Writing Effective Security Abuse Cases

Thitima Srivatanakul, John A. Clark and Fiona Polack

University of York Technical Report YCS-2004-375

University of York

Department of Computer Science

14 May 2004

Abstract

We grow increasingly dependent on the appropriate operation of computer-based systems. One aspect of such systems is security. As systems become more complex current means of analysis will probably prove ineffective. In the safety domain a variety of analysis techniques has emerged over many years. These have proved surprisingly effective. Since the safety and security domains share many similarities, various authors have suggested that safety techniques might usefully find application in security. This report takes one such technique, HAZOPs, and applies it to one widely used informal design component – UML's use cases.

Table of Contents

List of Figures

List of Tables

1. Introduction

Computer security is increasingly problematic; threats to systems and the data stored on them grow as computer networking and computer literacy increase. Computers are now an obvious target for the disgruntled employee, the rejected partner, the disenfranchised constituent, and the fraudster. With increased opportunity, attacks become more novel, complex and unpredictable.

To make matters worse, the criticality of computer systems is increasing. Many organisations rely on stored data, business or management programs, internet connections and the web/email facilities that the internet supports. Security breaches have the potential to cause dramatic losses of money, confidence, reputation, or even of life.

To combat the increased number and severity of security risks, systems development needs to analyse security with particular care and rigour. Security analysis should be an essential part of all computer-related engineering practices. Although security was one of the first domains to embrace formal development techniques and processes, much security analysis remains informal. At the requirements level most analysis is informal; with the increased complexity of emerging systems there is a pressing need to add rigour to the process of engineering security requirements.

Some requirements may be expressed as 'use cases'. This paper investigates the systematic use of a technique from the safety domain (HAZOPs) and shows how it can be applied to use cases to elicit and explore their security aspects.

1.1 Motivation

Research in security has emphasized particular aspects of security or technologies required to implement security mechanisms. Confidentiality properties, for example, have been analysed mathematically by many authors and there has been a vast amount of research into cryptography. However, these are only partial solutions to the security problem. Much security research has been sponsored by Governmental organisations and the research tends to reflect Governmental security concerns. Modern systems, over a range of application domains, have security issues. Reaching for pre-defined templates of what were relevant security problems and their solutions will often not suffice. Much more subtlety will be needed to determine in what security for a system should consist and how assurance can be gained that the developed system provides it.

It is universally *asserted* that security must be built-in, but proper integration of security development and analysis processes with the development of the rest of the system is not common. Bolt-on 'security' is a frequent approach. One can understand this lack of proper integration. System development is hard enough under benign conditions, let alone in the presence of malice. Attacks may come from all directions, from exploiting radio frequency leakage, through power consumption of a smart card leaking key information, to more social/personnel aspects such as what has recently been described as *cognitive hacking* [23]. Expertise across the relevant domains is

generally not possessed by any one person. Furthermore, for modern complex systems we might question whether 'expertise' is really enough. Expertise depends on previous experience. Many modern systems give rise to new security concerns and experts have a hard time coping with novel systems. Commonly used analysis techniques (e.g. checklists/profiles) are too rigid to uncover new threats.

Integrating security with the design process is further complicated because developments are usually iterative and so security issues must be addressed iteratively (with all that entails). Furthermore, evolution of systems is a natural part of the lifecycle and presents significant challenges for secure systems. Maintaining rigour under change is hard and expensive; much more so if certification requires externally provided evidence, such as that provided by independent evaluation.

The above problems are not unique to security. Safety critical and safety related systems present similar problems, but safety analysis and development would appear more tightly integrated. Part of the reason for this, perhaps, is that the safety domain has developed many simple but effective analysis tools whose results are seen to affect the design process in a significant way. These tools are typically understood by non-safety experts (though their effective use requires skill), allowing various stakeholders to bring their knowledge to bear in the service of safety, or allowing experts to exercise their craft with increased rigour. Might such safety techniques usefully be applied to security?

1.2 Solutions from safety

The scale and scope of security risk in modern computer systems requires systematic, rigorous techniques that are part of standard development processes. Techniques applied in the development of safety-critical systems tend to be well-defined and systematic. They have proven effective over many years.

Safety and security have many similarities. Both are concerned with the management of risk. Safety typically concerns itself with reducing the risk of loss of life or environmental damage to tolerable levels. Security concerns itself with engineering acceptable levels of risk to various assets (and is most typically concerned with threats to the confidentiality, integrity, and availability of information, and threats to the supporting infrastructure). Development concepts and practices from one domain have found interpretation or direct application in the other. For example, notions of graded assurance were first seen in the security domain; the US DoD's Trusted Computer Security Evaluation Criteria [65] (a.k.a. the 'Orange Book', due to the colour of its cover) required increased functionality and increased rigour in design to cope with various levels of system risk. The concept of graded assurance migrated to the safety domain, where functionality and assurance were separated (and so very simple functionality could be required to have high assurance). This separation of functionality and assurance has been adopted in recent security standards, such as the 'Common Criteria' [18]. Formal methods found application in the 1970s and 1980s primarily in security and subsequently became a mainstay of much safety-oriented research.

Perhaps the most important similarity is that rigorous arguments are generally required to justify the deployment of safe and secure systems. Systems must be *demonstrably* safe or secure. In recent years, the notion of a *safety case* has emerged -

a rigorous, well-structured and understandable marshalling of evidence that a system is suitable to be deployed, i.e. is acceptably safe. Safety analysts already have numerous techniques at their disposal for generating and recording evidence and these can be adapted for security assurance evidence. Fault Tree Analysis (FTA) [58] has inspired security Threat Trees [2] and Attack Trees [62]. These approaches systematically decompose goals, representing the findings in a tree structure. MOAT [42] is a security argumentation technique that uses FTA notations to express its assurance arguments. More sophisticated argumentation [57] is available via derivatives of the Goal Structured Notation [39].

Techniques for the investigation of *deviations* (unintended or unexpected behaviour [59]), developed for safety critical systems, also appear in security work. The abuse case model [53][54] expresses deviation from normal system use by specific actors. This work takes as its starting point *use cases*, a requirements technique from the *Unified Modeling Language* (UML) and related development methods. This clearly satisfies the need to integrate security analysis with conventional systems development; UML is increasingly the notation of choice in systems development. However, the technique again focuses on expressing the effect of known security threats; it is not capable of systematically seeking other possible threats. Systematic deviational techniques such as HAZOP [60][68], successfully used in safety critical systems, are little used in security. One doctoral study [31] applies HAZOP to protocol requirements analysis, to systematically investigate unknown threats and attacks on protocols.

We propose an extension of abuse cases, providing a more rigorous approach to analysing security. We apply the principles of HAZOP, tailored to a security perspective, to an existing functional requirements representation (i.e. use cases). The approach systematically mutates the model and its elements, thus prompting identification of a wide range of threats and other non-functional requirements.

The next section of the report discusses the concept and uses of HAZOP in more detail. Section 3 introduces and elaborates use case. Section 4 outlines the proposed method and the necessary interpretations of HAZOP guidewords. Section 5 provides guidance on the derivation of the use case deviants. Section 6 presents a simple case study of an e-commerce system. The report concludes with a summary of our findings.

2. HAZOP: A rigorous approach to security analysis

Security analysis must determine cost-effective controls to make a system's assets secure against credible threats. These threats may include loss of confidentiality, integrity, availability etc. Damage may also include secondary damage, e.g. to reputation. The analysis must explore the vulnerability of the system to these threats, the probability that such vulnerabilities may be exploited to form attacks, and determine the potential impact of attacks. Cost-effective counter-measures are then sought. Clearly, efficient identification of threats is key to the development of secure systems. HAZOP is a technique widely used in safety; it presents a possible approach to systematic security threat identification.

2.1 HAZOP

HAZOP is the technique of Hazard and Operability Analysis (HAZOP). It is a qualitative technique, developed by chemical producers to systematically identify potential hazards and operability problems in designs for new chemical plants. Traditional approaches based on checklists and intuition were capable of checking, for example, the effects of a valve not opening or not closing; HAZOP guidewords encouraged other (more subtle) scenarios to be identified, such as the valve being part-open, or sluggish in its opening. A systematic description of the technique applied to safety in computer and other systems is given in [68][60].

The basic principle of HAZOP is that hazards arise due to ***deviations*** from normal or intended behaviours. Like identification of systems requirements, HAZOP is best conducted as a team analysis activity, ideally using people with various backgrounds. The team, led by a HAZOP leader, systematically investigates each of the system elements to identify deviations from the design intent. Each of the identified deviants is then further investigated to ascertain possible causes and effects. The identification of deviations is prompted by a set of guidewords/guidephrases, where each related guideword/guidephrase is applied to each attribute. A pipe in a chemical plant might have fluid flow as an attribute. The guideword "NO/NONE" prompts us to consider what would happen if no fluid flowed (for whatever reason). Similarly, for a petrol tank, with attribute content "petrol", the guide phrase "OTHER THAN" prompts us to consider what would happen if the wrong kind of petrol were present (for example, use of leaded petrol in an engine designed for unleaded fuel might wreck that engine), or fluid other than petrol were used.

The technique is flexible. For example, it is possible to apply it to manual or automated procedures. A replacement procedure for an aircraft windscreen might have a step requiring the engineer to fasten the screen with identified bolts. What would happen if the wrong *kind* ("OTHER THAN") of bolt were used, or even if the whole step were omitted ("NONE") for a particular bolt? An aircraft engine maintenance procedure might require the level of oil to be checked in an engine with oil caps being securely replaced after the checks. What would happen if such caps were not replaced (use of 'NOT')? [1]

[1] Accidents have actually arisen with such causes. An aeroplane landed with the pilot unconscious having been half sucked out of the cockpit at 17000 feet having lost the windscreen after replacement.

The generic set of guidewords of HAZOP, as incorporated in UK Defence Standard OO-58 [68], is provided in Table 1.

Guidewords	Meaning
NO or NOT or NONE	None of the design intent is achieved
MORE	Quantitative increase in a parameter
LESS	Quantitative decrease in a parameter
AS WELL AS or MORE THAN	An additional activity occurs
PART OF	Only some of the design intent is achieved
REVERSE	Logical opposite of the design intention occurs
OTHER THAN	Complete substitution. Another activity takes place.

Table 1: HAZOP guide words [68].

A key part of HAZOP analysis is to interpret the guidewords for the context of interest. For example, Table 2 shows variations or interpretations of the guidewords for application to the timing of events or actions.

Guidewords	Meaning	Interpretation
Timing of Event or Action	**NO**	Event/action never takes place.
	EARLY	Event/action takes place before it is expected.
	LATE	Event/action takes place after it is expected.
	BEFORE	Happens before another event or action that is expected to precede it.
	AFTER	Happens after another event or action that is expected to come after it.

Table 2: Example guide word interpretations for timing [68].

HAZOP has gained wide acceptance: 00-58 [68] recommends HAZOP for hazard analysis in any safe system development; it has been successfully applied to the safety analysis and operation of chemical plants, aircraft, and many other products.

HAZOP has also been applied to the security domain. For example, Foster [31] has applied it to the generation and analysis of security protocols requirements. She stated that protocol development is relatively unstructured and as a consequence can produce protocols vulnerable to attack. The technique prompts the analyst to consider unknown threats and attacks on protocols.

---

84 out of 90 bolts were of smaller than required diameter. (BAC One Eleven over Oxfordshire 1990) [43]. In another incident, a pilot managed an emergency landing after oil drained out of all engines; one engineer had maintained all four aircraft engines and omitted to replace the oil caps.

Winther et. al. [69] has adapted HAZOP guidewords to generate new guidelines specific to security attributes. However, the derived guidewords are too restricted, and are not flexible enough to bring out most of the analysts' creativity. We have indicated a general approach to applying HAZOPs for security. Specialising the technique for a particular domain allows increased rigour and indeed efficiency.

3. Use cases

The use case technique, now popularised as part of the UML [61], was proposed by Jacobson [38] to describe behaviours of a system from a user's perspective. Typically, a use case characterises interactions between an actor (usually a user, but possibly another system) and a system. There is a visual representation, the use case diagram, and a tabular format. Figure 1 depicts the fundamental elements of the use case in diagrammatic form. The actor (a stick-man) interacts (an association line) with a use case (an ellipse).

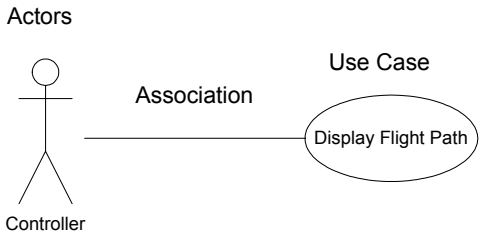

Figure 1: Fundamental use case elements.

Actors represent external users or other systems that have an interaction or association with the system. An actor characterises the role that users or other external systems may have in relation to the system.

- Actors are *everything that needs to exchange information with the system* [38]. Human and non-human interaction with the system can be expressed, and actors can represent individual or collective roles.

- Association lines connect an actor with the use case. This represents an interaction (communication) or association between the actor and that use case. The interaction may represent the exchange of information between the actor and the system or the invocation of the system's operation by the actor and vice versa.

- Use cases represent the system's behaviour from the external perspective, the task that the system must achieve on behalf of the actor(s) with which it interacts. Different authors document use cases in different ways. In general, the description comprises pre-condition, post-condition and sequences of actions (scenarios) including variants. The pre-condition defines the states that the system must be in for the use case to execute. The post-condition defines the state properties required after executing the use case.

The tabular form of a use case is a text description of how an interaction is accomplished. The different outcomes of a use case can be expressed using scenarios (action steps). There is little agreement as to what should be included in the tables.

In UML (and the related methods), use cases are almost always used to capture and express functional and late requirements. They represent the system and its

interactions, in a simple form that is easily understood by the different parties involved. In the Unified Process [45], the expression of use cases is the foundation for subsequent development activities, as the captured requirements are fed into the specification and design process.

Recent work has applied use case modelling to requirements analyses other than the purely functional. Abuse and misuse cases are proposed as (intuitive) means to capture and analyse security requirements [53][54][63]. Sindre [63] notes that both security and safety requirements can be elicited from use case diagram and scenarios. Douglass [26] shows that use case modelling can be used to document some non-functional requirements, for example, by annotating each action in use case scenarios with its timing constraints. Work by Allenby et. al. [1] on the elicitation and analysis of safety requirements also uses HAZOP on use case scenarios. The analysis results in a tabular form describing the likely catastrophic failures, their causes and effects.

Use cases, like abuse or mis-use cases, are potentially useful in the analysis of security requirements. The system interfaces presented to an attacker may be characterised by use cases. The attacker may 'stay within the rules' (but do so in a particularly clever way), or may choose to deviate from what is intended as acceptable behaviour. Attacks may often be regarded as 'exceptional flows' in a use case. Existing abuse case approaches do not give any systematic way of generating such unusual deviations. In this report we show how HAZOPs can be used to generate alternative or exceptional behaviours in a systematic way.

We regard our approach as a useful tool. Since not all system requirements will be recorded as use cases, other analyses have to be carried out too. Though HAZOPs may prove useful in those contexts too, in this report we restrict ourselves only to the analysis of use cases.

4. Method procedures and guidelines

Our approach starts from system use cases. These might be taken from the initial phase of the system development. In normal usage, the use cases would model only high-level functional requirements. The aim of our approach is to systematically analyse the system to identify potential threats and security requirements.

Our technique applies the HAZOP guidewords, with suitable interpretations, to elements of the use case. The elements incorporated are those that could be subject to deviations leading to meaningful results.

4.1 The analysis process

The essential steps of the analysis process are as follows.

1. From a use case, identify and record:
    a. intent and capability of actors;
    b. associations between actors and use cases;
    c. components from use case description: pre-condition, main flow of events, alternative flows of events (i.e. normal but perhaps less likely and exceptional /error flow of events) and post-condition.

2. Apply HAZOP guidewords with the appropriate interpretation to each element identified in step 1, to suggest deviations.

3. Evaluate whether the identified deviations violate, or could violate, any stated security properties. Investigate possible causes of the deviations.

4. Identify consequences that may arise from the deviations (extract affected assets from the identified consequences.)

5. Categorise the deviations identified, and generalise each group.

6. Provide recommendations or requirements on the identified problems/threats.

4.2 Use case elements and their security deviations

A use case describes a sequence of events, typically expressed by scenarios. Actions are expressed under various conditions, responding to a request from one of the actors that has an association with it. We identify elements that are subject to deviations for each of the main elements of a use case.

Actors

Each actor possesses different characteristics. Actor roles can be distinguished by their *intent* and *capability*. The characteristics determine the deviations that are possible for each actor; the ways in which actors deviate from their normal role may have different impacts on the system. Deviations from the actor's intent or actions

(deviations from goals), whether intentional or accidental, can reveal new threats to the system. Sometimes, new malicious actors are also identified by considering intentional deviation from expected actor behaviour.

The deviation of actors needs to take notice of the different potential resources and skills of individuals (or systems) playing the roles represented as actors. For example, a cyber-terrorist network usually has more ability to cause attacks than an individual teenage hacker, in terms of access to resources and skills. In gambling or gaming environments, actors may have differing local computational capabilities. Similarly, most actors will not have supercomputing facilities, but in some cases this may make a difference. Actors may differ in their ability to screen incoming information (e.g. for email viruses). Actors may also differ in the protocols they are able to support etc. A network administrator is able to access many more files or programs than a normal user. For human actors, one person may play different roles in the system and the roles a person may possess can be limited or restricted. For example, the same person is not allowed to both process and approve a payment.

The *intent* of an actor is an interesting concept. An actor may obey all security policy rules and participate in a seemingly innocuous interaction. However, it may be that the real intent is to signal information via a covert channel.

Associations

An association of the actor with a use case models the external interactions with the system through the functionality modelled in the use case. The significance of restricting access to an operation to a particular group of users (an actor), and of assignment of access controls to a particular user or actor could be highlighted by this part of the analysis. In secure systems, access to functionality depends on actor roles. We need to have a clear idea of which actors should be able to access which use cases for which purposes. Such control may be enforced in several ways. For example, access controls may form an explicit part of the use case (i.e. suitable authorisation checks are made by the system), or else physical access to particular terminals may be restricted to authorised personnel. (Such alternative refinements of system goals are addressed in [56]).

Availability is now a security goal for many systems. HAZOP application to associations such as NO association might reveal potential causes such as simple equipment malfunctioning (e.g. a keyboard refusing to acknowledge particular key presses, or else network failure). Inappropriate configuration may also result in associations not being effected.

Unintended associations are a particular problem. Use cases record only intended associations. An association represents an actor's ability to exchange matter or information across some interface. Intentional interactions use identified associations. It is intended that the channels used by the actors are the only relevant ones. However, physics often implements unintended associations. A communications cable may provide a point-to-point channel, but may also emit electromagnetic radiation that can be picked up by a suitably equipped eavesdropper. In a similar vein, an electromagnetic 'pulse gun' may seriously harm or even destroy equipment from short range. Electromagnetic interference may also be a problem without any deliberate action. Covert channels can be considered as unintended associations. More

generally, consideration of unintended associations points to a need for issues of zonal analysis to be considered [64]. However, at this point we indicate a general opportunity to consider such matters at the use case level (even if only briefly, or to highlight the need for particular analyses later in the development).

Several associations may be current for an actor; the actor may indeed engage in parallel instance of the same use case. In on-line gambling, this may allow for instances of collusion. It is possible also that uses cases executed in quick succession may cause problems (e.g. in a distributed cash system it may be possible to carry out a second withdrawal before a central database has processed a successful confirmation of the first, leading to withdrawal limits being exceeded).

Use Cases

The use case pre- and post-condition both represent states of the system at certain sets of times. Deviations from these normally-reached states causes deviant interactions and could result in exceptions to the flow of events. We can address the variations from the normal behaviour, by considering the deviations of each step in the use case and investigating the causes.

4.3 HAZOP security guidewords

Tables 3 to 5 show the interpretations of HAZOP guidewords that we propose for application to the use case elements and their features. In interpreting the guide words some measure of *scoping* will be needed, for example "OTHER THAN" applied to an action or result could cover a huge number of possibilities. We leave such pragmatic decisions scooping to the analyst.

An actor has an action, representing an intent, and a capability, representing the skills and resources at their disposal.

Element: Actors

Features	Guide words	Interpretations
Action (Intent)	**NO**	The action/intent does not take place.
	MORE	More action is achieved. This may be one of the following: *Sequential Repeat* – the same actions take place repeatedly. *Parallel Repeat* - – the same actions take place concurrently. Extreme Intent – some scalar attribute of the action is affected (e.g. extreme parameter values are used in service invocations).
	LESS	Less action is achieved than intended. The intended actions are incomplete or insufficient.
	AS WELL AS	*As* well as the intended or normal action, some unexpected supplementary or contradictory actions occur or are intended.
	OTHER THAN	The action achieves an incorrect result. Alternatively, the actor may use facilities for purposes other than those intended, i.e. abuse of privilege (some OTHER THAN actions may lead to exceptional scenarios).
Capability	**NO**	Lack of the capability to perform the action.
	MORE	More general capability, allowing more to be achieved than intended.
	LESS	Less general capability, allowing less to be achieved than is required.
	PART OF	The actor has only part of, or is missing a specific part of the capability.
	AS WELL AS	As well as the specific capabilities required, the actor has other specific capabilities.

Table 3. The interpretations of guide words for an actor.

An association is not considered to have additional features. However, more guide words are appropriate:

Element: Association

Features	Guide words	Interpretations
Associations	**NO**	Association does not/can not take place.
	MORE	*Superfluous* – Interface permits greater functionality to a particular actor than is required. More functionality is available. Association is not constrained as required. Further divisions are: *In-parallel with* – More functionality is provided/occurs simultaneously with the permitted ones. *In- sequence with* – More functionality provided/occurs before or after the permitted ones.
	LESS	Interface permits less functionality to a particular actor than is required. Association is over-constrained.
	AS WELL AS	Associations to a particular use case take place with other actors as well.
	REVERSE	Interaction takes place in the reverse direction.
	OTHER THAN	Wrong association is defined. *Swapping roles* – Swapping of associations between actors or individuals.

Table 4. The interpretations of guide words for an association.

Finally, for the use case itself, the temporal guide words are included:

Element: Use Case Elements

Features	Guide words	Interpretations
State (defined in a pre-condition or a post- condition)	**NO**	The state or condition does not take place or is not detected.
	AS WELL AS	Additional conditions apply. This may mean that more stringent checks are made, or else a more restrictive state results than is strictly required. Errors of commission might be considered here.
	PART OF	Only a subset of the required conditions applies. This might for example, result from incomplete checks (e.g. access control checks or integrity checks), by incomplete implementation (a program doesn't do all it should), or because the consequences of system behaviour are not fully understood.
	OTHER THAN	An incorrect condition applies. Perhaps the wrong data is used.
Action	**NO**	No action takes place.
	MORE	More action is achieved. A stronger post-condition is achieved. More actions could be carried out: *Repeat* – the same actions take place repeatedly. *Superfluous* – the system does different additional actions to those intended or required. A Trojan horse usually implements more functionality than is apparent, for example. Additionally, an action may take place for longer than required.
	LESS	Less is achieved than intended. A weaker post-condition could result. For example, an action is *incomplete*, or an action takes place for a shorter time than required, or an action stopped earlier than expected.
	OTHER THAN	An incorrect action takes place. An action not intended or required takes place instead. For example, *wrong* detail is provided or a *wrong* button is pressed.
Sequence of Actions (scenarios)	**LESS**	Less is achieved than intended. For example, *Drop* – Miss one or more parts of action. Additionally, a sequence of action takes place for a shorter time than required.
	AS WELL AS	The sequence does the intended actions plus others.
	REVERSE	The sequence of actions takes place in reverse order (and other out-of-order concepts).
	EARLY	The action sequence or its components takes place before it is expected (timing).
	LATE	The action sequence or its components takes place after it is expected (timing).
	BEFORE	The action sequence or its components happens before another action that is expected to precede it.
	AFTER	The action sequence or its components happens after another action that is expected to come after it.
	OTHER THAN	An incorrect action sequence takes place.

Table 5. The interpretations of the guidewords for a use case.

5. Deriving use case deviants

The guidewords alone do not give a clear idea of how to derive deviants from use cases. This section provides additional guidance for performing the use-case-based security analysis, based on experience of applying the technique. Note that, in applying the guidewords, much repetition is expected. Analysts must ensure that any apparent repetition is indeed repetition and not a subtle variation of a previously-identified threat.

The derivation of deviants must consider at least the three fundamental security properties (i.e. confidentiality, integrity and availability). The emphasis will vary between applications. The guidance also includes high-level patterns of analysis.

Viewpoint considerations/Stakeholders' interests

Stakeholders are individuals or organisations that have an interest in the system, even though they are not a direct part of the system, and may not directly interact with the system. One (partial) definition of a stakeholder is an authority (not a hacker!) that has an authorised ability to cause the system to cease to exist or cease to operate. Each stakeholder in a system has a viewpoint, representing his or her interest in the system. Stakeholders' interests might be in conflict.

Security analysis must take account of different viewpoints, and of the seniority or importance of the stakeholders. For example, some stakeholders are concerned about the integrity of classified information; others may find confidentiality of personal data more important. Deviations from all stakeholder interests should be considered.

Role/actor mapping

Deviation analysis must consider the possibility of unexpected interactions through shared and multiple roles. A computer system is built to support the roles of humans or of machines. An actor characterises the role that the users or other external systems play in relation to the computer system. Individuals within roles are not distinguished, and an individual may play different roles at different times.

A multi-user role always has potential deviations in which the actual user within a role initiating some interaction is not the same as the actual user receiving a response.

The use case definition of actors and roles may be confused, allowing individuals to operate outside their intended or authorised roles.

Business Rules

A business rule defines a business aspects or constraint on a system. It consists of policy, some constraints and a validation audit. The business rule influences the way that the interactions with the system are specified.

Business rules should include statements of the required security policy (e.g. auditing, integrity or confidentiality of data). However, in practice, security policies are at best implicit. The use case analysis helps to make explicit the security policy, and the system requirements must then be modified so that the functional requirements respect these policies.

When deriving the deviations, it is essential to consider what sort of business rules the system does or should enforce.

Well-known sources of security violations

Although the systematic HAZOP approach is a significant improvement on earlier intuitive analyses, it is important not to ignore or neglect the "checklists" of accumulated wisdom in security. Three common sources of security violation are timing, HCI, and presentation.

- Timing information is not always explicit in use case diagrams and descriptions. However, whenever timing may be associated with events or actions, timing deviations must be taken into account. Typically, timing issues include response time (time between input to output) and repetition time (time between successive updates or outputs). Deviations from timing are events that occur later, earlier, sooner or longer than expected. In terms of security, this may leak information or encourage odd or unfortunate user actions.

- Poor HCI design, for example implicit system response or lack of acknowledgement message, may be misleading to users of a system. Security problems may also arise if the HCI functions overload the system, or if the users cannot understand how to use the system as intended. Use case analysis occurs too early in development to include a detailed study of the HCI and its security implications. However, the consideration of known HCI pitfalls may help to identify security "anti-requirement": things that the system must not do if it is to be acceptably secure.

- Presentation issues include the way in which data is presented to users by a system. For example, intent and actions may be influenced by the order or format of data listed on the screen. Essentially, presentation is part of the conversion of data into information; varying the presentation changes the information that a user can extract. The most obvious security pitfall is perhaps inference. However, there are many more subtle presentational effects to be considered when deriving deviations.

6. Example application

This section presents a simple case study. The use cases describing an apparently-innocuous web-ordering system are analysed to systematically reveal threats.

6.1 System description

A website hosts a company's product catalogue that anyone using the internet can access and browse. A customer who are registered with the company can order goods via the website. To order goods, the registered customer must provide sufficient payment and delivery detail. If customers who are not registered but wish to purchase, the ordering facility provides an initial registration procedure.

Orders are processed by an operator, who logs on to the host system. The operator may change his/her login password when required.

The use case diagram is shown in Figure 2.

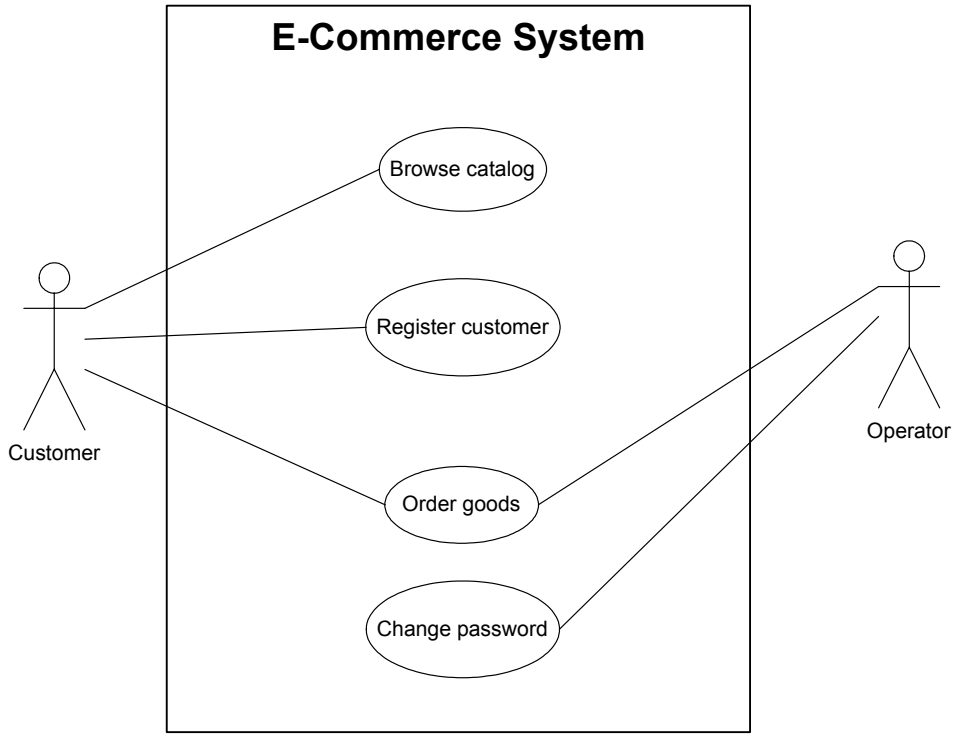

Figure 2: Simple e-commerce system use case diagram.

Table 6 lists the stakeholders in the web ordering system.

Stakeholders-Interests list

Stakeholders	Interest
Company owner	Profits and reputation of the company
System manager	System's performance and operation of staff.
Developer	Correct operation of the program.

Table 6: Stakeholders-Interest list.

Table 7 summarises the intents of the use case actors.

Actor-Goal list

Actor	Goal
Customer	Browse catalogue
	Register
	Order goods
Operator	Process order goods

Table 7: Actor-Goal list.

Use Case Descriptions

The following descriptions elaborate each use case in Figure 2.

Use case name:	Browse Catalogue
Goal:	To explore the lists of goods available from the system.
Actor(s):	Customer
Preconditions:	The customer has access to the internet.
Main flow of events:	1. The customer enters the e-commerce website. 2. The customer selects the Browse Catalogue section. 3. The system displays lists of products to the customer. 4. The customer browses the catalogue for a particular product. 5. The customer finds the product.
Alternate flows:	User cannot find product he/she wanted. Use case ends.
Post conditions:	The product is found.

Use case name:	Register customer
Goal:	To register a customer identity with the system.
Actor(s):	Customer
Preconditions:	The customer has access to the website. The customer has not registered before.
Main flow of events:	1. The customer enters the Register Customer section. 2. The system displays the new customer registration form. 3. The customer provides registration details. 4. The customer submits the registration form. 5. The system updates its registration data information.
Alternate flows:	The customer has already registered. Use case ends.
Post conditions:	The customer is registered and the details are saved to a database.

Use case name:	Order goods
Goal:	To order goods from the system.
Actor(s):	Customer Operator
Preconditions:	The customer is registered to order goods. The customer has entered registration details e.g. user name and password (the customer is logged on to the ordering section).
Main flow of events:	1. The customer enters Order Goods section. 2. The system displays the customer's account detail. 3. For each product that the customer wishes to order, the customer enters its identity. 4. The customer provides delivery details. 5. The system calculates and displays the price of the goods ordered. 6. The customer submits payment details. 7. The system confirms the result of transaction. 8. The operator collects the detail of the order. 9. The operator processes the order.
Alternate flows:	
Post conditions:	The order and its detail are entered on the system and the order is processed.

Use case name:	Change Password
Goal:	Change the password for the login
Actor(s):	Operator
Preconditions:	The operator is logged in.
Main flow of events:	1. The operator enters his/her current password 2. The system validates the password. 3. The operator enters a new password, twice, as prompted by the system. 4. The operator confirms the change. 5. The system saves the new password.
Alternate flows:	A1. If the operator's old password is incorrect, an error message should be displayed and the password should not be changed. A2. If the two entries of the new passwords do not match, the operator is prompted to re-enter them.
Post conditions:	The password of the operator is changed and updated in the database.

6.2 Analysis

The use case descriptions document functional requirements for the e-commerce system, capturing the interactions between the actors and the system. From the use cases and their descriptions, we can identify the actors' intents (these can be taken from the actor-goal list), actors' capabilities, associations of customer and operator with the use cases, the preconditions and post conditions of the use case, and a sequence of events of order goods. Each of the elements and features is subject to deviation. Table 8 to Table 11 show the analysis results for the system.

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
Use Case: Browse Catalogue					
Actor					
Customer's Action – *Browse Catalogue*	Not Relevant				
Customer's Capability – *Browse Catalogue*	**NO** – Customer does not have the ability to browse the catalogue.	• Physical – lack of computer system, lack of access to internet • Knowledge - does not know the website, no knowledge of using an internet system.	The system misses opportunity for one potential customer.	• Steal computer set • Block internet account • Provide wrong info about the website. • Modify website URL	
	MORE – Customer has more ability than required.	• Access hidden links.	Access of unwanted/ confidential information.		Any confidential data must not be made available for public access.
	LESS – Customer does not have much ability.	• Physical – lack of appropriate computer system, lack of access to internet • Knowledge - does not know the website, no knowledge of using an internet system.	The system misses opportunity for one potential customer.	• Steal computer set • Block internet access	Timeouts may come into force.
Associations					
Association - *Customer* and *Browse Catalogue*	**NO** – Customer does not have association with Browse Catalogue use case.	Same as NO in Customer's Capability			

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	OTHER THAN – Other people can browse the catalogue. Not of security relevance, any person is legitimate to browse the catalogue.				
Use Case Elements					
Pre-condition - *The customer has access to the internet*	**NO** - The customer cannot access to the internet.	• Loss of network and other physical equipment. • Internet account expires. • ISP not available • Wrong configuration of network.	• Customer's frustration • The system misses opportunity for one potential customer.	Denial-of-service	Strictly, this is a problem for the customer. Can make server side as flexible as possible.
Step 1- *The customer enters the e-commerce website.*	**NO** – The customer cannot enter the system website.	• The e-commerce server is down (or malfunctioning in some way). • Loss of network. • The system is blocked/overloaded • Local error	• Customer disappointment. • Customer may get bored and may not want to access the site again.	• Block system. • Interception of all info passed • Denial-of-service attack	
	MORE – applied to customer. More customers enter the web site.	• Possibly legitimate access by many interested users. • Massive simulated requests to enter the site by malicious processes.	• Service may be severely impeded. User browsing sessions may be very slow or else not accepted.	• Inadvertent or deliberate overloading of system (denial of service attack)	This is tricky. We could make *browsing* subject to log on (unusual). We could monitor requests for signs of actual user behaviour (rather than program based requests of a malicious agent)

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	OTHER THAN – the customer attempts to enter the indicated web site but is directed to another (possibly spoof of the original)	• Malicious handling at local host. • Domain squatting on addresses along the path. • Use of similar names for company, leading to customers being misled (e.g. via search engine returns).	• Goes to wrong site generally. Customer is simply engaging in a spoofed or plainly wrong interaction with obvious consequences. • Spoofed sites have obvious problems. Reputation of company may (wrongly) be tarnished.	Manipulation of host or routing	Be on lookout for attempted infringements of name. Search for links with similar names to that of the company.
Step 2- *The customer enters to Browse Catalogue section.*	**NO** – The customer cannot enter to the section.	• Link is not available • Firewall • Link is available but there has been denial of service	• Customer's disappointment • Customer may get bored and may not want to access the site again.	Fabrication of website	
	OTHER THAN – The customer enters a section other than Browse.	• Internal error on web site. • Communications are altered in transit to reflect different request (but to same web site).	Various – ranging from annoyance and bewilderment through to accessing sensitive information the customer shouldn't.	• Largely internal system inconsistency. • Manipulation of communications.	
Step 3- *The system displays lists of products to the customer.*	**NO** – The system was unable to display the products.	• The server is down (or part of system is down if distributed server). • The system is blocked. • Browser mismatches. • Error in database.	Customer's annoyance Plus, the company is not able to advertise any products.	• Block system • Virus/hacking	There may be good legal reasons why this is should be the case! Is there material which should have age limit imposed?

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	MORE – The system shows more than the lists of products than it is supposed to.	• Staff mistake. • Data (status e.g. sale/not yet on sale) of the associated product is incorrect. • Added product from the outsider (e.g. advertise on this website) • Search filters not working appropriately.	• Data release may have unfortunate effects: product not on sale is being displayed, and will become unavailable when the customer orders it, leading to customer dissatisfaction. Product displayed with old price and old detail, which the company needs to sell at the price it advertises. The organisation's reputation is effected if the product on sale is not suitable or illegal. Overloading with irrelevant information.	• Modification of message (communication channel). • Modification of website (file). • Modification of the storage files of product.	This really illustrates the importance of having accurate information on a web-site. There are all manner of commercial /legal reasons to do so. What exactly is meant by "is supposed to"? Are there legal constraints on what should be displayed? Illustrates need to enforce sales policy within the law. This has implications for the details we store with actual customers.

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	LESS – The system shows lists of products less than it is supposed to.	• Staff mistake. • Data (status e.g. sale/not yet on sale) of the associated product is incorrect. • Outsider's fabrication • Corruption of message passing.	• Supplier's frustration • Lose the opportunity to sell a particular product.	• Modification of message (communication channel). • Modification of website (file). • Modification of the storage files of product.	Begs the question as to how lists are requested. At this stage it is not stated whether this is simply by hyperlinks or else by search criteria.
	REVERSE – The system shows lists of product in a reverse or unusual order.	• Possible manipulation of communications but principal cause is simply deliberate action server side.	• A different (including reverse order) may influence selection to buy.	• Malicious server management/development .	Psychological issues must be taken into consideration. Who are the customers who have products here – are some being favoured? Can we make money openly from this? Note – some producers may actually pay to appear on a web site.
	OTHER THAN – The system shows something else or incorrect data associated with products.	• Outsider's fabrication • Replacement or redirect to other fake website to obtain customer's information. • Page not available (internal misconfiguration). • Internal server data/configuration errors.	• Customer's disappointment to see what is not expected. • Customer may be misled by erroneous data. The server management may be legally obliged to sell at whatever price was advertised for example.	• Fabrication of website • Man-in-the-middle attack • Incompetence interface design (threat). • Internal configuration management error (accidental or deliberate).	Need controlled access to commercially sensitive information on server database. Whole new set of use cases (requirements) needed.

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	AS WELL AS – details are supplied to customer but also to a third party.	• Comms monitoring • Profiling activities server side.	• Possible breach of privacy.	Passive monitoring on the net. Active monitoring by server (without knowledge of customer)	What is the policy on privacy? Most likely issue here is the storing of profile data. What are legal issues?
Step 4– *The customer browses the catalogue for a particular product.*	**NO** – N/A				
Step 5 – *The customer finds the product.*	**NO** – The customer cannot find the product he or she is looking for.	• The product is not available within the system (normal case). • Information of the product was tampered with. • Packet was corrupted/or manipulated. • Information sent is not complete • Mismatch search criteria	• Lose the opportunity to sell a particular product. • Customer dissatisfaction.	• Modification (corruption) of message (communication channel). • Modification of website (file). • Modification of the storage files of product.	Provide search results on similar searched words.
	OTHER THAN – The customer mistakes some other products for the product looking for.	• Unclear description of the product • Trick the customer in choosing something different. •	• Customer's frustration when receive product that was not intentionally ordered for. • The company must do more work if the product if returned and refund requested.	Distraction/manipulating the customer	Something of an HCI issue. Need to consider interface.

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
Post-condition – *The product/s is found.*	**NO** – The product is not found.	• The product is not available within the system (normal case). • Information of the product was tampered with. • Packet was corrupted/or manipulated. • Information sent is not complete • Mismatch search criteria	• Lose the opportunity to sell a particular product. • Customer dissatisfaction.	• Modification (corruption) of message (communication channel). • Modification of website (file). • Modification of the storage files of product.	

Table 8: Analysis of 'Browse Catalogue' use case.

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
Use Case: Register customer					
Actor					
Customer's action – *Register customer*	**NO** – Register action did not take place. Fail to register.	• Interception by an intruder • Loss of network/server is down • Customer does not complete the registration by accident. • Decline registration	• Customer losses trust in the company, if he has already submitted the detail but was not updated. • Need to register again.	Interception of message passed through network.	This can make the customer lose trust in the company's overall service and might not want to do more important service (e.g. purchasing the products) with the company.
	MORE – Register customer more than required.	• Repeated registrations (intentionally). • Duplicated or replayed of the message. • The system might not allow to change detail, so register again with different detail.	• Increase work load to the system, and may result in the unavailability of the service. • Block other communications with the system. • Take advantage of "one per customer" offers.	• Replay of message. • Repeated registration (Flood system).	Should allow customer to change detail after registration.
	LESS – N/A				
	OTHER THAN – Register someone other than oneself.	• Phantom user registration. • Customer provides wrong information detail • Customer registers on behalf of other person. • Customer uses other person's detail (steal info).	Wrong user information received, resulting in inaccurate data gathered (e.g. number of customer interested in the product)	Obtain other person's detail.	This raises the issues whether phantom users are of concern or not. Will it affect any offers or benefits?

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
Customer's capability – *Register customer*	**NO** – Customer does not have ability to register customer.	• Physical –loss connection. • Does not have registration information. • Confusion in what information to fill in.	Not be able to process any order from the web-site.		This is the problem of the customer to provide sufficient information. The system could prompt required fields to be filled in, leaving others as optional.
	MORE – Customer has more ability to register. (Register using different identities or addresses)	• Deviate some details such as names or addresses. • Swap places of surname and given name. • Different phonetic spellings of non-English names.	• The customer can take advantage of 'one-per-customer' offers or 'one-per-household' offers. • Profit loss for the company	Fraudulent multiple registrations.	There is a possible need to detect and deal with fraudulent multiple registration.
	LESS – Customer has less ability to register.	Duplicate of names in the database.	• Customer disappointment • The company loses one potential customer.		This is an issue to be further discussed on how to distinguish people having the same names (possibly using emails).
Association					
Association - *Customer* and *Register Customer*	**NO** – Association with the particular customer does not take place.	• Connection is blocked. •	• Customer disappointment • The company loses one potential customer.	Block connection Flood system	
	NO – Associations with the customers do not take place.	• Loss of network and other physical equipment. • Server is down	• Customer disappointment • The company loses potential customers.	Block connection Flood system Virus/hacking	
	MORE – N/A	N/A			
	LESS – N/A	N/A			

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	OTHER THAN – Association with others (e.g. customer/or to an operator/or to an intruder)	Normal operation (any body can register)			This raises issues on registration policy. Can a company worker be a potential customer?
Use Case Element					
Step 1 – *The customer enters to the Register Customer section.*	**NO** – The customer cannot enter to the Register Customer section.	• Link is not available • The web page is corrupted/removed.	• Customer disappointment. • Customer may get bored and may not want to access the site again.	Fabrication of website	
	MORE – The customer enters to the section more than once.	Keep refreshing or loading the same page.	Not a problem.		
	LESS – N/A				
	OTHER THAN – The customer enters to some other section rather than the Register Customer section.	• The page was replaced by an intruder. • Incorrect configuration/update (accidentally or maliciously)	• Customer's confusion • Company's reputation • Confidential details can be revealed out, if the customer fills in the form which thought to be authentic.	• Modification of website (file). • Modification of the storage files of product.	
Step 2 – *The system displays the new customer registration form.*	**NO** –The new customer registration form is not displayed.	• Server is down • Interception by intruder • Wrong configuration/ naming.	• Company loses opportunity for new potential customer. • Customer cannot register leading to customer dissatisfaction.	Interception	
	MORE – More information is displayed than intent on the form.	• Wrong form is displayed • Confidential data is displayed	• Customer's confusion • Confidential data can be seen by other people passing by the monitor display.	Fabrication of web-site.	Only sufficient information should be displayed.

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	LESS – Incomplete form is displayed.	• Interception • Wrong form is displayed	• Insufficient information received, may be useless or drop out from the registration resulting in customer's dissatisfaction. • Wrong information is kept in the system.	• Interception • Unauthorised modification of related files	Enough information should be displayed.
	OTHER THAN – Other form is displayed	• The page was replaced by an intruder. • Redirect to a phantom web page. • Mistake made by staff	• Customer's confusion • Company's reputation • Confidential details can be revealed out, if the customer fills in the form which thought to be authentic.	Unauthorised modification of files /names	
	OTHER THAN – a form is displayed by agent other than the system.	• Man in the middle attack. Or agent masquerading as the system.	• Minor problem if new info is offensive but real problem comes next.	Man in the middle attack.	
Step 3 – *The customer provides registration details.*	**NO** – The registration detail is not provided.	Customer inaction.	• Delayed registration • Failed registration		What happens if this is too late – a timeout?
	MORE – More registration detail is provided than required.	Not a problem.			
	LESS – Less registration detail is provided than required.	Customer is reluctant to provide real/enough information.	Insufficient information is registered.		Prevent by using mandatory fields (use system to check).

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	OTHER THAN – The customer provides some other detail.	Same as OTHER THAN in Customer's action	Wrong user information received, resulting in inaccurate data gathered (e.g. number of customer interested in the product)	Using inappropriate identify i.e. right identity, wrong detail or wrong identity but right/wrong detail.	Potentially major problem. What are the details? What onus is there to check accuracy by other means? Age, address etc. Issues of faked identity? Legal issues (data protection Act etc.)
Step 4 – *The customer submits Registration details.*	**NO** – The customer does not submit registration details.	• Customer inaction/ refuses/changes his mind. • Computer hangs	No new registration detail is stored.	Distract or trick customer	Minor
	MORE – The customer submits more than once.	• Customer mistakenly presses twice (over-eager customer) • Duplicate on network • Replay of message.	• The same customer detail is updated twice. • The system ignores the duplicate information.	Replay of message	Issue here is really one of change. What is the policy on changing details once provided? This needs to be authenticated in some way otherwise A can have his/her details removed by B. This is connected with above. There is the possibility of user submitting multiple but different details.
	LESS – N/A				

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	OTHER THAN – The customer clicks on another button.	• Mistake other buttons for the 'Register Button'	• Loss of information entered. • Exposure of information to others.	Distract or trick customer.	Minor.
	OTHER THAN – The customer submits detail to someone else.	• Interception by intruder	• Disclosure of information	Eavesdropping Interception	
	AS WELL AS – The detail is sent while something else occurs	• Internet gets disconnected. • Interruption by an intruder • Customer stops and wants to cancel • Extra event is initiated by customer or an intruder • Information may be stored in local buffer or else be retrievable by some means. • Information is simply sent to an intruder	• Customer is not sure whether the detail is sent through or not. • Customer sends the information again if he/she still wants to carry out resulting in duplication of records. • Intruder may get details	• Distract or trick customer.	A page displaying confirmation that the information is received would lessen worries from the customers.
Step 5 - *The system updates the information.*	**NO** – The information is not updated.	• Connection loss before sending information • Interception • Error in server side (programming error, physical loss, or from malicious actions)	• Customer sends the information again. • Customer is not sure whether the detail is sent through or not.	Interception	
	MORE – The information is updated more than once.	• Error in server side • Duplicated or replayed of the information	The same customer detail is updated twice.	• Replay of information • Unauthorised modification of program on server side.	

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	LESS – N/A				
	OTHER THAN – The system updates wrong information.	• Message is corrupted by an intruder. • Error in the update program.	The system obtains wrong information	• Corrupt of message • Unauthorised modification of program on server side.	
Post-conditions – *The customer is registered and the details saved to a database.*	**NO** – Customer is not registered to database.	• Physical problems • Software problems (e.g. database) • Duplicate account registered • Incomplete information provided. • Message corruption	• Customer's frustration. • The company lose opportunity to sell.	Maliciously attack physical or software components.	
	MORE – Customer is registered to database more than once.	• Update wrong by the program • Duplicate message received	The same customer detail is updated twice.	• Unauthorised modification of program on server side. • Replay of message	Is there any policy against duplicate customers?
	OTHER THAN - Incomplete information is registered to database.	• Update wrong by the program • Wrong/incomplete information is received.	The system receives wrong information	• Unauthorised modification of program on server side. • Modification of message via network.	

Table 9: Analysis of 'Register Customer' use case.

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
Use Case: Order goods					
Actor					
Customer's action – *Order goods*	**NO** – The customer does not order goods.	• Unsatisfied with the product selections/price • Complicated interface • Distraction by the company's opponent advertisement	• Lose an opportunity to sell • Customer's disappointment	Hacking	Provision of a user-friendly interface design. Provision of security mechanism in preventing unauthorised pop-up windows.
	MORE – The customer excessively order goods	• Not the owner of the account/or card he's using • Prank order with fake account.	• The company sent order without getting paid later on. • The owner of the payment becomes furious when finds out that the order was not initiated by him. • Time wasted when verify payment detail with card company and turn out to be fake. • Increase workload to the company staff	• Obtain password and card detail (steal, bribery, social engineering). • Get access to other computer, while logging on to the system.	This raises the issues of whether the payment should be received before delivery or not.
	AS WELL AS – The customer provides insufficient/incorrect detail when submitting order.	• Complicated interface • Detail not available • Typing mistake • Distraction	• Processing of payment would be unsuccessful. • Customer's disappointment • Increase workload to the company staff.	• Steal/virus • Distraction	Checks on mandatory field prior to accepting the request.

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	OTHER THAN – The customer intends to achieve something else rather than to order goods.	Malicious intents	Wrong statistics data on the number of customer access Disruption on the service Increase workload	Flood/block the system Impersonating as customer	This raises the issues of whether to allow users to simultaneously log on using the same accounts.
Customer's capability – *Order goods*	**NO** – The customer has no ability to order goods.	• Physical – loss connection. • Does not have account/ payment information • The customer has not been registered (registration detail not available)	Company misses the opportunity for one potential customer.	• Disruption of connections • Steal card/payment info.	This raises the issues of what method of payment should there be available for the customers.
Operator's action – *Process the order*	**NO** – The operator does not process the order.	• Lack of responsibility in work • Order does not come through to the operator. • Order is lost/or has been modified.	• Customer waits definitely for the goods order. • Customer's disappointment • Financial loss to the company	• Unauthorised modification of the message via network • Unauthorised deletion of order received.	Should inform the customer on the approximate delivery date.
	MORE – The operator processes the order more than require.	• Operator error (accidentally e.g. operator's misinterpretation) • Software error • Corrupt of the message (by an intruder or network) • Greedy operator, pretends that he/she has received the order more than it was sent.	• Financial loss e.g. the system sent out two orders but only get paid for one • Customer needs to pay twice if the both payment transactions are processed.	• Corrupt the message. • Intentionally make mistake by the operator.	
	LESS – N/A				

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	OTHER THAN – The operator processes order with wrong information.	• Operator error • Modification of information in the database • Greedy operator, pretends that he/she has receives wrong information • Corrupt of the message via network or an intruder.	• The system sends out wrong order, leading to customer's disappointment. • The customer needs to pay more than actually needs to.	• Corrupt the message. • Intentionally make mistake by the operator.	
	OTHER THAN – The operator process order when not received.	• Fabrication of order	An incorrect order is sent out.	Fabrication of order.	
Operator's capability – *Process the order*	**NO** – The operator has no ability to process the order.	• No privilege to do so. • The supplier connection is not available.	• Slow down the process. • Operator's frustration	Modification of the user's privilege.	Monitoring system should be useful for this purpose.
	MORE – The operator has ability to do more than processing the order (e.g. modify order).	• Inaccurate/or no privileged is set. • Database is not protected. • Error in the program	Operator is able to modify the order leading to financial loss and customer's frustration.	Unauthorised modification in database and software code.	Assign appropriate privilege
	LESS – N/A				
	OTHER THAN – N/A				
	AS WELL AS – The operator has ability to some other process involving ordering of goods.	• Responsibility/role is assigned incorrectly • Privilege assignment is incorrect.	Operator is able to modify the order leading to financial loss and customer's frustration.	Unauthorised modification in database and software code.	Assign appropriate privilege
Association					
Association – *Order goods* and *Operator*	NO – Order hasn't passed to operator when expected.	• Loss of network • Interception	• Customer waits indefinitely for the goods ordered. • Other competitor may have better offer and offer to the customer, if the information is revealed.	Interception	

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	MORE –More than one operator process the same order.	• Operator error • Software error • Greedy company, pretends to have receive more order	• Customer receives more than one order but pays for only one, resulting to financial loss of the company. • Customer needs to pay for more than one order, if the payment transactions are processed.	Unauthorised modification	Need to protect customer.
	LESS –N/A				
	OTHER THAN – Order goods is associated to someone other than the operator.	• Hacking to the system • Intruder has access to the system (computer system or documentation) • Fake operator, impersonating real operator • Order goods is sent to wrong place (across network)	Disclosure of information (e.g. account information, goods ordered info, payment detail, credit card number and etc.)	• Hacking • Impersonating	
	AS WELL AS - Order goods is associated to someone as well	• Interception • Man in the middle attack. • Operator reveals the detail (accidentally and maliciously)	Same as above. But the operator does not have knowledge that something has happened. Seems to still be normal.	• Interception • Man in the middle attack. • Malicious insider	
Association – *Order goods and Customers*	**NO** – Association with the customer does not take place.	Connection is blocked	Customer's disappointment	Block connection to the system.	
Use Case Elements					
Step 1 - *The customer enters Order Goods section.*	**NO** – The customer cannot enter Order Goods section.	• Link is not available • The web page for order good section is removed.	• Customer disappointment. Customer may get bored and may not want to access the site again.	Fabrication of website.	

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	MORE - The customer enters to the section more than once.	Keep refreshing and loading the same page.	Not a problem.		
	LESS – N/A				
	OTHER THAN - The customer enters to some other section rather than the Order Goods section.	• The page was replaced by an intruder • Incorrect configuration/update (accidentally/maliciously)	• Customer's confusion • Company's reputation	• Modification of website (file). • Modification of the storage files of product.	
Step 2 - *The system displays customer's account detail.*	**NO** – The customer's account detail is not displayed.	• Account does not match • Wrong data is sent to retrieve info • Interception • Message corrupt	• Customer will not be able to order goods leading to customer frustration or attempt to try to re-register again. • Disclosure of customer's account detail.	• Interception • Tap communication. • Corrupt the message.	
	MORE – The system displays customer's account detail more than necessary (e.g. card detail/security info).	• Software error • Programming mistake	Disclosure of customer's account detail.	Unauthorised modification of software	Only sufficient information is needed to display.
	LESS – Essential detail is not shown.	• Software error • Programming mistake • Interception	• Disclosure of customer's account detail. • Customer's confusion	Unauthorised modification of software	Enough information is needed to display.
	OTHER THAN – The account detail displayed is inaccurate.	• Modification of account detail • Interpretation of the account information is wrong. (Software fault)	Customer confusion to whether the detail is correct or not.	Same as above	
	OTHER THAN – The system displays other customer's account details.	• Retrieval of the wrong detail (software error) • Wrong data is sent to retrieve info.	Disclosure of other customer's account detail.	Same as above	

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
Step 3 - *For each item that the customer wishes to order, the customer enters its detail.*	**NO** –The customer does not enter the goods detail.	• Customer inaction	N/A		
	MORE – The goods entered are more than intended.	• Incorrect update of the goods to the list of goods selected. • Customer clicks on the 'submit' or 'add' button more than once.	Customer's frustration when get paid for goods not wanted.	Unauthorised modification.	The system should provide the list of the selected goods for the customer to view at any time.
	LESS – The goods entered are fewer than intended.	• Incorrect update of the goods to the list of goods selected. • Customer does not click on the 'submit' or 'add' button.	Customer's frustration when receive fewer order	• Unauthorised modification. • Interception	Same as above
	OTHER THAN – The customer enter wrong goods detail.	Unclear product description	Customer's frustration when receive wrong order	Unauthorised modification.	Same as above.
Step 4 - *The customer provides delivery details.*	**NO** – The delivery detail is not provided.	N/A			
	MORE – N/A				
	LESS – Insufficient delivery detail is provided.	• Customer provides insufficient detail • Interception • Message is corrupted.	• The order cannot be delivered. • Goods are sent to wrong address.	• Interception • Corrupt the message	Mandatory field check.
	OTHER THAN – The customer provides incorrect delivery detail.	Trick customer to make mistake.	Same as above	Same as above	Delivery confirmation page.

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	OTHER THAN – Delivery detail is provided by other person.	• Intruder enters his/her own delivery detail instead of the customer • Modification of the message by the intruder	Goods are sent to another address, while the payment is being deducted from the owner's account, resulting to customer dissatisfaction	Unauthorised modification of the message	
Step 5 - *The system calculates and displays the price of the goods ordered.*	**NO** – The system does not or fails to calculate/display the amount due.	Error in the program Interception	Customer does not know how much is due and may be reluctant to submit any payment details.	• Unauthorised modification of the program • Interception	
Step 6 - *The customer submits payment detail.*	**NO** – The customer submits payment detail to the system unsuccessfully.	• Loss of internet connection • Server is down • Customer inaction	• Customer is worried whether or not the payment has reached the system. • Other malicious person might attempt to make payment or capture payment detail if the customer is not around the PC.	Block system.	Confirmation page.
	MORE – The payment detail is sent out more than once.	• Customer accidentally submits more than once. • Replayed of message by an intruder.	Payment transaction is made twice.	Replay of message	
	OTHER THAN – Incorrect payment detail is sent out (e.g. value of payment, card number)	• Modification of message by an intruder • Customer enters incorrect detail	• Customer has pay more than require • The payment's validation fails resulting in customer's dissatisfaction.	Corrupt the message	Payment should be checked before any orders can be processed.
	OTHER THAN – The customer submits payment detail to other systems/users (message is revealed).	• No encryption of messages • Payment detail is sent to wrong place.	• Disclosure of payment detail to a third party. • Order is not processed if the system has not received the payment detail.	• Tap communication. • Interception	

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	OTHER THAN – Payment detail is submitted but not by the customer.	• Intruder fakes a payment message.	The payment transaction is rejected by card company.	Modification of message.	
	AS WELL AS – Other process occurs when payment is submitted.	• Loss of internet connection • Intruder initiates other process • Customer initiates other process.	Customer has no idea if the payment has already taken place.	Block system	Confirmation page.
Step 7 - *The system confirms the result of transaction.*	**NO** – The system does not confirm the result of the transaction	• The system is intentionally blocked from sending back the message so that the payment can be carried out without the customer knowing. • Loss of network and other communications	Customer's worry	Block system	Confirmation page or send a confirmation e-mail to the customer.
	MORE – The system confirms result more than once.	Duplicate of message	Customer is worried on how many transactions are made for the payment.	Replay of message.	
Step 8 - *The operator processes the order.*	**NO** – Same as above in operator's action.				
Post-condition - *The order and its detail are received and processed.*	**NO** – The data is not received.	• Loss of network • Interception • System is blocked	No ordering and purchasing taken place.	• Block system • Interception	
	MORE – Superfluous information is received.	Modification of the message	Wrong delivery/payment detail causing customer's annoyance towards the company.	Modification of message.	

Table 10: Analysis of 'Order Goods' use case.

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
Use Case: Change password					
Actor					
Operator's action – *Change password*	**MORE** – Operator repeatedly change password	Operator wants to use the same password still but was forced to change, so needs to change until he can use his original password.	Vulnerability to attacks if same password is used.	Social engineering, to convince the operator to use the same password.	Strictly enforce the password policy.
Operator's capability – *Change password*	**NO** – The operator has no ability to change the password.	• Operator cannot remember old password • The old password is incorrect.	Vulnerability to attacks if same password is used.		
Association					
Association – *Operator* and *Change password*	**OTHER THAN** – Change of password by someone else.	• Obtain of old password • Change in password files or database.	Unauthorised person may freely use the system for his/her own benefits.	Steal, social engineering, bribe.	Passwords in database/files should also be protected or encrypted.
	NO – No association between 'Change password' and the operator.	• Operator does not change password.	Vulnerability to attacks if same password is used.	Social engineering, to convince the operator to use the same password.	Enforce the passwords to be changed regularly
Use Case Element					
Pre-conditions – *The operator is logged on.*	**NO** – The operator cannot log in.	• Loss of network/ software problems • Account and/or password are incorrect • The operator was blocked • Account is disabled	No activities can be done.	• Block system • Modification of the stored account information.	

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
	MORE – More than one user/operator log on with the same account.	Obtain account information and password.	Unauthorised person may freely use the system for his/her own benefits.	Steal, social engineering, bribe.	Enforce only one user to be logged on at a time.
	LESS – N/A				
	OTHER THAN – The account is logged on by someone else.	Obtain account information and password.	Unauthorised person may freely use the system for his/her own benefits.	Steal, social engineering, bribe.	
Step 1. - *The operator enters his old password*	**NO** – N/A				
	MORE – The password is entered with many attempts.	• Intruder repeatedly guesses the password.	Unauthorised person may freely use the system for his/her own benefits.		Time-out and block the system if a number of attempts have exceeded.
	LESS – N/A				
	OTHER THAN – The operator enters an incorrect password.	Incorrect password is entered.	The password validation fails		
Step 2 - *The system validates password.*	**NO** – The system was unable to validate password.	• The password does not match the existing password in the system. • Password stored was changed by an intruder	The password validation fails. The operator is not able to access the system.	Modification of the stored account information.	
Step 3 - *The operator enters his new password twice*	**NO** – N/A				
	OTHER THAN – The operator enters wrong password in the second time	Typing mistake	Password will not be changed.		
	OTHER THAN – The operator enters a password twice which is not expected.	• Wrong position of hand on keyboard • Character is sent wrong from keyboard • The password is stored incorrectly.	Operator will be unable to logon.	Modification of the stored account information.	

Elements	Guideword-Deviation	Cause	Effects	Threats involved.	Comments
Step 4 - *The operator confirms the change*	**NO** – The change detail was not sent out.	Loss of network and connection	Operator will be unable to logon.	Block system	
	MORE – The change is sent out more than once.	The operator clicks the button twice.	Not a problem.		The system should prevent 'double clicking' action.
	OTHER THAN – The operator confirms the change for detail not expected (the detail sent out is wrong).	Modification of the message after transmitting.	Operator will be unable to logon.	Modification of the message	
Step 5 - *The system saves password to the system.*	**NO** – The password is not changed.	• Program error • Password input contains invalid character	Operator will be unable to logon.		
	OTHER THAN – The system saves wrong value of password.	• Program error • Intruder	Operator will be unable to logon.	• Spoofing • Modification of the stored account information.	
Post-condition – *Password is changed.*	Same as above				

Table 11: Analysis of 'Change Password' use case.

6.3 The results

The analysis identifies unexpected behaviours of the system and the actor, as the results from the application of the HAZOP guide words to the use case elements. Causes and consequences of each behaviour are identified. Further comments and recommendations can be investigated. This section summarises results obtained from the case study analysis.

The results highlight potential threats to the system. For example, impersonation, message corruption or unauthorised accesses to the system (by social engineering, bribery, stealing, hacking etc.) are identified as possible threats. The likelihood of these threats should be subject to further analysis, so that correct measures can be provided.

An issue raised from the discussion during the analysis is the order of the product lists displayed to the customer. The use case for browsing a catalogue stipulated that a list of relevant products be returned to the customer. One of the attributes of a list is the order in which the items appear (as anyone familiar with search engines will know). Applying OTHER THAN on this attribute led to discussions of the possible effects of different ordering and possible means of resolution.

Another example of security-related issue raised is the realisation that the user identity representation is non-trivial. As part of the registration process a user was required to submit his or her 'details'. Although our use case checked that the exact details had not already been registered, minor variants could easily be accepted as those of distinct customers. For example, Jeffrey Herebeacker and Jeffrey Hearbehacker would be considered distinct. There may be good pragmatic reasons why customers should not have multiple identities, but there are security issues too. What if we wanted to have of 'one-per-customer' offers? Similarly, the intent of 'one-per-household' offers could be circumvented easily by adopting minor variants of street name (the postal service in the UK is known for its ability to deliver very badly addressed post). This suggested that a more sophisticated and fuzzy notion of equality of details might be needed. Thus, although it might be 'obvious' when two sets of details are the same and when they are not, simply considering the application of OTHER THAN to details prompts us to consider in more detail what we really want, and reveals potential deficiencies in the mechanism we have chosen to implement. Though the technique does not solve the issue, it successfully highlights it for consideration.

Several other potential attacks identified in the analysis are due to the vulnerabilities of the payment process and the handling of confidential details. These suggest to the developer that the security policy for how the payment and the confidential details are to be handled should be made explicit and analysed further.

Additional functional requirements arise from the analysis. Some of these requirements are derived to help prevent or mitigate the likelihood of the vulnerabilities. The following are some of the derived requirements extracted from Table 8 to Table 11.

1. The system needs to check mandatory fields (e.g. all required payment details are filled in).

2. There is a need to provide functions to allow customers to update/modify their detail (in order to avoid unnecessary repeated registration).
3. Acknowledgement messages should be provided to users to confirm actions taken (messages sent). Thus the technique highlights issues relevant for good HCI design.
4. There is a need to provide/display confirmation details before processing the payment of the transaction.
5. There is a need to provide session time out (to protect against users wandering away from terminals etc).
6. Modification of web-site data should only be done by an authorised person in an appropriate manner. (This may be obvious but was certainly not explicit.)
7. There is a need for communications protection. Again, this may seem obvious, but the work exposed various assumptions being made about confidentiality and integrity of such communications.
8. There is a need to specify and enforce a policy on password maintenance.
9. There is a need to consider how abnormal or incomplete transactions should be handled. Additionally, there are obvious needs for the maintenance of a secure audit trail for appropriate accountability.

7. Discussion and conclusions

7.1 Summary of major findings

HAZOP provides a systematic approach to reasoning about the high-level security of a system modelled in use cases. It can be seen as a useful tool in the security analysis armoury. The worked study we have presented here is small but has allowed us to draw conclusions on the utility of the general technique. Below we summarise our observations on the experience of applying HAZOPs to the e-commerce example.

- **HAZOPS helps the derivation of security requirements and policy.** The analysis process is an effective means of teasing out security requirements. At their simplest these will be additional functional requirements. However, we found that the analysis process often provoked discussion about higher-level policy issues. In some environments the security policy may be implicit and in many cases will be incomplete. HAZOPS prompts the analysis team to think in ways they would not otherwise have done.

- **HAZOPS highlights issues.** HAZOPS provides problems and issues, not solutions! It forces the analyst to consider unusual scenarios. Sometimes the issues thrown up have no clear solution (e.g. one could deal with the provision of multiple identities by ignoring them and not offering 'one per customer' offers). Also, in trying to interpret what is meant by particular natural language descriptions one occasionally finds lower level design possibilities being discussed. Though strictly inapplicable at the use case level, lower level implementation issues can be unearthed and recorded for consideration. Inevitably, much design is iterative and in many developments there is some degree of working ahead (and so aspects of implementation will proceed before higher levels have strictly been finished). Working ahead with forethought should save misguided effort being spent.

- **HAZOPs can be applied to all use case elements**. Associations are often glossed over in normal developments but have significant security relevance, raising fundamental questions about who should have accesses to which services. An assumption is typically made in use cases that the actor remains constant throughout a transaction. However, this assumption can be subjected to perturbation, leading to issues of masquerading and impersonation etc. The very existence of association as a concept leads to this sort of thinking. Once an element is identified it can be challenged by perturbation and the consequences considered. Insight is typically needed to generate the scenarios. Although not all security problems will be identified via consideration of perturbations of the core use cases, the method does seem to yield useful information and prompts a much more systematic analysis.

- **Abstraction from communications hides threats**. In our web-based system the principals in transactions are distributed. The language in the use cases often obscures the underlying communications needed. Thus, 'the system presents to the customer', really involves sending the appropriate data across the web,

though this is nowhere mentioned. This abstraction hides attacks based on message interception/spoofing. We found in practice that we did address some communications security issues. Also, in uses cases, simple acknowledgement messages are often omitted. This too can hide or even create threats. If no acknowledgement is actually implemented, a user may induce a transaction again (e.g. buying air tickets for a second time over the web, thinking the first transaction initiated has been aborted).

- **Viewpoint prompts are useful!** We have found it very useful to wear various 'hats' in carrying out the analysis. These are what are usually referred to as viewpoints. Even simple considerations such confidentiality, availability, integrity, accountability and timing prompt the analysts to highlight issues in these areas. Analysts are free to generate these as appropriate. One could easily imagine environmental viewpoints (e.g. temperature, which may have a distinct effect on smart card operations), a passive eavesdropper viewpoint, a maintenance viewpoint, or a cleaner's viewpoint!

- **There are more actors than you think.** Determining what you have forgotten is hard! The analysts must attempt to consciously search for such additional considerations. Thus, for example, cleaners did not appear in any of the previous analysis, and yet they may have physical access to terminals and servers.

- **Elicitation of attack patterns.** The HAZOP based approach helps in elicitation of patterns of attack, as well as analysing and creating a process of developing a pattern library. For example, analysis reveals implicit protocols between the user and the system – and deviations from these protocols; this could form a *pattern* that can be applied generally.

- **Uses cases are programs too!** Although we have deliberately applied the technique to high-level scenarios, this is not essential. A normal program is a sequence of actions, with preconditions and preconditions. The HAZOPs approach can be applied to lower levels of design/implementation (mutatis mutandis).

The use of the technique itself is also observed. The following are some observations made on the results and procedure in carrying out the techniques:

- **There is a lot of repetition: we need a way of summarising the useful findings.** The results produced are very repetitive. This is probably because the same guidewords are applied to each element and action step of the use case, some of which are very similar. Generalization of the identified threats would be one possible approach in summarising the findings.

- **Team-based analysis.** Any team-based analysis approach is more powerful than a single-user equivalent; creativity is encouraged, and alternative views are aired and discussed. Because the systematic analysis of the use cases was carried out by a team (here the three authors), the coverage is wider and deeper than an initial exploration by one of the authors. It is also observed that roles in the team are important. Different roles have different interests on the system (i.e. privacy,

availability and confidentiality). This would help the discussion and raises issues that we might not normally consider.

- **Issues of scoping: individually apply guidewords to each term.** For more detailed analysis of a particular function or behaviour, each individual term or sub-statement could be thoroughly analysed by applying the HAZOP guidewords to. This creates more possible deviations to a particular function. However, we need to see if the findings are worth the effort.

- **Use case-based analysis.** Of course, the technique makes use of use case modelling as the underpinning of the analysis. The thoroughness of the analysis highly depends on how detail and accurate the use case is constructed and the level of abstraction you would like to achieve.

- **Flexibility.** Guidewords are adjustable. Interpretations can be modified according to certain types of the system.

- **Time-consuming and tedious.** Most systematic approach consumes more time, however producing a more thorough result. It is preferably to spend more time and effort to identify what is significant to the system at the early stage of the development. This sort of observation seems intrinsic to the technique. It applies also to some other thorough forms of test, e.g. mutation testing. Automated support for carrying out and recording the analysis. Indeed, since ideas often come quickly, a dedicated recorder who does not participate in the technical discussions may well be appropriate. In our efforts, we typically recorded ideas on a whiteboard and recorded them formally afterwards.

7.2 Conclusions

The role of security analysis is becoming increasingly important. Systems are becoming more complex and are now operating in environment with higher risks. Consequences of such systems failure are of high concern. The method, presented in the paper, provides a more rigorous approach in carrying out security analysis. It is intended to supplement other forms of analysis and should be integrated with it.

The simple case study has demonstrated that it is possible and beneficial to adapt HAZOP (technique widely used in safety community) to Use Case (which is primarily used for capturing functional requirements) to provide a more systematic approach for security analysis. The future work in this area could include: (1) generation of attack patterns from the analysis results, (2) further application to other descriptive types (e.g. procedures, informal descriptions, protocols described using message sequence charts), (3) generalisation of threats results and (4) implementation of tool support.

8. References

[1] Allenby, K. and Kelly, T.P., 2001. Deriving Safety Requirements using Scenarios. In: *the 5th IEEE International Symposium on Requirements Engineering (RE'01)*, IEEE Computer Society Press.

[2] Amoroso, E., 1994. *Fundamentals of Computer Security Technology*. Prentice-Hall. ISBN 0-13-305541-8.

[3] Anderson, R. and Stallings, W., 1996. Why Cryptosystems Fail. *Practical Cryptography for Data Internetworks*, IEEE Computer Society Press.

[4] Anderson, R., 1999. The Formal Verification of a Payment System. *In:* M. Hinchey and J. Bowen, ed. *Industrial-Strength Formal Methods in Practice*. Springer-Verlag, London.

[5] Anderson, R., 2001. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, Inc. ISBN 0-471-38922-6.

[6] Ball, C. and Kim, R., 1991. *An Object Oriented Analysis Of Air Traffic Control*. MITRE Corporation, McLean, Virginia. Available from: www.mitrecaasd.org/library/tech_docs/pre1999/wp90w542/

[7] Barcalow, J. and Yoder, J., 1997. Architectural Patterns for Enabling Application Security. *Fourth Conference on Patterns Languages of Programs (PLoP '97)* Monticello, Illinois, September 1997.

[8] Baskerville, R., 1993. Information Systems Security Design Methods: Implications for Information Systems Developments. *ACM Computing Surveys*, 25(4), 375-414.

[9] Bell, D. E., Lapadula L. J., 1974. *Secure computer systems: Mathematical foundations and model*. MITRE Corp., Tech. Rep. M74-244.

[10] Biba, K. J., 1977. *Integrity considerations for Secure computer Systems*. MITRE Corp., ESD-TR 76-372.

[11] Blair, B.G., 2002. *Accidental or Unauthorized Launch*. Available from: http://www.clw.org/pub/clw/coalition/chap2b.htm.

[12] Boeing. *Multi-engine maintenance*. Aero Magazine. Available from: http://www.boeing.com/commercial/aeromagazine/aero_05/textonly/m02txt.html

[13] Brewer, D. and Nash M., 1989. The Chinese Wall Security Policy. *In*: *Proceedings of IEEE Symposium Security & Privacy*, IEEE Comp Soc Press, 206-214.

[14] Brewer, D.F.C., 1993. Applying Security Techniques to Achieve Safety. *In: Directions in Safety-Critical Systems, Proceedings of the Safety-Critical Systems Symposium, Bristol*, Springer-Verlag London Ltd.

[15] Burns, A., McDermid, J. et al., 1992. On the Meaning of Safety and Security. *The Computer Journal*, 35(1).

[16] C & A Security Risk Analysis Group, 2002. *Introduction to Security Risk Analysis and Security Risk Assessment.* Available from: http://www.security-risk-analysis.com/

[17] Carroll, J.M., 1996. *Computer Security, 3rd Edition*. Butterworth-Heinemann. ISBN 0-750-69600-1

[18] Common Criteria Information Board, 1997. *Common Criteria for Information Technology Security Evaluation, version 2.0*, October 1997

[19] Commission of European Communities, 1991. *Information Technology Security Evaluation Criteria (ITSEC), version 1.2*, Commission of European Communities, June 1991.

[20] Clark, D.D. and Wilson, D.R., 1987. A Comparison of Commercial and Military Computer Security Policies. *In: Proceedings of IEEE Symposium on Security and Privacy*, IEEE Comp Soc Press, 184-194.

[21] Craigen, D., Gerhart, S. et al., 1993. *An international survey of industrial applications of formal methods; vol. 1: Purpose, approach, analysis and conclusions*. National Institute Standards and Technology, Technical Report NIST GCR 93/626.

[22] Cullyer, J., 1993. The Technology of Safety and Security. *In: The Computer Bulletin*, vol. 5.

[23] Cybenko, G., Giani, A. and Thompson, P., 2003. *Cognitive Hacking: A Battle for the Mind*. Computer, pp.5056.

[24] Damianou N., Dulay N., Lupu E., Sloman M., 2001. The Ponder Policy Specification Language. *In: Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks*, Bristol, UK, January 2001. Springer-Verlag LNCS, 29-31.

[25] Denning, D.E., 1999. *Information Warfare and Security*. Addison Wesley.

[26] Douglas, B.P. 2000. *Real-Time UML (2nd ed.): Developing efficient objects for embedded systems*. Addison Wesley Longman.

[27] US Department of Defence, 1985. *Trusted Computer System Evaluation Criteria (TCSEC)*, DOD 5200.28-STD, December 1985.

[28] Ellison, R., Linger, R., et al., 1999. Survivable Network System Analysis: A Case Study. *IEEE Software*, 70-71.

[29] The Federal Aviation Administration. Available from: http://www.faa.gov

[30] Fisch, E.A. and White, G.B., 2000. *Secure Computers and Networks: Analysis, Design and implementation*. CRC Press LLC. ISBN 0-8493-1868-8.

[31] Foster N., Jacob J., 2001. *Requirements Gathering and Analysis for Security Protocols*. Department of Computer Science, The University of York, UK.

[32] Herrmann, P. and Krumm, H., 2001. Object-Oriented Security Analysis and Modeling. *In: Proceedings of the 9th International Conference on Telecommunication Systems - Modeling and Analysis*, 21-32.

[33] Hoare, C.A.R., 1985. *Communicating Sequential Processes*. Prentice Hall.

[34] Hoyt, D. B., 1973. *Computer Security Handbook*. Macmillan, New York. ISBN 0-024-69910-1.

[35] International Civil Aviation Organization. Available from: http://www.icao.org/

[36] Isaksen, U., Bowen, J.P. and Nissanke, N., 1996. *System and Software Safety in Critical Systems*. Technical Report RUCS/97/TR/062/A, Department of Computer Science, The University of Reading, UK, 1997.

[37] *ISO Standard 17799*, 2000. Based on BS7799 British Standards Institute Standard for Information Security Management, last published in 1999.

[38] Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G., 1992. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley.

[39] Kelly, T.P. and McDermid, J.A. 1998 Safety Case Patterns – Reusing Successful Arguments, *In: Proceedings of IEE Colloquium on Understanding Patterns and Their Application to System Engineering*, London, UK.

[40] Kelly, T.P., 1999. *Arguing Safety – A Systematic Approach to Managing Safety Cases*. Thesis (DPhil). Department of Computer Science, University of York, UK.

[41] Kelly, T.P., 2002. *Kelly's Publication Web Page*. Available from: http://www-users.cs.york.ac.uk/~tpk/pubs.html

[42] Kienzle, D.M. and Wulf, W.A., 1997. A Practical Approach to Security Assessment. *In: Proceedings of the 1997 New Security Paradigms Workshop*, England, September 1997.

[43] King, D.F. *Learning Lessons the (not quite so) hard way*. Incidents, the road to human factors in engineering. Available from: http://www.hf.faa.gov/docs/508/docs/king15.pdf

[44] Kletz, T., 1992. *Hazop and Hazan: Identifying and Assessing Process Industry Hazards, 3rd edition*. Institution of Chemical Engineers. ISBN 0-85295-285-6.

[45] Kruchten, P. 2000. *The Rational Unified Process: An Introduction (2nd Edition)* Addison-Wesley.

[46] Laprie, J.C., 1992. Dependability: Basic Concepts and Terminology. *Dependable Computing and Fault Tolerance*, vol.5.

[47] Leveson, N.G., Harvey, P.R., 1983. Software Fault Tree Analysis. *The Journal of Systems and Software*, 80-99.

[48] Leveson, N.G., 1986. Software safety: Why, what, and how. *Computing Surveys*, 18(2), 125-163.

[49] Leveson, N.G., 1995. *Safeware, System Safety and Computers*, Addison-Wesley Publishing Company Inc. ISBN 0-201-11972-2.

[50] Longley, D. and Shain, M., 1987. *Data and Computer Security, Dictionary of Standards, Concepts and Terms*. MacMillan Publishers Ltd.

[51] Lynch, J., 2001. *Applying Safety Critical Systems Engineering Techniques to Secure Systems*. Dissertation (MSc). Department of Computer Science, University of York, UK.

[52] McDermid, J.A., Nicholson, M, Pumfrey, D. and Fenelon, P., 1995. Experience with the application of HAZOP to Computer-Based Systems. *In: Proceeding of the 10th annual conference on computer assurance (COMPASS 95)*. Gaithersburg, MD, 37-48.

[53] McDermott, J. and Fox, C. 1999. Using Abuse Case Models for Security Requirement Analysis. *In*: *Proceedings of 15th Annual Computer Security Applications Conference*, Phoenix, Arizona.

[54] McDermott, J., 2001. Abuse-Case-Based Assurance Arguments. *In: 17th Annual Computer Security Applications Conference*. December 10-14, 2001

[55] Mead, N.R., Ellison, R.J. et. al., 2000. *Survivable Network Analysis Method*, SEI, CMU.

[56] Moffett, J. and Nuseibeh, B. 2003 *A Framework for Security Requirements engineering*. Report YCS 368, Department of Computer Science, University of York. Available from: http://www-users.cs.york.ac.uk/~jdm/olpapers.htm

[57] Moore, A.P., 1997. *The JMCIS information flow improvement (JIFI) assurance strategy*. Technical Report 500-190, Centre for High Assurance Computer Systems Information Technology Division, Naval Research Laboratory, Washington, D.C. http://chacs.nrl.navy.mil/publications/CHACS/1997/jifi_web/node14.html

[58] NUREG 0492. Fault Tree Handbook. U.S. Nuclear Regulatory Commission.

[59] Pumfrey, D.J., 1999. *The Principled Design of Computer System Safety Analyses. Thesis (DPhil)*. PhD thesis, Department of Computer Science, University of York, UK.

[60] Redmill, F., Chudleigh, M. and Catmur, J., 1999. *System Safety: HAZOP and Software HAZOP*. John Wiley and Sons Ltd; ISBN: 0471982806

[61] Rumbaugh, J., Jacobson, I. and Booch, G., 1999. *The UML Reference Manual*. Addison Wesley.

[62] Schneier, B., 1999. Attack Trees. *Dr. Dobb's Journal* December 1999. Available from: http://www.ddj.com/articles/1999/9912/

[63] Sindre, G. and Opdahl, A.L., 2000. Eliciting Security Requirements by Misuse Cases. *In: Proceedings of TOOLS Pacific 2000*, 20-23 Nov. 2000 (120-131).

[64] Srivatanakul, T., Clark, J., and Polack, F. 2004. Security Zonal Analysis. Technical Report, YCS-374, Department of Computer Science, University of York, 2004.

[65] Trusted Computer System Evaluation Criteria, a.k.a. "The Orange Book" DoD 5200.28-STD, December 1985 (supersedes 1983 CSC-STD-00l-83). http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html

[66] UK Ministry of Defence, 1996. *Defence Standard 00-56 Issue 1: Safety Management Requirements for Defence Systems*.

[67] UK Ministry of Defence, 1996. *Defence Standard 00-56 Issue 2: Safety Management Requirements for Defence Systems*.

[68] UK Ministry of Defence, 1996. *Defence Standard 00-58: HAZOP Studies on Systems Containing Programmable Electronics*.

[69] Winther, R., Johnsen, O-A., and Gran, B.A., 2001. Security Assessments for Safety Critical Systems using HAZOPs. *In: Proceedings of SAFECOMP 2001*, Budapest, Hungary.