Combat fire support: the Tze'elim A case study

Avi Harel, Ergolight

ergolight@gmail.com

Combat fire support (CFS)

A basic model describing CFS comprises an interaction between a controller and a servicer. The controller is a combat unit in need for fire support, and the server is another combat unit assisting the controller, providing the combat fire, shooting at the enemy.

During a CFS session, the server situation needs to adapt to the controller. A situation in which the server complies with the controller may be regarded as consistent with respect to the controller.

The Tze'elim A case study - 1990

The Tze'elim Training base is located in Israel. The accident occurred while a reserve unit was conducting an exercise involving the capture of an enemy position. An artillery shell exploded near a group of soldiers on a training exercise, killing five and wounding ten. The accident resulted from a wrong command by the artillery officer at the front unit, applying the command code of the first stage of the plan, also to the second stage, after the front unit had already settled at the target position.

Accident Root-Cause Analysis (RCA)

Accident proofing should be based on RCA. Safety in the CFS relies on situational consistency. The operation might fail when the situation does not comply with the controller's needs. In this case, the supporting unit was shooting at TGT1, while the controller's need was to shoot at TGT2.

The analysis indicated that the safety measures were not sufficient, and that the accident could have been prevented had the exercise managers controlled the activity according to the plan. In this particular case, the primary scenarios were the stages of the exercise.

Operational errors

Traditional RCA is reactive, namely, specific to an incident. In reactive RCA we look for a single main important cause of the specific accident. The conclusions are often circumstantial, justified by quoting Murphy's Law. This accident is commonly attributed to a coding error.

Analysis of many accidents has shown that the term human error is just a name for operational failure that the human operator was not able to prevent (cf. Dekker, 2007).

To eliminate human errors, we need to understand how the operation fails. The challenge is to get enough evidence to understand how errors are generated. In this case, an error in the command code was regarded as an operator's slip, but it should have rather been regarded as a design mistake, namely, lack of warning about the code error.

Proactive RCA

Professional RCA should be proactive. A proactive version of Murphy's Law is that failure results from design mistakes, and therefore errors should be expected and prevented by design. In this case, the challenge is to detect and block erroneous codes.

RCA conclusions

The accident of this case study is typical of systems that do not maintain the primary scenario variable. In such cases, the scenarios are fuzzy, and consequently the server might assume a wrong scenario. If the scenario is fuzzy, then the system design may not include means to detect problematic situations. In this case, the system did not include means enabling to detect that the target coordinates were not compliant with the exercise stage. This is typical of CFS design in which the operational scenario is not defined explicitly. The exercise stages were defined clearly, but the system was not automated, and therefore the stages were not integrated in the exercise control.

Modeling the fire support

The coordination model is based on the cybernetics model of feedback control loops, as described in the STAMP paradigm, and developed in the STPA methodology (Leveson, 2004). The engineering challenges of CFS are to obtain and employ models of normal operation and of operational failure.

CFS may be described in terms of a controller server interaction, in which the server is a fire support unit, and the controller is the front unit that needs the fire support. Normal interaction is task driven, and subject to proper coordination between the controller and the server. The controller issues commands or requests to the server, and the server provides situation and activity reports.

Operational risks and hazards

Operational risks may be classified as either expected or unexpected. The risks of Friendly Fire Accidents (FFA) 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, due to confusion or due to the ‘irony of automation’. Unintentional human triggers, such as in the Tze’elim A accident, are called slips (Norman, 1983).

Situational risks

Situational risks may be classified as either external or internal:

- 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 CFS, the performance envelope consists of the visibility and safety zone, ammunition limitations and restrictions, etc.
- 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.

In normal design, almost all nontrivial system units are prone to situational errors, in terms of accessibility or availability. For example, if a critical feature, such as launching a shell or a missile, is enabled or accessible in the wrong scenario, then the operation might fail due to a design mistake of under-constrained activity. The Tze’elim A accident is due to missing constraints about the target of the CFS.

Situation coordination is error prone, as the number of possible situations grows exponentially with the number of state machines employed in the system operation. 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: when activated unintentionally, the results are unexpected.

Activity risks

Activity risks are mode errors, namely, failures due to operating while the mode (operational state) of a system unit does not comply with the operational scenario. 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).

Mode errors may be due to enabling operation in exceptional or in fuzzy situations. In such situations, the activity intended for a particular scenario might be risky in other scenarios.

Protection methods

The framework

To design the coordination between the front unit and the supporting unit we may apply the principle of multiple layer defense, as demonstrated using the Swiss Cheese illustration (Reason, 1997). In the context of CFS, we may apply following approaches:

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

Preventive: scenario-driven mode setting

In normal operation, the situation is scenario driven. The controller and the server 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. This model assumes that both the controller and the server assume the same scenario, implying that the scenario should be implemented as a concrete system entity that the system should handle.

This case study demonstrates the need for scenario-driven setting of the service situation. (cf. Harel, 2021). The accident is due to failure to maintain situation compliance following a change of the operational scenario. This kind of accident is typical of systems that do not maintain the primary scenario variable. In such cases, the scenarios are fuzzy, and consequently different system components might assume different scenarios. In such cases, the situation is not consistent. In this accident, the target points could have been associated uniquely with the exercise stage according to the plan, and the active point would have been derived directly from the active stage. This is possible if the exercise management is computerized.

Proactive: impact validation

The accident could have been prevented had the CFS units followed a validation protocol. According to this protocol, the supporting unit should have informed the front unit about the intended target coordinates, the front unit should have verified that the target coordinates suited the fire needs and provided a warning if they didn't.

This method was integrated in the 1993 revision of the safety guidelines following the Tze'elim B accident.

Reactive: compliance verification

Both the front unit and the supporting unit could have verified that the command complies with the plan. The verification may be automated, if the CFS exercise is managed by a computer, which traces the exercise and detects instances when the command does not comply with the exercise stage.

The abstract form of this method is that the exercise management should be computerized, and the computer should trace the execution and alert when the execution diverts from the plan.

Implementation

The key to enabling the coordination is defining the constraints in terms of rules. Many rules are generic, and customizable. This feature enables reducing the costs of eliminating operational risks.

Situational rules

To enable the situation adaptation, the scenario should be defined explicitly, and implemented as a primary system variable. Otherwise, different system components might assume different scenarios, and the corresponding situations might be different. The scenarios should be defined such that a scenario change involves:

- Enforcing consistent change of the adaptable variables
- Verification of the consistency of unadaptable variables.

In this case study, the scenario change was from stage 1 to stage 2. The scenario change implies changing the target point. The rules for scenario-mode compliance should be:

- While in stage 1, the target point should be TGT1
- While in stage 2, the target point should be TGT2

The operation in the incident involved violation of the second rule, namely, the target point in stage 2 was TGT1 instead of TGT2. Apparently, the design of the CFS did not include the proper verification tests.

Enforcing the situational rules

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.

The focus of situation verification is on the mode compliance with the scenario according to the rules and on disabling the operation when they do not comply with the scenario. The design challenge is to specify the rules defining the scenario-mode compliance and the reaction to violation of these rules.

Activity rules

Activity may be defined in terms of changes in the system situation. Accordingly, activity rules are about situational changes. The activity rule applicable to this case study is:

- On change to the stage 2, the system should prompt the operators to change to TGT2.

Error proofing

To be on the safe side, we should protect the system from all risky situations, because we cannot tell when one of them might be disastrous. Practically, this implies that error proofing ought to be a key topic of systems engineering.

Enforcing situation awareness

To enforce the operator's awareness of the rule violation we need to apply two mechanisms:

- Situational verification: ongoing notification, as long as the situation does not comply with the rules
- Activity verification: an alarm on changing from normal to exceptional situation.

Learning

The conclusion was that the safety rules should be revised and should include a chapter on proactive impact validation. Unfortunately, the safety authorities hesitated about the way they should implement the revision, thus enabling the Tze'elim B accident in 1992 (Harel, 2024).

Conclusions

The article presents a model of normal CFS and of FFA due to diversion from the normal procedures.

The Tze'elim A accident demonstrates a need for early detection of exceptional situations, and also a method for detecting them. The method demonstrated here is based on rules for associating the situation of the supporting unit with the scenarios.

The vision proposed here is that system engineering standards may include a chapter on when and how to apply the method for scenario-driven operation.

References

Dekker, S 2007. *Just Culture: Balancing Justice with Accountability*, Ashgate.

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

Harel, A 2024. Combat fire support: the Tze'elim B case study

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

Norman, DA 1983. Design Rules Based on Analyses of Human Error. *Communications of the Association for Computing Machinery,* 26, 254-258.

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