Synchronization control: the Therac 25 case study

Avi Harel, Ergolight

ergolight@gmail.com

http://radonc.wdfiles.com/local--files/radiation-accident-therac25/therac25.pdf

Abstract

The Therac-25 accidents are a cautionary tale about the importance of safety in the integration of medical devices, demonstrating the need for early detection of inconsistent situations. The article presents a method for consistency verification, based on a model of synchronization control. Synchronization is required when the controller changes the operative scenario, and the change involves changes in the situation of two or more services. Synchronization problems are due to enabling critical events during the synchronization. The method highlights the benefits of generic rules for associating the system situations with the scenarios.

The Therac 25 case study

The Therac-25 was a radiation therapy machine that malfunctioned, due to software race conditions. The machine was involved with at least six known accidents between 1985 and 1987, in which patients were given massive overdoses of radiation.

Circumstantial investigation

According to Murphy's Law, failure is often attributed to bad luck. According to the theory of Black Swans, coincidence is not predictable (Taleb, 2008). According to theories of organizational accidents, organizations prefer to blame the operators instead of admitting the developer's role (Reason, 1997). Rather, the engineering approach is that accidents should be regarded as a design mistake, namely, lack of protection from the unexpected. Mishaps should be prevented by design.

The Therac 25 accidents are commonly attributed to replacing the hardware interlock with software that was not designed and tested properly. However, in most safety-oriented studies the conclusions are circumstantial, focusing on organizational and disciplinary factors. Notably, Leveson & Turner (1993) identified the following causes for the Therac 25 accidents:

1. Overconfidence in software
2. Confusing reliability with safety
3. Lack of defensive design
4. Failure to eliminate root causes
5. Complacency

6. Unrealistic risk assessment
7. Inadequate investigation
8. Inadequate follow up on accident reports
9. Inadequate software learning practices
10. Software reuse
11. Safe vs. friendly user interfaces
12. User and government oversight and standards

Apparently, circumstantial investigation does not teach how to design similar systems such that this kind of accident does not repeat. In particular, the investigation did not provide any insight into questions such as:

- Why did the administrators not know about the accidents?
- Why did the accident repeat with other patients?
- Why did the accident repeat in several locations?

Reactive root cause analysis (RCA)

Accidents may be prevented based on RCA. Traditional RCA is reactive, namely, specific to an incident. In reactive RCA we look for a single main important cause of the accidents. System operation might fail when the situation does not comply with the controller's needs. In the investigation of this case study in was found that the machine had two treatment modes:

- E-beam therapy, applying low intensity electron-beam directly, over short periods of time
- X-ray therapy, applying X rays decelerated from high intensity electron beam via a rotating tray with accessories for converting E energy to X energy.

The root cause was a synchronization issue within the software that created a dangerous mismatch between the intended and delivered radiation dose. When operating in direct electron-beam therapy mode, a low-powered electron beam was emitted directly from the machine, then spread to safe concentration using scanning magnets. When operating in X-ray mode, the machine was designed to rotate a tray into the path of the electron beam, in order to shape and moderate the power of the beam.

In the accidents of the case study, the E-beam was activated in an inconsistent situation. Here is the sequence of operation resulting in the accidents (Casey, 1998):

1. Operator intends to enter e for E-beam treatment
2. Inadvertently, she enters x, activating the X-Ray treatment
3. Consequently:
   - The electron beam is set to high intensity
   - The tray starts rotating to enable E to X energy conversion
4. The operator changes the mode to E-beam treatment, while in synchronization
5. The change is too quick, less than 8 seconds, the nominal sync time
   - The system fails to change the E-beam intensity to low.

6. The computer screen showed that the mode was E-beam treatment
   - It does not show that the E intensity is high
7. The operator pressed b to operate the beam on
   - The patient received the high intensity E beam while the tray is still out.
8. The screen shows Malfunction 54 error
9. The operator resets the machine and presses b again
   - Again, the patient received the high intensity E beam while the tray is still out.
10. The screen repeats the Malfunction 54 message
11. The operator presses b again, the third time
    - And again, the patient received the high intensity E beam while the tray is still out.

The investigation did not provide answers to a key question, which is why was the operator not aware of the inconsistent situation? Why did the system not inform the operator about the exception?

Proactive RCA

Safety-critical operation relies on situational consistency. The six documented accidents occurred when the high-intensity electron-beam was activated for X-ray therapy without the tray getting into its place; the machine's software did not detect that this had occurred. Three software faults are to blame:

- The software did not provide an interlock that should replace the hardware interlock, which was applied in earlier versions of the Therac 25 system
- The software enabled race between the E-beam and the X-ray treatment
- The software enabled activity during the synchronization

The accidents are due to failure to maintain situation compliance following a change of the operational scenario.

RCA conclusions

This case study demonstrates the need for scenario-driven setting of the service situation (cf. Harel, 2021).

The engineering challenge is to get enough evidence to understand how exceptions are generated. In this case study, the system did not include means to verify that the system situation was consistent. The accidents of this case study are typical of systems that do not maintain the primary scenario variable, namely, when the operational scenario is not defined explicitly. In such cases, the scenarios are fuzzy, and the system design may not include means for detecting problematic situations. Consequently, the services might assume a wrong scenario.

A way to enforce situational consistency is by defining rules for situational compliance with the controller. The scenarios should be defined explicitly and instrumented, for reference by situations classified as normal.

Modeling the system coordination

The engineering challenges of synchronization control are to obtain and employ models of normal operation and operational failure.

Controller multi-service interaction

The basic coordination model is of controller multi-service interaction. Often, the controller consists of two components: an operator, in charge of deciding what and how to operate, and a user interface, used for exchanging information with the services.

Normal interaction is task driven, and subject to proper coordination between the controller and the servers. The controller issues commands or requests to the services, and the services provide reports on their situation and activity.

Operational risks and hazards

An operational risk may be classified as either expected or unexpected. The risks of overdose are well understood, and they may be regarded as expected. However, the ways hazards are generated might be unexpected. Expected hazards are associated with triggers, situations, and activity.

Triggers

Triggers may be classified according to their actuators: human or technological. Human operators are error prone. A human factors version of Murphy's Law is: if the design enables the operators to fail, eventually they will. Technological triggers are much rarer because they are captured during the system verification process.

Human triggers may be either unintentional, or due to confusion. In this case study the trigger was the human operator, who was too fast in correcting the erroneous selection of the E-beam treatment.

In traditional interaction design we empower the human operators, enabling them to access rarely used critical features, to cope with unexpected situations. These features are error prone in terms of feature accessibility or availability: when activated unintentionally, the results are unexpected. For example, if a risky feature, such as high intensity E-beam, is enabled, then the operation might result in an accident due to a design mistake of under-constrained activity, resulting in operator's beta type error.

Situational risks

Situation coordination is error prone, as the number of possible situations grows exponentially with the number of state machines employed in the system operation.

Situational risks may be classified as either external or internal:

- Internal risks are about controllable variables, such as service performance and modes, which the operators may set and change. Internally, the situations must comply with the scenario, as perceived in the controller. Internal risks are due to diversion from the situations defined as normal.
- External risks are about uncontrollable variables, such as environmental conditions and other contextual parameters. Externally, normal operation must be in the performance envelope. External risks are due to approaching the performance boundaries, defined as limits of performance variables. In the context of radiotherapy, the performance envelope consists of the operational conditions, including the audibility and visibility of the system alarms.

Scenario-driven coordination

A situation may be regarded as consistent if it complies with the scenario. This definition implies that controller-service interactions should be dominated by the controller. Scenarios are defined at the controller, and the situation consistency is enforced by a mapping from scenarios to a vector space of the process states. Accordingly, a situation is consistent with a scenario if it is identical to a projection of such mappings. In the Therac 25 example:

- The scenarios are E-beam and X-ray treatment
- The situation space consists of the beam intensity and the tray position (in, out)
- The situational mapping: treatment ➔ (beam intensity, tray position)

In this case study, we have two consistent situations, defined by the mappings:

- E-beam therapy ➔ (low intensity, tray out)
- X-ray therapy ➔ (high intensity, tray in).

Activity risks

Almost all system units, and almost all system features, are prone to activity risks due to mistakes in constraining the system situation (cf. Harel, 2021). Activity risks demonstrated in the Therac 25 accidents are incidents and race conditions.

- An incident is an instance of operating in exceptional situation, not supported by the operation design. In the case study, the exceptional situation was (high intensity, tray out).
- A race condition is an instance of inconsistent scenarios, resulting in inconsistent service operation. In the case study, the electron beam was of high intensity, which was suited to the X-ray treatment, and not to the E-beam, which was the intended treatment.

Incidents and race conditions are often due to enabling operation in inconsistent or situations or in fuzzy scenarios.

The synchronization model

The synchronization model is applicable to events of changing the operational scenario through interactions involving two or more services, which should prepare for a subsequent task. Synchronization is required when the controller changes the operative scenario, and the change involves changes in the situation of two or more services (processes). A synchronization session begins with a command to change the scenario. Following the command, the services need to change their operational modes, according to a scenario-driven mapping scheme.

The session ends when all the services have completed their mode transition successfully. Only after all the preparations are finished, may the system resume operation.

Synchronization risks

During the synchronization, the system situation is inconsistent, therefore, any activity at this stage is unpredictable. In the Therac 25 accidents, the operators changed the scenario from E-beam to X-ray. According to the mapping scheme, the services changed the situation from (low intensity, tray out) to (high intensity, tray in). The accidents occurred when the situation was not consistent, before the situation changed was completed.

The synchronization fails when reaching a timeout before all of the processes have completed the mode transitions. The system should notify the operators about such events.

Protection strategies

The protection framework

The protection plan is based on the cybernetics model of feedback control loops, as described in the STAMP paradigm, and developed in the STPA methodology (Leveson, 2004).

To design the coordination between the controller and the services we may apply the principle of multiple layer defense, as demonstrated using the Swiss Cheese illustration (Reason, 1997). In the context of the synchronization challenge, we may apply three layers, according to the following approaches:

1. Preventive: scenario-driven mode setting,
2. Proactive: impact validation, and
3. Reactive: compliance verification.

To be on the safe side, we should protect the system from all risky situations and from all risky activities, because we cannot tell when one of them might be disastrous.

Preventive: scenario-driven mode setting

In normal operation, the situation is scenario driven. The controller and the services are coordinated by the operational scenario. Scenarios may be used as situation vectors, namely,

pointers to the set of state machines, thus reducing the situational complexity from exponential to linear.

The scenario should be implemented as a concrete system entity that the system should handle, ensuring that the situation is well defined. In particular, the concrete scenario eliminates the risk of race condition, because at any time, only one scenario is operative, and there is no race.

In the Therac 25 case study, the implementation of the operational scenario is the radiotherapy treatment. Key design challenges include:

- Define normal situations as projections from scenarios to the situation space
- Disable any activity during the situation transition
- Notify the operators about the transition and its completion
- Alert in case of exceptional transition time

Proactive: validating the expected impact

The accident could have been prevented had the Therac 25 units followed a validation protocol. According to this protocol, the controller could have queried the services about their situation. In response, the controller should have verified that the services' situations suited the synchronization needs. If the synchronization needs are not satisfied, the controller should have disabled the operation and provided a warning to the operators.

Reactive: compliance verification

The controller should react to the operator's command according to the sync state: during the synchronization, the command should be delayed until after the synchronization is completed. During and after the synchronization, the operators should be notified about the sync state.

To enforce the operator's awareness of the operational situation, we need to apply two mechanisms:

- Situational verification: ongoing notification, as long as the situation is exceptional
- Activity verification: an alarm on changing from normal to an exceptional situation.

Rule-based implementation

The key to enabling the synchronization control is by defining the constraints in terms of rules. Many rules are generic, and customizable. Generic rules enable reducing the costs of eliminating operational risks.

Activity tracking

To enable detection of risky situations, the system should include means for activity tracking. Activity tracking is essential for learning from accidents. A well-known example of such add-ons is the black box used in airplane accident investigation. An example of an add-on designed for

Windows applications was the Ergolight tracker, which was part of the usability testing suite, proposed by Harel (1999). This kind of tracker enables to accelerate the learning, by capturing near misses, namely, capturing and reporting on exceptional situations before they turn to accidents.

Activity trackers may be integrated in the software design or added to an existing design.

Scenario definition

The risk of design based on implicit scenarios is that the system situation is not consistent, in the sense that different system components might assume different scenarios.

To enable the situational coordination, the scenario should be defined explicitly, and implemented as a primary system variable. The scenarios should be defined such that a scenario change involves:

- Enforcing consistent change of the controllable variables
- Verification of the consistency of uncontrollable variables.

A scenario definition may be regarded as optimal if it is not redundant.

Situational rules

The basic rules about the situation consistency about the compliance of the situation with the scenario:

- Consistency: The situation should be identical to the vector projected from the scenario
- Protection: While the situation does not comply with the scenario, all risky activities should be disabled.

In the Therac 25 case study, the relevant scenarios were E-beam and X-ray treatments. The customized rules for setting the controllable variables should be:

- While in the E-beam scenario, the services' modes should be (low intensity, tray out).
- While in the X-ray scenario, the services' modes should be (high intensity, tray in).

To enable the situation verification, the system requirement should specify the operational scenarios, the system design should manage the scenarios, and the software program should verify that the implementation complies with the design.

Situation verification

The situation verification may rely on the skills of the human operators. Rules about enforcing the situation awareness are:

- The controller should notify the operators about the compliance

- While the situation does not comply with the scenario, the controller should provide a salient notification about it.

These are generic rules that may be customized for specific synchronizations.

A design challenge is to specify the reaction to violation of the consistency rules. Apparently, the Therac 25 software design missed the requirements for situation verification.

Activity rules

The operational activity consists of events, each associated with a transition between system situations. Therefore, the activity rules are derived from the situational rules. In this case study, prior to the transition, the active scenario was E-beam treatment, and the system situation was consistent with the mapping from the scenario, namely, (low intensity, tray out). Then, after the operator changed the scenario to X-ray treatment, after the synchronization session has been completed, the situation was consistent with the mapping from the X-ray scenario, namely, (high intensity, tray in).

The activity rule derived from the situational rules is that on changing from E-beam to X-ray treatment, the services' situation should change from (low intensity, tray out) to (high intensity, tray in).

Activity verification

The controller should inform the operators about situation changes. The verification rules included are:

- Services feedback on receiving the command for changing their states
- Services notification on completion of the state transition.
- Controller notification on completion of all the state transitions.

These are generic rules, that should be customized for particular synchronizations.

Conclusions

The Therac 25 accidents demonstrate the need for early detection of inconsistent situations. Synchronization problems are due to enabling critical events during the synchronization. The article presents a rule-based model of synchronization control, demonstrated with the Therac-25 case study, and a method for eliminating synchronization risks based on this model. The method highlights the benefits of generic rules for associating the system situation with the scenarios.

The vision proposed here is that standards on integration engineering may include a chapter on when and how to apply the method for synchronization control.

References

Casey, S 1998. *Set Phasers on Stun: And Other True Tales of Design, Technology, and Human Error (2nd ed)*. Aegean Publishing Company. pp. 11–16.

Harel, A 1999. Automatic Operation Logging and Usability Validation. *Proceedings of HCI International '99*, Munich, Germany

Harel, A 2021. Scenario-based modelling DOI: 10.13140/RG.2.2.12834.35523

Leveson, N 2004. A new accident model for engineering safer systems. *Safety science 42* (4), 237-270, *2004*. 3052

Leveson NG, Turner CS, 1993. An investigation of the Therac-25 accidents, *Computer 26* (7), 18-41

Reason, J 1997. *Managing the Risks of Organizational Accidents*, Ashgate.

Taleb, NN 2008. *The black swan*. Penguin Books.