Tool Support for Scenario-based Functional Allocation

Alistair Sutcliffe, Jae-Eun Shin, Andreas Gregoriades
Centre for HCI Design, Department of Computation
University of Manchester Institute of Science and Technology (UMIST)
P.O.Box 88, Manchester, M60 1QD, UK
a.g.sutcliffe@co.umist.ac.uk

Abstract: Tool support is described for analyzing requirements and creating conceptual models from scenarios. A schema of scenario-based knowledge is proposed that extends the **i*** ontology with concepts to represent the system environment and natural language semantics to categorize arguments. Modelling tools are introduced to support the process of transforming scenarios into models and requirements. We illustrate use of the tools by analysis of the London Ambulance case study. The advisor guides the analyst with contextual appropriate advice on functional allocation of agent roles and generic requirements to avoid errors and system failure. The advice is based on research in human reliability engineering.

Keywords: scenarios, functional allocation, socio-technical systems, human error.

Introduction

Scenarios have received considerable attention as a means of eliciting and validating requirements (Carroll, 1995; Potts, Takahashi & Anton, 1994; Rolland et al., 1998). In requirements elicitation, models are derived from scenarios by a process of generalization, while in requirements validation, scenarios can be used as examples to test a model or requirements specification. However, there are few methods or tools that help the transformation of scenarios to models or support the use of scenarios in requirements validation.

One use of scenarios is to capture information about the system environment (e.g. Kyng, 1995) which is often ignored in conceptual models. Yu and Mylopoulos (Yu, 1997) emphasize the need to model the system environment, since lack of domain knowledge frequently leads to inadequate requirements and hence system failures Curtis, Krasner & Iscoe, 1988). The ***i**** framework (Yu, 1997) was developed for modelling and reasoning about the impact of organizational environments on information systems, and ***i**** does provide reasoning mechanisms for validating relationships between agents, tasks and goals; however, we argue that requirements analysis tools should go further and provide advice on issues such as functional allocation and socio-technical system design. In previous work we investigated taxonomies of influencing factors and proposed scenario-based techniques for diagnosing problems in communication and functional allocation in socio-technical systems (Sutcliffe, 2000; Sutcliffe et al., 1998). To assist this modelling, we introduce tools that support the process of transforming scenarios into models and requirements specifications. These tools are based on schema of scenario-based knowledge, explained in the following section. The tools are illustrated by analysis of the London Ambulance case study.

Knowledge Representation Schema for Scenarios

Scenarios have many definitions and even more diverse content (Carroll, 2000, 1995; Cocchiarella, 1995), so a general purpose ontology of knowledge (Hovy, 2001; Sowa, 2000) might seem to be an appropriate choice. However, we wish to build upon existing conceptual modelling languages (e.g. UML) and ***i**** in particular because this is established in RE. Our schema, therefore, contains concepts that are familiar in many modelling languages (i.e. agents, objects, tasks, goals), but it adds new constructs for modelling the system environment and, more radically, for argument and communication. We propose a unified schema that represents arguments expressed in natural language and the domain of discourse (i.e. the modelled world) to which those arguments pertain. The motivation for this is simple. Scenarios frequently report opinions and causal arguments that explain aspects of the modelled world. Capturing and analyzing such arguments is often critical to discovering accurate requirements.

A schema of scenario components was derived from the review of relevant literature (Carroll, 1995; Carroll et al., 1994; Daren, Harrison & Wright, 2000; Mylopoulos, 1998; Sutcliffe et al., 1998, Mylopoulos, 1998), ontologies and knowledge representation (Chung & Nixon, 1995; Guarino, 1997; Sowa, 2000; Van Heijst, Schreiber & Wielinga, 1997; Waterson & Preese, 1999). The schema categorizes scenario narratives into five areas (*Actors & Structures, Intentions, Tasks, Environment*, and *Communication*) and three levels (*Strategic, Tactical,* and *Operational*).

Semantics to express structures and properties of the system environments and argumentation were drawn from functional theories of language (Mann & Thompson, 1988) to augment the semantics in the ***i**** model (Mylopoulos, 1998). Concepts are first order modelling primitives which have properties. The concepts and relationships in each of the five areas are as follows:

- Actors & structures: agent, attribute, group, organization, physical structure, role; *properties* of agents: motivation, capability, dependability, power, reputation, responsibility, trust
- Intentions: goal, objective, policy, strategy; *properties* of goals: importance, quality
- Activity-related: action, event, object, procedure, resource, state, task; *properties* of tasks: predictability, complexity, criticality
- Environmental: social, economic and physical environments including location; *properties* of environment: predictability, interruptions, weather state, stress, climate, noise; *properties* of social environment: management culture, time pressure, stress, inclusiveness
- Communication: argument, attitude, background context, causation, consequence, decision, elaboration, evidence, issue, interpretation, hypothesis, justification, motivation, position, viewpoint.

The schema concepts and relationships are shown in Figure 1. The actors, intentions, task and environment components all represent the modelled world and are connected to communication components by user-defined relationships. The communication area is not explicitly coupled to the modelled world because it represents arguments about the domain of discourse. In many cases, segments in scenario narratives may refer to argument and to properties of concepts in the modelled domain; for instance, a narrative that argues for system users being well trained can be described by properties of agents (their motivation) as well as given a motivating argument for their training.

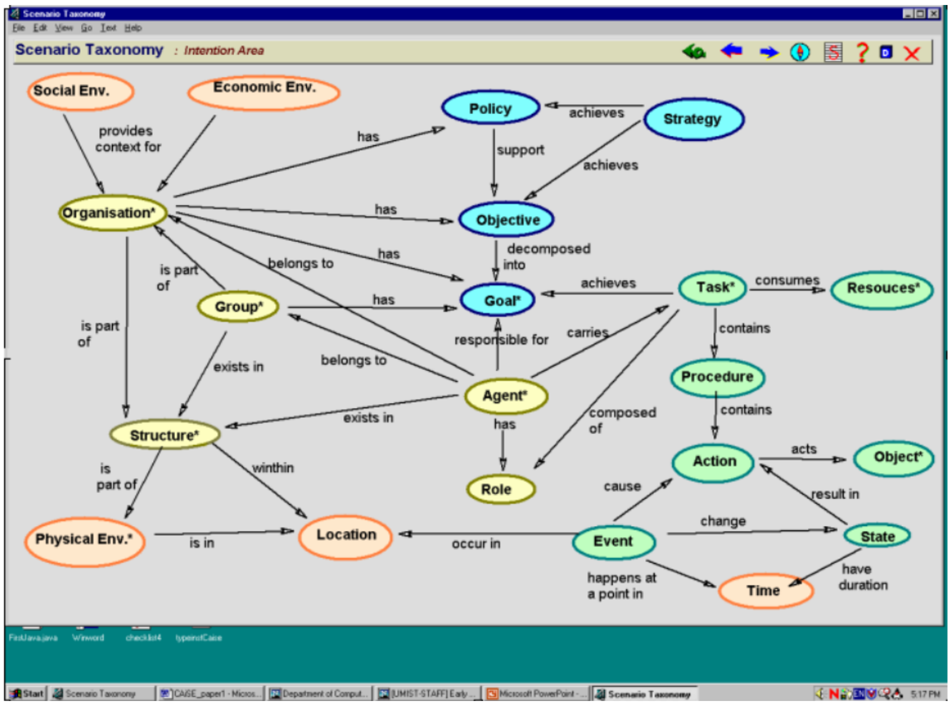


Figure 1 - Hypertext tool showing schema map interface with domain model components and relationships at the tactical and strategic level.

Hypertext Tool for Scenario Management: An object-oriented authoring system (Asymetrix's Tool-Book Instructor II), a hypertext tool, was used to construct a concept map interface of the scenario schema, as illustrated in Figure 1. The user could access definitions as a pop up "tool tip" text to explain each component with examples and synonyms to help understanding.

Scenarios are marked up using the annotator editor which also functions as a model editor so that model components can be linked to scenario narrative segments. We illustrate use of the scenario modelling tools with the computer-aided dispatch (CAD) system in the London Ambulance Service (LAS). The aim is to show how a scenario modelling tool can be used in requirements analysis to identify key design issues by a retrospective analysis of the LAS CAD system starting with a narrative that presents a scenario-like summary of the system failure. (Finkelstein & Dowell, 1996)

The annotation editor provides a direct manipulation interface so the user can highlight a segment of scenario text and then point to the schema component that describes it. This causes markup tags to be placed in the selected text, *e.g. <agent> text editor </agent>*. Text can be selected several times so part of a scenario narrative can be linked to communication as well as domain model components, *e.g. <justify> to support the user's task it was necessary to create a tool which was the <agent> text editor </agent></justify>.*

The annotation editor could also be used to create models by a simple pick and place dialogue. This allows the user to create a model of the domain with agent, task, object instances and then link the model components to their corresponding origin in one or more scenarios. The LAS system model derived from this analysis is shown in Figure 2. Links in the annotation editor provide traceability so cause-consequence arguments in the scenario can be traced to the relevant system model components. For example, the frustration experienced by the ambulance crews which led them to poor reporting of call status relates to the relationship between the Crew agent and the Reporting goal/task.

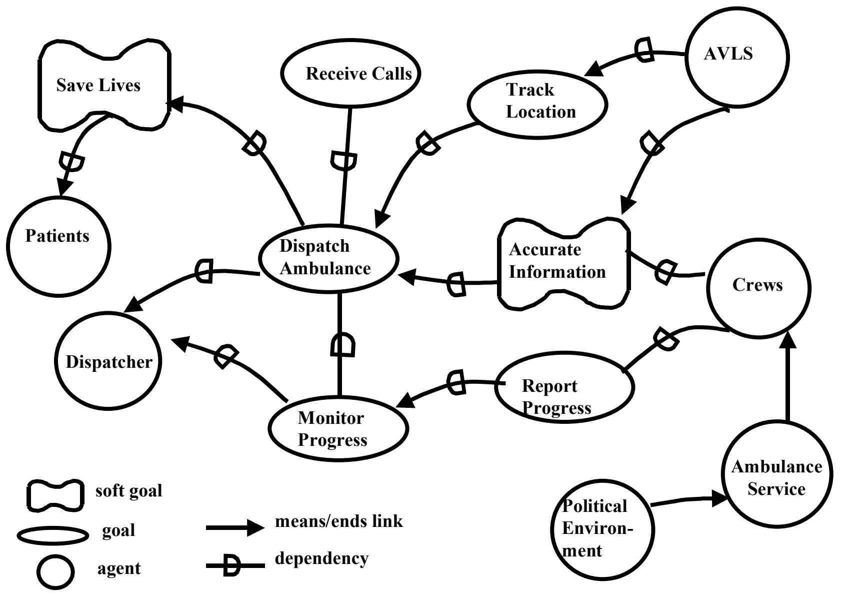


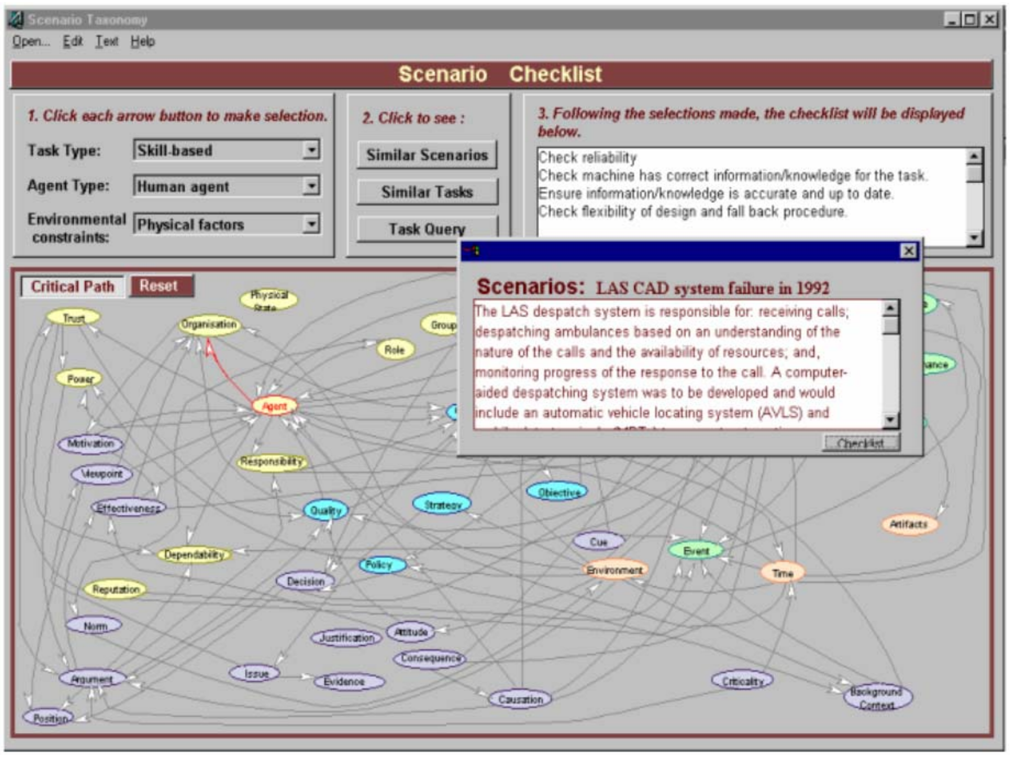
Figure 2 - Model of the LAS system in adapted ***i**** notation. Only goals and agents are shown for simplicity, with additions of organization and environment components taken from our scenario schema. Causal influences on the crews are modelled as means-ends links.

Scenario Analysis Advisor: The scenario analysis advisor uses rules and relationships between schema components to produce advice on human factor problems and generic requirements that indicate solutions to those problems. Three types of advice are available:

- Functional allocation issues: this advice concerns the trade-off decisions about which functional requirements should be fully automated, partially automated or left as manual tasks. This advice is accessed when querying goal/task or agent components in models or scenarios. The knowledge is drawn from the HCI literature (Bailey, 1982; Wright, Dearden & Fields, 2000) and contains warnings about flexibility of processes, the predictability of events, workload estimation techniques and social issues such as human reactions to changes in responsibility and authority brought about by automation.

- System reliability and human error: this advice pertains particularly to task/goal-agent relationships but it may also be accessed via organization-agent relationships. The knowledge for this is drawn from the human reliability engineering literature (Hollnagel, 1998; Leveson, 1995; Reason, 2000) and covers typical errors that may occur in certain task-agent combinations with generic requirements to prevent or contain such problems.
- General analysis information about socio-technical system problems attached to relations between agents, tasks, goals, organizations, and the social/political environment. This knowledge is taken from the RE and studies of socio-technical systems.

The advice is accessed via browsing on the schema map and selecting components. Alternatively, nodes in marked-up scenarios can be selected to access advice dialogues. To illustrate the process, assuming the user has selected four nodes: agent, task, structure and physical environment; first the system requests further information about the properties of the selected nodes as shown in Figure 3. For instance the task properties are entered as high or low estimates for the level of complexity, the level of training and familiarity with the task, the agent types (e.g. human or machine agent), and the environmental properties (e.g. interruptions, weather conditions). This information is used by the system to narrow its search for appropriate advice.



Task Type: Skill-based
Agent Type: Human agent
Environmental constraints: Physical factors

Check reliability
Check machine has correct information/knowledge for the task.
Ensure information/knowledge is accurate and up to date.
Check flexibility of design and fall back procedure.

Scenarios: LAS CAD system failure in 1992

The LAS despatch system is responsible for: receiving calls; despatching ambulances based on an understanding of the nature of the calls and the availability of resources; and, monitoring progress of the response to the call. A computer-aided despatching system was to be developed and would include an automatic vehicle locating system (AVLS) and

Figure 3 - Scenario analysis advisor, showing input of properties for the selected relationship (1. top left) and advice (3. top right hand message box).

If functional allocation advice is chosen with agent and task nodes selected then the system provides the advice illustrated in Table 1 which shows the setting of the properties of the schema components, and the advice and its justification with respect to those settings.

Rules indicate potential constraints on agent-task combination such as the dangers of allocating complex tasks to poorly motivated agents. If the organization node that the agents belong to and properties of management culture are set to poor, this will set the agents' motivation low, so the influence of property settings is propagated along schema relationships. Table 2 shows error prevention advice for the task-agent relationship.

Advice is created by rules that follow the schema links from task and agent to structures and organization (the immediate system environment) and then to physical and social environment nodes. The error advice database is organized using the schema to enable access to the appropriate information. The advice is generated by rules that link the preconditions to types of error. A sample of the rules is given below, with the general format followed by an example:

- Functional allocation rules:
  If task <property=H/L> *Then* allocate to <Machine or Human or Collaborative (machine support)>: e.g. *If* task <complexity=L> *Then* allocate to <Machine agent>
  If agent <property = H/L> *Then* allocate to <Machine or Human (training advice)>: e.g. *If* agent <capability = H> *Then* allocate to < Human agent>
- Error reliability rules:*If* organisation <property = H/L> *Then* agent <property = H/L>: e.g. *If* organisation <incentive = L> *Then* agent <motivation = L>
  If agent <property = H/L> *Then* errors <(slips/mistakes) probable/not probable>: e.g. *If* agent <motivation = L> *Then* slips are likely
  If physical environment <property = H/L> *Then* errors <(slips/mistakes) probable/not probable>: e.g. *If* physical environment <time pressure = H> *Then* slips are likely.

The number of rules that trigger error predictions is counted to increase a confidence rating, e.g. slips are likely (influencing factors = 3/8).

Table 1 - Functional allocation advice for Tasks and Agents according to their property settings.

Component	Properties	if High, then Implications	if Low, then Implications
Task	Complexity	Capable and well trained operators; allocate to humans	Little training, suitable for automation
	Predictability	Automate, if not too complex	Allocate to humans
	Importance/criticality	Motivate operators, back-up, recovery and fail safe design	Less training and error prevention needed
Agent	Motivation	Allocate demanding tasks to humans	Manual allocation for non-critical simpler tasks
	Capability	Check time for complex tasks Human operation	Automate for simple, predictable tasks
	Task knowledge	Skilled tasks Human operation	Decision support; training necessary
	Dependability	Allocate critical tasks to humans, or automate	Automate; Humans for simpler, non-critical tasks

Table 2 - System and human reliability advice for task-agent relationships.

Component	Properties	if High, then Implications	if Low, then Implications
Task	Complexity	Mistakes unless operators are well trained	Slips when operators become bored
	Predictability	Slips in routine operation; beware time pressure	Mistakes in reacting to usual events; training and simulation help
	Importance/criticality	Time pressure, fatigue and stress cause errors	Slips unless well motivated
Agent	Motivation	Mistakes less likely, slips still occur	Prone to mistakes and slips
	Capability	Fewer errors if well trained	Errors unless trained and given simple tasks
	Dependability	Errors less likely unless time pressure, tired or stressed	Prone to mistakes, lapses and slips

The error advice points to high likelihood of mistakes which are errors in intention and planning, while slips and lapses are failures of attention and concentration. Slips and lapses occur in skilled operation when the user agent is familiar with the task, whereas mistakes occur more frequently in tasks that require more judgment and decision making. Time pressure, fatigue and stress increase error rates, even when agents are well motivated, capable and reliable (Reason, 1990).

Case Study: London Ambulance Service

The LAS scenario was investigated using the analysis advisor. First, a general checklist is provided followed by more specific advice on critical design issues through understanding relationships between them. The task type for "Dispatch Ambulances" is set to *Critical = H*, *Complexity = H* and *Predictability = M*. This task requires judgment and knowledge about identifying calls and ambulance resource availability. Since full automation of the dispatch task was planned, the agent type is set to machine. For the environment properties, time pressure and interruptions were set to high while the other environmental factors like the weather could also be poor, so the following advice checklist was displayed:

- Function allocation advice is:
  Check reliability.
  Check machine has correct information/knowledge for the task.
  Ensure information/knowledge is accurate and up to date.
  Check flexibility of automated design and fallback procedures.
  With implications for the human role:
- Check change in responsibility is acceptable to the user.
  Investigate the assignment of authority of agents for tasks/goals.
  Question the impact on users' motivation and morale of changes in responsibility and authority.
  Investigate users' trust in technology; if it is poor, then operation may be ineffective.
- And implications of physical environment for task effectiveness:
  Investigate time for decision making.
  Check for interruptions and flexibility in handling unexpected events.
  Investigate weather impact on task operation.

Clearly such advice was not followed during the development of the CAD system, so it is possible that provision of this knowledge might have prevented some of the design mistakes. The CAD system did not have the correct information in the gazetteer of London streets and the location of the ambulances was inaccurate because radio blackspots prevented the system from tracking the vehicles in some locations. Furthermore, the information on call progress was inaccurate because of the crews' failure to report calls via mobile data terminals (MDTs). Implementation of the system changed the responsibility and authority of the dispatcher controllers because the system made the choices for them, with little opportunity to override decisions. This created an inflexible system. The motivation and morale of the dispatchers was probably impacted before the system went live, but rapidly became worse when the unreliability of the system became obvious.

In the second example, the report task and ambulance crew node is selected with advice on potential system errors and human reliability. This delivers the following guidance organized in four areas where errors may occur: the human agent (i.e. the ambulance crew users), design of the computer system (in this case the mobile data terminal) and the environment in which the system is used. The task in this case involved the crews entering progress reports into the MDTs. The properties settings of the task and environment were:

Task: Critical = High, Complexity = L, Skill = H and Predictability = M
Environment: Time pressure = H, Stress = H, Predictability = L

The task is critical because accuracy of the CAD system databases depends on it. Although the task of reporting in itself is not complex, its predictability can vary as some calls do not go to plan and the time pressure is created by crews having to attend to a higher priority task first, such as giving first aid, and getting the patient to hospital. The task is assumed to be a trained skill. The analysis advice is accompanied by generic requirements as follows:

- Human Error: slips and lapses are likely with skilled tasks. Check that training is adequate, and that the skill has been practised recently.
  generic requirements: to prevent/remedy lapses, use timeouts to check progress, provide reminders, status indicators, keep routines short; to prevent/remedy slips, trap slips with validation routines, minimize distractions, make task operations clear, ensure objects acted upon cannot be confused with others, provide reminders, undo facilities, editing facilities to correct errors.
- User Interface Design
  generic requirements: predictable and consistent user interface, same layout of screens,

consistent commands, simple actions, clear and unambiguous feedback, ergonomic requirements of visibility, audibility of output.

- Environment influences: slips and lapses
  generic requirements: minimize distractions and interruptions, e.g. non-essential noise, extraneous visual stimuli, non-essential communication. Ensure user has sufficient time to complete task without being pressured. Minimize fatigue and stress, which adversely affect user concentration. Investigate user motivation.
- Social/political environment
  generic requirements: management culture should motivate users to complete tasks by encouragement and incentives. Goals and standards of performance should be clearly communicated and addressed by training. Users should feel that they own the system and are involved with its success. Avoid authoritarian management styles if possible.

In this case the advice was pertinent to the LAS system at both levels. There were several user interface design defects in the MDT terminals which made them difficult to use, such as the order of entering call progress and poor visibility of the displays. The crews didn't use their MDTs effectively because of motivational problems caused by a poor managerial culture which did not involve crews in the design of the system. Furthermore, no incentives were given for effective use, and training was inadequate. Finally, when failures began to accumulate in the system, crews became stressed and tired which led to more slips and errors. Careful design was necessary because the crews' working environment was prone to interruptions and time pressures so error prevention should have been built into the user interface; for instance, by making the reporting sequence clear. This last point is not made explicit in the LAS report; however, it can be inferred as another contributing factor from the above checklist.

Lessons Learned

The scenario annotator/advisor is currently a prototype/concept demonstrator which we created to gain feedback on the suitability of developing this approach. The feedback obtained so far from demonstrations of the system to industrial users has been reasonably encouraging; however, several problems have emerged. Firstly, the advice often requires human factors knowledge to interpret it. Our reaction to this problem is twofold. Firstly, we intended the system to be used by software engineers who have received at least some HCI training, and secondly to make the advice easier to understand, although this will make it more verbose. The second problem was anticipated from the outset: that marking up scenarios is a labour-intensive task which leads to the question about whether the annotation and traceability between scenarios and models will provide sufficient added value for the effort. As yet we have no answer to this point; however, to persuade industrial users to try out the system, and hence allow us to capture effectiveness data, the next step is to add an information extraction tool to partially automate the markup. Information extraction tools work by being trained to recognize text segments using rules that combine domain specific vocabularies with discourses marker phrases; e.g. *because, as a result*, etc., point to *cause-consequence* components. Another direction is to restrict markup by only documenting parts of a scenario narrative that relate to important requirements. Other problems are concerning the readability of the complex schema graphs and understanding their semantics, although these problems were alleviated by follow-up explanation, so an explanation facility is another extension. The concept of integrating communication/argumentation and the modelled domain was considered worthwhile as documentation on scenarios design discussion and models tended to be kept separately, making traceability difficult. Reaction to the system advice was most favourable overall, although the users pointed out that this could be driven directly from the schema graph without the scenarios.

Conclusions

The contribution that the scenario advisor tool has made so far is to explore the feasibility of tool support for eliciting conceptual models by generalization from scenarios and delivering advice on human factors issues in system development. The focus on human error was motivated by our previous work modelling the causes of system failure and human error using Bayesian Belief Networks (Galliers, Sutcliffe & Minocha, 1999; Sutcliffe, 1993). BBN models enable error probabilities to be predicted for system operators and software components by running system models against scenarios describing the environment, agents and task. However, BBN models hide the knowledge that motivated their construction, so in validation studies users requested more explicit representation of that knowledge. We have reverse engineered the knowledge out of the BBN to make it available as a checklist. One future test of the advisor prototype is to try it in combination with the BBN tool. The advice contained in the current systems is preliminary and will be improved by tailoring it with more domain-specific evidence; however, our initial intention was to evaluate the feasibility of tool-based assistance for functional allocation.

The second contribution of this work is to propose a role for model-driven advice in computer aided systems engineering. Our source of advice in the safety critical systems and human factors literature (Bailey, 1982; Hollnagel, 1998; Reason, 2000) needs to be imported into mainstream system development since many requirements failures, which the LAS system illustrates, could be prevented by more systematic analysis of functional allocation and potential causes of error. Furthermore, we believe that embedding such advice in model editors allows it to be delivered in the appropriate context during modelling activity. Further validation tests of our existing preliminary prototype are the next step to assess the utility and effectiveness of a scenario annotator/advisor tool.

Acknowledgements

This research was supported by EPSRC Systems Integration Programme SIMP project (Systems Integration for Major Projects). Special thanks to David Corrall in BAE Systems for his valuable comments and help with this research.

References

Bailey, R.W. (1982). *Human Performance Engineering: A Guide for System Designers*. Prentice Hall, Englewood Cliffs NJ.

Carroll, J.M.(1995). *Scenario-based Design: Envisioning Work and Technology in System Development*. John Wiley, New York.

Carroll, J.M. (2000). *Making Use: Scenario-based Design of Human-computer Interactions*. MIT Press, Cambridge, MA.

Carroll, J.M., Mack, R.L., Robertson, S.P, and Rosson, M.B. (1994). Binding Objects to Scenarios of Use. *International Journal of Human-Computer Studies* 41:243-276.

Chung, L., and Nixon, B.A. (1995). Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach. In *Proceedings of the 17th International Conference on Systems Engineering*. IEEE Computer Society Press, Los Alamitos, CA. 25-37.

Cocchiarella, N.B. (1995). Knowledge Representation in Conceptual Realism. *International Journal of Human-Computer Studies* 43:697-721.

Curtis, B., Krasner, H., and Iscoe, N. (1988). A Field Study of the Software Design Process for Large Systems. *Communications of the ACM* 31(11):1268-1287.

Daren, A, Harrison, M. and Wright, R. (2000). Allocation of Function: Scenarios, Context and the Economics of Effort. *International Journal of Human-Computer Studies*,. 52: 289-318.

Finkelstein, A., and Dowell, J. (1996). A Comedy of Errors: the London Ambulance Service Case Study. In *Proceedings of the 8th International Workshop on Software Specification & Design IWSSD-8*, IEEE Computer Society Press, Los Alamitos, CA. 2-4.

Galliers, J.R; Sutcliffe, A.G; and Minocha, S. (1999). An Impact Analysis Method for Safety-critical User Interface Design. *ACM Transactions on Computer-Human Interaction* 6:341-369.

Guarino, N. (1997). Understanding, Building and Using Ontologies. *International Journal of Human-Computer Studies* 6**:**293-310.

Hollnagel, E. (1998). *Cognitive Reliability and Error Analysis Method: CREAM*. Elsevier, Amsterdam.

Hovy, E.H. (2001). Comparing Sets of Semantic Relations in Ontologies. In R. Green and S.H. Myaeng (Eds) *Semantics of Relationships*.

Kyng, M. (1995). Creating context for design. In J.M. Carroll (Ed.) *Scenario-based Design*. Wiley, New York. 85-108.

Leveson, N.G. (1995). *Safeware: System Safety and Computers*. Addison Wesley, Reading, MA.

Mann, W., and Thompson, S. (1988). Rhetorical Structure Theory: Toward a Functional Theory of Text Organization *Text* 8:243-281

Mylopoulos, J. (1998). Information Modelling in the Time of the Revolution. *Information Systems* 23:127-155.

Potts, C., Takahashi, K, and Anton, A.I. (1994). Inquiry-based Requirements Analysis. *IEEE Software* 11:21-32.

Reason, J. (1990). *Human Error*. Cambridge University Press, Cambridge.

Reason, J. (2000). *Managing the Risks of Organizational Accidents*. Ashgate, London.

Rolland, C. Arhur, B.C., Cauvel, C., Ralyte, J., Sutcliffe, A.G., Maiden, N., Jarke, M., Haumer, P., Pohl, K., Dubois, E., and Heymans, P. (1998). A Proposal for a Scenario Classification Framework. *Requirements Engineering* 3:23-47.

Sowa, J. F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, Pacific Grove, CA.

Sutcliffe, A.G. (2000). Requirements Analysis for Socio-technical System Design. *Information Systems* 23:213-233.

Sutcliffe, A.G., Maiden, N.A.M., Minocha, S., and Manuel, D. (1998). Supporting Scenario-based Requirements Engineering. *IEEE Transactions on Software Engineering* 24:1072-1088.

Sutcliffe, A.G. (1993). *Modelling Business Goals and Requirements Acquisition: report HCID/93/10.* City University, London.

Van Heijst, G., Schreiber, A.T.., and Wielinga, B.J. (1997). Using Explicit Ontologies in KBS Development. *International Journal of Human-Computer Studies*. 45:183-292.

Waterson, A., and Preece, A. (1999). Verifying Ontological Commitment in Knowledge-based Systems. *Knowledge-based Systems* 12:45-54.

Wright, P.C., Dearden, A.M., and Fields, R. (1997). Function Allocation: A Perspective from Studies of Work Practice. *International Journal of Human-Computer Studies* 52:335-356.

Yu, E. (1997). Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In *Proceedings of the 3rd IEEE Int. Symposium on Requirements Engineering*. IEEE Computer Society Press, Los Alamitos, CA. 226-235.