

# *DesignExpert*: A Knowledge-Based Tool for Developing System-Wide Properties

Lee Scott Ehrhart<sup>†</sup>, Tanya Korelsky<sup>°</sup>, Daryl McCullough<sup>★</sup>  
Joe McEnerney<sup>°</sup>, Scott Overmyer<sup>†</sup>, Owen Rambow<sup>★</sup>  
Franklin Webber<sup>°</sup>, Robert Flo<sup>‡</sup>, and Douglas White<sup>‡</sup>

★ CoGenTex, Inc  
840 Hanshaw Road, Suite 11  
Ithaca, NY 14850-1589

‡ AFRL Rome  
Rome, NY

† Drexel University  
Philadelphia, PA

° Key Software  
Ithaca, NY

Contact: [owen@cogentex.com](mailto:owen@cogentex.com)

Length: 4633 words and about 4.5 pages of figures

## Abstract

Many failures of software systems may be traced to inadequate development of system-wide attributes, such as system security, reliability, and usability. Despite the recognized importance of system-wide requirements, commercially available computer-aided software engineering (CASE) tools do not provide any special assistance for satisfying them, instead concentrating on support for developing and rapidly prototyping a system's functionality. This paper presents *DesignExpert*, a knowledge-based tool intended to aid system developers and other stakeholders to effectively address system-wide requirements. The tool elicits requirements, analyzes needs, generates design alternatives, and suggests evaluation strategies.

## 1 Introduction

After more than three decades of experience and tremendous advances in hardware and software technologies, organizations with critical system requirements are finding that the systems delivered still fail to meet the operational need. In many cases, the problem lies in a failure to correctly identify and address the system-wide requirements (such as usability, security, fault-tolerance, maintainability, and real-time requirements) that define the quality and effectiveness of system performance.

This paper presents *DesignExpert*,<sup>1</sup> a computer-aided software engineering (CASE) tool prototype which aids system developers and other stakeholders to effectively deal with system-wide requirements. To our knowledge, it is the only CASE tool to specifically address several types of system-wide requirements.

The paper presents an overview of system-wide requirements and the related life cycle development issues in Section 2. Section 3 follows with an overview of *DesignExpert*, and Sections 4, 5, and 6, respectively, discuss in detail the three domains of expertise it supports: human-computer interaction, security, and fault tolerance. The paper concludes with a brief reference to an imminent formal evaluation effort (whose results will be discussed in the final paper). Given the complexities of the three areas of expertise covered by *DesignExpert*, we cannot in this paper give a full exposition of the way in which *DesignExpert* reasons and the knowledge it contains – that must wait for more specialized expositions.

---

<sup>1</sup>*DesignExpert* is a collaborative development effort jointly undertaken between CoGenTex, Inc. (Prime Contractor), Drexel University and Key Software, Inc. (Contractors), funded by USAFRL. The HCIA component is based on the DesignPro system (Ehrhart et al., 1996).

## 2 System-Wide Requirements

### 2.1 Importance of System-Wide Requirements

System-wide requirements, often called nonfunctional requirements, define and constrain most or all of the functional modules of a system rather than a small subset (Davis, 1994). Common examples of system-wide requirements are the performance, usability, or maintainability of a system. *DesignExpert* concentrates on three particular system-wide requirements: security, fault tolerance, and human-computer interaction. Each of these requirements is both non-functional and system-wide:

- A system is secure if it protects data in specific ways, from unauthorized disclosure or tampering, or from unavailability. Security differs from functionality because data protection may be needed regardless of how the system is expected to process its data. Security can sometimes be enforced in a single module, a gateway to the system, but more typically it depends on the interaction of many modules that form the trusted part of the system. A mistake in implementing one of these modules can easily compromise the security of the entire system.
- Similarly, a system is considered fault tolerant if it will continue to function correctly in spite of one or more failures of its components. Fault tolerance should not change a system's functionality much, if at all, but rather should protect that functionality from specific kinds of failures. Because failures can occur in any module, fault tolerance is necessarily a system-wide concern.
- Finally, human-computer interaction design involves the allocation of tasks between operator and system, presentation of information, and the ergonomics of interaction. Thus, human-computer interaction involves more than the functionality of the system's external interfaces — it involves the overall system concept.

Thus, each of these requirements depends on most or all of the system; none could be satisfied simply by adding or fixing a single module unless most or all of the other modules were already designed with the system-wide requirement in mind. Any new module added to the system would be affected by these system-wide requirements. As a result, while system-wide requirements are important throughout the software engineering process, they are particularly important at the early stages: requirements gathering and formulation, and design.

### 2.2 Communicating about System-Wide Requirements

System engineering is a complex cognitive task involving many stakeholders with specialized knowledge, needs, and expectations. As in all cooperative human problem-solving tasks,

communication among stakeholders plays a crucial role in successful system engineering. Communication is particularly important during the requirements gathering and analysis phase, since this phase involves the elicitation, interpretation, and formalization of informally expressed domain knowledge, and it involves a great number of stakeholders of different technical background (clients, domain experts, requirements engineers, system engineers, end users). In addition, since system-wide requirements can imply a wide array of very specialized knowledge, we must assume that most of the stakeholders involved in systems engineering are not experts in all the relevant types of system-wide requirements. Lack of expertise makes communication even more difficult, and even more crucial: for example, a client will need to know at all stages of the requirements engineering process why certain security requirements have been posited by the requirements engineer, and he or she needs the requirements explained in a manner that is comprehensible to him or her.

Finally, the standardization of documentation procedures (such as those defined by DoD STD 2167A or IEEE Std 1498) places considerable burden on the participants in the system engineering effort, especially in those domains in which they are not experts.

### **2.3 The Need for Support in the Development of System-Wide Requirements**

While system functionality typically can be changed by modifying a few modules, the non-functional requirements constrain how functionality is delivered by the system. Because a system-wide requirement affects many or all modules in a system, “retrofitting” the system to satisfy a new requirement usually has a considerable cost. There is a pressing need for low-cost design, development, and testing environments that permit assessments about which investments in system-wide requirements make sense and which should be avoided.

Specifically, we have identified the following needs:

- Requirements engineers who are not specialists in a particular type of system-wide requirement may not know what information to elicit from clients and domain experts.
- Requirements engineers may have difficulty in drafting relevant documents because they do not know what needs to be recorded, nor what language is appropriate.
- Clients (such as Government program managers), requirements engineers, and designers may have problems estimating what requirements can be realistically met within the given constraints of time and budget.
- System designers may not know how to address certain system-wide requirements.

However, despite the recognized importance of system-wide requirements, commercially available computer-aided software engineering (CASE) tools do not provide any special assistance for satisfying them, instead concentrating on support for developing and rapidly

prototyping a system's functionality. Meanwhile, a considerable body of knowledge about system-wide requirements has been accumulated and can be made available to developers. The goal of *DesignExpert* is to make this knowledge available to different stakeholders in convenient manner.

### 3 System Overview

*DesignExpert* is a knowledge-based CASE tool prototype composed of three assistants, each giving advice in its own domain of specialization:

- The Security Assistant (SA) provides advice on issues relating to data security.
- The Fault-Tolerance Assistant (FTA) provides advice on employing hardware and software resources to achieve a given level of fault-tolerance.
- The Human-Computer Interaction Assistant (HCIA) provides advice on the design and evaluation of information presentation and interaction protocols.

The three assistants are conceptually independent of each other. This means that the user could use a single one of the three assistants; there is no need to use all three assistants at each session or for each project. At the same time, they share basic concepts of interaction. Each elicits information from the analyst by requesting answers to specific questions about the system to be designed. Each provides a recommendation and/or analysis of the data.

*DesignExpert* addresses all phases of systems engineering, but concentrates on the requirements analysis and design phases. The tool itself does not presuppose any particular software engineering process and can be used in the context of any processes, as long as the process has steps corresponding to the standard phases. It can also interoperate with other CASE tools. In addition, the flexible report generation facility can easily be adapted to help satisfy the reporting and documentation needs of any specific process.

To increase its utility, *DesignExpert* is implemented using a platform-independent architecture. The advisory function is implemented in CLIPS 6.0. The user interface is implemented in HTML using CGI and JAVA to handle input/output and serve sessions. This interface architecture permits economical use of multimedia on UNIX, Windows, and Macintosh platforms.

## 4 The Security Assistant: Supporting System Integrity

### 4.1 SA Overview

The Security Assistant (SA) supports the formulation of security requirements and choice of security mechanisms to meet those requirements.<sup>2</sup> In this process, SA helps the user balance the cost of countermeasures and the likelihood that a threat will be actualized, on the one hand, with the value of assets and the importance of the system mission, on the other.

A security requirement is a written formulation of the objective of opposing some attack on, threat to or vulnerability of the given system. System security requirements derive from several sources, among which the most noteworthy are:

- Mission needs and concepts of operations
- Risk and threat assessments of potential deployment sites
- Requirements stipulated for similar systems
- Organizational security policies
- Mandatory requirements (for government systems)

The SA uses a series of HTML surveys (questionnaires) to gather security-related information about a system's functional architecture and its deployment environment into the SA repository and then analyzes that data.

The SA produces two kinds of output:

- The SA automatically generates reports in fluent English that summarize the information entered into the repository.
- Given the information about the system functionality and its deployment environment, the SA provides a critique of chosen security measures and recommends countermeasures of appropriate strength to offset known or potential risks. This critique is also presented to the user in the form of a report automatically generated in fluent English.

The Security Assistant is designed to provide support to users during the pre-implementation phases for new systems and maintenance and upgrade phases for legacy systems. Several different groups of users might benefit from the SA's advice:

---

<sup>2</sup>For a more complete exposition of the theoretical approach underlying the SA and the FTA, see (Webber et al., 1998).

1. Requirements engineers can use SA to maintain a representation of the intended system's security requirements as the system's requirements evolve (and, later, as the system itself evolves). They can use SA to ascertain that they have gathered the relevant information. The report generation capability facilitates the communication about requirements with other stakeholders and the documentation of the requirements.
2. System designers benefit by including security as a system wide property intertwined with functional and architectural design and not as an add on. Moreover, this group benefits because SA can handle generic security concerns thus giving them the freedom to concentrate on the really unique security problems that are specific to each system.
3. Government system certifiers, who are responsible for determining the residual risk associated with deploying these systems for a given mission in a given environment often have to treat each case separately. Moreover, they must contend with a mountain of required paper work as well as perform technical analyses.
4. System Administrators and IT Managers benefit by having a tool that can help organize and analyze systems and networks that they are responsible for maintaining. These professional are frequently tasked with enhancing such systems by deploying new applications or hosts. This can cause security compromises for either new or old components. These decision makers need advice that reflects the IT system's mission as well as the risks to which it is subjected.

## 4.2 SA's methodology of system description

Modern distributed systems applications typically consist of hardware and software components that are themselves systems. This leads to the notion of "system of systems". From a security standpoint this is a very important perspective since not only must component systems be trustworthy but also they must be combined in such a way that the resulting system of systems is also trustworthy. The Security Assistant has been designed with this perspective in mind.

A system under study is described as a set of interrelated functional components which are deployed (and possibly replicated) at sites. Viewed from the topmost level, the target system has a global operational environment which is made up of the site environments. Threats to security come from these environments and can arise from natural or human sources. It is this threat space that characterizes the risks run by the target system. Ultimately, security of the target system is achieved by determining and implementing a Security Policy which counters the threats and enables the system missions to serve its clients while protecting its assets.

Although a target system security policy is the ultimate goal, elements of it are often known in advance or are dictated by organization policy. For these reasons, several security policy elements appear as both inputs and outputs of the SA.

In SA's methodology, the security-related information about the system is gathered at four levels:

1. The system-wide level. This survey gathers information about the system's functional architecture, its missions and clients, its main assets, its system-wide security characteristics, and the list of sites that constitute its deployment environment.
2. The site level. A site is a collection of hosts that are usually concentrated geographically in one region, logically related by common purpose, physically connected to each other by a LAN, pre-exist the target system, and often perform functions unrelated to the target system. The site survey gathers information about what system assets are deployed at the site, detailed information about the site's physical, computing and communications environment, as well as its detailed security characteristics.
3. The host level. Hosts are connected to the site to provide the platforms onto which target system functional components are deployed. Hosts are described by specifying hardware configuration, operating system, the set of deployed target system software components which implement system assets described on the higher levels, and the suite of other installed applications (if any) and libraries.
4. The software component level. The software component level provides the user with a place to characterize how host resources are shared between the target system software and peer software on the same host.

The system-wide survey describes the target system from the functional point of view, while the lower-level surveys gather the deployment information.

### **4.3 Automatic reasoning for security analysis and advice**

SA uses an expert system to analyze and critique user descriptions of a system under study. Knowledge of security derived from expert experience, publications and DoD standards such as the TCSEC and the Common Criteria has been encoded in rules that are activated by the presence or absence of relevant facts in the SA's repository.

By determining what types of damage an asset or mission can sustain, what kind of attack methods are possible, how likely it is that an attack can be mounted and what system defenses are available to counter attack methods it is possible to formulate inference engine rules that reflect rules of thumb like the following:

- Make sure that every asset is protected from every known potential attack method.
- Balance the cost of the system defenses with the asset values and the mission importance.

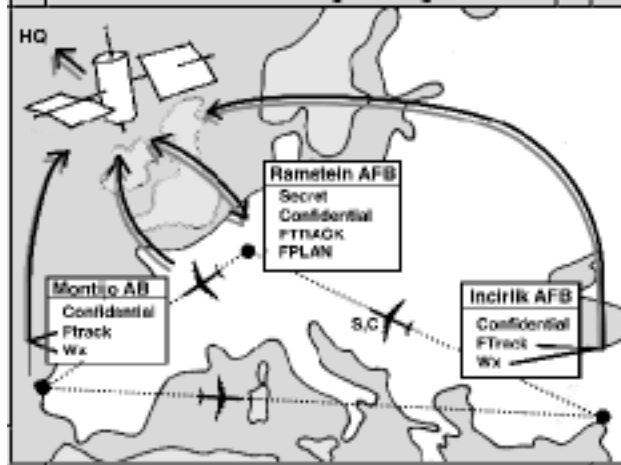


Figure 1: Sample MPATS System architecture

#### 4.4 Example Use of Security Assistant

In this section we describe a scenario of using Security Assistant for a Mission Planning and Tracking System (MPATS). MPATS is an experimental secure distributed application (implemented but not deployed). The purpose of MPATS is to plan and track military airlift missions such as the one shown below. It consists of several components distributed across several sites (see Figure 1). The boxes represent sites and list the functional components deployed at that site.

In our scenario, the user is either a requirements analyst or a certifier operating during the requirements analysis phase. The SA user first describes MPATS on the system-wide level, using the suite of system-wide surveys. The security environment survey is shown in Figure 2.

The user can validate or document the entered system-wide information using an automatically generated summary of this information (see Figure 3 for the summary of the information entered in the survey of Figure 2).

The current automated security analysis works on the system-wide level and takes into account only a part of the site security-related deployment information supplied by the user. The attributes that are most important for the automated analysis are: site network connections, security mechanisms at the site and site hostility degree.

The user then asks the SA to analyze the system. The SA makes three recommendations, as shown in Figure 4. The automatically generated explanation of the reasoning underlying the first recommendation is shown in Figure 5.<sup>3</sup>

If more detailed deployment information is available to the user, the user can use host and

<sup>3</sup>For a detailed account of the *DesignExpert* explanation facility, see (Barzilay et al., 1998).

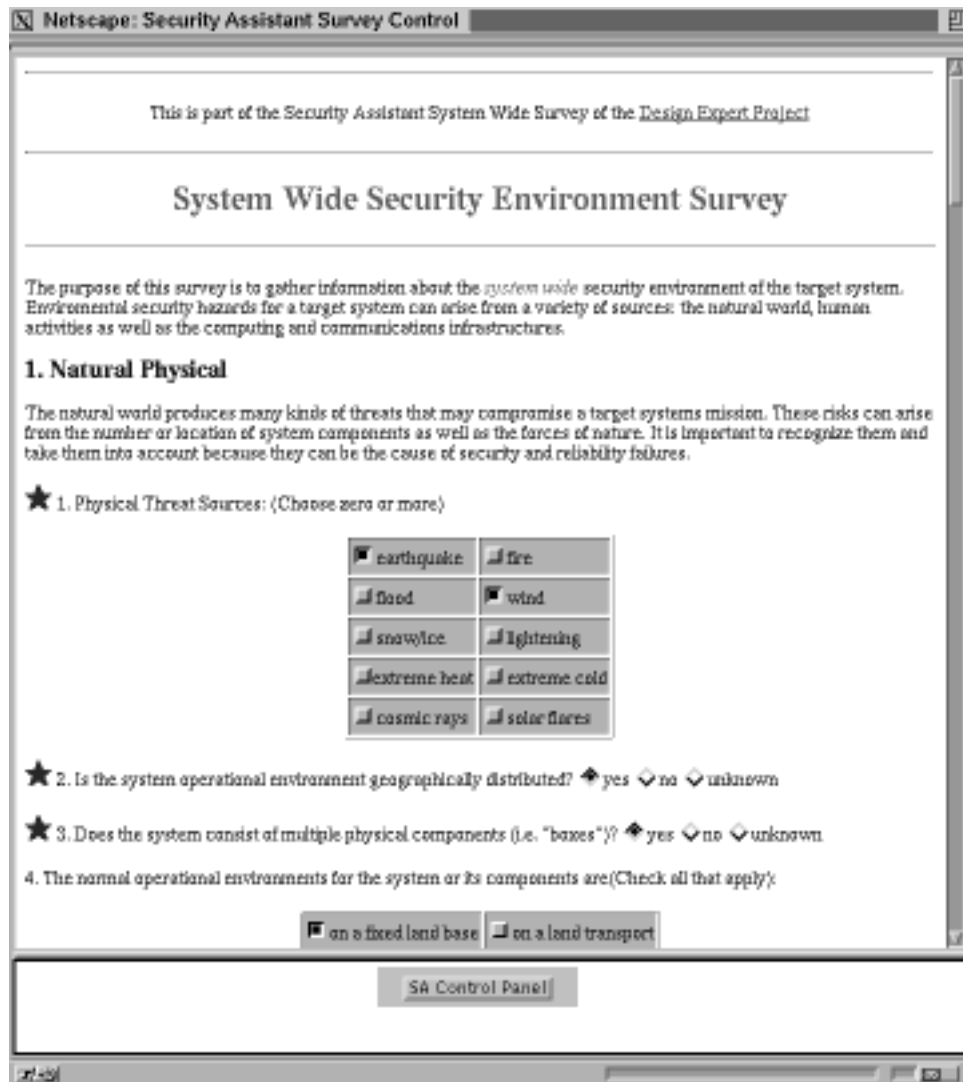


Figure 2: SA survey screen for system-wide security environment (excerpt)

software component surveys to enter this information into SA's repository. Figure 6 shows an excerpt from the Host survey, and Figure 7 the corresponding summary.

## 5 The Fault Tolerance Assistant: Supporting System Reliability

The Fault Tolerance Assistant (FTA) is *DesignExpert's* support for developing reliable distributed computer systems. Using FTA helps to posit realistic reliability and availability requirements, and increases confidence that a system's design satisfies its reliability and

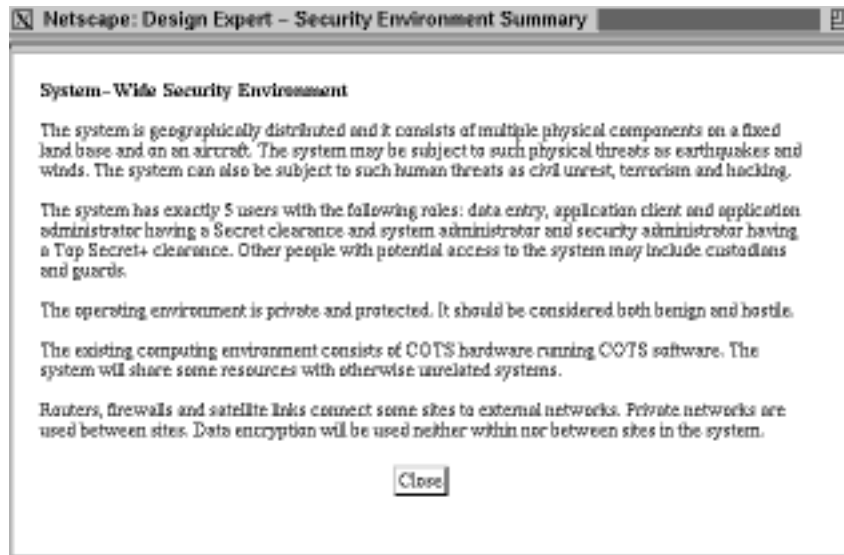


Figure 3: Summary of SA survey for system-wide security environment

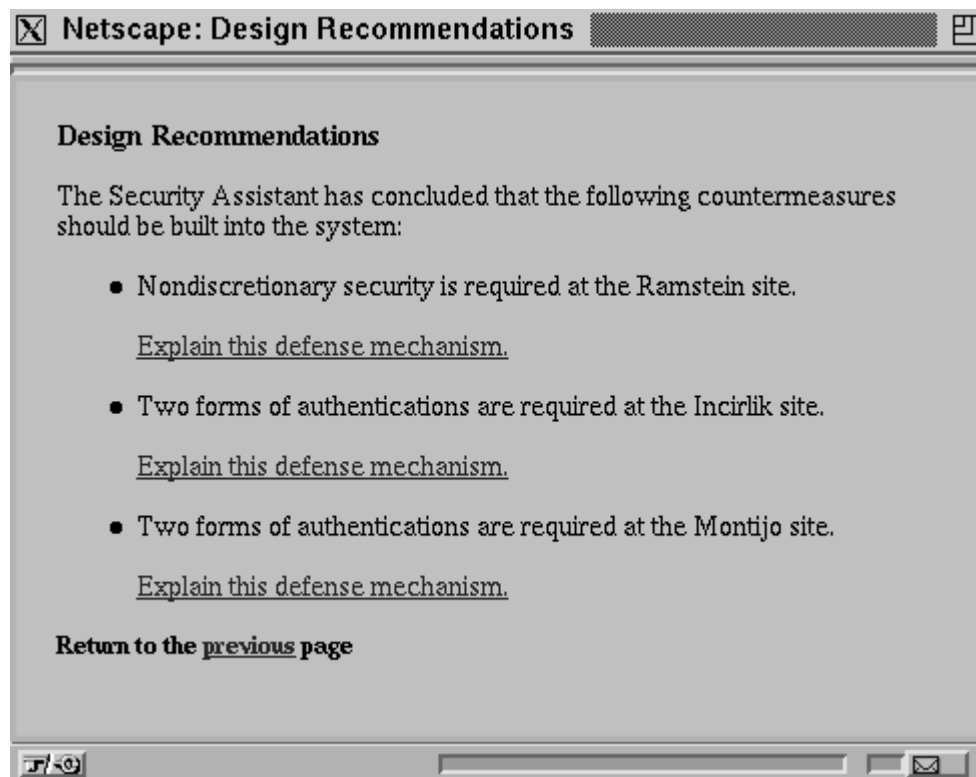


Figure 4: SA recommendations for MPATS system

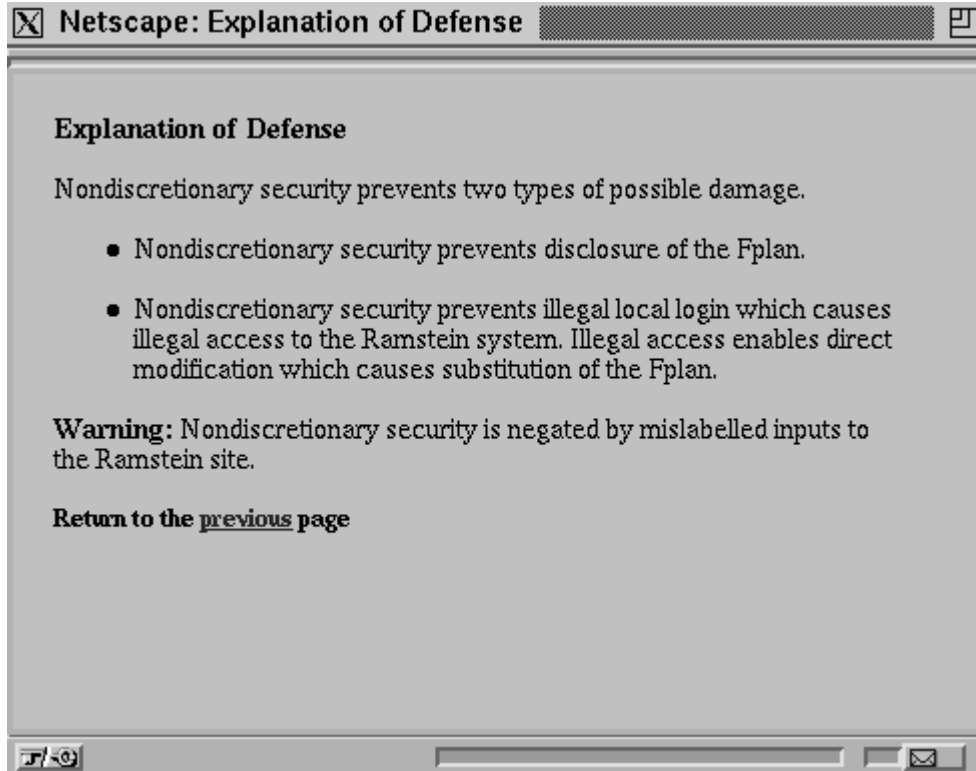


Figure 5: Automatically generated explanation of SA recommendation

availability requirements.

FTA surveys its user's requirements for system reliability and availability. A *reliability* requirement tells how likely it is for a system to give continuously correct operation for a given duration. An *availability* requirement tells how likely it is for a system to be operating correctly at any given time. Several similar requirements can also be specified in FTA (Siewiorek and Swarz, 1982):

- the mission time;
- the mean time to failure (MTTF);
- the mean time between failures (MTBF);
- the mean time to repair (MTTR).

The availability, MTBF, and MTTR requirements depend on the possibility that failed components of the system can be repaired, while the others do not.

To analyze the system requirements, the FTA user must supply an abstract model of the system. This model is hierarchical: every component and subcomponent may be composed of other subcomponents. The system model includes both hardware and software.

## 4. Host Security Attributes

|   |  |
|---|--|
| Mode of Operation                               | Multi-level <input type="checkbox"/>   |
| Highest Classification on Host                  | Secret <input type="checkbox"/>  |
| Authentication Devices<br>(choose zero or more) | <input type="checkbox"/> Behavioral Monitor<br><input type="checkbox"/> Finger Print Scan<br><input type="checkbox"/> Iris (eye) Scanner<br><input checked="" type="checkbox"/> Smart Card   |
| Miscellaneous<br>(choose zero or more)          | <input checked="" type="checkbox"/> Host Shared with Peers.<br><input checked="" type="checkbox"/> Audit Is Turned On.<br><input checked="" type="checkbox"/> Auditing Host for Site.<br><input checked="" type="checkbox"/> All OS Patches Installed.<br><input checked="" type="checkbox"/> All CERT Advice Followed.<br><input type="checkbox"/> Host Is Mobile.<br><input checked="" type="checkbox"/> Host Is in a Secure Room.<br><input checked="" type="checkbox"/> Host Is TEMPEST Shielded.<br><input type="checkbox"/> Host Hardware Is Tamper Proof. |

Figure 6: SA survey screen for the host (excerpt)

Netscape: Design Expert - Security Environment Summary

### Host Survey

**Host Profile**

Host Gauss is a SUN Microsystems desktop workstation, running the SUN CMW operating system. It has the following peripheral devices: a color monitor, an external disk, and a keyboard.

**Host Connectivity and Usage**

Gauss is connected to SIRPNet via a router. It is an MPATS system host and the following MPATS functions are deployed on it: FPLAN, FTRACK, and Wx (Weather Forecasting).

**Host Security Attributes**

Gauss is a multilevel host with a highest classification level of SECRET. It uses Smart Card as the authentication mechanism.

Close

Figure 7: Summary of SA survey for a host

A textual representation of the model is built in the Acme architecture description language (Garlan et al., 1997). We chose the Acme language to be the standard architecture description for FTA because it is a simple language designed to allow easy translation to the architecture description languages used by other tools. The Acme language also can be easily extended by associating attributes with each component of an architecture.

The system model can include the following attributes for each component:

- replication factors;
- failure and repair rates;
- number of failures tolerated in each replicated ensemble;
- the failure model (e.g., Crash; Byzantine) assumed and the failure model guaranteed;
- the assignment of each software component to the hardware component on which it runs.

FTA produces these outputs:

1. A Monte Carlo simulation of the system model. This simulation assumes that each atomic element of the system model undergoes failure and repair at constant rates. The steps in the simulation can be made visual using a graphical architecture editor developed at Key Software.
2. Estimates of reliability, availability, and other measures of goodness of a given system model. To estimate reliability, mission time, and MTTF, the simulation is run many times and the results averaged. To estimate availability, MTBF, and MTTR, a very long simulation is run in which the system can fail and be repaired many times and averages are computed from that single run. FTA also estimates the uncertainty in each of these averages.
3. A critique of the attributes in the system model. FTA looks for several kinds of inconsistencies in these attributes, including: whether the fault tolerance specifications can be achieved given the model of failures; whether obvious failure correlations exist; whether the failure model assumed by a component  $X$  can be guaranteed by the components on which  $X$  depends.
4. Recommendations for increasing reliability. Improvements in reliability can sometimes be had by increasing the degree of replication and sometimes by changing the allocation of software components to processors (Nieuwenhuis, 1990). FTA estimates the potential for such improvements using a “quick and dirty” analysis rather than using the computationally-intensive (but more accurate) Monte Carlo simulation.

## FTA Analysis

---

Latest Run

### Run #1

#### System Requirements and Simulation Results

- Reliability:
    - required: 95% for a 12 hour mission
    - estimated: 97.0652 +/- 0.49505%
    - design satisfies this requirement
  - MTTF:
    - required: 500 hours
    - estimated: 619.205 +/- 63.1334 hours
    - design may satisfy this requirement
    - try running the simulation again to improve the precision of this estimate
  - Steady-State Availability:
    - required: 99%
    - estimated: 99.0174 +/- 0.15066%
    - design may satisfy this requirement
    - try running the simulation again to improve the precision of this estimate
- 

Close

Figure 8: Sample FTA analysis

For a sample of an FTA analysis output, see Figure 8.

FTA's design advice should lead to better system models and ultimately to a sound judgment of whether particular reliability requirements can be satisfied. If a model satisfies the requirements, the number and kind of components in the model can be used to estimate the final cost of building the real system.

The users of the FTA include the following groups:

- Customers (such as Government program engineers) and requirements engineers can estimate whether requirements for fault-tolerance are realistic.
- System designers can ascertain that a system's design satisfies its reliability and availability requirements.

FTA's requirements survey can be reached from *DesignExpert's* Security Assistant (SA) (see section 4), thus coupling the two assistants together.

## 6 The Human-Computer Interaction Assistant (HCIA)

Human-computer interaction design embodies most of the system concept that is “available” to the user to guide his/her mental model of the system. For example, the HCI design incorporates such critical system design factors as:

- representation of information regarding the situational elements external to the system (support systems, environment, threats, etc.);
- representation of system states and feedback to the operator on results of actions taken;
- allocation of tasks between the human users and the computer as determined by the dynamics of the situation and the requirements of the methods selected to support task performance; and
- modes in which users may interact with all of this information to explore situations, develop hypotheses, generate options, select among alternatives, and implement their decisions.

The goals of the HCIA include support for the following issues:

1. Cognitive Engineering: Design of displays and interaction routines consistent with what we know about human cognitive structures and processing; displays that are consistent with human data organization and “storage,” and the way humans frame problems, make inferences, generate options, and implement action;
2. Organizational Focus: Permitting the user to examine HCI designs in the context of the larger system issues, such as its integration with other systems and coordination across users, teams and organizations;
3. Technology exploitation: Suggesting the innovative use of image, text and graphics processing technology to support real-time animation, user or system-directed searches, multidimensional displays, and “virtual” processing environments to improve the transparency of the interaction between the users and their tasks;
4. Designing for error: design of information displays and interaction protocols for preventing and controlling the impacts of operator error to improve system performance and reliability; and
5. Evaluation: Suggesting techniques for evaluating the proposed HCI design and new system concepts to identify potential risks not only to overall system performance, but also to the system development effort.

At the core of the Human-Computer Interaction Assistant (HCIA) is the concept of “interaction” with computers, information, problems, algorithms, displays, etc. We regard interaction as any exchange between human and digital-electronic systems. Using the “interaction” paradigm, the HCIA knowledge base synthesizes findings from research in the cognitive sciences with knowledge about the information display and interaction technologies to support human-computer interaction analysis, prototyping, and evaluation.

The HCIA user first identifies the kinds of cognitive tasks the users may be required to perform and examine the factors affecting performance. If a task affects decision performance, it is necessary to find out what characteristics of the task do so. Meister (1981) identifies five task dimensions that may affect performance:

- Functional requirements (cognition, perception, etc.)
- Complexity
- Mental workload
- Temporal factors (pace, duration, sequence, etc.)
- Criticality

Cognitive task taxonomies, such as those found in (Fleischman and Quaintance, 1984) and (Rasmussen et al., 1990) are used as a filter to identify and categorize basic cognitive tasks with respect to these dimensions. In addition, Andriole and Adelman (1995) present a taxonomic discussion of human information processing and inferencing tasks with respect to the potential cognitive errors associated with each. These sources and others were used in the design of the HCIA to define a set of task characteristics (e.g., task complexity, etc.) for which the user sets parameter values using a simple categorization of High-Medium-Low.

More precisely, the HCIA queries the user for information pertaining to four areas:

- users — experience, training, organizational roles;
- tasks — high-level functions, performance goals, decision task characteristics (e.g., timing, criticality, etc.);
- organizational context — organizational goals, missions, control structures, communication modes, “culture”; and
- environmental context — when, where, how, and under what conditions the system will be used.

In addition, the HCIA requires the user to further bound the design options by identifying:

**DesignExpert**

Requirements Design Issues Design Advice  
Evaluation Submit HCIA Home

HCI Requirements Definition

**Task Characteristics**

Radio buttons are used to define the characteristic and its impact on the system. Select the characteristic to view a description and create annotations.

|                                      | Low                              | Medium                           | High                             | Unknown               |
|--------------------------------------|----------------------------------|----------------------------------|----------------------------------|-----------------------|
| <a href="#">Task Familiarity</a>     | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| <a href="#">Visual Inspection</a>    | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |
| <a href="#">Time Threshold</a>       | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> |
| <a href="#">Complexity</a>           | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |
| <a href="#">Problem Boundaries</a>   | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| <a href="#">Information Conflict</a> | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |
| <a href="#">Proceduralism</a>        | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| <a href="#">Direction</a>            | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |
| <a href="#">Criticality</a>          | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |

Save/Submit Reset

Figure 9: Sample HCIA questionnaire for task characteristics

- any system-level constraints (such as hardware platform, storage limitations, operating system, etc.);
- and project variables (such as contract flexibility and project team composition).

To aid in this definition, the system provides descriptions of each task characteristic and an explanation of the criteria for assigning the parameter value. Space is also provided for user annotation to explain the designer's rationale for the rating given the characteristic in the context of the project. As an example, the HCIA questionnaire for task characteristics is shown in Figure 9.

Once the profiling is accomplished and constraints are specified, the analyst submits the requirements to the HCIA knowledge base for design suggestions regarding technologies and techniques for user-machine dialog, information presentation, and other interface and interaction design features. HCIA presents design advice for specific technologies and techniques in three categories.

1. data and information coding;

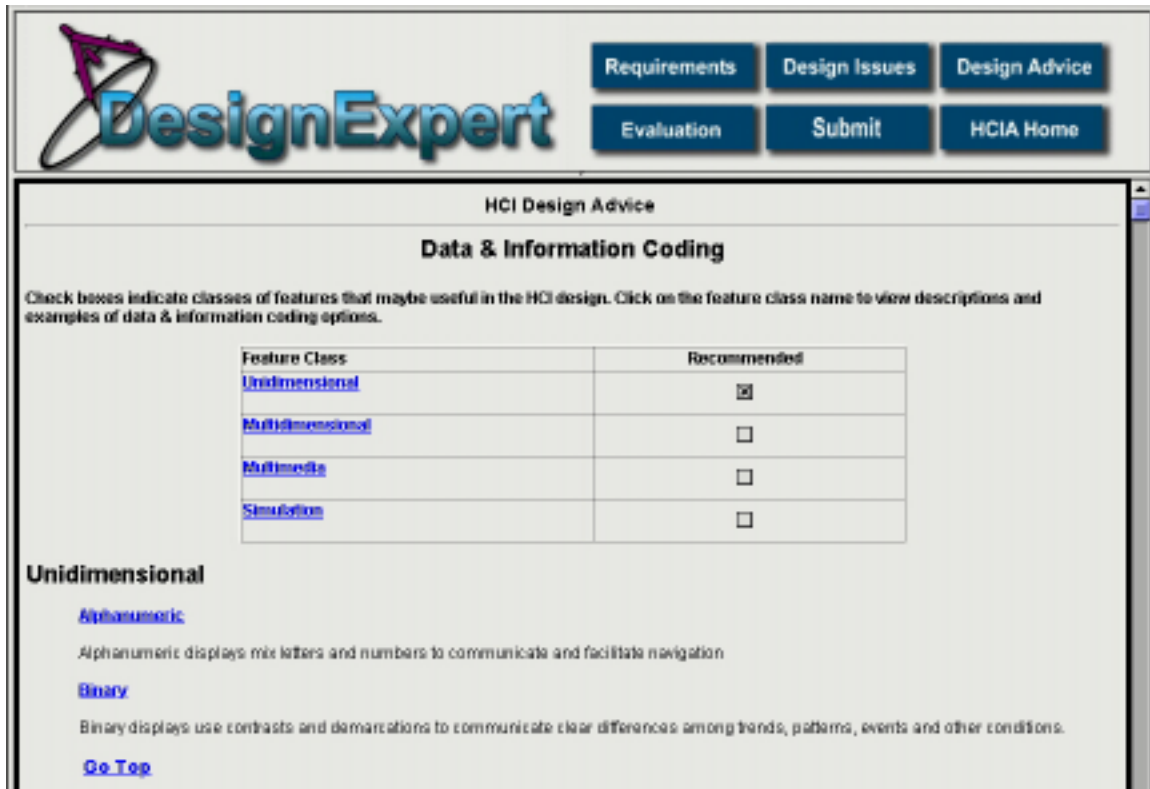


Figure 10: Sample HCIA design advice for data and information coding

2. display design and features;
3. and dialog and interaction styles

The HCIA presents definitions and an example for each possible feature and a recommendation as to its appropriateness for the system under development. Figure 10 shows a sample HCIA design advice for data and information coding.

To complete the life cycle support, the HCIA provides reference information on the critical design evaluation issues and suggests approaches to evaluation of the HCI design, user performance, and system usability. The evaluation screens identify and describe a range of objective and subjective methods for assessing performance and workload (a sample screen is shown in Figure 11). Throughout the HCIA, HTML links connect the user to further information available online.

The HCIA is designed to be used by analysts and design decision makers, including domain knowledgeable customers (e.g., operational end-users, domain experts, etc.) and software development practitioners (e.g., project managers, analysts, designers, programmers, quality experts, etc.).

**DesignExpert**

Requirements Design Advice Evaluation Submit HCIA Home

### HCI Evaluation Advice: Objective Measures

This section presents advice on objective methods for evaluating system usability and HCI design quality.

- [Performance](#)
- [Workload Assessment](#)

#### Performance Measures

##### Task Performance Time

Time data of various types may be captured to establish a baseline or to compare task allocation/support options.

**Reaction Time** - Time between the occurrence of an event (stimulus) requiring an action and the start of the action (response) demanded by the event. Reaction time is measured to see how quickly the user/team can respond to an event for which failure to respond is critical. Reaction time may be analyzed to predict minimum response times required or maximum response times expected. Measurement requires that a discrete time may be identified for both the stimulus event and the beginning of the response.

**Task Duration** - Time from the initiating stimulus to the time the task or function is completed. Duration is measured to assess the time required to complete a task. Task duration data may be analyzed to predict minimum and/or maximum required for task performance. Duration measurement may not be extremely precise to be useful for comparison purposes.

##### Error Rate/Accuracy

Error/accuracy data are the most common performance measures collected. Performance accuracy is usually dependent upon the time available for task completion; thus, error rates may be expressed in terms of errors committed per time allotted.

In many situations the definition of what constitutes an "error" is not clear. For example, researchers may need to capture errors of omission as well as errors overtly committed. In situations where "ground truth" or objectively correct answers cannot be provided, the assessment of response accuracy becomes a subjective issue. In these cases, researchers may need to establish a consistent means of assessing the relative accuracy of a given response based upon expert judgment or review.

To make use of error data, it is important that not only the error event is captured, but also the reason for error. Establishing error cause often entail a categorizing error according the characteristics of the error, the error context, the result of the error, etc.

##### Time Necessary to Train Users/Operators

Training time may be measured for baseline or comparative purposes by conducting a training case study. In the simplest model, representative users (individuals or

Figure 11: Sample HCIA evaluation advice

The HCIA rule base can be extended to reflect significant changes in interface technology or interaction paradigms. The extension of the knowledge base requires either an understanding of the cognitive system engineering principles and related research that support them or sufficient experience and expertise to make the necessary judgments.

## 7 Summary

The *DesignExpert* approach derives requirements from models of system and user needs at varying levels of abstraction to enhance understanding and consideration of system-wide requirements across the system development life cycle. In the earliest phases of development, a broad-brush approach may be used by planners and analysts to estimate feasibility and scope projects. The initial profiles and models can be elaborated during requirements analysis for greater specificity and used to generate and evaluate design alternatives. The report facilities extend the documentation capabilities and provide a summary of requirements and design issues for use by the implementation and testing teams.

The final prototypes of the three assistants have been implemented. *DesignExpert* will be formally evaluated in the immediate future (using both a focus-group evaluation and a use-based evaluation; for the SA, the use-based evaluation will involve government system certifiers at MITRE). We intend to report on the results of the formal evaluation in the final paper.

## Bibliography

- Andriole, S. J. and Adelman, L. (1995). *Cognitive Systems Engineering for User-Computer Interface Design, Prototyping and Evaluation*. Lawrence Erlbaum, Hillsdale, NJ.
- Barzilay, Regina; McCullough, Daryl; Rambow, Owen; DeCristofaro, Jonathan; Korelsky, Tanya; and Lavoie, Benoit (1998). A new approach to expert system explanations. In *Proceedings of the 8th International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario.
- Davis, Alan M. (1994). *Software Requirements: Objects, Functions, and States*. Prentice-Hall, Inc., Upper Saddle River, NJ, 2nd edition.
- Ehrhart, Lee Scott; Andriole, S. J.; Monsanto, C.; Hibberd, B. J.; and Flo, R. (1996). Designpro: Expert advice for information interaction design, prototyping, and evaluation. In *Proceedings of 1996 Sixth Annual Dual-Use Technologies Applications Conference*, Syracuse, NY.
- Fleischman, Edwin A. and Quaintance, Marilyn K. (1984). *Taxonomies of Human Performance: The Description of Human Tasks*. Academic Press, New York.
- Garlan, David; Monroe, Robert T.; and Wile, David (1997). Acme: An architecture description interchange language. In *Proc. CASCON '97*.
- Meister, David (1981). *Behavioral Research and Government Policy: Civilian and Military R & D*. Pergamon Press, New York.
- Nieuwenhuis, L.J.M. (1990). Static allocation of process replicas in fault tolerant computing systems. In *IEEE Symp. Fault Tolerant Comp.*
- Rasmussen, J.; Pejtersen, A.-M.; and Goodstein, L.P. (1990). Taxonomy for cognitive work analysis. Technical Report Riso M-2871, Riso National Laboratory, Roskilde, Denmark.
- Siewiorek, Daniel and Swarz, Robert (1982). *The Theory and Practice of Reliable System Design*. Digital Press.
- Webber, Franklin; McEnerney, Joseph; and Kwiat, Kevin (1998). The DesignExpert approach to developing fault-tolerant and secure systems. In *4th Int'l Conf. on Reliability and Quality in Design*.