

Effective Security Requirements Analysis: HAZOP and Use Cases

Thitima Srivatanakul*, John A. Clark, and Fiona Polack

Department of Computer Science, University of York,
Heslington, York, YO10 5DD, UK.
[jill,jac,fiona]@cs.york.ac.uk

Abstract. Use cases are widely used for functional requirements elicitation. However, *security* non-functional requirements are often neglected in this requirements analysis process. As systems become increasingly complex current means of analysis will probably prove ineffective. In the safety domain a variety of effective analysis techniques have emerged over many years. Since the safety and security domains share many similarities, various authors have suggested that safety techniques might usefully find application in security. This paper takes one such technique, HAZOP, and applies it to one widely used functional requirement elicitation component, UML use cases, in order to provide systematic analysis of potential security issues at the start of system development.

Keywords: HAZOP, requirement analysis, security analysis, use case

1 Background

Threats to computer systems and data grow as computer networking and literacy increase. To combat the rising number and severity of security risks, developers need to acknowledge the risks and analyse security with particular care. Security research has emphasised aspects or technologies that implement security mechanisms. Confidentiality properties, for example, have been formalised mathematically by many authors and there is much research into cryptography (see references in [12]). However, these are only partial solutions. Pre-defined solutions to past problems do not always suffice and ad hoc approaches to security leave too many loopholes. Subtlety is needed to determine in what security for a system should consist and how assurance can be gained that the developed system provides it. This is particularly so for complex or novel systems.

In the safety domain, analysts also deal with large, complex and novel systems, but have numerous simple but effective techniques for generating and recording evidence; their use is closely integrated with the development process. Fault Tree Analysis (FTA) [11] has inspired security Threat Trees [2] and Attack Trees [13]. These systematically decompose goals, representing findings in

* Thitima Srivatanakul is funded by the Royal Thai Government.

a tree structure. MOAT [6] is a security argumentation technique that uses FTA notations to express its assurance arguments. More sophisticated argumentation [8] is available via derivatives of the Goal Structured Notation [5].

Systematic deviational techniques such as HAZOP (Hazard and Operability Analysis) [10,9], successfully used in safety critical systems, are little used in security. The basic principle of HAZOP is that hazards take place due to deviations from normal or intended behaviours. Winther et.al. [16] has adapted HAZOP guidewords to generate guidelines specific to security attributes. However, the derived guidewords are not flexible enough to bring out the analysts' creativity. Perhaps the most significant deviational activity in security is the Flaw Hypothesis Method [15], one form of penetration testing. It uncovers weaknesses by hypothesising suspected flaws, thus providing a method for assessing vulnerabilities. However, the approach lacks rigour. The process, especially the flaw generation, relies heavily on analysts' domain expertise and skill; it does not encourage proactive analysis.

We have devised a HAZOP general approach for security. Specialising the technique for a particular domain may allow increased rigour and efficiency.

2 HAZOP-Based Security Analysis on Use Case Model

The use case technique [4], now popularised as part of the UML, describes behaviours of a system and actors from a user's perspective. In UML (and the related methods), use cases are usually used to capture and express functional requirements. They represent the system and its interactions, in a simple abstract form that is easily understood by the different parties involved. 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 other than the purely functional. Abuse and misuse cases are proposed as (intuitive) means to capture and analyse security requirements [7,14]. Sindre [14] notes that both security and safety requirements can be elicited from use case diagram and scenarios. Douglass [3] 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 timing constraints. Allenby [1] uses HAZOP on use case scenarios to elicit and analyse safety requirements; the results, in a tabular form, describe possible catastrophic failures, their causes and effects.

Use, abuse and misuse cases are potentially useful for analysis of security requirements. The interface presented to an attacker can often be characterised by use cases. The attacker may 'stay within the rules' (but do so in a particularly clever way) or deviate from what is intended as acceptable behaviour. Attacks may be regarded as 'exceptional flows' in a use case. Current abuse case approaches do not offer a systematic way of generating such unusual deviations.

Our technique extends abuse cases, providing a more rigorous approach. By applying the principles of HAZOP, tailored to a security perspective, to an existing functional requirements representation (i.e. use cases), we systematically

mutate the model and its elements, thus prompting identification of a wide range of threats and other non-functional requirements. HAZOP is thus used to generate alternative behaviours in a systematic way.

2.1 Use Case Elements and Their Security Deviations

Our technique applies HAZOP guidewords to elements of a use case that could be deviated in realisable ways. A key part of HAZOP analysis is to interpret the guidewords for the context of interest. We propose interpretations of guidewords for the attributes of each of the main elements of a use case that are subject to deviations (see Appendix A).

Actors are anything that needs to exchange information with the system [4]. An actor characterises the role that users or external systems have in relation to the system. Actors can represent individual or collective roles.

A person may play different roles and the roles of a person may be restricted. For example, the same person is not allowed to both process and approve a payment. Actor roles can be distinguished by their intent/action and capability. These determine 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 intent or actions (deviations from goals), whether intentional or accidental, can reveal new threats to the system. New malicious actors may be identified by considering intentional deviation from expected actor behaviour. The intent of an actor is interesting – an actor may obey all security rules and participate in a seemingly innocuous interaction, when the real intent is to signal information via a covert channel.

The deviation of actors needs to take note of the 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 teenage hacker, in terms of access to resources and skills.

Associations of actors with use cases model the roles that interact with the system through the functionality modelled in the use case. The interaction may represent the exchange of information or invocation of system operations. In secure systems, access control depends on role; we need a clear idea of which actors should be able to access which use cases for what purpose. The significance of restricting operation access to a particular actor, and of assignment of access controls to particular users could be highlighted by this part of the analysis.

Use cases record intended associations for intentional interactions. It is intended that the associations model the only channels used by the actors. Covert channels can be thought of as unintended associations. For example an association implemented by a communications cable provides a point-to-point channel but physics implements an unintended channel, the emission of electromagnetic radiation that can be monitored by an eavesdropper. We try to identify such general concerns through our analysis.

Actors may have concurrent associations or engage in parallel instance of the same use case. It is possible also that uses cases executed in quick succession may cause problems. In on-line gambling, for instance, repeated sequential or

parallel engagement may simply act as a denial of service mechanism, or may allow collusion among betters.

Use Cases represent tasks that the system must achieve on behalf of the associated actors. Use case documentation usually comprises at least pre-condition, post-condition and sequences of actions (scenarios), including variants.

In the execution of a scenario, deviant interactions could result in exceptions to the flow of events. We can address the causes and effects of variation by considering the deviations of each step in the use case. The use case pre- and post-condition represent states of the system at certain sets of times. Deviation from these normal states may be possible, and may affect security. Analysis must elicit possible causes of such deviant states.

2.2 Analysis Process

A HAZOP analysis is conducted by a team of analysts with varied backgrounds, led by a HAZOP leader. The team systematically investigates each use case element to identify deviation from requirements. Each deviant is further investigated for possible causes and effects. The process has the following steps:

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

2.3 Deriving Use Case Deviants

The guidewords alone do not explain how to derive use case deviants. This section provides additional guidance for performing the HAZOP-based analysis, based on our experience of applying the technique.

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 it. A viewpoint represents a stakeholder's interest in the system. Security analysis must take account of any conflicting viewpoints and the relative importance of stakeholders, but deviations from all stakeholder interests should be considered.

In addition, derivation of deviants considers as viewpoints the three fundamental security properties (confidentiality, integrity and availability).

Role/actor mapping – An actor characterises a role played in relation to the computer system. Individuals within roles are not distinguished, and an individual may play different roles. Deviation analysis must consider the possibility of unexpected interactions through shared and multiple roles. A multi-user role always has potential deviants in which the user in a role who initiates an interaction is not the same as the user receiving the response. The definition of actors and roles may be confused, allowing individuals to operate outside their intended or authorised roles.

Business rules define business constraints on a system, including policies and validation. When deriving the deviations, it is essential to consider what sort of business rules the system does or should enforce. In practice, security policies are often implicit. The use case analysis may help to make explicit the security policy, which may require system requirements to be modified so that functional requirements respect to the policy.

Common knowledge and security violations – Although the systematic HAZOP approach is a significant improvement on intuitive analyses, it is important not to ignore the existing ‘checklists’ of accumulated wisdom in security.

3 Example Application

This section presents a simple case study. An apparently-innocuous web ordering system, Figure 1, is systematically analysed to reveal threats.

A website hosts a company’s product catalogue that anyone using the internet can access and browse. A customer is a person registered with the company who can order goods via the website. To order goods, the registered customer must

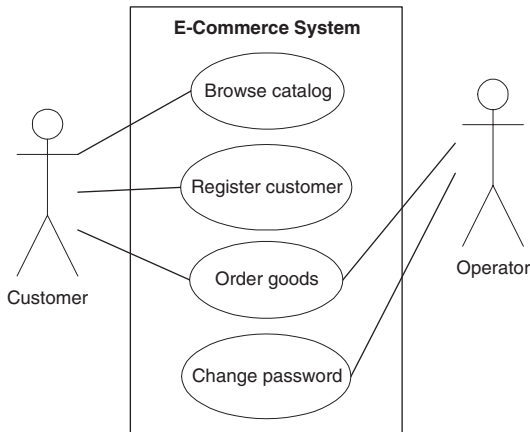


Fig. 1. Simple e-commerce system use case diagram

provide sufficient payment and delivery detail. If an un-registered contact wishes to place an order, the ordering facility provides an initial registration procedure. Orders are processed by an operator, logged on to the host system. The operator password can be changed as required.

Stakeholders' interests include:

1. Company owner - Interest in profits, and reputation of the company.
2. System manager - Interest in system's performance and operation of staff.
3. Developer - Interest in correct operation of the program.

The intents or goals of the Customer actor include *browse catalogue*, *register*, *order goods*, *pay for goods*. The operator's goal is to *process order goods*.

3.1 Analysis

We chose *order goods* to demonstrate how the technique is applied. The description of the use case is given in Table 1.

Table 1. *Order Goods* use case description

Use Case Name	Order goods
Goal:	To order goods from the system.
Actor(s):	Customer Operator
Preconditions:	1. The customer is registered to order goods. 2. The customer has entered registration details e.g. user name (i.e. 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 desired product, 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.
Post conditions:	Order details are entered on the system and the order processed.

The use case description documents functional requirement for goods ordering, capturing the interactions between the actors and the system. It shows what functions the system needs, to satisfy the requirement.

From the use case and its description, we identify actor intent (customer – order goods, pay for goods; operator – process order), actor capability (customer – capability to order goods; operator – capability to process order), associations of customer and operator with *order goods* use case, pre- and post-conditions and the event sequence of *order goods*. Each element is subject to deviation.

3.2 Results

We provide a fragment of the analysis of *order goods*. Table 2 shows the results for the customer intent, Order Goods.

Results in Table 2 highlight potential threats to the system e.g. prank orders, impersonation or unauthorised access to the system by whatever means; deviations from the use case description also reveal issues. For example,

event 6 in the use case, *the customer submits payment detail* (Table 1), deviated using the OTHER THAN guideword, gives rise to *incorrect payment is sent (value or card number)*. This could be caused by an intruder modifying the message, message corruption, or simply by the customer entering incorrect details, perhaps due to distraction or poor interface design.

Applying the guideword, LATE, to event 7, we realise that, if the system does not confirm within a reasonable time, a customer might resubmit details and thus order more than once, a usability issue. A malicious attacker might delay network messages corresponding to event 6 to cause the customer to resubmit.

In the web-based example used here, the principals in transactions are distributed. The language in the use cases often obscures the underlying communications needed. Thus, *system presents to customer* involves sending the appropriate data across the web, though this is not stated. This abstraction hides attacks based on message interception/spoofing. We found in practice that we were prompted to address some communications security issues.

Another intriguing security issue was uncovered by analysis of the registration process, during which a user is required to submit *details*. The use case checks that the details are not those of an already-registered customer. Applying OTHER THAN to *details*, it became apparent that a user could, maliciously or accidentally, obtain multiple identities if name or address details were subtly altered. This might compromise the company stakeholder's wishes to administer "one per customer" or "one per household" offers. This raises issues for the implementation detail-equality determination and the business offers policy.

In addition to exposing assumptions, the analysis raises a number of security-related requirements, such as checks on mandatory fields, and messages for acknowledgement and confirmation of details. From the discussion, other business-related issues emerge, such as the influence on sales results of different orderings of products in displayed lists.

4 Discussion

HAZOP provides a systematic approach to reasoning about the high-level security of a system modelled in use cases, and is thus a useful tool in the security analysis armoury. The study of which part is presented here allows us to draw conclusions on the utility of the general technique. Below we summarise our observations on the experience of applying HAZOP to the e-commerce example.

HAZOP helps the derivation of security requirements and policy.

The analysis process is an effective means of teasing out security requirements.

Table 2. Analysis result from customer intent – Order Goods

Customer's intent	Order Goods
Guideword	NO - The customer does not order goods
Cause	Unsatisfied with the product selections/price Complicated interface Distraction by the company's opponent advertisement
Effects	Lose an opportunity to sell Customer's disappointment
Threats involved	Hacking
Recommendation	User friendly interface design Provision of mechanism to prevent unauthorised pop-up windows
Guideword	MORE - The customer excessively order goods
Cause	Not the owner of the account/or credit card Prank order with fake account
Effects	- 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. - Wasted time trying to verify payment detail for a fake card. - Increase workload to the company staff.
Threats involved	Obtain password and card detail (steal, bribery, social engineering). Impersonating as customer. Get access to other computer, while logging on to the system.
Recommendation	Consideration of policy on payment – before or after delivery.
Guideword	LESS - N/A
Guideword	AS WELL AS - The customer provides insufficient/incorrect detail when submitting order
Cause	Complicated interface Detail not available Typing mistake Distraction
Effects	- Processing of payment would be unsuccessful. - Customer's disappointment. - Increase workload to the company staff.
Threats involved	Steal/virus Distraction
Recommendation	Checks on mandatory field prior to accepting the request.
Guideword	OTHER THAN - The customer intends to achieve something other than to order goods.
Cause	Malicious intents
Effects	Wrong statistics data on the number of customer access Disruption on the service Increase workload
Threats involved	Flood/block the system Impersonating as customer
Recommendation	This raises the issues of whether to allow users to simultaneously log on using the same accounts.

Our illustration reveals issues on payment policy, and rules for simultaneous log-ins on one registration. At its simplest, the results provide additional functional requirements. However, we found that the analysis process often provoked discussion about higher-level policy issues. HAZOP prompts the analysis team to

think in ways they would not otherwise have done. In some environments the security policy may be implicit and in many cases will be incomplete.

HAZOP highlights issues. HAZOP reveals problems and issues, not solutions. It forces the analyst to consider unusual scenarios. Sometimes the issues thrown up have no clear solution. Though strictly inapplicable at the use case level, in trying to interpret particular natural language descriptions, lower level design possibilities may be unearthed and recorded. Inevitably, much design is iterative with some degree of working ahead; working ahead with forethought should reduce misguided effort.

HAZOP can be applied to all use case elements. Once an element is identified it can be challenged by perturbation and the consequences considered. 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.

Abstraction from communications hides threats. In our system the principals in transactions are distributed. The language in the use cases obscures the underlying communications needed. This abstraction hides attacks based on message interception/spoofing. Also, in uses cases, simple acknowledgement messages are often omitted. This too can hide or even create threats.

Viewpoint prompts are useful. We have found the use of stakeholder viewpoints useful. Considerations such as confidentiality, availability, integrity, accountability and timing also prompt the analysts to highlight issues in these areas. Analysts are free to generate viewpoints as appropriate.

There are more actors than you think. Determining what you have forgotten is hard. The analysts must consciously search for 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.

The use of the technique itself is also observed. As a team-based analysis approach, the technique is more powerful than a single-user equivalent; creativity is encouraged and alternative views are discussed. However, there is a lot of repetition in the results. This is probably because the same guidewords are applied to each element of the use case, some of which are quite similar.

Most systematic approaches, including this one, are tedious and time consuming, but produce a thorough result. It is preferable to spend time and effort to identify what is significant to the system at an early stage of development.

The technique's results arise from deviating models or descriptions. Therefore, the thoroughness of the analysis depends on the quality of the model (the use case) and its level of abstraction.

5 Conclusion and Future Work

The role of security analysis is becoming increasingly important. Systems are becoming more complex and are operating in environments with higher risks. Consequences of system failure are of high concern. The method presented here provides a rigorous approach to security analysis. It is intended to supplement and integrate with other forms of analysis. Although not all security problems can be identified by consideration of perturbations of core use cases, the method does seem to yield useful information and prompts a systematic analysis.

The simple case study has demonstrated that it is possible and beneficial to adapt HAZOP (widely used in safety) to Use Cases (primarily used for capturing functional requirements) to provide a more systematic approach for security analysis. 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 threat results and (4) implementation of tool support.

References

1. K. Allenby and T.P. Kelly. Deriving safety requirements using scenarios. In *5th IEEE International Symposium on Requirements Engineering (RE'01)*. IEEE Computer Society Press, 2001.
2. E. Amoroso. *Fundamentals of Computer Security Technology*. Prentice-Hall, 1994.
3. B.P. Douglass. *Real-time UML (2nd ed.): Developing efficient objects for embedded systems*. Addison-Wesley Longman Ltd., 2000.
4. I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992.
5. T.P. Kelly. *Arguing Safety - A Systematic Approach to Managing Safety Cases*. PhD thesis, Dept. of Comp. Science, University of York, UK, 1999.
6. D.M. Kienzle and W.A. Wulf. A practical approach to security assessment. In *Proc. of the 1997 New Security Paradigms Workshop, England, 1997*.
7. J. McDermott. Abuse-case-based assurance arguments. In *17th Annual Computer Security Applications Conference.*, 2001.
8. A.P. Moore. The JMCIS information flow improvement (JIFI) assurance strategy. Technical Report 500-190, Center for High Assurance Computer Systems Information Technology Division, Naval Research Lab., Washington, D.C., May 1997.
9. UK Ministry of Defence. Defence Standard 00-58: HAZOP Studies on Systems Containing Programmable Electronics. 1996.
10. F. Redmill, M. Chudleigh, and J. Catmur. *System Safety: HAZOP and Software HAZOP*. John Wiley & Sons, 1999.
11. N.H. Roberts, W.E. Vesely, D.F. Haasl, and F.F. Goldberg. *Fault Tree Handbook*. System and Reliability Research Office of U.S. Nuclear Regulatory Commission, Washington, DC., 1981.
12. B. Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.
13. B. Schneier. Attack Trees. *Dr. Dobbs's Journal*, 1999.
14. G. Sindre and A.L. Opdahl. Eliciting security requirements by misuse cases. In *Proc. of TOOLS Pacific 2000*, pages 120–131, 2000.

15. C. Weissman. Security Penetration Testing Guideline: A Chapter of the Handbook for the Computer Security Certification of Trusted Systems. Technical report, NRL TM-8889/000/01, 1995.
16. R. Winther, O-A. Johnsen, and B.A. Gran. Security assessments for safety critical systems using hazops. In *Proc. of SAFECOMP 2001*, Budapest, Hungary, 2001.

A Security HAZOP Guidewords for Use Case Model

A.1 Actors Guidewords

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.
- AS WELL AS : Parallel action - as well as the intended or normal action, some unexpected supplementary 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.

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.

A.2 Association Guidewords

- NO : Association does not/can not take place.
- MORE : Superfluous - Interface permits greater functionality to a particular actor than is required. 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.

A.3 Use Case Elements Guidewords

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 apply. 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. This may be one of the following:
 - Repeat - the same actions take place repeatedly.
 - Superfluous - the system does additional actions to those intended or required.
- LESS : Less is achieved than intended. 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.

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 take 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.