Chapter 5

Procedures, programs and their impact on dependability

Denis Besnard

University of Newcastle upon Tyne

1 Introduction

The dependability of a system can be enhanced by structuring it so as to limit the flow of errors, so that faults do not result in failures. This concept applies to both the mechanical and the human components of a computer-based system (see Chapter 1 of this volume). The present chapter looks at the impact of rules (programs and procedures) on dependability. More precisely, we look into the interaction that rules permit between humans and machines, highlight the strengths and weaknesses of their collaboration and assess the contribution of this collaboration to dependability. These are issues of importance because they shape the structure of the interactions within the organisation and impact on dependability. For instance, such aspects as stability or survivability of computer-based systems depend on how well rules match the cooperative capabilities of technical and non-technical components, the environment and its (expected or unexpected) events.

2 Rules from an operational viewpoint

There is a need to define how a given component should behave when performing a given task so that this component can be controlled, guided and its behaviour predicted. Rules prescribe a particular action in response to triggering conditions. Most of the time these conditions are recurrent or their occurrence has been anticipated. Therefore, the encoding of the required actions will save resources and provide guidance when these conditions occur. For the time being, let us define a rule.

> *A rule is a sequence of instructions defining an acceptable response to a particular need in a particular context. A rule can have a human or a machine as a recipient and can include a precise objective to be reached and how to reach it.*

2.1 Rules for normal conditions

Assume initially that a computer-based system is in a normal state. The rules defined for its components are intended to ensure that the system stays in this state. Consider the example of a plane flying at cruising altitude with no active alarm. During flight legs, there is a procedure that requires the monitoring of altitude and fuel as a routine scan. This eases the early detection of problems. Also, between flight legs, some radio communications and navigation actions are required to follow the various vectors assigned by Air Traffic Control (ATC). The decomposition of the flight into documented sub-objectives (e.g. reaching waypoints) facilitates moving from one normal state to another normal state (e.g. from cruise to descent).

2.2 Rules for abnormal conditions

When an abnormal state occurs (e.g. an accident) that is covered by rules (e.g. an emergency procedure), these rules will usually be aimed at reverting the overall system to a normal state. If this is not possible, then preventing further damage becomes the priority.

Rules can aim at returning to normal conditions. To continue with our example of a commercial aircraft, assume that the pilot cannot follow the descent path and decides to overshoot ("go around"). This is a non-normal situation that is anticipated during system design and is therefore covered by rules. These rules comprise procedures for the pilots (activating the go-around mode, among many others) and the running of the related program by the aircraft (automatically apply maximum thrust and seek maximum climb rate).

Rules can also have the role of containing further damage.

2.3 When there is no rule: The United Airlines DC-10 flight 232

On July 19, 1989, United Airlines flight 232 bound for Denver crash-landed at Sioux City Airport, Iowa. One hundred and twelve people were killed and 184 survived [5]. The aircraft was forced to land after a metallurgical defect in the fan disc of the tail-mounted engine (#2) caused its catastrophic disintegration. The severity of this failure was such that the crew had no control over ailerons, rudder, elevators, flaps, slats, spoilers, or steering and braking of the wheels. The only control the crew had was over the throttle controls of the two, wing-mounted engines. By varying these throttle controls, they were able, to a limited extent, to control the aircraft in both horizontal and vertical planes. This was done with the help of another DC-10 pilot who was onboard as a passenger and was brought to the cockpit. Flying an aircraft this way is understandably not common practice and several airmanship principles were violated on this flight (e.g. steering with engine thrust, having a third pilot in the cockpit). But, by performing these rule adaptations, the crew were able to reach the airport –where the rescue teams where on standby– and save many lives. This event exhibits the neutral nature of some violations. These can be beneficial to system safety when humans have a valid mental model of the functioning of the system [1]. It allows them to implement ad hoc control modes and, to some extent, cope with unexpected configurations.

3 What or whom are rules for?

3.1 Rules for machines: programs

Even machines that are well-equipped with large numbers of sensors do not perfectly capture the operational context at hand. Even very advanced computer-based systems (e.g. experimental flight deck systems) cannot be guaranteed to capture all contextual changes of relevance to the handling of a given situation. As a result, the execution of well-designed programs can trigger unwanted behaviours. It follows that the correctness of a program cannot be disconnected from the context in which it will be implemented. The crash of the Ariane 5 launcher [3] highlights the importance of this point. The rocket self-destroyed in June 1996 because the amplitude of its movements could not be handled by the navigation system (initially developed for the smaller launcher Ariane 4). The reviews and tests failed to identify that the data generated by Ariane 5 would a) fall beyond the acceptable range and b) make the main on-board navigation computer shut itself down. The resulting off-track trajectory then necessitated the in-flight self-destruction of the launcher.

3.2 Rules for humans: procedures

Although it might seem obvious that machines do not take initiatives and therefore need programs, the issue is not clear-cut when it comes to humans. Procedures fulfil a different need than programs. Indeed, they offer a way of bounding the search space for the people involved. For instance, procedures on a production line attempt to ensure that equipment/parts/information are in the place required at the right time. This has serious implications in critical systems. The Apollo-13 CO_2 emergency scrubbers are such an example. They had to be engineered during the mission after the accidental loss of the contents of oxygen tanks [4]. The intention was that the crew could then optimise the recycling of the air available in the module. The need for such scrubbers was not anticipated by design teams. As a result, no procedure existed to cover this emergency case. It follows that a high number of design possibilities (which materials to use, which are available to the crew, how to assemble them, etc.) had to be envisaged and tested. In the end, everyone involved had to look at everything. This case demonstrates that lack of procedures (e.g. an emergency case that was not envisaged at the design stage) implies huge costs in terms of time and efforts.

3.3 Rules for humans using machines: manuals

[This section was omitted]

3.4 The creation of rules

[This section was omitted]

4 Undependability in rules

4.1 Undependability in procedures

Humans sometimes make erroneous adaptations of rules which can then have dramatic consequences for the control of critical processes. The following example describes a violation that occurred in a nuclear fuel production plant. There is a limited amount of uranium that can be put together without initiating fission. A *criticality* event happens when this critical mass is exceeded. A chain reaction then occurs, generating potentially lethal radiation. On December 30, 1999, in Tokaimura (Japan), a *criticality* accident occurred at the JCO nuclear fuel processing plant, causing the death of two workers [2]. A team had to process seven batches of uranium in order to produce a uranium solution. The tank required to process this solution is called a buffer column. Its dimensions were 17.5cm in diameter and 2.2m high, a shape that diminishes the risks of *criticality* events. The inside of this tank was known to be difficult to clean. In addition, the bottom of the column was located only 10cm above the floor, causing the uranium solution to be difficult to collect. Thus, workers illegally opted to use another tank called a precipitation tank. This tank was larger and situated 1m above the floor. Moreover, it was equipped with a stir propeller making it easier to use for homogenising the uranium solution.

The workers were not aware of the risks of a *critical* event associated with the pouring of the seven batches into a *criticality*-unsafe tank. This error contributed to the accident which was rooted in a complex combination of deviant organisational practices. Among these featured the pressures from the managerial team to increase the production without enough regard to safety implications and crew training. This policy impacted on the safety culture developed by the workers, providing them with excessive liberty, even for critical procedures.

4.2 Undependability due to program(mer)s

[This section was omitted]

4.3 Undependability due to tensions between components

[This section was omitted]

4.4 Undependability due to brittleness

[This section was omitted]

5 Dependability through human-machine cooperation

Since the systems we are concerned with in this chapter are composed of technical and human components, their interaction adds another dimension to the role and application of rules. Namely, the singularities of humans and machines offer a very powerful complementarity that allows computer-based systems to adapt to virtually any situation, including the unexpected ones. The human-machine collaboration is not an equal and fixed sharing of capacities, regardless of the nature of the situation at hand. Instead, humans sometimes need to rely fully on machines for the execution of tasks they cannot perform themselves (e.g. replacing fuel rods in the core of a nuclear reactor). Conversely, there is sometimes a need for the automation to be "unplugged" or ignored. It follows from the above that the interaction of humans and machines, through procedures and programs, spread over a continuum. Within the latter, the final system's dependability (or lack thereof) can be described in terms of a more or less perfect complementarity of very different components' skills. Namely, we believe that humans' and machines' application of rules are most of the time cooperative rather than exclusive from one another.

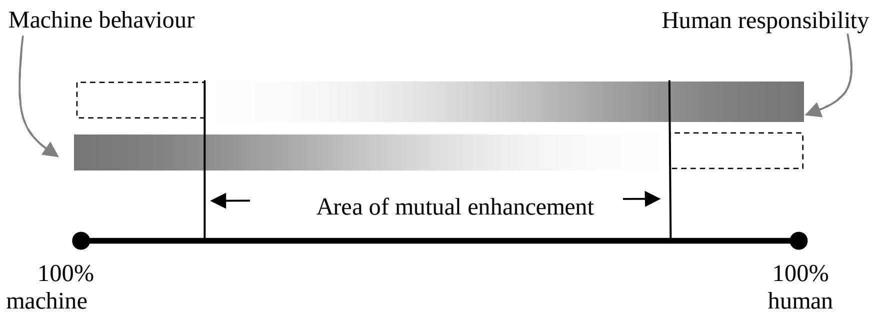

Fig. 1. The continuum of human-machine cooperation

Fig. 1 describes a continuum of potential cooperation between humans and machines. In modern computer-based systems, humans and machines operate simultaneously on the same set of tasks. Only in relatively rare occasions do machines and humans behave independently of their counterpart. This is represented by the dotted-line rectangles in Fig. 1.

The ability to think and adapt is a key property of humans. On the other hand, machines (especially computers) will apply rules (i.e. run their programs) very reliably. Surely, each of these two system's components has weaknesses. However, the collaboration of these two different types of components provides a complementarity that enhances the dependability of systems in general. These issues are discussed below.

5.1 Computers enhancing human performance

Computers can support humans in simple ways, yet provide very valuable help. For instance, computers are machines that in some circumstances provide such actions as *undo*. The latter is the electronic version of trial-and-error. Given that this is a basic human learning strategy, the *undo* function happens to be a powerful and intuitive error recovery tool. Also, some automatic checks can be run on particular user's actions such as file deletions, by asking the user for confirmation that the requested action is indeed intended. In automotive systems, computers run ABS systems or traction control software, thereby contributing to safety. Also, in highly critical systems such as aircraft piloting, software assistants allow pilots to be warned in advance when particular actions are required or when deadlines for actions are approaching. Indeed, modern experimental flightdeck systems incorporate assistance to pilots via model-driven assistant systems that track the pilots' behaviour and compare it to reference plans stored in a knowledge base. This comparison is performed by dynamically modelling the operational context and allows the inferring of context-sensitive operational objectives, flagging of errors, and provides support and assistance (see Chapter 6).

However elaborate such systems are, they remain imperfect. They indeed capture contextual features but remain brittle in that they are not designed to handle any kind of situation or exception. They nonetheless demonstrate the importance given by designers of cooperative systems to the recording of the context for the execution of rules.

5.2 Human recovery from the computer application of an irrelevant rule

Undependability can be caused by changes in the context not being captured by the computer. In this particular case, human components compensate for brittleness by preventing the application of a well-designed, yet irrelevant rule. Examples of the above abound in aviation where pilots turn off irrelevant alarms or prevent the onboard computer from performing undue changes to the navigation plan (see Chapter 6).

6 Final considerations

At least to some extent, the dependability of computer-based systems depends on how humans and machines deal with rules. In the case of simple problems (e.g. reading a sensor and sending its value) under normal conditions, it might be possible for a programmer to design a piece of code for which the specifications are perfectly defined. However, the reality is that computers are now made to interact with humans in the control and supervision of critical missions. Therefore, the need for dependable programs is vital.

Here is a list of issues that designers of critical computer-based systems should consider:

- *Brittleness*.
- *Human behaviour is not dictated by rules*.
- *Humans and machines succeed and fail for different reasons.*
- *Exception handling*.
- *Humans and computers are complementary components*.
- *Misconception*s.

7 Conclusion

Rules (programs and procedures) share several features. They are designed to structure the way the work is done, to provide support in response to the performing of a task; they are intended to limit the propagation of failures from a component system to a larger one. From a cognitive point of view, rules (especially procedures) support humans' actions. From a dependability point of view, procedures are an error limitation mechanism. In this respect, the type of errors, their severity and their frequency must be known in order to provide a risk-sensitive support to their recipient.

The dependability of computer-based systems cannot be attributed to the performance of humans or machines alone. Instead, it is how rules make these two components interact with each other that contributes greatly to the overall system's dependability. Within this framework, rules have a coordination function: they contribute to maintaining the system in a normal state, and to its reverting to such a state when deviations occur. This is a systemic view of fault tolerance and fault recovery in which undesirable states are handled by machines and humans complementing each other (Section 0). This idea has been supported by first considering the nature of rules (Section 1), their recipient and role (Section 2), and the cases of undependability that rules sometimes trigger (Section 0).

References

[1] Besnard D, Greathead D (2003) A cognitive approach to safe violations. Cognition, Technology & Work, 5, 272-282

[2] Furuta K, Sasou K, Kubota R, Ujita H, Shuto Y, Yagi E (2000) Analysis report. Cognition Technology & Work, 2, 182-203

[3] Lions J L (1996) Ariane 5 flight 501 failure. Report by the inquiry board. Available online at http://www.cs.berkeley.edu/~demmel/ma221/ariane5rep.html (last accessed: 09/06/2005)

[4] NASA (1970) Report of the Apollo 13 Review Board. Available online at http://history.nasa.gov/ap13rb/ap13index.htm (last accessed: 30/06/2005)

[5] NTSB (1990) Aircraft accident report. United Airlines flight 232. Mc Donnell Douglas DC-10-10. Sioux Gateway airport. Sioux City, Iowa, July 19, 1989. National Transportation Safety Board, Washington DC, USA