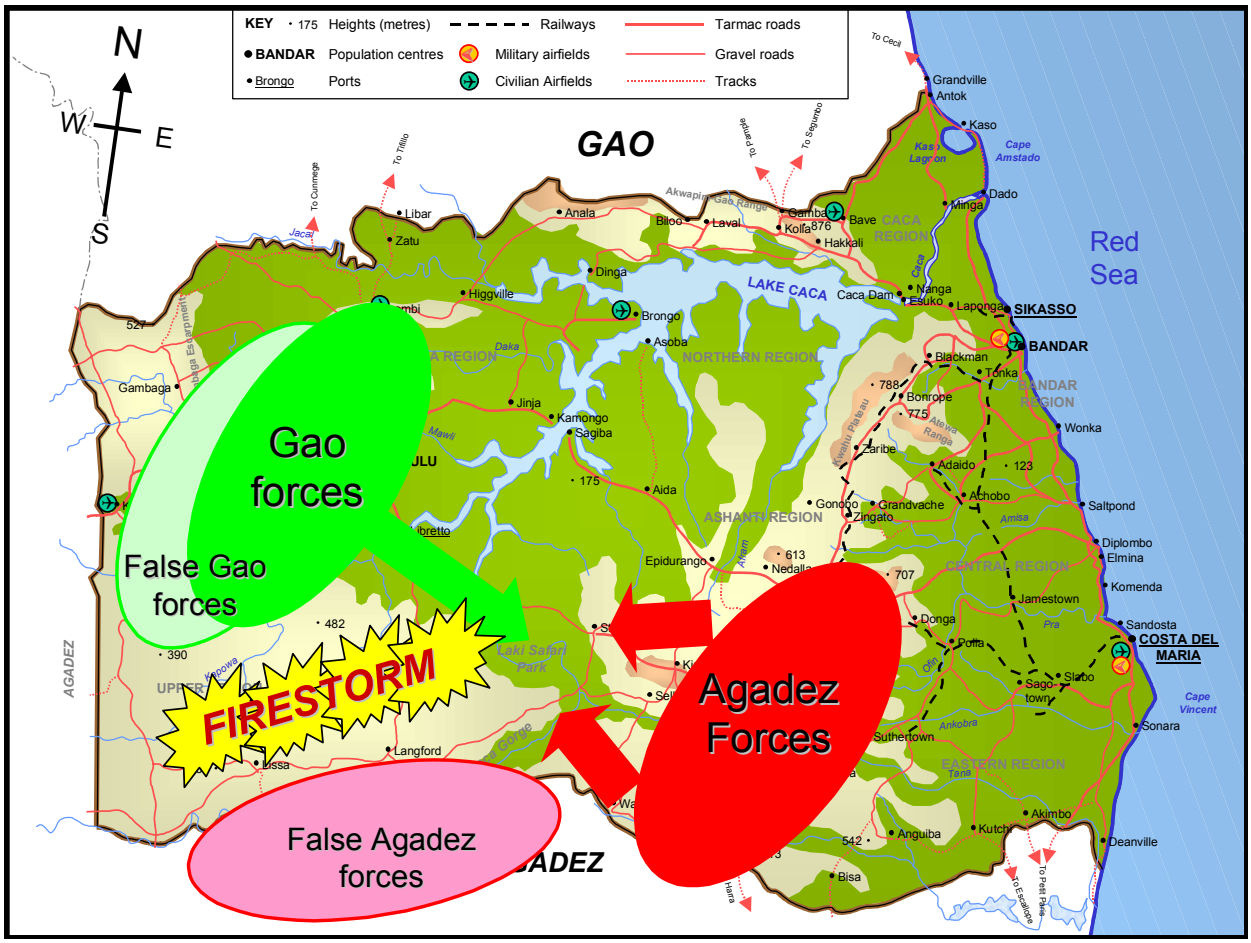


Figure 2 - Map of Binni showing Firestorm Deception

This Binni Coalition (multi-national) operation needs to rapidly configure various incompatible, 'come-as-you-are' or foreign systems into a cohesive whole within an open, heterogeneous diverse and dispersed environment. This scenario provides a suitable test for the software agent experiments, where the run-time composability of software agents is a very close metaphor for the dynamic uncertainty of Coalition Operations. The complexity of the situation must not be underestimated and is best illustrated by looking at the Binni Coalition Command Structure shown in Figure 3 below.

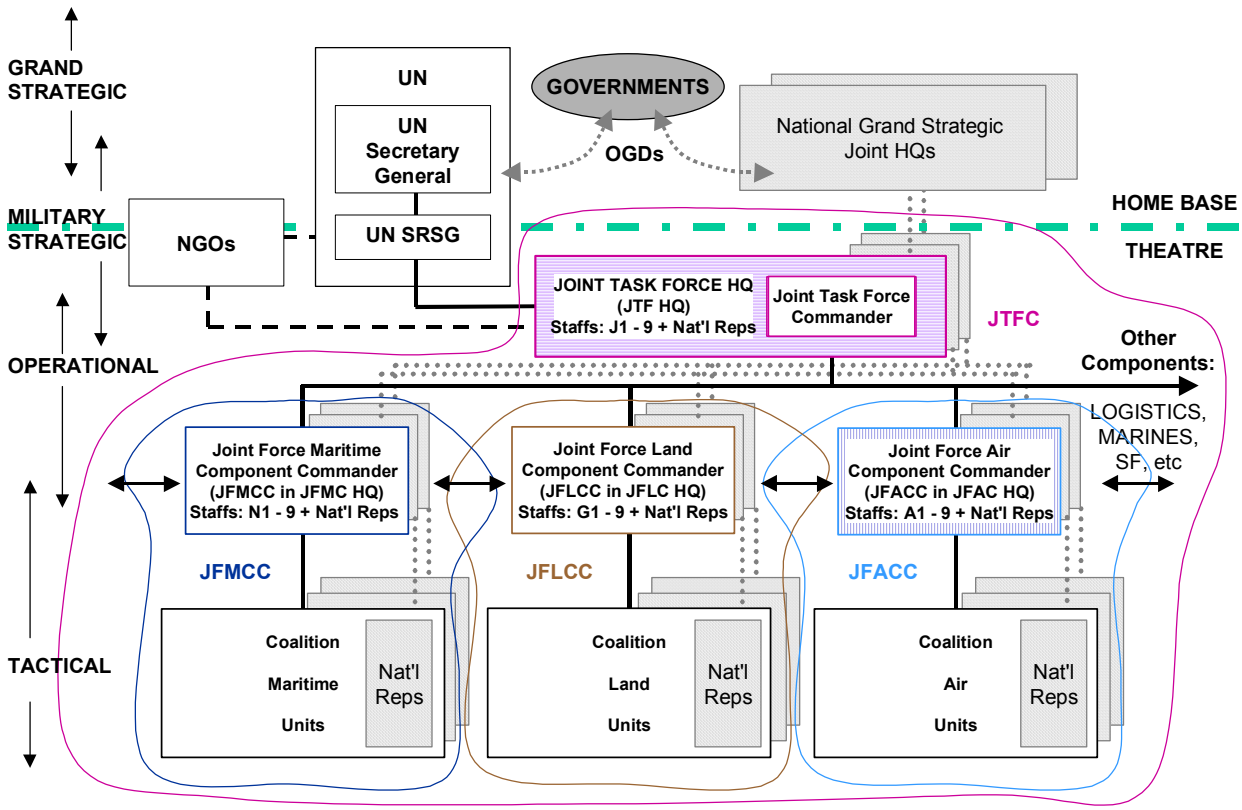


Figure 3 - Representative Coalition Command Structure

This is a representative and realistic Coalition command structure involving the UN, Governments, Other Government Departments (OGDs - such as the Foreign Office), Non-Government Organisations (NGOs - such as Oxfam), representatives of all the Coalition Countries (with their own 'ghosted' Command Structures) and the Coalition HQs and subordinate fighting forces. The solid black lines on the diagram show the legal lines of command authority (the 'command chain') and accountability. Dashed lines show an advisory / negotiating role. This is the kind of Coalition structure which would be agreed by the participants and no part of it is 'owned' by any specific country. Other possible Components (Logistics, Special Forces (SF) etc) are not shown. Note that the 'Levels of Command' overlap and their boundaries are not rigidly defined - as a general rule though, the JTFHQ and Component HQs span the critical operational / tactical boundary, which can *roughly* be equated to the planning / execution boundary.

Section 4 - Corresponding Software Agent Architecture

Human 'Domains'

Integrating the use of information across a Coalition is not just a matter of employing technology - it involves the creation of a coherent 'interoperability of the mind' at the human level as well as exploiting appropriate doctrine, organisations, personnel and procedures. In a Coalition many social and cultural factors, therefore, come into play. The SACIS and CoAX Teams realised that the mapping between the human and technical worlds was not straightforward and that, from the human perspective, four kinds of 'domains' could be identified as follows:

- Organisational Domains: These relate to a span of control or legal authority of a command entity where information would be shared across the domain at a common security level. The SACIS and CoAX work has focussed on the Joint level of command - in particular on two organisation domains: the Joint Task Force HQ (JTF HQ) and the Joint Force Air Component HQ (JFAC HQ). Technically, the JTF HQ and the JFAC HQ are sub-domains of the Coalition Organisational structure shown in Figure 3.
- Country Domains. Each of the National command chains would be a separate, self-contained 'domain' with its own processes, information, security regime etc. The interface with the Coalition is often through the National Representatives (liaison officers) who carry out any necessary 'translation' and act as a 'safety gap' for security reasons. Figure 3 shows each Nation 'ghosting' all of the Command HQs - in practice each Nation will provide different degrees of command 'presence' throughout the Battlespace.
- Functional Domains. These relate to a set of entities collaborating on a common task. Such domains may be virtual (ie exist only in Cyberspace), are often informal and may come and go as the military imperative changes. A more formal functional domain would be the Intelligence community which spans various levels of command.
- Individual Human Domains of Responsibility. In simple terms Commanders have responsibility for the effective running of their own HQ *and* all the subordinate ones (in practice they delegate this authority). Hence the individual human domains of influence may overlap - shown with the shapes with irregular boundaries on Figure 3 above.

These types of domains are not entirely exclusive and there are many different levels of overlap and interaction depending on the viewpoint taken. It is this complexity at the human level that create difficulties for technical systems.

Software Agent Domains

A software agent domain could be more tightly defined in technical terms as follows:

> "Software agent domains are bounded objects (which can interoperate via intermediate structures) with clear identities and ethos. Each domain contains entities and structures working collaboratively and sharing information, processes and 'control' procedures such as security regimes and policies."

Hence, the SACIS and CoAX Teams maintain that software agent domains provide a better mapping to the human domains described above than some other technical approaches. The next section explains how the software-agent domains were defined and how they operate.

In a software-agent-enabled infrastructure (such as the CoABS Grid) agent domain management services are defined. These services will evolve from and enhance existing services available within the software agent framework. An agent domain consists of a unique instance of a domain manager (DM) along with any agents that are registered to it. The function of a domain manager is to: 1) manage agent registration, 2) serve as a single point of administration for policy management. That is, the domain manager could configure, re-configure, store, publish and enforce polices that exist for that domain. Domains assure those who deploy agents systems that there is policy uniformity across multiple platforms and hosts, as long as semantically equivalent monitoring and enforcement mechanisms are available across those platforms and hosts. Under these conditions, it would follow that a given domain could extend across host boundaries and, conversely, multiple domains could exist concurrently on the same host. With respect to platform independence, it should be possible for agents running on the same platform to be in different domains (for example, a resident and a visiting mobile agent running on the same platform may belong to different domains having more or less restrictive security privileges). In addition, domains contain match-makers (MM) which work together to provide information about local and remote agent services that are available.

A policy is a machine-readable set of statements in which some element (such as an agent) of an agent system declares a specification intended to describe or govern its interaction with other elements of the agent system. For example, an agent may declare a policy that all messages it exchanges with other agents must be encrypted, or that certain timing and message sequencing constraints must be observed when requesting a particular kind of service from that agent. The latter is an example of a conversation policy. A domain manager has policies such as:

- no agents registered to its domain may communicate with agents outside the domain;
- no agent can consume more than a given fraction of some system resource;
- agents must respond to messages from the domain manager within a given time frame;
- agents with higher priority tasks pre-empt lower priority ones.

The policy is expressed in a persistent machine-readable format, which could be interpreted by a platform-specific policy enforcement mechanism. At the current time, the representation is very simple, but a more powerful representation in DAML is currently being defined. Policy and policy-enforcement mechanisms could be defined in multiple locations in a given implementation. The separation of policy specification from policy-enforcement mechanisms allows policies to be dynamically reconfigurable, and relatively more flexible, fine-grained, and extensible. Agent developers can build applications whose policies can change without necessarily requiring changes in source code. The rationale for using declarative policies to describe and govern behaviour in agent systems includes the following claims: easier recognition of non-normative behaviour, policy reuse, operational efficiency, ability to respond to changing conditions, and the possibility of off-line verification.

Software Agent Domains in SACIS and CoAX

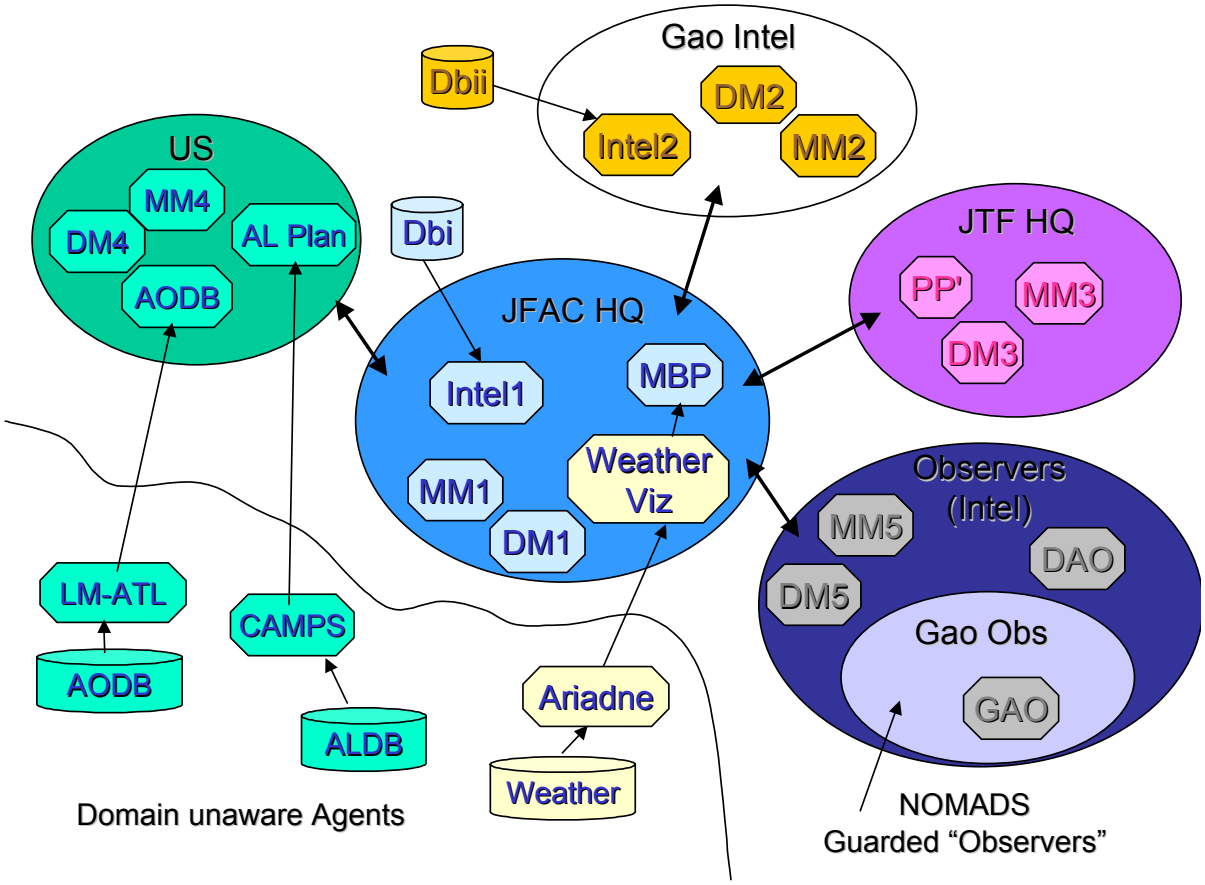


Figure 4 - Software Agent Domain Mappings

The SACIS and CoAX Teams have set up a demonstration containing 23 separate software agents grouped into 6 agent domains (including one sub-domain). The interoperability of the agents and systems was made possible using the inter-agent communications services provided by the DARPA CoABS Grid, with the policies enforced by KAoS[14] domain management services. Figure 4 below shows that within each agent domain there is a matchmaker (MM) and domain manager (DM), and then a set of software agents and agent-enabled tools / applications, which are described further below.

The SACIS and CoAX teams have carried out a number of experiments. In the one described here two legacy systems from different nations were brought together and agent-enabled, along with an agent process management tool. In addition, a number of agent-wrapped databases, agent-wrapped public domain Internet information and agent domain and policy administration and malicious agent management tools were provided. The supporting infrastructure was built on the DARPA Grid.

In the JFAC HQ the Joint Force Air Component Commander's (JFACC) staff use a UK tool called the Master Battle Planner (MBP). The MBP assists the planners by providing them with a visualisation - which relates directly to their way of thinking because it is an Expressive System [11] - on which they can manipulate the air intelligence information, assets, targets and missions etc to build up an Air Battle Plan. The MBP has been given a software agent wrapper to enable it to bring together information from the various Coalition partners in near real-time - and this provides a significant improvement in overall capability.

There is also a weather cell in the JFAC HQ which can acquire information from the Internet via the Ariadne (see http://www.isi.edu/info-agents/ariadne/) software agent. Ariadne runs off-site and gathers information from a number of weather web-sites. WeatherViz shows a world map, a region-of-interest map and the weather reports received. This information is put onto the agent information Grid where it is picked up by MBP and other agents and can be used by the operators to inform planning - again, a significant improvement in capability.

The Process Panel (PP), see Figure 5, in the JTF HQ domain is an agent-based tool for providing the Joint Task Force Commander (JTFC) with a campaign 'process' visualisation. One window shows the main military events in a schematic view, whilst the others show other views such as task breakdown and process products. Colours are used to indicate the status of the events. This tool can be used to plan, evaluate, schedule, review and monitor courses of action and, through the use of software agents, the PP can 'sense' remotely the status of activities which increases Coalition situational awareness and provides a warning of possible critical path events.

[14] Knowledgeable Agent-oriented System - from IHMC / UWF and Boeing.

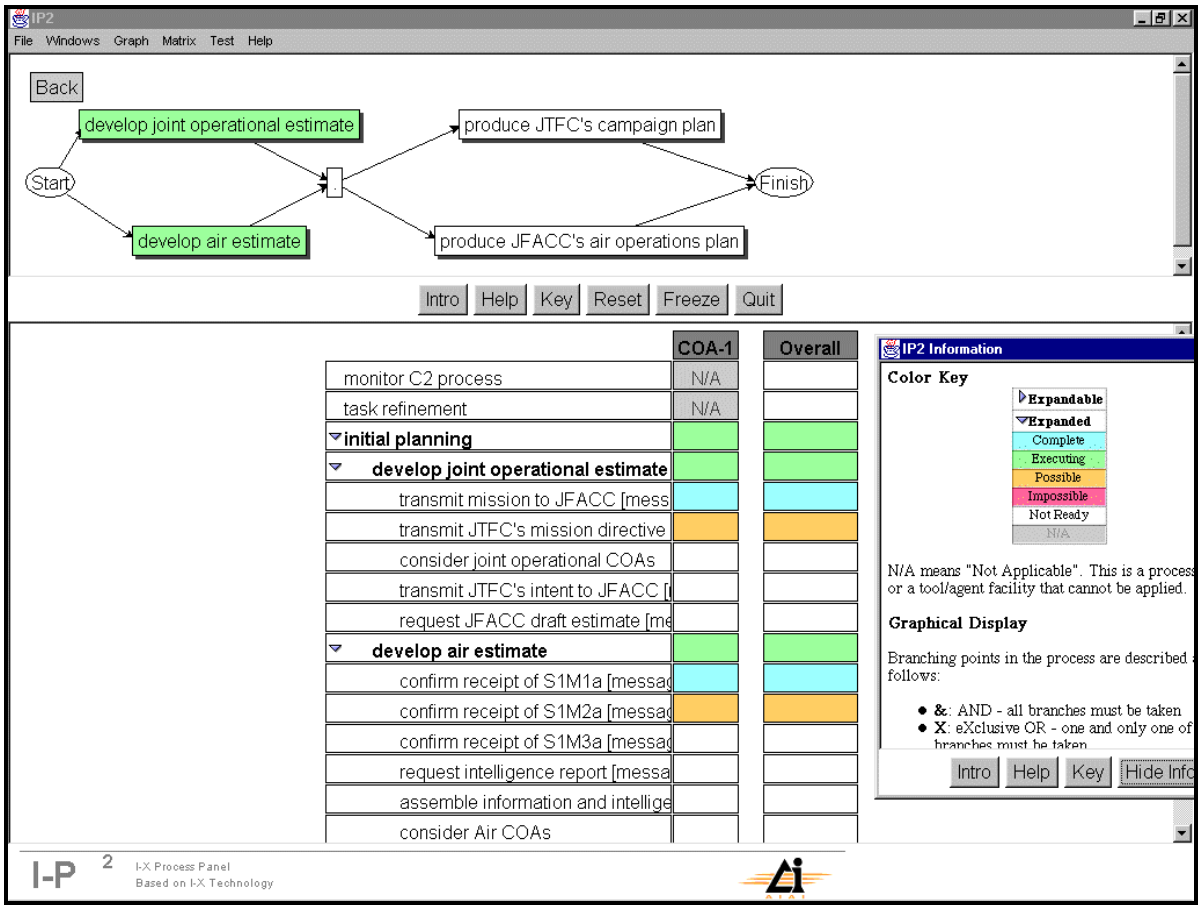


Figure 5 - The Process Panel

The US country domain has access to an agent-enabled tool called the Consolidated Air Mobility Planning System (CAMPS). This consists of a map display and a number of tools for managing airlift assets, their home bases and potential destinations, information about loads and for creating airlift missions. The domain also contains an agent-wrapped air operations data base (AODB) which has US Intelligence and Operations information in it. Some of this information is made available to the Coalition via the software agent wrapper - improving Coalition situational awareness and reducing the chance of conflicts.

In addition, there is a functional domain dealing with feeds from observers in the field. These observers are from several countries, from different parts of the organisation and involve different flavours of agent (eg the Dartmouth Agadez Observer (DAO) is a surrogate agent for D'agents from Dartmouth College, USA). There are also Gao observer agents but, as there is some distrust of these agents, they are put in a more secure, guarded domain which exploits the security and resource control aspects of NOMADS. In particular, the resource control mechanisms can protect hosts from denial-of-service attacks from malicious agents. This use of

software agents *has* allowed greater interoperability, but a risk assessment shows that appropriate measures have also been introduced to protect security.

Section 5 - Experiment Story Board

Several demonstrations have been given of the results of the SACIS / CoAX work and screen movies and briefings are available on the CoAX web site mentioned in Section 1. In addition, the demonstration was successfully given to military staffs in Miami in February 2001 leading to a request for the Team to bid for participation in Millennium Challenge 2002 and JEFX 2002. The demonstrations have used a militarily plausible story board which showed Coalition, Country and Observers information being located, fused and employed to meet military imperatives during initial planning in the Binni scenario.

The demonstration begins in the JFAC HQ where the Air Component Commander's staff bring together information from the various Coalition partners. They use the MBP, whose wrapper agent detects the human activity and which then sends out requests for updates to the various Coalition agent information resources. Once information has been received, the MBP (see Figure 6 below) displays icons for enemy potential targets such as airbases, ground forces, and SAM sites; and for friendly air units, ships and airspace regions.

The JFACC Staffs have also received the JTFC's Guidance documents and they acknowledge their receipt on the MBP. This triggers an agent which informs the JTFC's Staffs of this via the Process Panel, which now shows that the documents have been received by the JFAC HQ, and changes the status of related tasks to show that they can now proceed. This means that software agents are enabling the JTFC to monitor the progression of the overall planning task throughout the Coalition and be alerted quickly if critical-path events are occurring.

Back in the JFAC HQ the staff gather more information about the Theatre of Operations. They need to know about the weather in the Firestorm area, the disposition of Gao and Agadez ground forces, and whether there are any other air missions in the area. An operator uses WeatherViz to select the appropriate weather station and requests an update from Ariadne (which extracts the relevant data from publicly available information on the Internet). Once this information arrives it is sent to the MBP and appears as an icon on the map display. This shows how the software-agent enabled infrastructure can allow operators to access suitable information as and when they need it, regardless of where it is located.

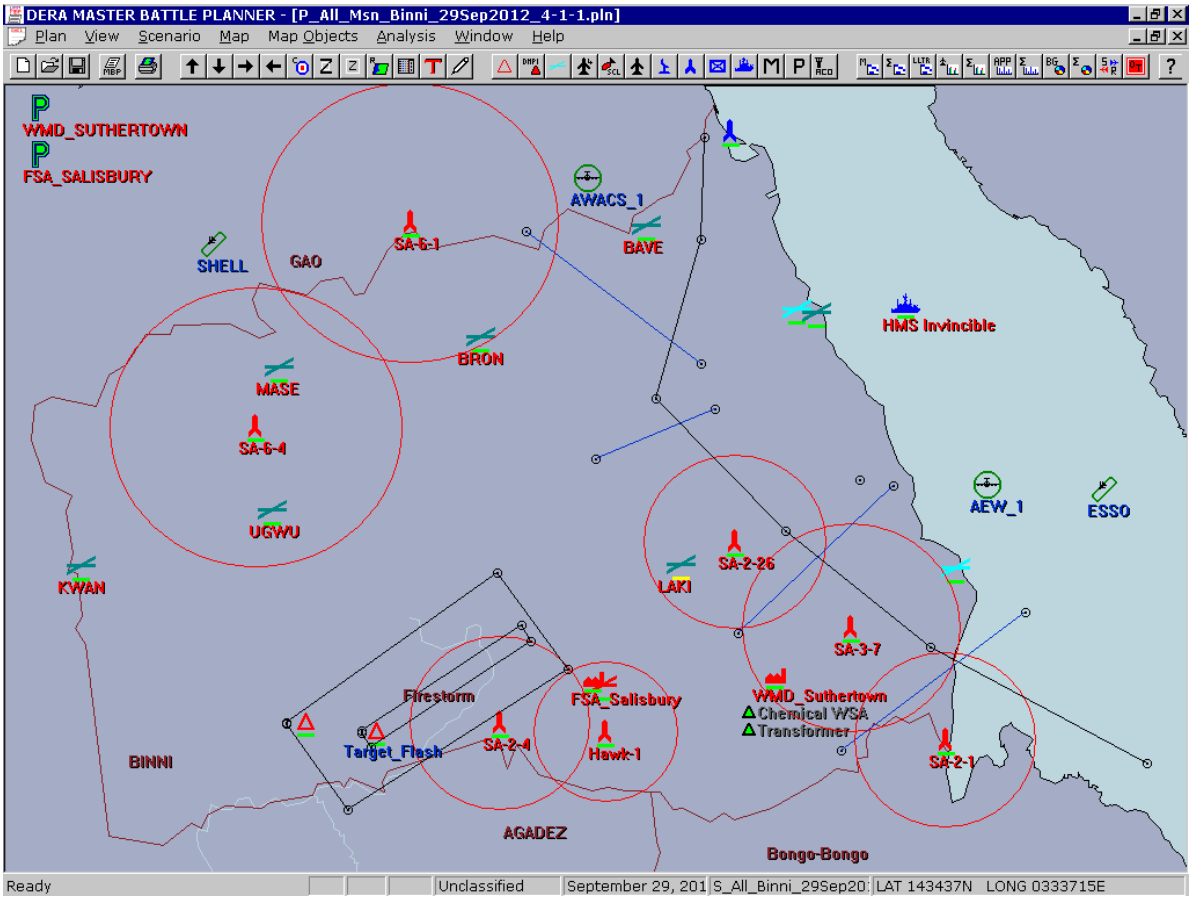


Figure 6 - Master Battle Planner

While this has been going on, in the USA Country Domain, work has started on planning the movement of critical weapons into the Theatre of Operations. The Firestorm requires the use of a new weapon and the USA intends to fly the weapons to Cyprus and then into Theatre. An operator uses the software agent GUI (which could be done remotely with the software agents mediating the interaction) on CAMPS to edit the load and number of aircraft to be sent. The plan is validated by CAMPS and a route generated from Cyprus into Binni. The operator accepts the plan and tells CAMPS to "Achieve it" which means that an order is generated containing all the relevant information. The missions generated are then sent through the agent infrastructure and displayed automatically in the MBP. In the JFAC HQ, when the planners subsequently generate the necessary offensive, defensive and support missions for the Firestorm, as they now know about the airlift missions and won't create other missions which would conflict with them.

Back in the JFAC HQ, the planners are examining the Firestorm options in detail. Based on initial Intel, the RECCE area and possible Firestorm target areas have been defined. However, unknown to the Coalition staffs, Gao doesn't like the Firestorm option and is trying to subvert it by feeding in misinformation about the location (see Figure 2) of Gao and Agadez forces. The Coalition has observers in the field who have been feeding in their observations, which are different to those from Gao. To firm-up the Firestorm options the planners check the currently

known positions of Gao and Agadez forces. They do this by double-clicking on several of the assets on MBP's map display. This triggers software agents to request other agents to return their up-to-date information on the assets - for some of the assets two dialog boxes pop-up containing different information - one from the Gao agents and one from the JFAC HQ's Intel agents. The planners realise the deception and decide to cut off the communication with the Gao agents.

The Coalition System Administrators have access to the KAoS Policy Admin Tool (KPAT) which is used to block access to the Gao domain. The purpose of KPAT is to provide an easy-to-use domain management interface for Coalition agent system administrators (SysAds) to control the behaviour of groups of agents dynamically. Coalition agent policies can be developed and verified in advance and stored in a policy library, or they can be created and enforced on-the-fly. By providing domain management through policies, rather than through requiring agent developers to write special purpose code, three benefits accrue:

- the burden on agent developers is reduced;
- changes in policy can be effected dynamically on software agents to change their behaviour at run-time without changes to the agent code;
- policies can be enforcement on malicious agents without them knowing anything about this in advance or even at all.

A policy is defined to block communication from the Gao agents and it is selected and committed - which changes the domain policies such that the Gao information is excluded. Once access to the Gao domain has been turned off, the planners refresh the MBP's display by requesting an update of information through the Coalition agent network. It is found that many of the ground unit positions alter. This is because they are now only using the accurate information from observers in the field and not on the misinformation from the Gao observer agent. This has demonstrated another feature of the agent infrastructure - robustness and the ability of the system to reconfigure automatically as levels of service change and as entities come and go.

The final part of the demonstration is driven by the fact that Gao is unhappy at this turn of events and covertly begins a denial-of-service attack. At the start of the UNWAFB deployment an 'observer' domain was created containing Dartmouth Gao Observer (DGO) and Dartmouth Agadez Observer (DAO) agents. Gao requested that one of its agents (Gao Agadez Observer, GAO) is put on the sensor platform that is hosting DAO, so that it can observe Agadez movements independently. Because there is some mistrust of Gao, permission is granted, but GAO is required to run under the NOMADS environment, which allows the SysAds to change and monitor the computing resources given to an agent. After the Gao Intel domain is cut off, Gao intends to avenge itself by launching a denial of service attack on the DAO host machine (by writing continuously to hard disk and using up CPU and network resources) by its GAO agent within the observer domain.

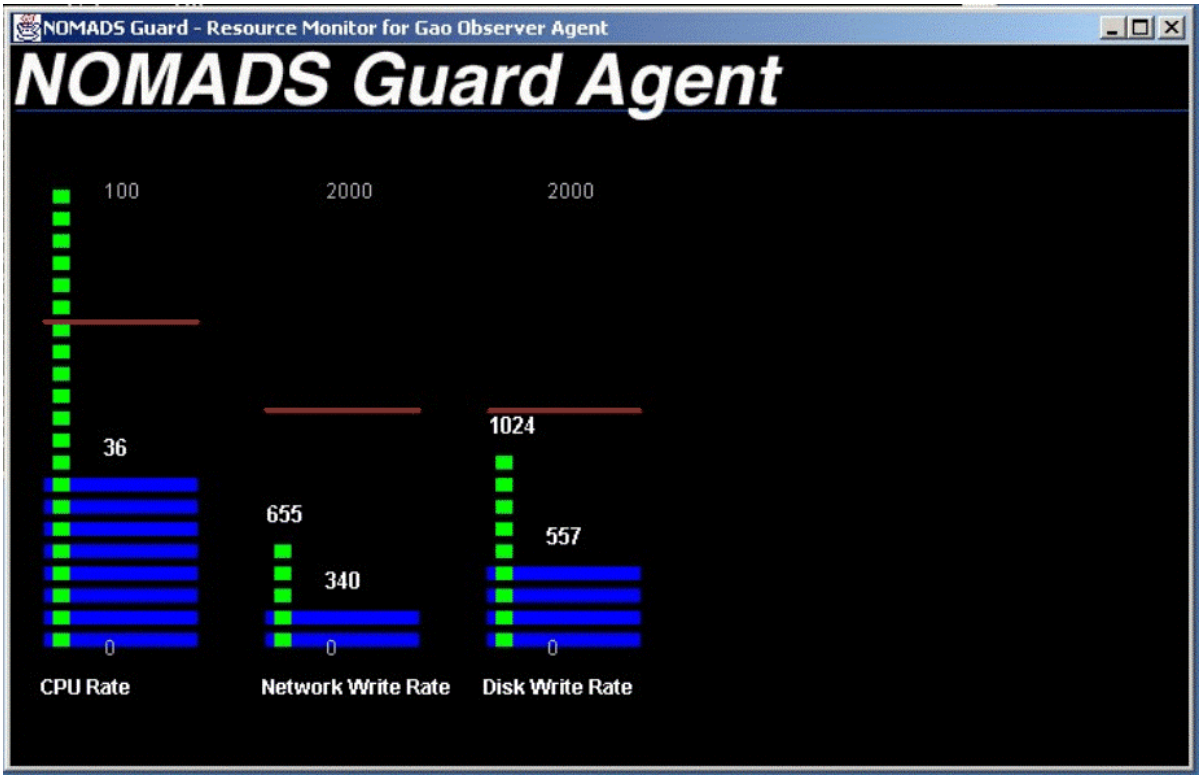


Figure 7 - The NOMADS Guard Display

A sentinel function of the NOMADS guard (see Figure 7 above) monitors the resources used by the GAO agent. At first, the average computing resource consumption of the GAO agent is reasonable, but when the denial-of-service attack is launched, the GAO agent consumes resources excessively. In consequence of the attack, the (friendly) DAO agent's ability to perform its tasks is slowed to a halt. The KAoS domain administration services must not only be able to respond to administrator-requested policy changes, but also automatic event-driven changes. A sentinel function of the NOMADS guard detects the denial-of-service attack. It automatically responds to the attack by reducing the resource limits set (the original limits before the attack are shown as the horizontal red bars in Figure 7) of the GAO agent and in effect neutralises the attack.

In summary, the demonstration has shown part of a realistic UN supported Coalition operation involving several command HQs, organisational, national and functional entities working together in a software agent enabled infrastructure with various levels of access and functionality. The demonstration has shown how legacy systems can be agent-wrapped and interoperability achieved with other agents tools and services - all supported by the CoABS Grid.

Section 6 - Assessment of Software Agents

Technical Progress to Date

To date, the key technical achievements of SACIS and CoAX are as follows:

- we have demonstrated a proof-of-concept, agent-based Coalition C2 architecture within a representative Coalition scenario;
- for the first time, we have shown legacy and previously stand-alone US and UK military systems inter-operating via agent technology;
- we have shown how agent organisation, behaviour, security and resources can be managed by explicit Coalition policy control;
- specifically, the demonstrations have shown certain features and benefits of an agent-enabled environment such as:
    - operators working collaboratively with agents;
    - agents working semi-autonomously 'in the background';
    - the robustness of an agent-enabled infrastructure (the DARPA CoABS Grid) - reconfiguring automatically;
    - access being provided (on demand) to remote, previously stand-alone and dispersed information and remote operation of applications;
    - agents' use of communication and computing resources being controlled;
    - the ability to create shared understanding and improved visualisation.

Assessment and Future Research Programme

The SACIS research started in April 1999 and the CoAX project officially began in February 2000 and we believe that we have made good early progress towards demonstrating the utility of agent technology in Coalition operations. We have put together a prototype Coalition C2 architecture that supports and embraces heterogeneity and have exercised this in an agent-based C2 demonstration that enacts Coalition activities within the Binni scenario. The CoABS Grid and KAoS domain management capabilities have allowed us to interoperate US and UK military systems as well as a variety of agent-based information resources. In should be noted that the CoABS Grid, in particular, has played a vital role in rapid integration of systems.

Assessment work funded by DARPA CoABS programme has reported favourably on the performance issues of agent-enabled infrastructures and the experiences of the SACIS and CoAX Teams have shown that the agent-wrapping of legacy systems and the integration of different agent systems at short notice is relatively straightforward. Indications are that agent wrapping is simpler where systems 'expose' more of their internal information and methods and work is also in hand in this area. In addition, a heterogeneous set of agents can be made to interoperate as long as implementers adhere to some minimum set of message and other standards. Also,

heterogeneity should be accepted and embraced as it is seen as being inevitable and can actually be beneficial in a number of cases - especially in security terms.

The next phases of the research work are designed to further enhance the software agents and provide a more exacting demonstration capability. These phases are described below:

***CoAX Binni 2001*: This work is due in July 2001 and will:**

- move on to supporting the execution phase of conflict (which is characterised by being more uncertain and which has more demanding performance requirements) with software agents - see Figure 8 below;
- take on and address C2 process tracking;
- improve domain management by allowing nested and overlapping domains;
- address both the planning and execution phases of a realistic military scenario;
- involve up to 45 agents in 7 domains that are part of a realistic and extended Coalition C2 architecture.

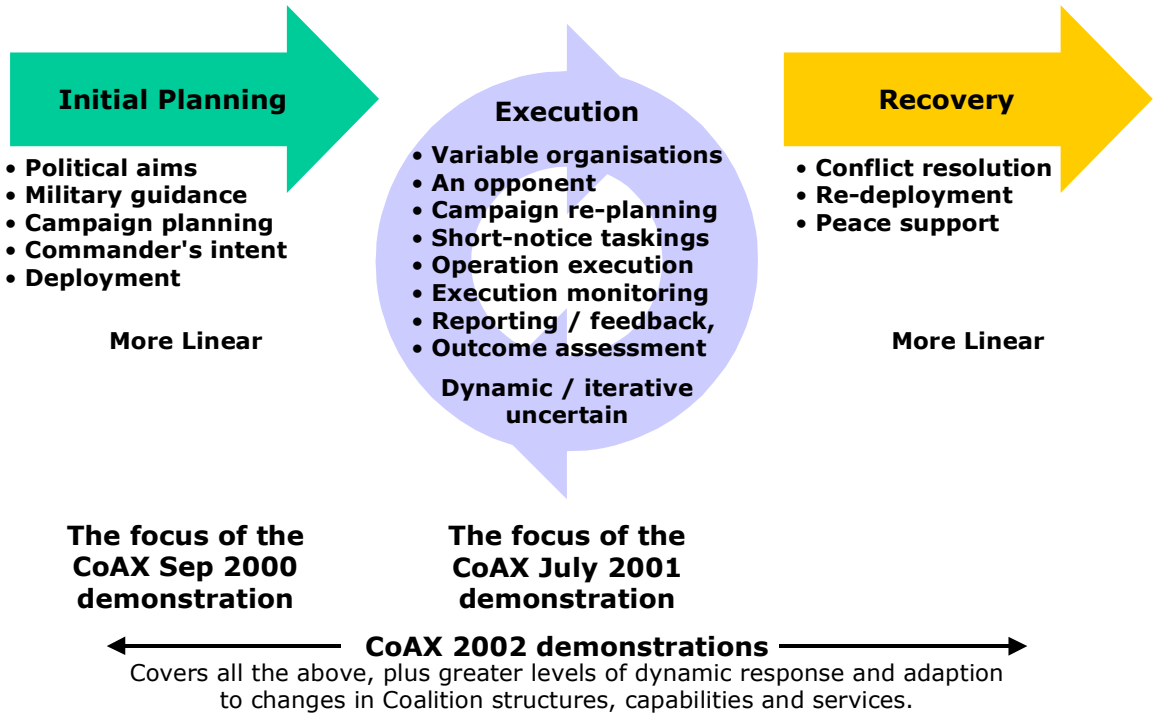


Figure 8 - Phases of an Operation and the CoAX demonstrations

***CoAX Binni 2002*: This work is due in April 2002 and will:**

- add the ability to monitor, plan and control the Coalition C2 processes and deal with events arising from execution;
- include obligation management, eg: ensure that agents are meeting their commitments;
- involve the dynamic creation of 'virtual Coalition organisation';
- see agents and domains added to Coalition structure 'on-the-fly';
- deal with dynamic Coalition tasks and processes;
- use a dynamic Coalition scenario, with partners joining / leaving unpredictably;
- cope with a dynamic, uncertain and event-driven execution environment.

This work has been submitted for inclusion in Millennium Challenge / JEFX 2002.

In the research so far, a certain amount of hardwiring and pre-agreed formats have been employed and in the next phase of the work semantic web technology will be used to investigating further how run-time interoperability can be achieved 'on-the-fly' as it is felt that semantic interoperability can significantly improve agent collaboration.

Military Implications of the Results

Software Agents can be viewed as entities acting on behalf of, mediating or supporting the actions of human users and having the ability to carry out tasks autonomously to achieve goals or assist the activities of the users. This research project has shown how software agents can carry out tasks which enable interoperability between information systems and infrastructure services brought together in a 'come-as-you-are' Coalition.

In the experiments so far, the software agents operated in a number of roles. They have worked 'in the background' – through matchmaking, domain management, process management and other agent services – to find, establish and maintain the infrastructure, information and procedural links necessary to achieve and support interoperability in a dynamically changing environment. In addition, they have worked collaboratively with human operators, mediating requests for information and formatting and displaying the results almost transparently.

Dealing effectively with unpredictable changes – owing, for example, to the destructive activities of opponents or because of systems failing and services being withdrawn – is a typical Coalition problem where software agents could make a significant contribution. So far, we have shown that a software agent infrastructure is robust and, to some extent, is 'self-healing'. Our aim is to investigate this further to show that software agents can provide agility, robustness, flexibility and additional functionality beyond that provided by the individual Coalition partners.

Our conclusion is that software agents, together with agent-based infrastructures and services provided by the CoABS Grid and KAoS, could play a key role in supporting Coalition operations. We think that this technology will provide the ability to bring together and integrate

systems quickly to aid in all aspects of Coalition operations, without sacrificing security and control. Our long-term goal is to use this technology in the creation, support and dynamic reconfiguration of virtual organisations - with Coalitions being an archetypal and timely example of an area where this technology is vitally needed.

Section 7 - Concluding Remarks

The central hypothesis that is being investigated in SACIS and CoAX is that the agent-based computing paradigm is a good fit to the kind of computational support needed in Coalition operations. Thus far, we have shown that agents can usefully share and manage access to information across a stylised Coalition architecture in support of planning. This has required our gaining knowledge and expertise in, for example, agent communication languages, agent infrastructures, agent policy management [12], and agent legacy-system integration.

Over the next year, SACIS and CoAX have a series of technical demonstrations planned of increasing complexity. Building on the baseline capabilities already established, in the next phases of CoAX our priorities will be on demonstrating how agent technology can deal with the dynamic aspects of Coalition formation and the uncertainties of the execution phase where we will face an opponent. We envisage showing partners joining or leaving, services becoming unavailable, as well as agents aiding the human problem-solving and decision-making process in response to external events.

One early lesson has been that the Cyberspace inhabited by the software agents should not be seen just as an information pipe between humans - it is a Battlespace in its own right. This indicates that 'Cyberspace Superiority' should be obtained (as for any other part of the Battlespace) and that to achieve this it is essential that the Coalition Forces are able to *act decisively inside Cyberspace*. As humans cannot physically enter Cyberspace, it may be that the only way that Cyberspace Superiority can be achieved is though the use of a variety of software agents acting on behalf of or mediating the actions of human users.

References

1. University of Maryland Baltimore County AgentWeb: http://agents.umbc.edu/

2. Jennings, N R, Sycara, K, and Wooldridge, M. 1998. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1:275-306.

3. Huhns, M N, and Singh, M P (editors) 1998. *Readings in Agents*. Morgan Kauffman: San Francisco, California.

4. Bradshaw, M (editor) 1997. *Software Agents*. AAAI Press: Menlo Park, California.

5. *IEEE Intelligent Systems* 1999. 14(2).

6. Shoham, Y. 1999. What we talk about when we talk about software agents. *IEEE Intelligent Systems*, 14(2): 28-31.

7. Labrou, Y, Finin, T, and Peng, Y. 1999. Agent communication languages: the current landscape. *IEEE Intelligent Systems*, 14(2): 45-52.

8. DARPA Agent Mark-Up Language: http://www.darpa.mil/iso/ABC/BAA0007PIP.htm and http://dtsn.darpa.mil/iso/programtemp.asp?mode=347 and http://www.daml.org/

9. Jennings, N R. An Agent-based Approach for Building Complex Software Systems. *Communications of the ACM*. Vol 44, No: 4. pp. 35 to 41. April 2001.

10. Rathmell, R.A. (1999) A Coalition Force Scenario 'Binni - Gateway to the Golden Bowl of Africa', In *Proceedings of the International Workshop on Knowledge-Based Planning for Coalition Forces*, (ed. Tate, A.) pp. 115-125, Edinburgh, Scotland, 10th-11th May 1999.

11. Pawson, R. Expressive Systems, 2000. http://expressive-systems.org/

12. Bradshaw, J., Suri, N., Kahn, M., Sage, P., Weishar, D. & Jeffers, R. Terraforming Cyberspace: Toward a policy-based grid infrastructure for secure, scalable, and robust execution of Java-based multi-agent systems. Proceedings of the Workshop on Agent-based Cluster and Grid Computing, *IEEE International Symposium on Cluster Computing and the Grid* (CCGrid2001), Brisbane, Australia, 14-18 May, 2001. (Enlarged version to appear in IEEE Computer, in press).

Acknowledgements

DERA work was carried out as part of the Technology group 10 of the UK MOD Corporate Research programme. We also acknowledge the contributions of the following CoAX partners:

- Ariadne Agents - Public Domain Weather provision (USC / ISI).
- Consolidated Air Mobility Planner (CAMPS) - legacy airlift tool (GITI / BBN / AFRL).
- Domain Management - Knowledgeable Agent-oriented System (KAoS) (UWF / IHMC and Boeing).
- EMAA / CAST agents - US Air Intelligence Provision (Pete Gerken, LM-ATL).
- Malicious Agent control - KAoS Policy Administration Tool (Boeing, UWF / IHMC).
- Master Battle Planner (MBP) agent-wrapped, legacy planning tools for Air Battle Planning (Chris Walker and Don Brealey, DERA).
- NOMADS Mobile Agent System (UWF / IHMC and Boeing).
- Observer D'agents (Dartmouth).
- Process and Task Management tools (AIAI).
- The DARPA "Grid" - an agent-enabled infrastructure (GITI, ISX).

Edinburgh work on the CoAX project is sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Command and Control Directorate under grant numbers F30602-99-1-0024. UWF / IHMC work on KAoS and NOMADS is supported by a contract from DARPA's Control of Agent-based Systems (CoABS) program (Contract F30602-98-C-0170), the DARPA Ultra*Log program, the NASA Cross-Enterprise and Intelligent Systems programs, and the National Technology Alliance.

Intentionally Blank

Appendix H: Coalition Agents Experiment: Multi-Agent Co-operation in an International Coalition Setting

David N. Allsopp, QinetiQ
Patrick Beautement, QinetiQ
Jeffrey M. Bradshaw, University of West Florida
Edmund H. Durfee, University of Michigan
Michael Kirton, QinetiQ
Craig A. Knoblock, University of Southern California
Niranjan Suri, University of West Florida,
Austin Tate, AIAI
Craig W. Thompson, Object Services and Consulting Inc.

Abstract: Military Coalitions are examples of large-scale multi-faceted virtual organizations, which sometimes need to be rapidly created and flexibly changed as circumstances alter. The Coalition Agents eXperiment (CoAX) aims to show that multi-agent systems are an effective way of dealing with the complexity of real-world problems, such as agile and robust Coalition operations and enabling interoperability between heterogeneous components including legacy and actual military systems. CoAX is an international collaboration carried out under the auspices of DARPA's Control of Agent-Based Systems (CoABS) program. Building on the CoABS Grid framework, the CoAX agent infrastructure groups agents into domains that reflect real-world organizational, functional and national boundaries, such that security and access to agents and information can be governed by policies at multiple levels. A series of staged demonstrations of increased complexity are being conducted in a realistic peace-enforcement scenario situated in 2012 in the fictitious African state of "Binni". These demonstrations show how agent technologies support the rapid, co-ordinated construction of a Coalition command system for intelligence gathering, for visualization, and for campaign, battle and mission planning and execution.

Citation: Allsopp, D.N., Beautement, P., Bradshaw, J.M., Durfee, E.H., Kirton, M., Knoblock, C.A., Suri, N. and Tate, A. and Thompson, C.W. (2002) "Coalition Agents Experiment: Multi-Agent Co-operation in an International Coalition Setting", Proceedings of the Second International Conference on Knowledge Systems for Coalition Operations (KSCO-2002), Toulouse, France.

Introduction and Background

Military Context

Success in military operations involves carrying out high-tempo, coherent, decisive actions faster than an opponent can react, resulting in decision dominance through the use of command agility. Command agility is about being flexible and adaptable so that fleeting opportunities can be grasped; the Commander issues clear intent and then delegates control to subordinates, allowing them the scope to exercise initiative. It also means being innovative, creative and unpredictable in a manner that (even if low-tempo) increases confusion in the mind of an opponent. This process is command led; human decision-making is primary and the role of technology is secondary. Shared understanding and Information Superiority are key enablers in this process and are fundamental to initiatives such as the UK's Command and Battlespace Management program, the US Joint BattleSpace Infosphere program and, more generally, Network-Centric Warfare (http://www.dodccrp.org/).

In addition to the problems of integrating single-service and Joint capabilities into a coherent force, the nature of Coalition (multi-national) operations implies some need to rapidly configure foreign or 'come-as-you-are' systems into a cohesive whole. Many problems in this environment can only be solved by organizational changes and by 'aligning' doctrine, concepts of operations and procedures. Due to the inevitable absence of pre-existing co-ordinated systems, Coalition scenarios require a rapid, flexible, on-the-fly approach that allows capabilities to be assembled at run-time. However, in addressing this requirement for interoperability, it is also crucial to address issues of security of data, control over semi-trusted software from other Coalition partners, and robustness of the resulting system (e.g. the ability to withstand denial-of-service attacks).

Currently, Coalition operations are often characterized by data overload, information starvation, labor intensive collection and co-ordination of information, and standalone stove-pipe command systems that use incompatible data formats. This leads to a horrendous technical integration task and gives commanders only scattered snapshots of the battlespace. This paper aims to show that the agent-based computing paradigm offers a promising new approach to dealing with such issues by embracing the open, heterogeneous, diverse and dispersed nature of the Coalition environment. In this paper, we show that software agents that act on behalf of human users enable military commanders to act decisively in cyberspace and thus contribute towards the achievement of 'Cyberspace Superiority', a critical component of warfare in the information age (Alberts et al, 2001).

Software Agent Technology

Software agents are currently receiving much attention in the research community. This interest is being driven by the phenomenal growth of the Internet and the World-Wide-Web. Agents can be viewed as semi-autonomous software designed to help people cope with the complexities of working collaboratively in a distributed information environment. This involves the agents communicating between the users and between themselves. The agents are used to find, format,

filter and share information, and work with users to make the information available wherever and whenever they need it. The agents are also able to proactively suggest courses of action, monitor mission progress, and recommend plan adjustments as circumstances unfold.

A community of agents can be seen as a set of distributed, asynchronous processes communicating and sharing information by message passing in some infrastructure. In this regard, an important output from DARPA's CoABS program is the CoABS Grid — a middleware layer based on Java / Jini technology that provides the computing infrastructure to integrate heterogeneous agent communities and systems rapidly (http://coabs.globalinfotek.com/).

A recent article (Jennings, 2001) argues that the agent paradigm is a good way of building complex software systems in general, and hence offers potential benefits in the Coalition setting. For example, legacy command systems could be provided with software agent wrappers that allow them to inter-operate and share information with other systems and agent applications in a loosely connected, heterogeneous architecture, underpinned by the CoABS Grid. The scenario used as the basis of the experiments to test this hypothesis is described in section 2.

Aims of the CoAX Project

This paper describes the progress of an international collaborative effort whose overall goals are to demonstrate that the agent-based computing paradigm offers a promising new approach to dealing with the technical issues of establishing coherent command and control (C2) in a Coalition organization. This collaborative effort, entitled CoAX (Coalition Agents eXperiment), is a Technology Integration Experiment under the auspices of DARPA's Control of Agent Based Systems (CoABS) program (http://www.aiai.ed.ac.uk/project/coax/). Specific hypotheses of the research program are that:

- agents are a useful metaphor for dealing with the complexity of real-world systems such as military operations;
- an agent-based C2 framework can support agile and robust Coalition operations;
- software agents can be used to enable interoperability between legacy or previously incompatible systems;
- the CoABS Grid can be used to rapidly integrate a wide variety of agents and systems — i.e., rapid creation of virtual organizations;
- domain policies can structure agent relationships and enforce Coalition policies;
- intelligent task and process management can improve agent collaboration;
- semantic web technology can improve agent interoperability between disparate Coalition command systems.

The CoAX team has built a software agent test-bed based on the CoABS Grid (http://coabs.globalinfotek.com/). This paper describes the work done, the demonstrations carried out so far, the scenario and storyboard used and some of the insights gained.

Structure of the Paper

The paper begins by describing the Coalition scenario and military command structure used in our demonstration experiments. Section 3 describes the corresponding agent architecture that was developed to reflect the military organizational structure. The events occurring in the storyboard used for the various demonstrations so far are described in Section 4. A preliminary assessment of software agent capabilities and a discussion of future research are provided in Section 5. Concluding remarks are given in Section 6.

A Representative Scenario and Coalition Command Structure

The CoAX work needed a suitably realistic scenario for its experiments and so we expanded the fictional "Binni" scenario (Rathmell, 1999) developed for The Technology Co-operation Programme. In this scenario the year is 2012 and global warming has altered the political balance of the world. The action is set in an area that is currently the Sudanese Plain (Figure 1). Previously uninhabited land in the Plain is now arable and the area has received large amounts of foreign investment. It is now called “The Golden Bowl of Africa”.

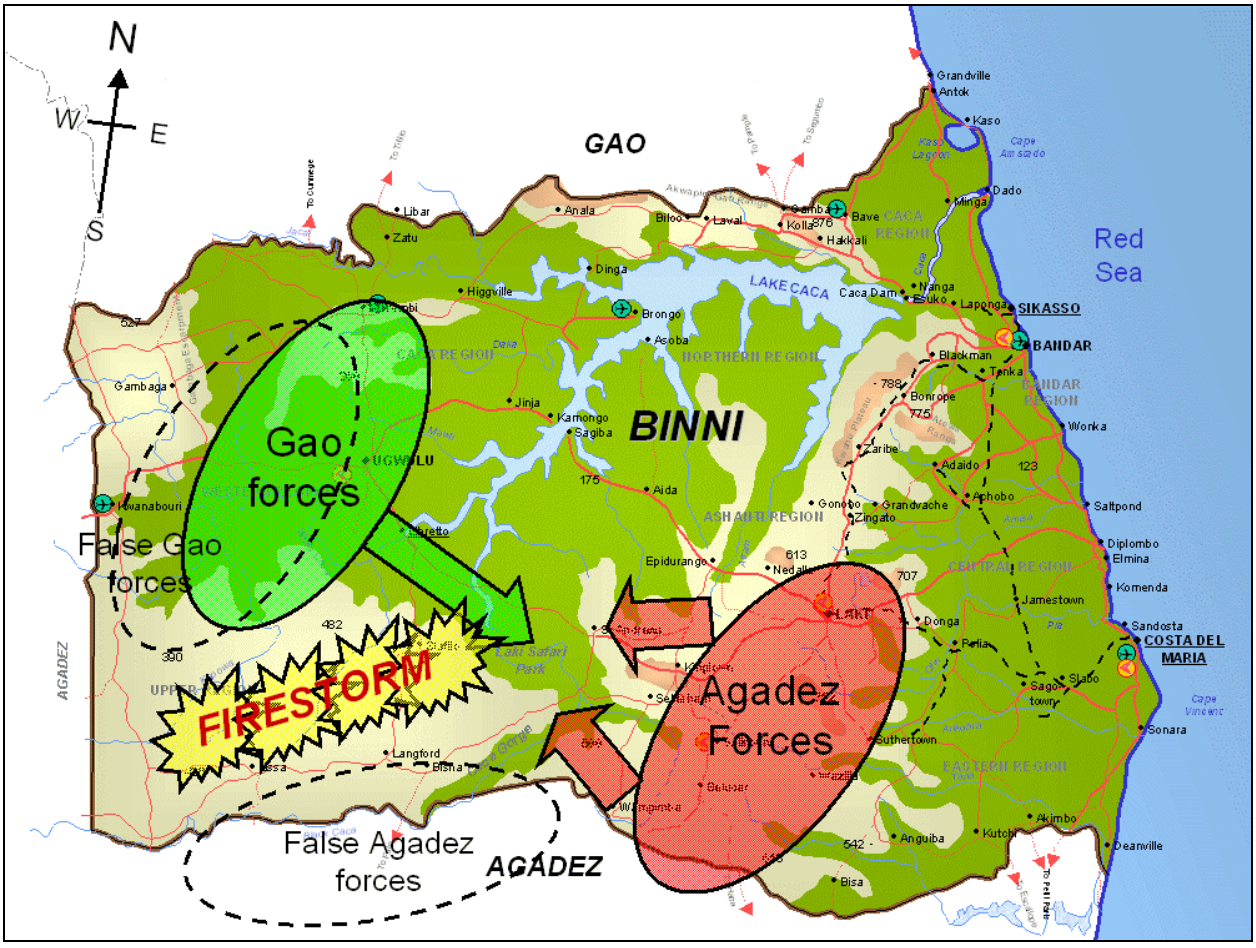


Figure 1. Map of Binni showing firestorm deception. Misinformation from Gao is intended to displace the firestorm to the west, allowing Gao and Agadez forces to clash in the region of the Laki Safari Park.

A conflict has developed between two countries in the area. To the north is Gao, which has expansionist aspirations but which is only moderately developed, with old equipment and with a mostly agrarian society. To the south is Agadez, a relatively well developed and fundamentalist country. Gao has managed to annex an area of land, called it Binni and has put in its own puppet government. This action has come under fierce attack from Agadez. Gao has played the 'threat of weapons of mass destruction from Agadez' card and has enlisted support from the UN who have deployed a force, the UN War Avoidance Force for Binni (UNWAFB), to stabilize the region. This basic scenario was adapted for a number of CoAX demonstrations (see Section 4), beginning with the initial planning phase, then moving onto shorter timescales and more dynamic, uncertain events for the execution phase.

Coalition Command Structure

This Binni Coalition operation needs to rapidly configure various incompatible, 'come-as-you-are' or foreign systems into a cohesive whole within an open, heterogeneous and dispersed environment. This scenario provides a suitable test for the software agent experiments, where run-time composability is a very close metaphor for the dynamic uncertainty of Coalition operations. The complexity of the situation must not be underestimated and is best illustrated by looking at the Binni Coalition Command Structure shown in Figure 2 below.

This is a representative and realistic Coalition command structure involving the UN, Governments, Other Government Departments (OGDs, such as the Foreign Office), Non-Government Organizations (NGOs, such as Oxfam), representatives of all the Coalition countries (with their own 'ghosted' Command Structures) and the Coalition HQs and subordinate fighting forces. The solid black lines on the diagram show the legal lines of authority (the command chain) and accountability. This is the kind of Coalition structure that would be agreed by the participants; no part of the formal command chain is owned by any specific country. Note that the 'levels of command' overlap and their boundaries are not rigidly defined. Dashed lines show an advisory / negotiating role.

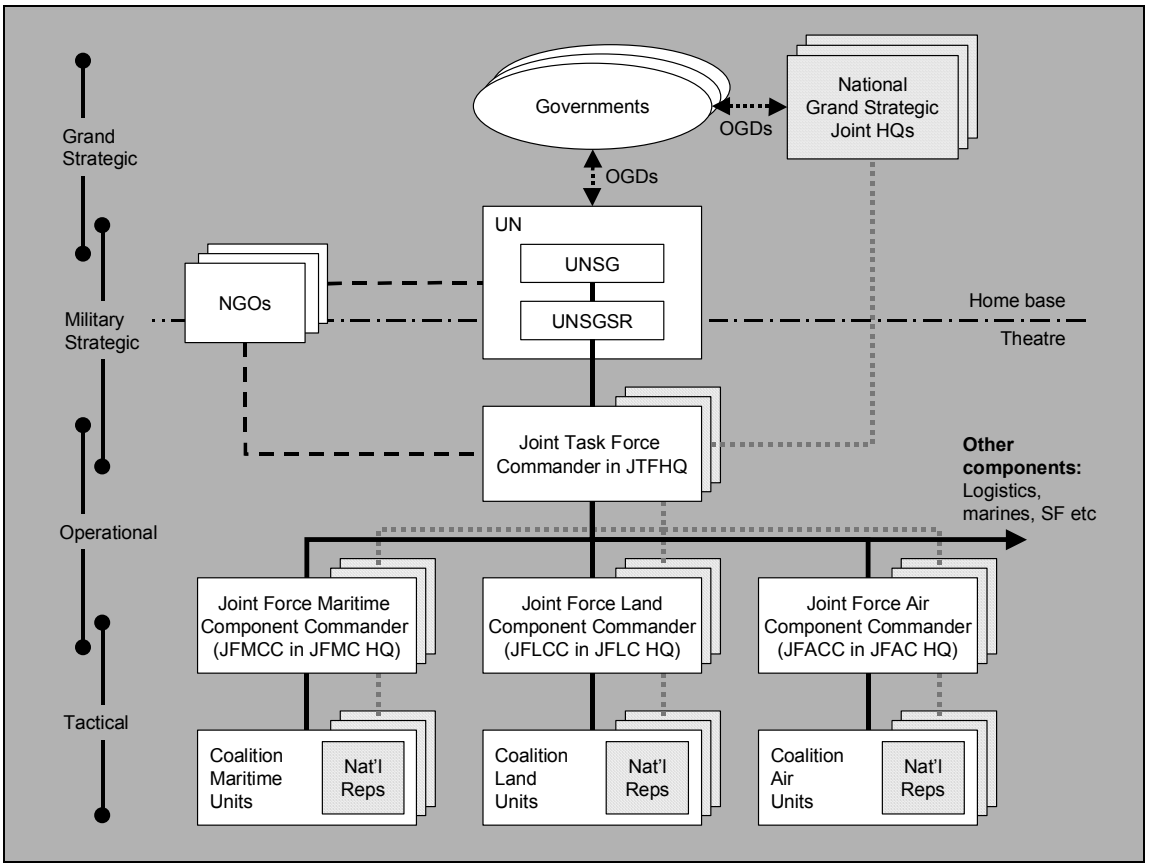


Figure 2: A representative Coalition structure, showing the chain of command down from the United Nations, including the 'ghosted' command structures of the participant nations, and Non-Government Organizations (NGOs). The approximate command level at which the various entities operate is indicated on the left.

Software Agent Architecture

Human Domains

Integrating information across a Coalition is not just a matter of employing technology — it involves the creation of a coherent 'interoperability of the mind' at the human level as well, where many social and cultural factors come into play. The mapping between the human and technical worlds is thus not straightforward. From the human perspective, we identified four kinds of 'domains':

- **Organizational Domains**: for example the Joint Task Force HQ (JTF HQ) ;
- **Country Domains**: each of the National command chains would be a separate, self-contained domain;
- **Functional Domains**: sets of entities collaborating on common tasks, for example Meteorology or Intelligence ;

- **Individual Human Domains of Responsibility**: Commanders have responsibility for their own HQ and all subordinate ones (in practice they delegate). Hence the individual human domains of influence may overlap.

These types of domains are not entirely exclusive and there are many different levels of overlap and interaction depending on the viewpoint taken. It is this complexity at the human level that creates difficulties for technical systems.

Software Agent Domains

COABS GRID INFRASTRUCTURE

At the most basic level, the agents and systems to be integrated require infrastructure for discovery of other agents, and messaging between agents. The CoABS Grid provides this. Based on Sun's "Jini" services which are themselves based upon Java's Remote Method Invocation, the Grid allows registration and advertisement of agent capabilities, and communication by message-passing. Agents on the Grid can be added or removed, or their advertisements updated, without reconfiguration of the network. Agents are automatically purged from the registry after a short time if they fail. Multiple lookup services may be used, located by multicast or unicast protocols. In addition, the Grid provides functionality such as logging, visualization, and more recently encryption of messages and agent authentication.

KAOS DOMAIN MANAGEMENT

The increased intelligence afforded by software agents is both a boon and a danger. By their ability to operate independently without constant human supervision, agents can perform tasks that would be impractical or impossible using traditional software applications. On the other hand, this additional autonomy, if unchecked, also has the potential of effecting severe damage to military operations in the case of buggy or malicious agents. The Knowledgeable Agent-oriented System (KAoS) provides services that help assure that agents from different developers and running on diverse platforms will always operate within the bounds of established policies and will be continually responsive to human control so that they can be safely deployed in operational settings (Bradshaw et al., 1997, 2001). KAoS services and tools are intended to allow for the specification, management, conflict resolution, and enforcement of policies within the specific contexts established by complex military organizational structures.

KAoS domain management services can be used to group agents into logical domains corresponding to organizational structures, administrative groups, and task-oriented teams. Within CoAX, these domains mirror the human domains described above, allowing for complex hierarchical, heterarchical, and overlapping structures. An agent domain consists of a unique instance of a domain manager (DM) along with any agents that are registered to it. Alternatively, an intensionally-defined domain consists of a set of agents sharing one or more common properties (e.g., the domain of all agents physically residing on some host). The function of a domain manager is to manage agent registration, and serve as a single point of administration

and enforcement for domain-wide, host-wide, VM-wide, VM-container-wide, or agent-specific policies.

DOMAIN POLICIES

A policy is a declarative constraint governing the behavior of one or more agents, even when those agents may not be domain-aware or where they may be buggy or malicious. For example, a policy may be declared that all messages exchanged among agents in the JFAC HQ domain must be encrypted, or that an agent cannot simultaneous belong to the US and the UK domain. A policy does not tell the agent how to perform its task; it rather specifies the conditions under which certain actions can be performed. By way of an analogy to traffic management, it is more like a set of individually-customizable stop signs and highway patrol officers that define and enforce the rules of the road than it is like a route planner that helps agents find their way to their destinations.

Policies governing authorization, encryption, access control, and resource control are part of KAoS domain management. However, due to our focus on agent systems our scope goes beyond these typical security concerns in significant ways. For example, KAoS pioneered the concept of agent conversation policies (Bradshaw et al., 1997). Teams of agents can be formed, maintained, and disbanded through the process of agent-to-agent communication using an appropriate semantics. In addition to conversation policies, we are developing representations and enforcement mechanisms for mobility policies, domain registration policies, and various forms of obligation policies. These policies are represented in ontologies using the DARPA Agent Markup Language (DAML), and an efficient description logic-based approach is used as the basis for much of the domain manager's reasoning to discover and resolve policy conflicts and to perform other kinds of policy analysis.

The separation of policy specification from policy-enforcement mechanisms allows policies to be dynamically re-configurable, and relatively more flexible, fine-grained, and extensible. Agent developers can build applications whose policies can change without necessarily requiring changes in source code. The rationale for using declarative policies to describe and govern behavior in agent systems includes the following claims: easier recognition of non-normative behavior, policy reuse, operational efficiency, ability to respond to changing conditions, and the possibility of off-line verification.

Software Agent Domains in CoAX

The CoAX demonstrations contain software agents grouped into agent domains using the CoABS Grid, with the policies enforced by KAoS domain management services. A typical domain configuration is shown in Figure 3.

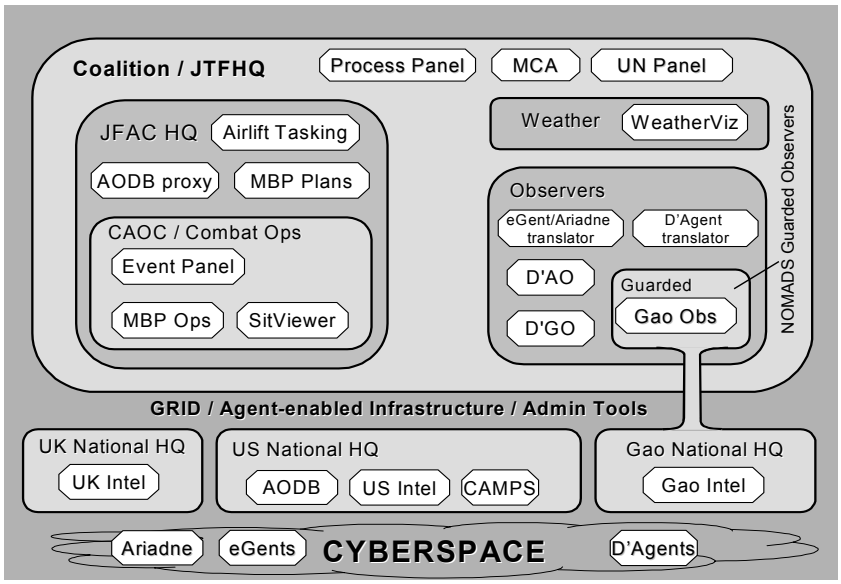


Figure 3. Typical CoAX domain structure; domains are indicated by rounded rectangles; agents by angled rectangles. Some agents are proxies for agents or legacy systems that are not themselves domain aware. Each domain would also contain a Domain Manager agent and a Matchmaker agent (omitted for clarity). Nesting of domains indicates a hierarchy of responsibility and policy control. The agent acronyms are expanded in the body text.

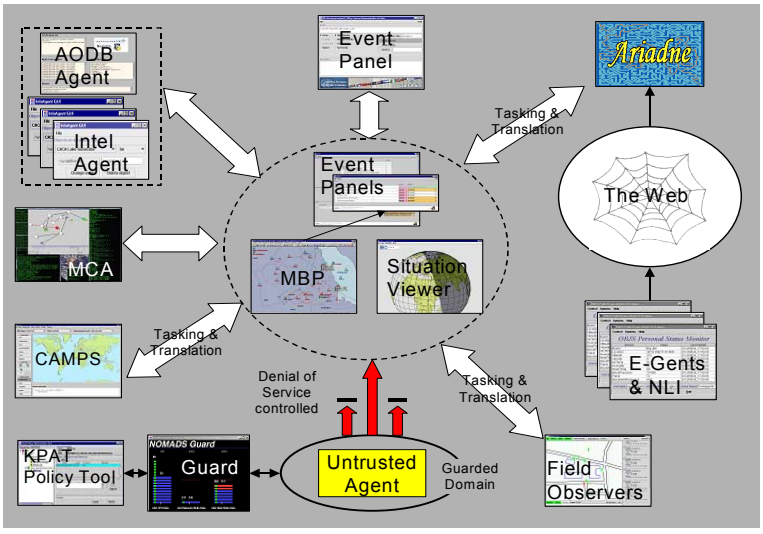


Figure 4: Overview of technologies and agents. The central visualization and planning tools find and acquire data (e.g. disposition of ground forces) and services (e.g. airlift scheduling and plan deconfliction) from the other agents and systems, in some cases via intermediate tasking and translation agents. MBP = Master Battle Planner, MCA = Multi-level Coordination Agent, KPAT = KAoS Policy Admin Tool, AODB = Air Operations Data Base, NLI = Natural Language Interface, CAMPS = Consolidated Air Mobility Planning System.

Demonstration Storyboard and Technologies
In this section we progress through the storyboard created for the Binni Scenario, and describe each of the agent systems and technologies brought into play for each part of the story. An overview of the interactions from the agent/system point of view is shown in Figure 4.

Population of Domains
Following the outbreak of hostilities, the UN has deployed their UN War Avoidance Force for Binni (UNWAFB), to stabilize the region. The active Coalition participants at this time are the UK, US and Gao.

In agent terms, a variety of agent domains are set up using the CoABS Grid infrastructure and the KAoS domain management services, representing the organizational structures (the JTF HQ and the JFAC HQ), the nations (UK, US, Gao) and various functional domains such as Weather and Observers. These domains are populated with a number of agents, which register with their Domain Manager and optionally advertise their services with their domain Matchmaker.

Data Gathering and Air Planning
After exploring options to separate the opposing forces and restore the peace in the region, the deployment of a large ground observation and peace enforcement force and other courses of action have been rejected, and a 'Firestorm' mission has been decided upon. This will clear land to enable simpler remote and ground observations with less risk to the Coalition peacekeepers. The Coalition undertakes initial information gathering and planning.

MASTER BATTLE PLANNER (MBP)

Air planning at the JFAC is performed using QinetiQ's MBP, a highly effective visual planning tool for air operations. MBP assists planners by providing them with an intuitive visualization on which they can manipulate the air intelligence information, assets, targets and missions, using a map-based graphical user interface (Figure 5). This enables an operator to build a battle scenario containing targets, offensive and defensive units, airspace measures and other objects using simple dialogs and point-and-click techniques on the map. Objects on the map can then be moved around, and their properties can be changed. Information such as the allegiance and status of units, and the ranges of units may also be displayed.

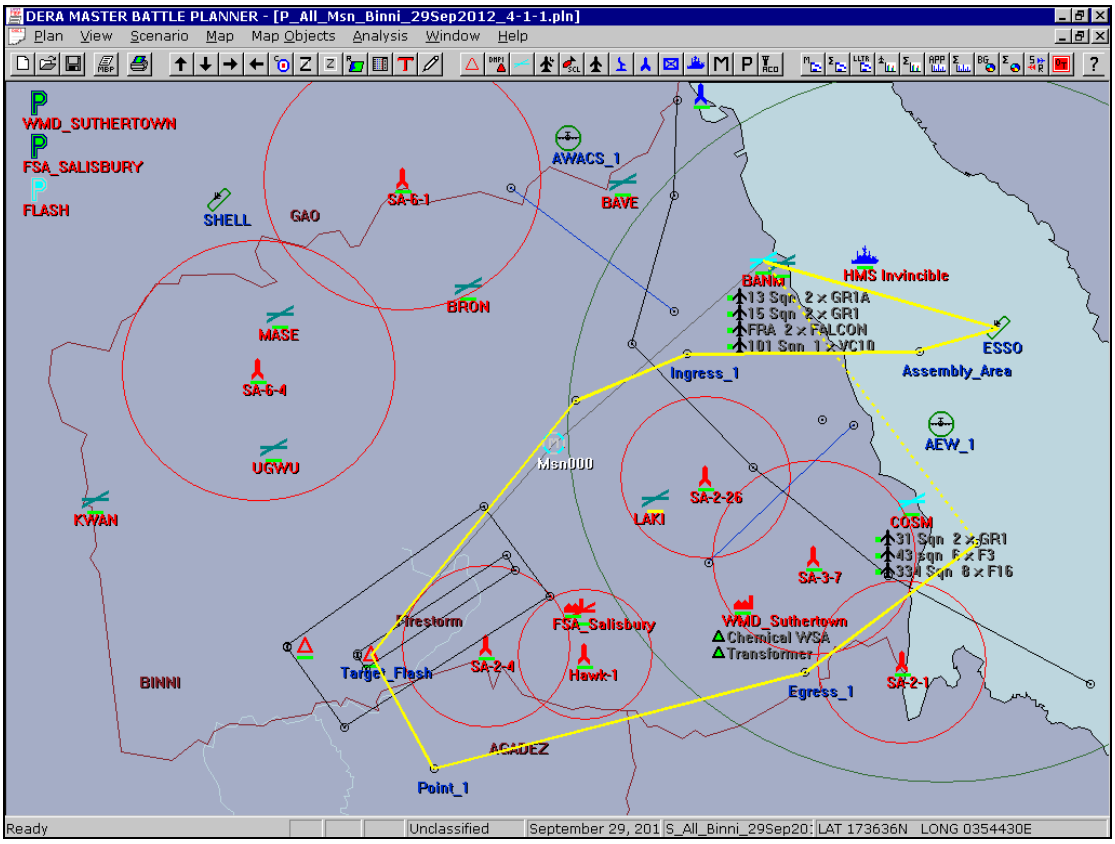


Figure 5: Master Battle Planner map display of the fictional countries of Binni, Gao, and Agadez. A selected mission is highlighted, proceeding from an airbase (BANM), to refueling tanker (ESSO), to the target via waypoints and airspaces, and back to base by a different route.

The operator can interact with these entities and can plan individual air missions (or more complex packages of missions) by dragging and dropping offensive units onto targets on the map. Supporting / defensive elements are added in the same way. The system gives the operator analytical tools to assess the planned air operations for:

- the best utilization of resources; (e.g. by highlighting air units that are over-tasked);
- time-phasing (through charts and animated 'fly-out');
- concordance with the military guidance given.

MBP is a monolithic C++ application, which has been agent-enabled by wrapping it in Java code, using the Java Native Interface. The agent-enabling of MBP allows it to receive all the scenario data (targets, assets, airspaces etc.) from multiple information-providing agents ('Intel Agents' — see Figure 4) and update this information in near-real time. Importantly, this process is integrated into the normal usage of MBP; when an operator views the status of an object, agents are automatically tasked to update the information. Agents may also 'push' changes of status to MBP. Information concerning other air missions can be accepted and merged with

missions planned within MBP, as described below. Missions can also be saved and exported, enabling other agents to reason with the data.

CONSOLIDATED AIR MOBILITY PLANNING SYSTEM (CAMPS)

The second real military system integrated into the demonstration is the Air Force Research Laboratory's CAMPS Mission Planner (Figure 6). CAMPS develops schedules for aircraft to pick up and deliver cargo within specified time windows. It takes into account constraints on aircraft capabilities, port handling capabilities, crew availability and work schedule rules, etc. Users of the planner develop plans (schedules) for aircraft to carry a particular cargo, specifying the intermediate ports, air refueling tracks and the kinds of crews that will be available. They can also specify a number of constraints on the airports potentially involved in the plans to be developed (Emerson & Burstein, 1999; Burstein et al, 2000).

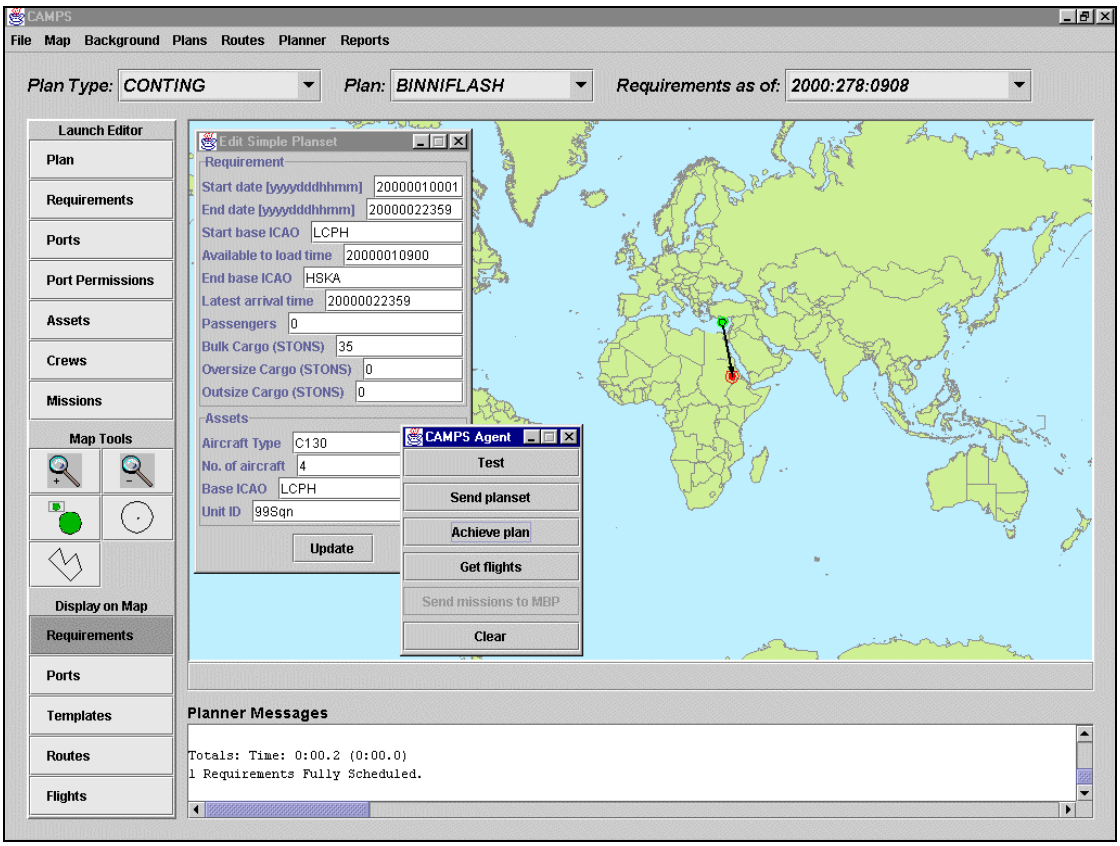


Figure 6: The CAMPS airlift planner, and the demonstration agent used to task the CAMPS agent with a simple requirement: movement of cargo from Cyprus into the fictional country of Binni.

In the demonstration scenario, CAMPS schedules airlifts of cargo into Binni. These airlift flights could conflict with offensive air missions, so the scheduled flights are requested from the CAMPS agent, and sent to MBP, forming part of the normal MBP air visualization. This

is achieved by an intermediate agent which tasks CAMPS, and also translates between the KQML messages used by CAMPS and the XML messages used by the MBP agent.

This is an interesting example, as only partial translation is possible; CAMPS and MBP differ fundamentally in their definition of air missions. A CAMPS mission consists of an arbitrary collection of flights, where a flight is a single journey from A to B by a single aircraft. However, an MBP mission consists of a starting point and a route, which must return to the starting point (perhaps by a different path), and may consist of multiple aircraft. CAMPS can therefore produce routes that have no fully valid representation in MBP, although they could be partially represented or indicated graphically.

ARIADNE

In a similar manner, weather information extracted from websites by the Ariadne system from the University of Southern California, Information Sciences Institute, is translated and forwarded to MBP, again forming part of the normal picture of the air situation. Ariadne facilitates access to web-based data sources via a wrapper / mediator approach (Knoblock and Minton, 1998). Wrappers that make web sources look like databases can be rapidly constructed; these interpret a request (expressed in SQL or some other structured language) against a web source and return a structured reply. The mediator software answers queries efficiently using these sources as if they formed a single database. Translation of the XML from Ariadne into the XML expected by MBP was handled by custom code, but can now be performed more easily using XSLT (Extensible Stylesheet Language Transformations); this latter technique is used elsewhere in the demonstration (section 4.2.6).

I-X PROCESS PANELS (I-P^2)

This Coalition planning process is supported using I-X process panels. I-X is a research program with a number of different aspects intended to create a well-founded approach to allow humans and computer systems to cooperate in the creation or modification of some product such as a plan, design or physical entity — i.e. it supports synthesis tasks. I-X may also be used to support more general collaborative activity. The I-X research draws on earlier work on O-Plan (Tate et al, 1998), <I-N-OVA> (Tate, 1996) and the Enterprise Project (Fraser and Tate, 1995) but seeks to make the framework generic and to clarify terminology, simplify the approach taken, and increase re-usability and applicability of the core ideas. Within CoAX, the I-X approach is being used to provide task and process support and event-response capabilities to various Coalition participants (Figure 7).

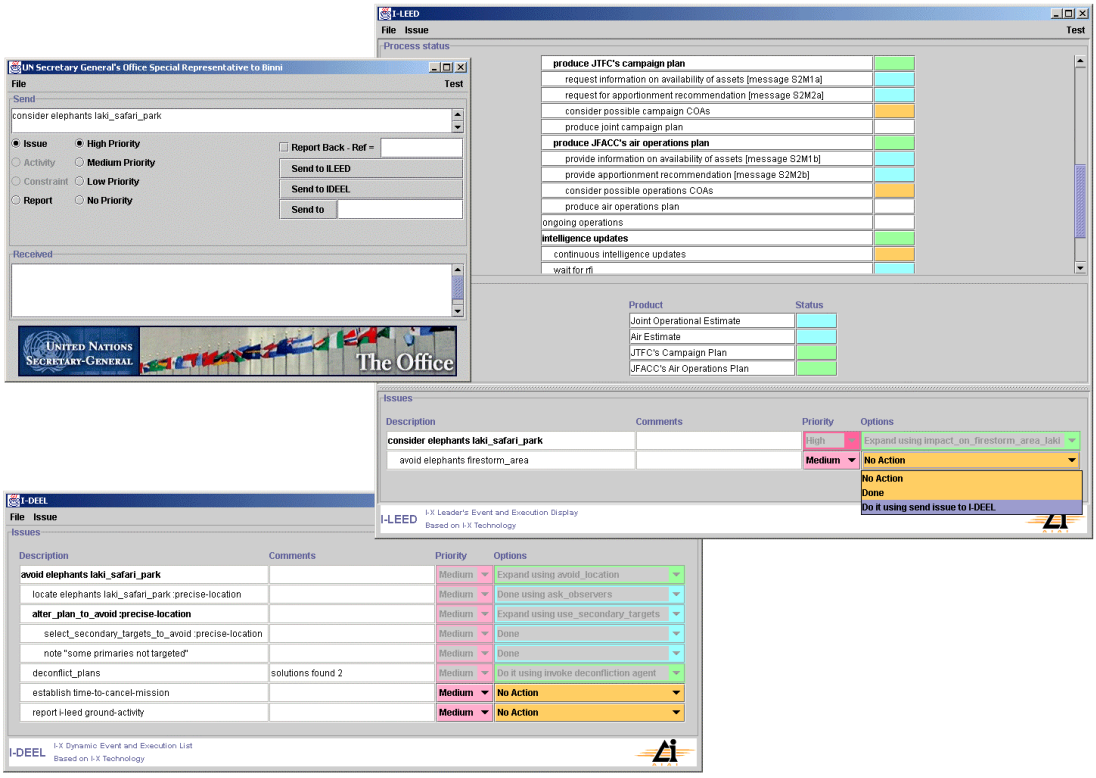


Figure 7: I-X Process and Event Panels

The aim of an I-X Process Panel (I-P^2) is to act as a workflow and messaging 'catch all' for its user. It can act in conjunction with other panels for other users if desired. A panel:

- Can take *any* requirement to:
  - Handle an issue;
  - Perform an activity;
  - (in future) Add a constraint.
- Deals with these via:
  - Manual (user) activity;
  - Internal capabilities;
  - External capabilities (invoke or query);
  - Reroute or delegate to other panels or agents (pass);
  - Plan and execute a composite of these capabilities (expand).
- Receives reports and interprets them to:
  - Understand current status of issues, activities and constraints;
  - Understand current world state, especially status of process products;
  - Help the user control the situation.
- Copes with partial knowledge.

RESOURCE CONTROL VIA DOMAIN POLICIES

Gao has host nation status within the Coalition but its intentions are unclear and it is distrusted. Special steps are taken to monitor the information passed to and from Gao within the Coalition. During the demonstration, misinformation feeds by Gao (intended to displace the firestorm to allow Gao to take an advantage and move forward) are detected and thwarted. Gao becomes belligerent and launches a denial of service attack against the Coalition's C3I infrastructure.

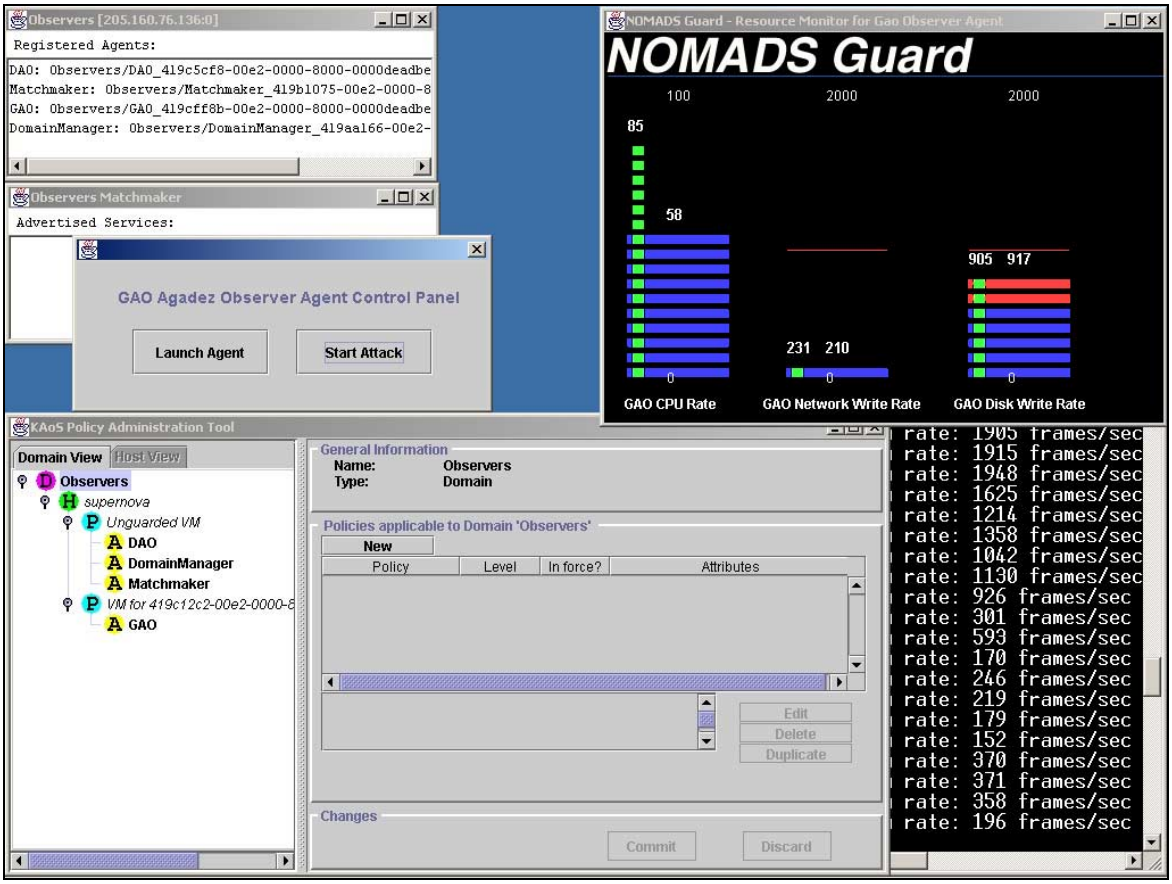


Figure 8: A denial-of-service attack by the Gao agent is starving other agents of resources (note the decreasing rate of processing in the console, bottom right). The Guard (top right) is monitoring the resource usage of the Gao agent. The excessive resource usage triggers a change in domain policy, and the resource limits enforced by the AromaVM are lowered. The policy can also be changed manually using KPAT, the KAoS Policy Administration Tool (bottom left).

The Gao agent in the demonstration is run under NOMADS, a mobile agent system from IHMC. The NOMADS project aims to develop a set of distributed agent-based systems using the Java language and environment. The agent code runs in a new Java Virtual Machine, the AromaVM. The AromaVM provides two key enhancements over standard Java VMs: the ability to capture

the execution state of threads and the ability to control resources consumed by threads. By capturing the execution state of threads, the NOMADS agent system provides strong or transparent mobility for agents.

In addition, the resource control mechanisms can be used for controlling and allocating resources used by agents as well as to protect against denial of service attacks by malicious agents. When the Gao agent exceeds certain resource limits, an automatic change in domain policy is triggered by a domain Guard, and the AromaVM is instructed to reduce the resources available to the malicious agent (Figure 8). An operator can manually reduce the limits further, using the KAoS Policy Admin Tool (KPAT).

DATA FEEDS FROM MOBILE DEVICES AND OBSERVERS

The firestorm mission has been planned and aircraft have already taken off. However the news media break a story that wildlife in an important safari park in Binni may be in danger as the park overlaps the firestorm area. With only an hour to go, the UN Secretary General's Special Representative to Binni asks the Joint Task Force Commander to consider the wildlife risk aspects of the planned approach. Dynamic information gathering and information feeds using agent technology are employed to create a real time feed of the position of some at-risk large mammals.

This urgent issue is noted and broken down into sub-tasks using the event panels. The progress of aircraft is monitored in near real-time on the Situation Viewer agent from QinetiQ, and the time left before aircraft are committed to their targets is determined from MBP. A search is made for information on the locations of animals in the safari park, and it is discovered that data are available on-line via agents running on monitoring devices attached to large mammals in the park. The agents are eGents (agents that communicate by email) developed by Object Services and Consulting, Inc (OBJS). Historical data from these devices is queried using a Natural Language Interface from OBJS. To aid the planners, a live data-feed is created from the safari park website, using Ariadne to extract data from the pages, and a translator agent using XSLT. The resulting message stream is sent to MBP and to the Situation Viewer agent, allowing the current position and track of the animals to be visualized (Figure 9).

Data about the movement of ground forces, from the D'Agents field observation system from Dartmouth College, are also transformed using another instance of the translator agent and visualized in the same way, allowing the coalition to identify a convergence of hostile forces on the Laki safari park area.

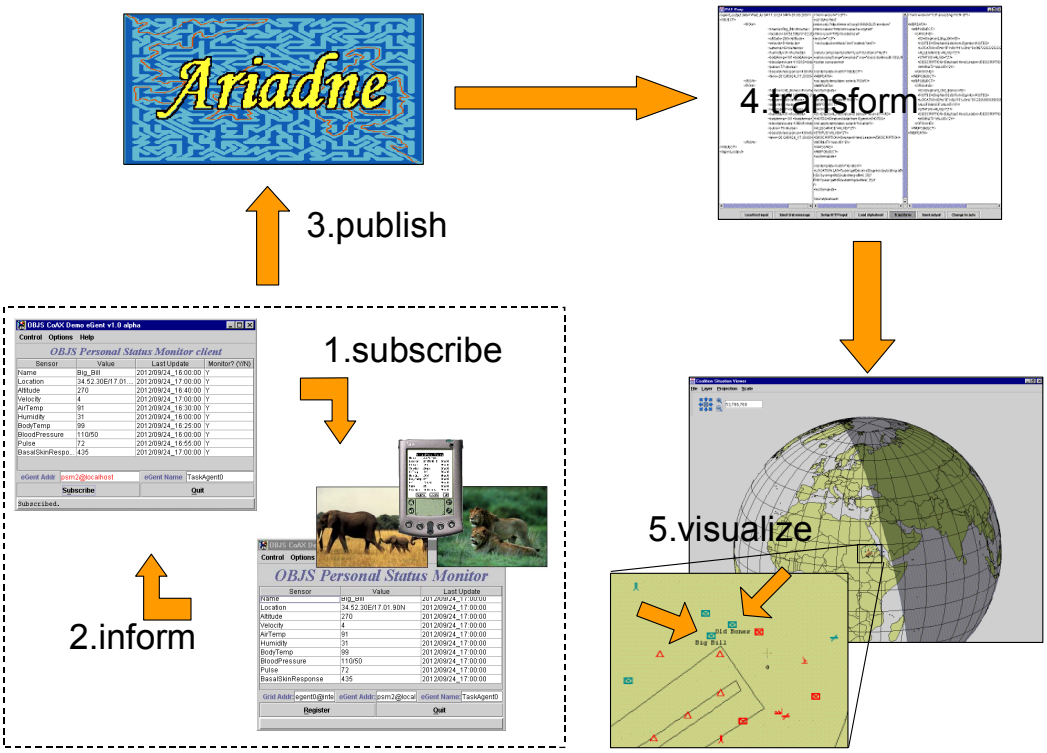


Figure 9: An eGent client subscribes to eGents running on mobile devices (wildlife tags). The data from these devices are published by the client on a web page. Ariadne extracts data from the webpages, and produces XML. The XML is transformed to another format by another agent using XSL Transformations, and finally sent to agents such as MBP and Situation Viewer for visualization.

PLAN EXPORT AND DECONFLICTION

After consideration it is decided to continue with the firestorm mission, but to re-plan as necessary to avoid risk to wildlife. Firestorm targets are adjusted in time or secondary targets selected as necessary for the first wave of firestorm bombing. The impacts of these changes on the Coalition's medical and humanitarian operations are automatically detected, and unintended conflicts between disjoint Coalition operations are avoided.

The air plans are revised using MBP, and then sent to a deconfliction agent to check them against planned activities in other Coalition HQs. The Multi-level Coordination Agent (MCA) from the University of Michigan processes the plans, using multiple levels of abstraction to generate solutions (Clement & Durfee, 1999). The planners are kept informed of progress via their I-X event panels, and can view the results on the MCA display when ready (figure 10). The plans are adjusted iteratively until the conflicts are resolved.

DYNAMIC FORCED MIGRATION (SCRAM) OF OBSERVER AGENTS

Agadez seeks to use this complication to seize the initiative and launches fighter attacks against a Coalition airborne high value asset (JSTARS) that is monitoring the operation. When this attack is detected, the JSTARS starts to regress, which implies that the observer agents on the JSTARS will not be able to continue providing information to the coalition.

In order to solve this problem, the administrator uses the forced migration (scram) capabilities of the NOMADS mobile agent system to move the observer agents from the JSTARS platform to a secondary ground station platform. The NOMADS system uses the state capture mechanisms in the Aroma VM to capture the full execution state of the agents on the JSTARS. Once captured, the execution state is sent to a new platform where the agents can be restarted without any loss of their ongoing computations (figure 11). This allows the observation agents to continue to operate on the ground station and provide information to the coalition even after the JSTARS regresses.

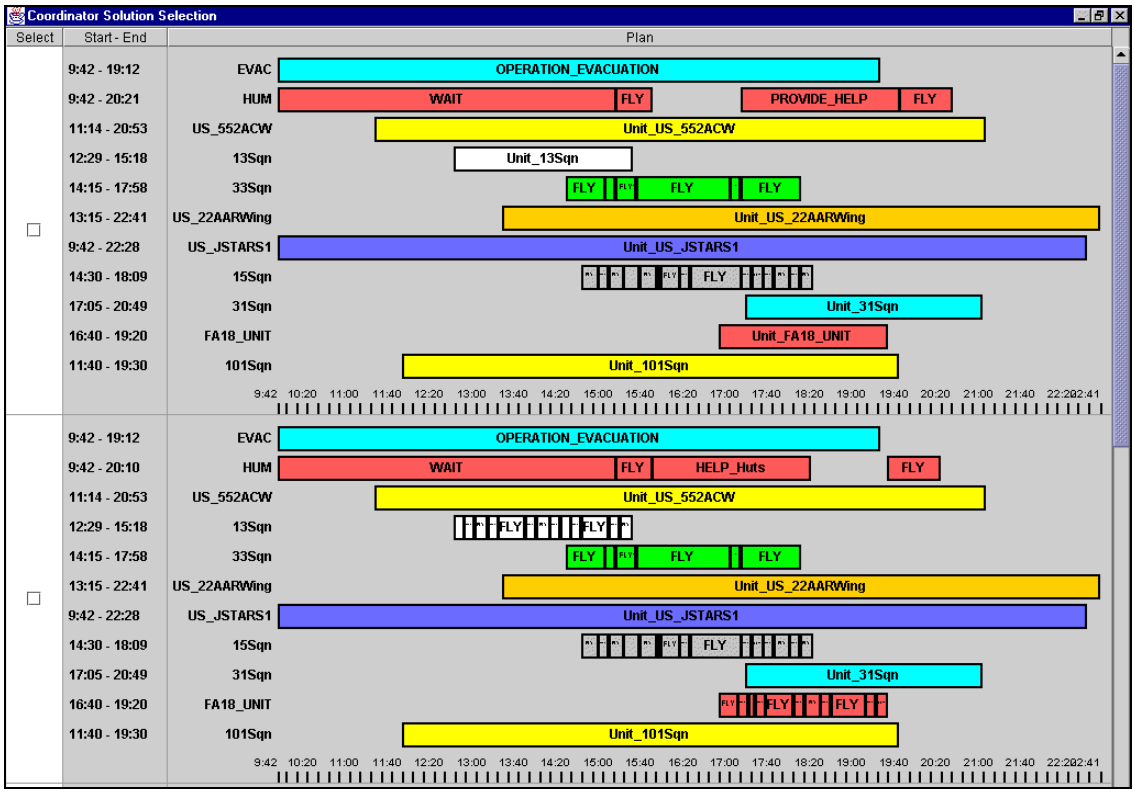


Figure 10: Deconfliction of Coalition plans by the Multi-level Coordination Agent. In the second solution (lower half) two missions (13Sqn and the FA18_UNIT) have been broken down to a lower level of abstraction to seek more optimal coordination

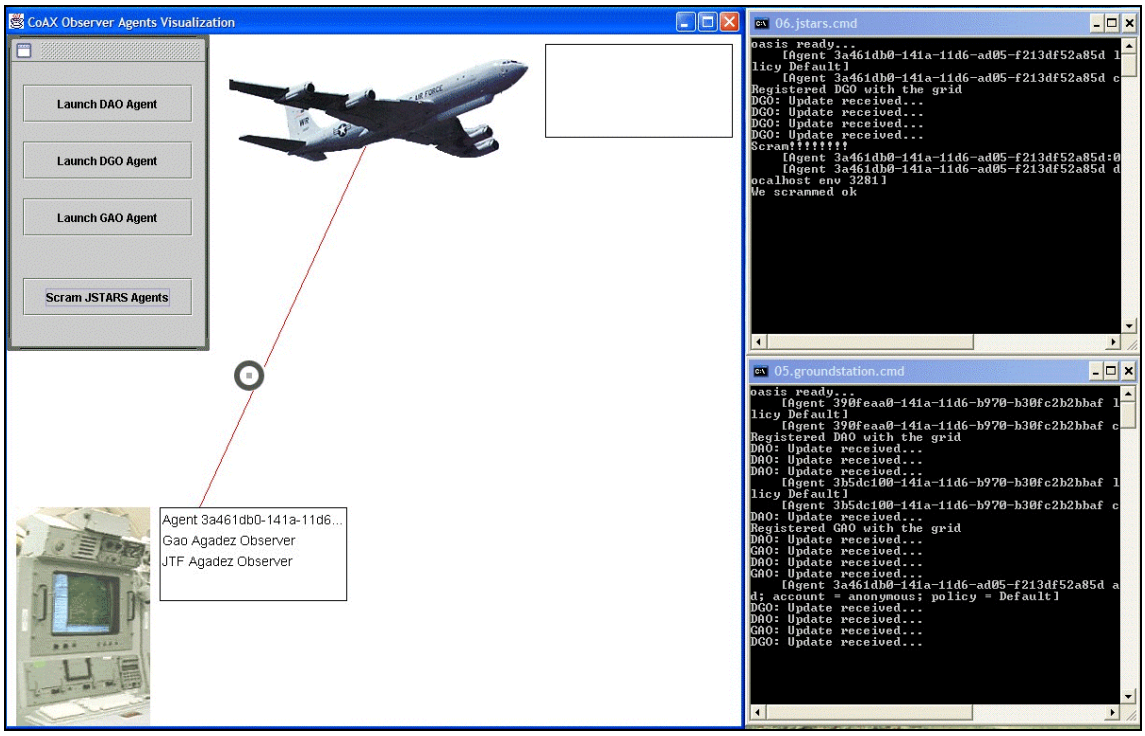


Figure 11: Forced migration of observer agents from mobile platform to ground station, using NOMADS and AromaVM. The updates from the DGO agent, initially on the JSTARS airborne platform (top right console) then start to appear on the new ground platform (lower right console).

Assessment of Software Agents

Technical Progress to Date

The CoAX project officially began in February 2000 and we believe that the demonstrations we have undertaken corroborate the hypotheses outlined in Section 1.3, demonstrating the utility of agent technology in Coalition operations. We have put together a prototype Coalition C2 architecture that supports and embraces heterogeneity and have exercised this in an agent-based C2 demonstration that enacts Coalition activities within the Binni scenario, including both the planning and execution phases of operations.

The CoABS Grid and KAoS domain management capabilities have allowed us to interoperate, for the first time, previously stand-alone US and UK military systems as well as a variety of agent-based information resources. In particular, the CoABS Grid has played a vital role in rapid and robust integration of systems. We have shown how agent organization, behavior, security and resources can be managed by explicit domain policy control.

Assessment work funded by the DARPA CoABS program has reported favorably on the performance issues of agent-enabled infrastructures and the experiences of the CoAX team have

shown that the agent-wrapping of legacy systems and the integration of different agent systems at short notice is relatively straightforward. This task is simpler where systems expose more of their internal information and methods. In addition, a heterogeneous set of agents can be made to interoperate as long as implementers adhere to some minimum set of message and other standards. Heterogeneity should be accepted and embraced as it is seen as being inevitable and can actually be beneficial in a number of cases — especially in security terms.

Dynamic task, process and event handling is an important aspect of collaboration and Coalition C2. In the CoAX demonstrations a process panel was used to indicate the start of the tasking and lead into the heart of the demonstration. In the execution phase of operations, process panels in the main commands or headquarters were more extensively used as they enabled a clearer military relevant view of what was happening between the agents in less technical language than would otherwise be visible. Process and event panels have been found to be helpful in keeping users informed of the current stage of collaboration, and maintaining a shared picture of the current state of the collaborative efforts.

Our experience is that an agent-enabled environment gives the ability to create shared understanding and improved visualization. Specific benefits were gained when agents worked semi-autonomously in the background to process information and support decision making collaboratively with operators, and when agents were integrated into existing tools so as not to disrupt familiar methods of operation.

Future Research Program

An aim only partially addressed in the current work is the construction and maintenance of a fully dynamic virtual Coalition organization. This would involve:

- domains and agents added to the Coalition structure ‘on-the-fly’;
- Coalition partners joining / leaving unpredictably;
- handling of dynamic Coalition tasks, processes and events.

Capabilities under investigation for future demonstrations include

- obligation management, e.g. ensure that agents are meeting their commitments;
- improved agent collaboration and run-time interoperability achieved using semantic web languages and technology (Allsopp et al, 2001a);
- richer domain organization and security policies (Bradshaw et. al., 2001);
- richer task, process and event management with more dynamically determined agent relationships (Tate et al., 2002);
- a variety of agents providing new types of data, and data-processing capabilities such as threat classification and track prediction.

Aspects of this work will be included in the Fleet Battle Experiment-Juliet 2002, part of the Millennium Challenge joint integrating experiment.

Military Implications of the Results

The CoAX research program has shown how software agents can carry out tasks that enable interoperability between information systems and infrastructure services brought together in a 'come-as-you-are' Coalition.

In the experiments so far, the software agents operated in a number of roles. They have worked 'in the background' — through matchmaking, domain management, process management and other agent services — to find, establish and maintain the infrastructure, information and procedural links necessary to achieve and support interoperability in a dynamically changing environment. In addition, they have worked collaboratively with human operators, mediating requests for information and formatting and displaying the results almost transparently.

Thus an agent-enabled environment helps create shared understanding and improves the situational awareness of military commanders. Moreover, it could make a significant contribution to the aims of Network-Centric Warfare which is defined as follows: an approach to the conduct of warfare that derives its power from the effective linking or networking of the warfighting enterprise. It is characterized by the ability of geographically dispersed forces to create a high level of shared battlespace awareness that can be exploited via self-synchronization and other network-centric operations to achieve commander's intent.

One early lesson has been that Cyberspace should not be seen just as an information pipe between humans — it is a Battlespace in its own right. This indicates that 'Cyberspace Superiority' should be obtained (as for any other part of the Battlespace) by ensuring that Coalition forces are able to act decisively through software agents acting on behalf of or mediating the actions of human users.

Dealing effectively with unpredictable changes — owing, for example, to the destructive activities of opponents or because of systems failing and services being withdrawn — is a typical Coalition problem where software agents could make a significant contribution. So far, we have shown that a software agent infrastructure is robust and, to some extent, is 'self-healing'. Our aim is to investigate this further to show that software agents can provide agility, robustness, flexibility and additional functionality beyond that provided by the individual Coalition partners.

Concluding Remarks

The central hypothesis being investigated in CoAX is that the agent-based computing paradigm is a good fit to the kind of computational support needed in Coalition operations. The evidence so far confirms this view: we have shown a number of disparate agent systems working together in a realistic Coalition application and indicated the value of the agent-based computing paradigm for rapidly creating such agent organizations. Agents can usefully share, and manage access to, information across a stylized Coalition architecture.

Our conclusion is that software agents, together with agent-based infrastructures and services provided by the CoABS Grid and KAoS, could play a key role in supporting Coalition operations. We think that this technology will provide the ability to bring together and integrate systems quickly to aid in all aspects of Coalition operations, without sacrificing security and control. Our long-term goal is to use this technology in the creation, support and dynamic reconfiguration of virtual organizations — with Coalitions being an archetypal and timely example of an area where this technology is vitally needed.

Acknowledgements

The authors gratefully acknowledge all those who contributed to the CoAX project, including Mark Burstein, Thom Bartold, Maggie Breedy, John Carson, Jeff Cox, Brad Clement, Rob Cranfill, Jeff Dalton, Pete Gerken, Bob Gray, Arne Grimstrup, Paul T. Groth, Greg Hill, Heather Holmback, Renia Jeffers, Martha Kahn, Shri Kulkarni, John Levine, Jean Oh, Pradeep Pappachan, Shahrukh Siddiqui, Jussi Stader, and Andrzej Uszok. The various projects that participated in CoAX were sponsored by the Defense Advanced Research Projects Agency (DARPA) and managed by the U.S. Air Force Research Laboratory, except work by QinetiQ, which was carried out as part of the Technology Group 10 of the UK Ministry of Defence Corporate Research Programme. The US Government and the contributors' organizations are authorized to reproduce and distribute reprints 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 official policies or endorsements, either express or implied, of DARPA, the US Government, the US Air Force Research Laboratory, the UK MoD, or the contributors' organizations.

References

Alberts, D. S., Garstka, J.J., Hayes, R.E., Signori, D. A. (2001) “Understanding Information-Age Warfare”, CCRP Publication Series, 2001. ISBN 1-893723-04-6

Allsopp, D.N., Beautement, P., Bradshaw, J.M., Carson, J., Kirton, M., Suri, N. and Tate, A. (2001) “Software Agents as Facilitators of Coherent Coalition Operations”, Sixth International Command and Control Research and Technology Symposium, US Naval Academy, Annapolis, Maryland, USA, 19-21 June 2001.

Allsopp, D.N., Beautement, P., Carson, J. and Kirton, M. (2001a) “Toward Semantic Interoperability in Agent-based Coalition Command Systems”, Proceedings of the First International Semantic Web Workshop, July 30-31, 2001, Stanford University, CA, USA, pp 209-228

Bradshaw, J.M., Suri, N., Kahn, M., Sage, P., Weishar, D. and Jeffers, R. (2001) “Terraforming Cyberspace: Toward a Policy-based Grid Infrastructure for Secure, Scalable, and Robust Execution of Java-based Multi-agent Systems”. IEEE Computer, 49-56, July 2001.

Bradshaw, J.M., Dutfield, S., Benoit, P. and Woolley, J.D. (1997) "KAoS: Toward an Industrial-Strength Generic Agent Architecture," Software Agents, AAAI Press/The MIT Press, Cambridge, Mass., pp. 375-418.

Burstein, M., Ferguson, G. and Allen, J. (2000) "Integrating Agent-Based Mixed-Initiative Control with an Existing Multi-Agent Planning System", Proceedings of the Fourth International Conference on MultiAgent Systems, Boston, MA, 2000.

Clement, B.J. and Durfee, E.H. (1999) "Top-Down Search for Coordinating the Hierarchical Plans of Multiple Agents", Proceedings of the Third International Conference on Autonomous Agents, pages 252-259, May 1999.

Emerson, T. and Burstein, M. (1999) "Development of a Constraint-based Airlift Scheduler by Program Synthesis from Formal Specifications", Proceedings of the 1999 Conference on Automated Software Engineering, Orlando, FL, September, 1999.

Fraser, J. and Tate, A. (1995) "The Enterprise Tool Set — An Open Enterprise Architecture", Proceedings of the Workshop on Intelligent Manufacturing Systems, International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada, August 1995.

Jennings, N R. (2001) "An Agent-based Approach for Building Complex Software Systems". Communications of the ACM. Vol 44, No: 4, 35-41. April 2001.

Knoblock, C. A., and Minton, S. (1998) "The Ariadne Approach to Web-based Information Integration", IEEE Intelligent Systems , 13(5), September/October 1998.

Rathmell, R.A. (1999) "A Coalition Force Scenario 'Binni — Gateway to the Golden Bowl of Africa'", Proceedings of the International Workshop on Knowledge-Based Planning for Coalition Forces, (ed. Tate, A.) pp. 115-125, Edinburgh, Scotland, 10th-11th May 1999.

Tate, A. (1996) "The <I-N-OVA> Constraint Model of Plans", Proceedings of the Third International Conference on Artificial Intelligence Planning Systems, (ed. Drabble, B.), pp. 221-228, Edinburgh, UK, May 1996, AAAI Press.

Tate, A., Dalton, J. and Levine, J. (1998) "Generation of Multiple Qualitatively Different Plan Options", Fourth International Conference on AI Planning Systems (AIPS-98), Pittsburgh, PA, USA, June 1998.

Tate, A., Dalton, J., and Stader, J. (2002) "I-P^2 — Intelligent Process Panels to Support Coalition Operations", in Proceedings of the Second International Conference on Knowledge Systems for Coalition Operations (KSCO-2002) (ed. Tate, A.), 23/24-Sep-2002, Toulouse, France.