Combat fire support: the Kandahar 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 Kandahar case study - 2001

Three American soldiers and five Afghans were killed when special forces troops called in an airstrike meant to hit Taliban positions on Dec. 5, 2001. Instead, a B-52 bomber dropped a 2,000-pound satellite-guided bomb on a battalion command post occupied by the American forces and Afghan allies. Pentagon officials later said the bomb went astray, because the Air Force combat controller who set the coordinates for the attack had changed the batteries on his GPS receiver, which reset the coordinates back to the user's own location rather than the Taliban position. The officials attributed the accident to training problems.

The analysis indicated that the accident could have been prevented had the design prohibit changing the GPS operation mode. The conclusion is that this and similar accidents could and may have been prevented by avoiding mode setting by default, and by applying the methods of impact validation and of scenario-driven operation.

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 the front unit, instead of the enemy.

The analysis indicated that the safety measures were not sufficient, and that the accident could have been prevented had the front unit controlled the GPS mode and the target point sent to the supporting unit. In this particular case, the primary scenarios were those associated with the GPS operation modes: target acquisition vs. navigation.

Operational failure

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 missing guidance about changing the GPS battery during a CFS procedure.

To eliminate failure, we need to understand how it is developed from an exception. The challenge is to get enough evidence to understand how exceptions are generated. In this case, the failure is attributed to coincidence, but it should have been regarded as a design mistake, namely, lack of protection from the unexpected.

Proactive RCA

Professional RCA should be proactive. A proactive version of Murphy's Law is that failure results from design mistakes, and therefore exceptions should be expected and prevented by design. In this case, the challenge is to detect and block the event sequence that ended up in the accident.

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 for detecting problematic situations. In this case study, the system did not include means to detect that the GPS mode was not compliant with the CFS scenario. This is typical of CFS design in which the operational scenario is not defined explicitly. The GPS control was not integrated in the system, and therefore the GPS modes were not integrated in the CFS 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'. In this case study the trigger was combined: it was the human operator, who followed the instruction to change the GPS battery, as indicated by the GPS.

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 (beta type). The Kandahar accident is due to missing constraints about the GPS modes.

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 GPS mode could have been associated uniquely with the operational scenario, and the active GPS mode would have been derived directly from the active scenario. This is possible if the CFS 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 shooting point and the front unit should have verified that the intended point suited the fire needs and provided a warning if it didn't.

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

Reactive: compliance verification

The supporting unit could have verified that the command complies with the operative scenario. The verification may be automated, if the CFS operation is managed by a computer, which traces the activity and detects instances when is does not comply with the scenario.

The abstract form of this method is that the operation management should be computerized, and the computer should trace the execution and alert in case of exception.

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 relevant scenarios were target acquisition and navigation. The active scenario was of target acquisition, and the derived GPS mode was accordingly. The rules for scenario-mode compliance should be:

- While in the target acquisition scenario, the GPS mode should be target acquisition
- While in the navigation scenario, the GPS mode should be navigation.

The operation in the incident involved violation of the first rule, namely, the scenario was target acquisition, and the GPS mode changed by default to navigation. 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 spontaneous change of the GPS mode, the system should alert about the inconsisten situation.

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.

Conclusions

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

The Kandahar 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

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

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

Harel, A 2024 (B). 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

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