Enforcing feature availability: the AF 296 case study

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

Ergolight consulting ergolight@gmail.com

The AF296 case study

The case study is about the disabled thrust control.

Air France Flight 296Q was a chartered flight of a new Airbus A320-111. On 26 June 1988, the plane crashed at the Habsheim airshow while making a low pass over the airfield. This was the first fly-by-wire.

The low-speed flyover, with landing gear down, was supposed to take place at an altitude of 100 feet; instead, due to a problematic flight protection envelope, the booster was disabled for eight seconds. Consequently, the plane performed the flyover at 30 ft, skimmed the treetops of the forest at the end of the runway and crashed. All 136 passengers survived the initial impact, but 3 then died of smoke inhalation from the subsequent fire.

Official reports concluded that the pilots flew too low, too slow, failed to see the forest, and accidentally flew into it. The captain, Michel Asseline, disputed the report and claimed an error in the fly-by-wire computer prevented him from applying thrust and pulling up. Five individuals, including the captain and first officer, were later found guilty of involuntary manslaughter. Captain Asseline, who maintained his innocence, went on to serve ten months in prison and a further ten months of probation.

Anatomy of operational failure

Traditional root-cause analysis (RCA)

In the AF296 accident, the controller was the side stick, and the service was the thrust. The failure was in the thrust availability to the pilot. The failure in the case study was that the service was not available when it was needed in emergency.

In this case study, the controller experience was Loss of Control (LOC). The engineering challenge about LOC accidents is to enforce the service availability by design. This case study suggests the need for a scenario-based availability plan.

In proactive failure RCA we define models based on the analysis, intended for availability assurance by design. The impact of the models is demonstrated and evaluated by hypothetical application to the case study.

Exceptions

In the case study, the failure was due to crossing the safety boundaries. The safety boundaries define the transition to exceptional situations. In the AF296 accident, the exception was disabled thrust at low altitude.

The problem in the case study was that the exceptions were not defined explicitly, and the performance boundaries were fuzzy.

Invisible risks

When the exceptions are fuzzy, it is possible to divert to the exceptions, and such diversion is unnoticed. In most of the systems, most of the failures are unknown, because neither the designers nor the customers can notice them, and therefore they are not aware of them. Only when the costs are high do we bother to look for ways to prevent repeating the diversion. Only a small part of the diversions is noticed, namely, when their costs are noticeable.

In case of an incident, when the designers notice a failure, they often attribute it to an operator's error. They are obliged to pay attention to a failure only in case of an accident, namely, when the costs are extremely high.

We should assume that the number of risky situations is huge, because we can see only those that are costly, and because we do not bother to detect and investigate low-cost events.

Operational errors

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). In the AF296 case, flying over the field was considered a pilot error, instead of a design mistake, disabling the thrust.

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.

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.

Modeling the system integration

The challenge of proactive RCA is to elicit common attributes of the case study to obtain a model of failure which applies to other industries.

In the case study, the system had two components: a human controller and a technological subsystem. In each of them, the controller could not activate a critical service, which was required in emergency.

The conclusions from the reactive failure RCA may be used proactively, to obtain a model of system failure, and subsequently, to obtain a methodology for ensuring feature availability. In the proactive version of failure RCA, we look for the enablers of the problematic sources, and for the ways of the situation diversion from normal to exceptional.

A model for describing systems such as in the case study may comprise at least two layers. In the context of availability challenges, the top layer comprises a human supervisor and a technological subsystem, and the technological subsystem comprises a controller and services.

The top layer

The top layer comprises concrete entities and data flowing between the entities.

The top-level entities include a human supervisor and a technological subsystem. In this case study, the technological subsystem is the Airbus 320A aircraft.

The top-level data includes functions applicable to the supervisor, scenarios defined by the supervisor, and tasks defined by the supervisor, applicable to the technological subsystem. The supervisor function in the case study is pulling up. The scenarios defined by the supervisor are pull up vs. protected mode. The subsystem tasks applicable for the case study are direct reflections of the supervisor functions.

This top-level model is typical of many case studies.

The technological subsystem

The technological subsystem comprises entities, processes, situations, and activities. In the case study above, the entities include a controller and a server. The controller is the side stick used for pullup control, and the service is the thrust.

Rule-driven coordination

A rule-based model describing proper operation may help with this task and may facilitate the implementation. In normal situations the controller and the service should be coordinated, according to safety rules. The rules are obtained in hindsight, based on the observation obtained in reactive RCA. For example, a rule defining hypothetical safe operation applicable to the case study could be:

> Rule: when pulling up the side stick, the thrust should be available,

To enforce the service availability when in need, the design should constrain the situations, to comply with the scenario.

Operational risks are associated with diversion from normal to exceptional situations. Exceptional situations are situations which are not supported by the design of normal operation.

Diversions

In normal operation, the system units are coordinated. A diversion from a normal situation to exceptional is called a coordination slip. An availability diversion is a change from coordinated to uncoordinated situation between a controller and a service.

A coordination slip may occur by a trigger, or through a lapse or drift. A trigger is just a name for an action diverting the operational situation from normal to exceptional. A trigger may result from a human slip, from a hardware failure, or from a software bug. In the case study, the accident may be attributed to coordination slips due to a trigger, resulting in disabling the thrust in low altitude, due to a design mistake.

The challenges are to prevent diversions, and to detect the diversion and to notify about it to the supervisor at the time it is generated.

Operational risks

Diversions may be classified as either expected or unexpected. Initially, before the accident, they are unexpected. In hindsight, they are expected and predictable. The challenge is to develop a model of predictable diversions and a model of unexpected diversions.

The system behavior depends on the context of the activity. For example, a control, such as a button, may activate one feature or another, depending on the state of a selector. In normal operation, we need to constrain the activity to suit the desired behavior. Operational risks are often associated with operator errors due to improper constraining:

- Over constraining might result in inability to perform a desired function (Alpha errors). The effect is LOC
- Sub constraining might enable unintentional activation of functions intended to be used in specific situations, such as at setup or in maintenance (Beta errors). The effect is unpredictable.

Both types of errors should be attributed to design mistakes. In the case study, the first slip was due to sub constraining (Beta error), and the effect was diversion to operating in over constrained conditions (Alpha error), in which a safety feature was not available.

The case study demonstrates the effect of wrong constraint: the activity was over constrained, preventing the thrust control.

The engineering challenges about LOC accidents is to enforce the service availability by design. The case study demonstrates the need to prevent any mistakes in constraining feature availability. Such mistakes may be detected in validation testing.

Enforcing feature availability

Principles

The design challenge is to enforce the availability of critical features. This is a special case of a more general problem of controller service coordination: the service mode should comply with the controller scenario. A feature which is critical for the operation of a unit in a certain scenario must be coordinated with that unit, in that scenario.

The model of availability failure may be used as a baseline for a methodology for availability assurance. The case study may demonstrate hypothetical realization of the protection principles and methods, which may be applied to systems like those in the case study.

The desired response to the trigger does not necessarily be the same for the two sources. It may depend on urgency in resuming coordinated activity. The scenario is superior to the service mode because it reflects the task imposed by the monitor. In a scenario-based design, the service mode should adapt to the scenario. In the case study, if the scenario has changed, the controller may activate the service automatically. If the service operators disabled it, the service should notify the controller about the change in the availability.

Diversion control

Availability diversion may result from a trigger originated by the controller or by the service. The challenge is to detect the trigger and to notify about it to the supervisor. A method used to design the coordination between processes is based on the principle of multiple layer defense, as demonstrated using the Swiss Cheese illustration. The layers of availability assurance are:

1. Preventing coordination slips
2. Exploratory decision making
3. Alerting on predictable coordination slips
4. Availability awareness
5. Rebounding from slips
6. Warning on unexpected coordination slips
7. Recovery
8. Escalation preview
9. Sustaining the slip.

Preventing coordination slips

Availability diversion may result from a trigger originated by the controller or by the service. A trigger is an action by the controller or the service, that eventually results in a diversion. The safest way to manage risks is to avoid them. To manage the risks of coordination slips, we may ensure that the system is coordinated. To prevent a diversion, we need to prevent that trigger. The design challenge is to direct the impact of the action such that it does not result in a trigger. We may constrain the operation to

comply with coordination rules. Hypothetical example from the case study: do not apply the protection envelope while the pilot is applying the pullup control.

Exploratory decision making

The motivation to prefer human control over automation is the operator's advantage in decision making. However, to enable proper decisions, the operators need to understand the risks of approaching the protection envelope, and the potential impact of the optional choices. Preview information may help the operators to decide on the safe option.

Preview information may help the decision of the controller operators, by prompting to avoid the diversion and by informing them of the time remaining before an action is required. Hypothetical example from the case study: the system could inform the pilot about the time remaining until the thrust is disabled.

Alerting on predictable coordination slips

Not all coordination slips may be avoided. For example, according to the human factors version of Murphy's Law, if the system enables human errors, then the operators are likely to err. To detect expected coordination slips, we need to include special probes in the activity design. The probes may be designed based on a model of the system coordination. Example of hypothetical detection of expected slips from the case study: the system could inform the pilot about the risks of operating with disabled thrust.

The coordination may be validated continuously; however, it is more practical if it is validated only in an event of change of the controller scenario or of the service availability.

Availability awareness

It might not be sufficient that the system detects and warns about the hazard. Any delay in detecting constraint violation might enable an accident. The discipline for assuring that the operators are aware of the hazard is HCD. The case study demonstrates the costs of detection delay: the pilot was not aware of the airplane being in the protection mode, in which the thrust was delayed.

The design challenge is to notify about a diversion at the time it is generated.

Warning on unexpected coordination slips

Not all coordination slips are expected. Diversions may be prevented only if they are predictable. When they are not predictable, the system may still detect coordination slips, and notify the operators about them. Hypothetical example from the case study: the system could notify the pilot when the protection envelope is applied, disabling the pilot control.

A method for detecting unexpected situations is by risk indicator, based on segmentation of continuous system variables, such as performance variables, or time measurement of process execution or state transition.

Recovery

Occasionally, the operators might fail to rebound from the exceptional situation. In case of a coordination failure both the controller and the service may react, depending on the source.

The reaction may be spontaneous recovery, by rebounding to the original coordinated state, or by enforcing a corrective change in the partner.

The recovery methods are by applying troubleshooting and recovery procedures, and by collaboration between the operators and the system, while in safe-mode operation.

Escalation preview

When operation in exceptional situations, it is important to notify the operators about it. In response the operators need to look for ways to recover from the exceptions. The time frame for recovery may be limited, and knowledge of the time frame is important for deciding well about the recovery procedure.

Sustaining the slip

Sometimes, the system design does not include sufficient means for troubleshooting, and the coordination practically fails. For these cases, the system should apply a last protection layer, which is by employing resilience procedures, such as safe-mode operation

Lessons

The challenge is that safety-critical features should be available when they are needed in case of hazard. The conclusion is that it is possible to enforce the availability of safety-critical features. The design should protect from disconnecting safety critical features and should provide warning when these features are disconnected.

References

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