Friend-foe coordination in combat fire support (CFS)

Avi Harel https://avi.har-el.com/

Ergolight consulting ergolight@gmail.com

Abstract

The article suggests a model of controller-server interaction for describing CFS. The design goal proposed is to eliminate the risks of friendly fire. Based on three case studies, the conclusion is that the key to preventing accidents is in managing the risks of operating in exceptional situations, in which the server is not coordinated with the controller. A protocol of scenario-based interaction may be employed to ensure that the interaction is always coordinated.

Objective

Worse than losing a battle are accidents of hitting your own allies, due to identification friend foe (IFF) mistakes. Particularly painful cases are Friendly Fire Accidents (FFA), namely, mistakes in which a supporting unit hits the front unit instead of the enemy. Typically, such accidents are commonly attributed to decision errors made by the human operators. The goal is to propose a model, comprising principles and a procedure for detecting and reporting on coordination problems.

Means for IFF were developed for air force and navy units. These means may apply also to CFS but are not applicable to ground forces. The article presents a model of CFS errors, and a framework for eliminating these errors by design.

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.

Case studies

The model was obtained by analysis of investigation reports of three published CFS accidents. Two of the accidents were in the Tze'elim Training base located in Israel (Harel, 2024 A,B), and the third was near Kandahar, in Afghanistan (Harel, 2024 C).

Tze'elim A accident- 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.

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. The conclusion was that the safety rules should be revised, but the safety authorities hesitated about the way they should implement the revision.

Tze'elim B accident – 1992

In retaliation for Iraqi Scud missile attacks during the Gulf War, The IDF's General Staff Reconnaissance Unit was preparing for the assassination of Saddam Hussein, hoping to eliminate the threat of potential nuclear, chemical, and biological weapons, as well as the missile capability to hit Israel, and the fear that Hussein would continue trying to develop such capabilities.

The preparation included a simulation of an ambush for Saddam Hussein by a guided missile. The plan had two parts: first, a dry run, then after setting target images, shooting with live missiles. Due to confusion at the fire support position, a live missile was shot while the exercise administrators were still setting the target images. Five soldiers were killed in this accident.

Again, the analysis indicated that the accident could have been prevented, had the exercise managers controlled the activity according to the plan. The conclusion was that the safety authorities should implement the revision of the safety rules.

Kandahar accident- 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.

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 Tze'elim A accident: In this case, the supporting unit was shooting at TGT1, while the controller's need was to shoot at TGT2.
- The Tze'elim B accident: In this case, the supporting unit was shooting live ammunition, while the controller's need was to shoot dummy ammunition.
- The Kandahar accident: In this 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.

Typically, people attribute accidents to operators' errors (Dekker, 2007).

- The Tze'elim A accident is commonly attributed to an operator's slip
- The Tze'elim B accident is commonly attributed to an operator's confusion
- The Kandahar accident may be attributed to unfortunate battery failure.

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

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. According to the US National Library of Medicine (NLM), RCA is "aimed at discovering the causes of close calls and adverse events for the purpose of identifying preventative measures" (Charles et al., 2016).

In the case studies, the challenge was to detect and block the event sequence that ended up in the accident.

RCA conclusions

The accidents of the case studies are 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. This is typical of CFS design in which the operational scenario is not defined explicitly.

- In the Tze'elim A and B accidents, the exercise stages were defined clearly, but the system was not automated, and therefore the stages were not integrated in the exercise control.
- In the Kandahar accident, 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'.

- The Tze'elim A accident: In this case, the trigger was a controller's slip, in the command code
- The Tze'elim B accident: In this case, the trigger was a server's confusion about the stage
- The Kandahar accident: In this case, the trigger was power reset at the server.

The common characteristic of the different types of triggers of these accidents is that they changed the situation from normal to exceptional.

Limitations of human control

Human errors are challenging, because humans are included in the system as flexible operators in emergencies, to enable coping with exceptional situations unseen at design time. However, they can rarely do it properly, due to their virtue of training-based reaction. According to the "irony of operation", in emergency, the operators are likely to react as trained in normal operation, instead as by calm, logical decision making. For example, if the system design includes a frequently used prompting to confirm risky operation, the human operator is likely to confirm the prompt automatically, before considering its applicability, as expected by the designers.

Situational risks

The situational risks may be classified as 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 be coordinated. In the context of fire support, the server situation should comply with the scenario, as perceived in the controller. Internal risks are due to diversion from the situations defined as normal.

The accidents in the case studies are 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.

Situational complexity

The number of possible situations grows exponentially with the number of state machines employed in the system operation; therefore, careless design of the situation coordination is error prone, as demonstrated in the case studies. Special coordination techniques, such as scenario-based situation assignment, must be employed to maintain situation coordination.

In normal operation, the coordination is scenario driven. The controller and the server are coordinated by the operational scenario. This model assumes that both the controller and the server assume the same scenario, implying that the scenario is a concrete system entity that the system should handle.

Activity risks

Activity risks are about 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, 2024).

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.

The problem of mode errors may further be extended to describe accidents due to operating in transient situations. In the context of FFA, the extension may refer to the case of a front unit controlling two services, one which is a supporting unit, and the other is an intelligence unit.

Risks of under-constrained activity

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 Tze'elim A accident is due to under constraining the human operator at the front unit

- The Tze’elim B accident is due to under constraining the human operator at the supporting unit
- The Kandahar accident is due to lack of constraints about the GPS modes.

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. Rather than alarming about diverting from the plan, the operation could be enforced to comply with the plan, by setting the operational modes according to the 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.

In hindsight, the case studies demonstrate the need for scenario-driven setting of the service situation. (cf. Harel, 2024):

- In the case of the Tze’elim A accident, the command codes could have been associated uniquely with the plan stage, and the command code would have been derived directly from the stage. This is possible if the exercise management is computerized.
- In the case of the Tze’elim B accident, the ammunition type could have been driven directly from the exercise stage. In the case of the Kandahar accident, the GPS mode should not have been set by default. Rather, the mode could have been set to ‘Target acquisition’ after changing the battery, directly from the fire command.
- In the Kandahar 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

In hindsight, the accidents of the case studies 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.

- The Tze'elim A accident could have been prevented had the supporting unit informed the front unit about the intended target coordinates, and the front unit had confirmed that the target coordinates suited the fire need. The officer at the front unit could examine the coordinates received from the supporting unit and would cancel the fire request.
- In the Tze'elim B accident, at the time of the accident, the impact suiting the front unit needs was dummy ammunition, according to the Dry Run stage, and the impact intended by the supporting unit was live ammunition, which could fit the Wet Run stage. Consequently, the front unit would cancel the action.
- In the Kandahar accident, the B-52 bomber could ask the front unit to confirm the coordinates of the target point, and the front unit could cancel the fire command.

Reactive: compliance verification

In hindsight, 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 it does not comply with the scenario.

- In the case of the Tze'elim A accident, the front unit should have verified that the fire command complies with the plan. They could do it, had the exercise management been computerized.
- In the case of Tze'elim B accident, the computer could detect that the supporting unit is about to fire live ammunition, which was not according to the exercise stage, and could alarm about it.
- In the case of Kandahar accident, the computer could verify that the GPS mode was set to 'Target acquisition', according to the rules of using the GPS, and alarm when it was set to 'Navigation' mode.

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.

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 the Tze'elim A case study is: On change to the stage 2, the system should prompt the operators to change to TGT2.
- The activity rule applicable to the Tze'elim B case study is: On change to wet run (and not earlier), the system should prompt the operators to change to live ammunition.
- The activity rule applicable to the Kandahar 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

Accidents are commonly attributed to human errors. To enable seamless operation, we need to understand the sources of incidents, and find ways to eliminate them.

The article presents a model of normal CFS and of the risks of FFA due to diversion from the normal procedures, a lists protection methods, and operational rules that may eliminate these risks.

Extensions

The problem of FFA is a special case of the controller service coordination problem. The extended problem is of compliance of the server with the scenario, defined by the controller. If the reason for the accident is that the server operational mode does not comply with scenario, then we often call it a mode error. Mode errors were noticed as key sources for well know accidents such as AF 296, airplane TO/GA , TMI, Torrey Canyon, and more.

References

Ryan Charles, Brandon Hood, Joseph M. Derosier, John W. Gosbee, Ying Li, Michelle S. Caird, J. Sybil Biermann, and Mark E. Hake 2016. How to perform a root cause analysis for workup and future prevention of medical errors: a review. Patient Saf Surg. 2016; 10: 20. doi: 10.1186/s13037-016-0107-8

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

Harel, A 2024. Challenges of Enabling Seamless Operation, Submitted to *HSI 2024*
DOI: 10.13140/RG.2.2.34261.40169

Harel, A 2024 (A). Combat fire support: the Tze'elim A case study,
DOI: 10.13140/RG.2.2.33070.73283

Harel, A 2024 (B). Combat fire support: the Tze'elim B case study
DOI: 10.13140/RG.2.2.16922.66248

Harel, A 2024 (C). Combat fire support: the Kandahar case study
DOI: 10.13140/RG.2.2.26988.99203

Leveson, NG 2004. A New Accident Model for Engineering Safer Systems. *Safety Science* 42(4):237-270, DOI: 10.1016/S0925-7535(03)00047-X

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