Development of Safe Systems

Jan Peleska

Bremen University — BISS — JP Software-Consulting

Overview

1. The Notion of Dependability
2. Safety Requirements
3. Model Design
4. Safety Requirements and Safety Mechanisms
5. Specific Properties of Safe SW Systems
6. Validation, Verification and Test of Safety Properties

1. The Notion of Dependability

Our objective: Development and verification of safety-critical reactive systems that continuously interact with their environment and control (parts of) the environment in order to perform the user-required services and enforce the required safety properties.

- **Dependability:** The trustworthiness of a computer system such that reliance can be justifiably placed on the service it delivers.
- **Service:** System behaviour as it is perceived by its users.
- **User:** Another system (human or physical) interacting with the target system.

Dependability Attributes

- **Availability:** readiness for usage
- **Reliability:** continuity of service
- **Safety:** avoidance of catastrophic consequences on the environment
- **Security:** prevention of unauthorized access and/or handling of information

  Security attributes:

  - confidentiality
  - integrity
  - availability

Dependability — Faults – Errors – Failures

- **Failure:** delivered service no longer complies with specification
- **Error:** part of the system state which is likely to lead to subsequent failure
- **Fault:** cause of an error

Dependability — Fault-Tolerance versus Fault-Avoidance

Classification of methods achieving dependability:

- **Fault-Tolerance:** provide a service complying with the specification in spite of faults
- **Fault-Avoidance (Fault-Prevention):** a-priori prevention of fault occurrence or introduction
- **Fault-Removal:** reduce the presence (number, seriousness) of faults
- **Fault-Forecasting:** estimate the present number, the future incidence, and the consequence of faults

Our Favourite Approach: Fault-Tolerance on HW level — Fault-Avoidance on specification, design and SW level

Dependability — Safety versus Liveness Properties

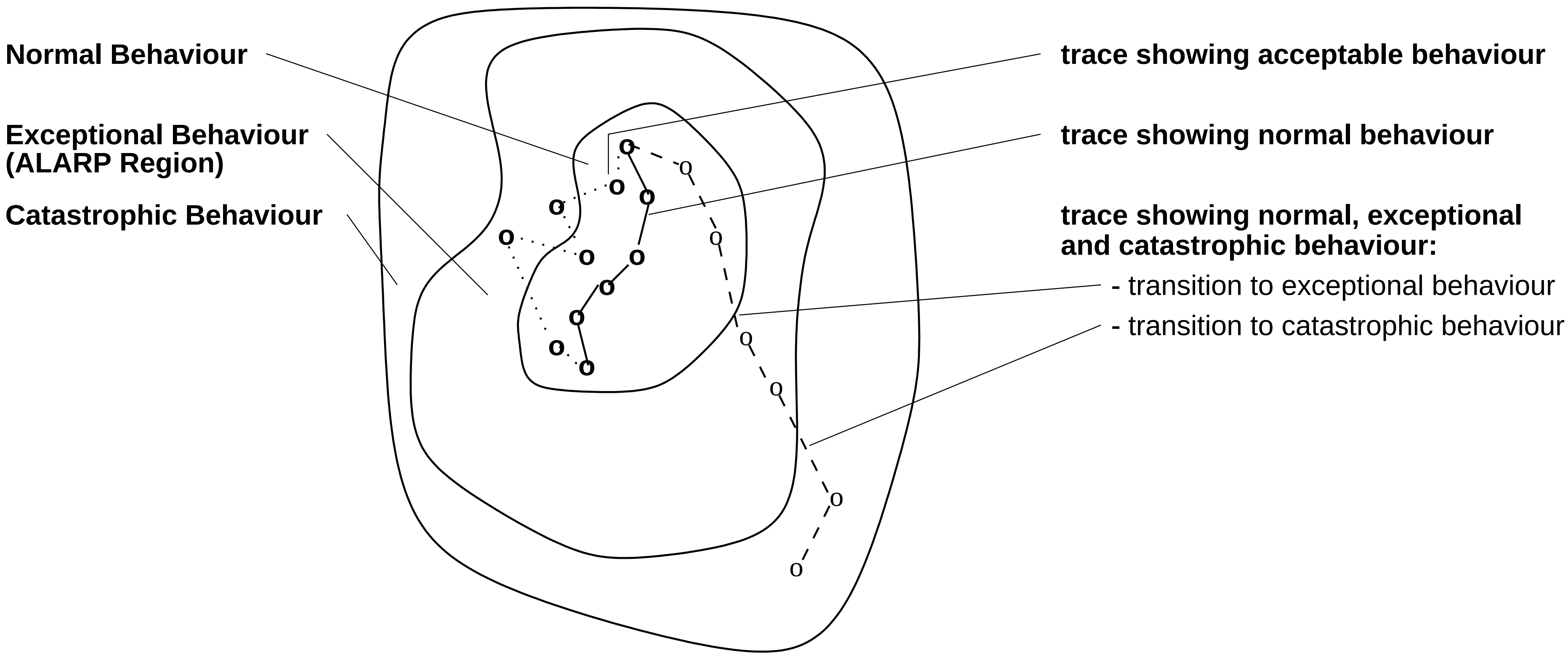
In computer science, specifications are classified as containing

- **Safety Properties S:** Any sequence of events or transitions etc. violating S contains a prefix all of whose infinite extensions violate S. (S always holds)
- **Liveness Properties L:** Any arbitrary finite sequence of events can be extended to an infinite sequence satisfying L. (L finally holds)

For safety-critical real-time systems, all dependability requirements should be specified as safety properties: Liveness properties can only guarantee that a service will FINALLY be delivered, which is not sufficient in the context of hard real-time systems.

Dependability — Classification of System Behaviour

System state space is partitioned into three areas. System execution is regarded as sequence of state transitions, possibly accompanied by visible actions (input/output).



2. Safety Requirements

Safety Requirements specify the subset of safety-critical system requirements.

- Safety requirements are often specified by separate authorities, not by the customer.
- User requirements specified by the customer may be in conflict with safety requirements.

The specification of safety requirements should be separated from user requirements.

Safety Requirements – Common Understanding of Standards

Safety requirements should be derived from a **Risk Analysis**, containing

- **Hazard Analysis:** list of possible hazards, their impact on the environment, their possible causes (e. g., sequences of faults leading to a hazard)
- **Risk Assessment:** quantitative or qualitative estimates for the probability that a hazard will occur

According to today's state of the art, a hazard analysis should at least be semi-formal, for example using fault-trees, event tree analysis of cause-consequence analysis.

Safety Requirements – Common Understanding of Standards

- relate severity of hazard, probability of occurrence and system behaviour in **Risk Diagram**
- derive required effort for validation, verification and test (VVT) of system components from the risk diagram and the impact of component failure on hazard occurrence

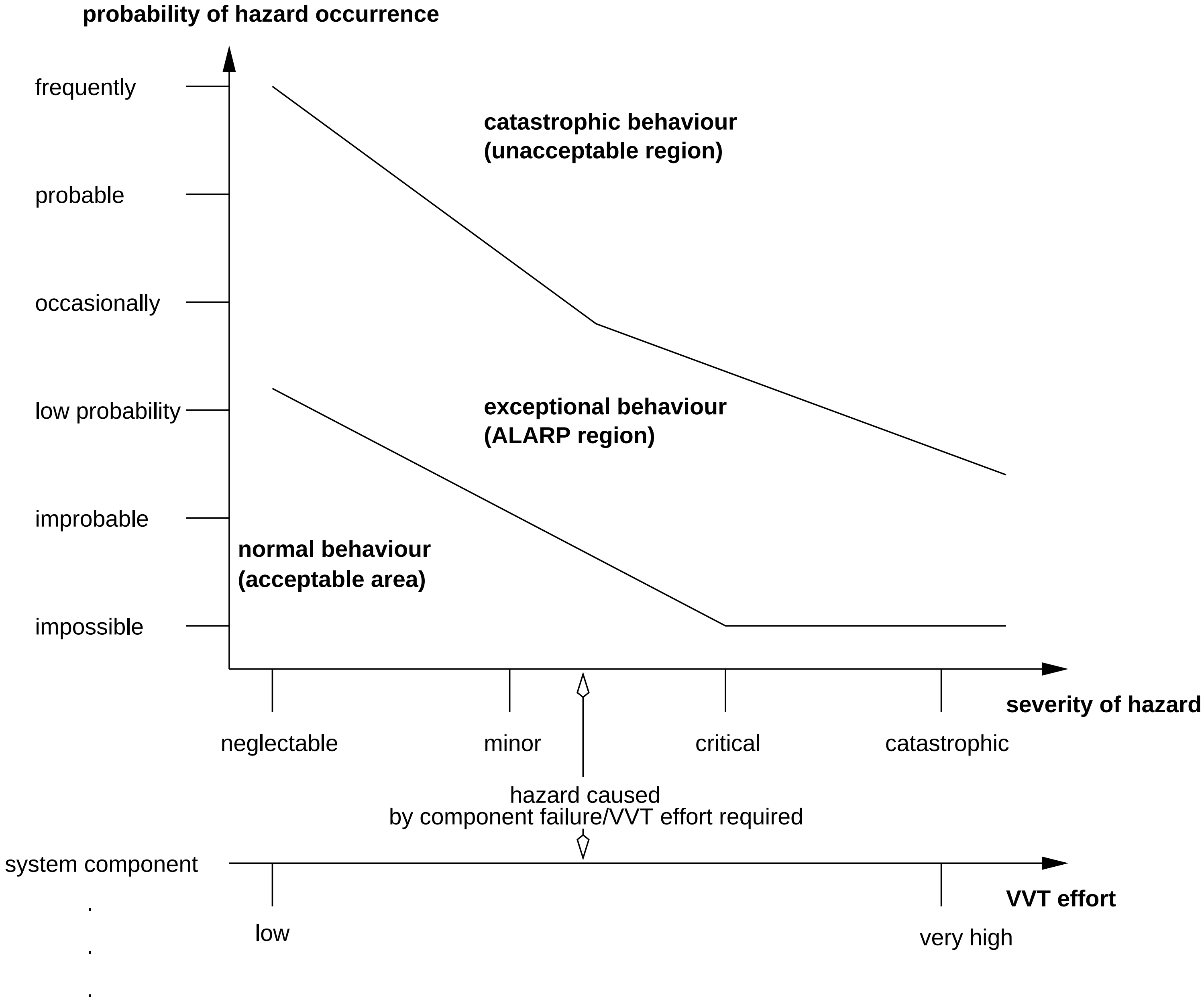
3. Model Design

Separation of safety requirements and user requirements suggests separation of system models:

- **Physical Model:** specifies how the environment to be controlled by the target system behaves

- **Controller Model:** specifies how the target system should behave in order to perform the user-required services and to enforce the safety requirements. Controller model should be devided into model of user requirements and model of safety requirements

- **Hazard Model:** specifies the anticipated hazards that may cause faults, errors and failures in the environment and the target system and may lead to violation of safety requirements for environment and target system


probability of hazard occurrence
frequently
probable
occasionally
low probability
improbable
impossible
catastrophic behaviour
(unacceptable region)
exceptional behaviour
(ALARP region)
normal behaviour
(acceptable area)
severity of hazard
neglectable
minor
critical
catastrophic
hazard caused
by component failure/VVT effort required
system component
.
.
.
VVT effort
low
very high

Model Design — Specification Methods

Specification Methods suitable for physical model and controller model can be classified according to

Method	DM	FM	UBM	TBM	VVT	CG
SA/RT/IM	+	+	·	-	·	·
STATECHARTS	·	+	+	·	·	·
SDL	·	+	+	·	·	+
CSP	·	·	+	+	+	+
CCS	·	·	+	+	+	+
LOTOS	·	·	+	-	+	+
Z	+	+	·	-	·	·
VDM	+	+	·	-	·	+

DM = Data Model — FM = Functional Model — UBM = Untimed Behavioural Model — TBM = Timed Behavioural Model — VVT = support for Validation Verification and Test — CG = support for code generation from specifications — + = good support — · = weak support — - = no support