WGRISK DIGREL Task Group Workshop
Helsinki, Oct. 25 – 28, 2011

# Survey of Failure Modes

*Tsong-Lun Chu*

*Energy Sciences and Technology Department*

*(631-344-2389, Chu@BNL.GOV)*

BROOKHAVEN NATIONAL LABORATORY

a passion for discovery

Office of Science
U.S. DEPARTMENT OF ENERGY

## Outline of Presentation

- Objectives
- Summary of failure mode taxonomies provided by participants
- Definition of levels of detail for hardware and software
- Consensus failure modes for software from DC worksh op
- Persisting scope and technical Issues
- Path forward

## Objectives

- Summarize the failure mode taxonomies provided by participants.
- Clarify definitions of different levels of detail for hardware and software.
- Discuss scope and technical issues.

# Eleven Organizations Provided Inputs

- NRG (Nuclear Research and Consultancy Group)
- EDF (Electricity of France)
- JNES (Japan Nuclear Energy Safety Organization)
- BNL (Brookhaven National Laboratory)
- NKS (Nordic Nuclear Energy Research)
- OSU (Ohio State University)
- ORNL (Oak Ridge National Laboratory)
- KAERI (Korean Atomic Energy Research Institute)
- VTT (Technical Research Centre of Finland)
- CNSC (Canadian Nuclear Safety Commission)
- IRSN (Institut de Radioprotection et de Surete Nucleaire)

## Summary of Taxonomy Inputs from Participants

- Two tables summarizing hardware and software failure modes including the failure modes identified by the software group at DC workshop were prepared.
- Each table further classifies the failure modes in terms of their levels of detail.
  - For hardware: system, channel, module, circuit board, and generic component levels
  - For software: system, channel, microprocessor, and sub-level
- Two additional columns provide information on modeling methods and data sources.

## Levels of Detail for Failure Mode Definition: Hardware

- For a digital protection system, the levels of detail at which failure modes were defined and used in the failure mode survey may include
  - the entire system, e.g., an RPS.
  - a channel or a division: an RPS may consist of multiple (redundant or diverse) channels or divisions;
  - a module, e.g., input, output, and processing modules;
  - a circuit board: a module may be implemented using circuit board(s); and
  - the generic components, e.g., A/D and D/A et al.
- In principle, failure modes of a control system can be defined at the same levels of detail
  - Often control systems do not have redundant channels.

## Levels of Detail for Failure Mode Definition: Software

- System Level:
  - For a digital protection system, at the system level, the software consists of the collection of software running on various microprocessors of the system and failure modes can be defined at this highest level.
- Channel Level:
  - For the redundant or diverse channels of an RPS, the collection of software running on the microprocessors of a single channel may also fail and cause the failure of that channel. Failure modes of all software belonging to a single channel can also be defined at this level as channel level failure modes.
- Microprocessor Level:
  - For the software program running on a particular microprocessor, the software is treated as an individual component like the microprocessor of a module.
- Sub-level:
  - The software that runs on a microprocessor may be complicated enough such that it can be further decomposed, to a so-called sub-level.

## Observations on Participant Inputs

- Some participants included information on failure effects (as a part of an FMEA); others did not.
- In general, most of the failure modes are developed specifically for protection systems only.
- There is more information on hardware failures than on software failures but a set of consensus software failure modes were developed at DC Workshop.
- Levels of detail at which failure modes were defined are very different, and at the same level of detail failure modes from different participants may still be different.
- Although there is no consensus modeling methods, event tree/fault tree approach appears to be the most popular one.
- CCF did not receive too much attention.

## Levels of Detail for Software Failure Mode Definition from DC Workshop, May 2011

- Failure modes can be defined at different levels of detail, i.e., RPS function level, trip signals level (high reactor pressure level), individual signal level (starting individual pump, not applicable to RPS but may be necessary for ESFAS).
- RPS software consists of data acquisition (e.g., communications, filtering, data validation, scaling), logic processing (e.g., standard elementary functions such as comparison or Boolean manipulation, application specific logic), voting (e.g., voting algorithm and communication), priority actuation logic (many be implemented only in hardware).

## Example Software Failure Modes Identified at DC Workshop, May 2011 (1)

- System Level Failure Modes *
  - For an RPS: failure to actuate (including failure to hold); spurious failure; possible others dependent upon additional functions judged to be safety related.
  - For load sequencing: failure to activate in time.
  - For an ESFAS: failure of trip signals such as a high reactor pressure level.

*: failure modes at this level of detail may also be categorized as channel level failure modes.

## Example Software Failure Modes Identified at DC Workshop, May 2011 (2)

- Microprocessor Level
  - For Data Acquisition*: incorrect value, incorrect validity, both, no value, no validity (may be subdivided, e.g., incorrect low or high).
  - For Logic Processing: failure to actuate (including failure to hold), spurious failure.
  - For Voting Logic: incorrect voting, no vote (will lead to failure to actuate [including failure to hold] and spurious failure).
  - For Priority Actuation Logic: incorrect priority, no priority (will lead to failure to actuate [including failure to hold] and spurious failure).

*It may be worthy considering failure modes for communication logic (which seems to be considered a part of data acquisition here) separately, considering its importance.

## Example Software Failure Modes Identified at DC Workshop, May 2011 (3)

- Sub-level failure modes are defined for software functional modules related to individual signals to hardware components such as pumps and valves.
- Example sub-level failure modes include failure of individual signals from an ESFAS to actuate pumps and valves.

## Persisting Issues: Modeling Methods and Data Sources

- "*The taxonomy will be the basis of future modeling and quantification efforts. It will also help define a structure for data collection.*" ~ The CAPS
- Most participants appear to have the fault tree modeling method in mind, though it is not clear if this method can capture all dependencies (e.g., communications between channels), fault tolerant features (e.g., self-diagnosis), and software-hardware interactions (e.g., changes to the software logic upon detection of a hardware failure)?
- Different modeling methods may require different levels of detail. Before a standardized method for modeling is agreed upon, can we determine the preferred level of detail of the failure modes or should we leave the level of detail as an open item?
- Software failures are conditional, i.e., on the environment it is being executed. How can this be captured by the existing modeling methods?

## Persisting Issues: How to Determine the Appropriate Level of Detail for Failure Mode Definition

- Modeling methods often define the levels of detail.
- Data availability is a realistic constraint on the level of detail.
- At the selected level of detail, the defined failure modes should be physically meaningful and the failure effects should be propagatable.
- Capability of capturing fault-tolerance features is desirable.
- Should CCF be defined at the same level as individual failures?

# Persisting Issues: Should Failure Effects Be Included in the Taxonomy?

- Failure effects are needed in order to include failure modes in a reliability model (so that the failure effects can possibly be propagated).
- Can generic failure effects be identified and described or should they only be determined based on a specific application when developing a reliability model?

## Persisting Issues: Definition of Failure Modes, Effects, Causes, and Mechanisms

- In the literature, these terms are often used inconsistently.
- Is there a standard that defines them? Should we endorse such a standard (or standards)?
- Should failure modes be defined in terms of functionality? In terms of functionality, is there any difference between software and hardware failure modes.
- Should failure modes be generic (i.e., apply to any digital protection or control system), be specific to a class of digital systems (e.g., reactor protection systems), or be specific only to a particular system?

## Persisting Issues: Completeness and Genericness of Failure Modes

- Do we want to discuss the issue using the inputs from the participants at each level of detail to ensure some kind of completeness and genericness?
- Common cause failure modes of hardware and software should be further discussed. What should be the level of detail?
- Can communication and synchronization, which are sometimes only auxiliary functions of a system, introduce CCF or dependent failures?

## What Additional Research is Needed on Failure Mode Taxonomy?

- Capturing interactions between hardware and software
- Do we need to treat application, platform, and operating system software separately?
- Assessment of "coverage", for example, the ability/probability of a watchdog timer to detect failures
- Software CCF between diverse digital protection systems
- Use of reliability physics modeling to arrive at data

## Path Forward

- Reach consensus on hardware failure modes and levels of detail.
- Should software failure modes of communication logic also be reviewed?
- For guideline development, should we define failure modes at a specific level of detail or at all levels of detail?
- How to assess the completeness and genericness of failure modes of hardware and software at each level of detail?
- Application of the failure mode taxonomy to case study.

## Backup Slides:

## Failure Modes of Generic Components for a Digital Feedwater Control System (DFWCS)

## Example Failure Modes of Generic Components for the DFWCS System (1)

- Hardware Common-Cause Failure (CCF): The hardware of the main CPU and backup CPU is identical. The occurrence of a hardware CCF may fail the entire system.
- Software: The main and backup CPUs run the same software and a software CCF may occur and fail the entire system. Two failure modes are considered: (1) the software on the main CPU seems to be running normally but sends erroneous output and (2) the software halts and hence, the CPU stops updating output.
- In addition to the CCFs of software, the above failure modes are also considered for the individual software running on the CPU modules considering the fact that the main and the backup CPU are in different modes (controlling and tracking modes) and might be running different portions of the software at any given time.

## Example Failure Modes of Generic Components for the DFWCS System (2)

- Microprocessor of the main CPU: Failure modes considered are (1) the microprocessor seems to be running normally but sends erroneous output and (2) the microprocessor stops updating output.
- Associated components of a microprocessor, such as the ISA bus, random-access memory RAM, ROM, Basic Input/Output System (BIOS), flash disk, buffer, and serial port. It is conservatively assumed that each component has only one failure mode, i.e., a loss of the component, which entails the loss of the functions performed by the component.
- Address logic: This is a generic digital component, also called a decoder. A microprocessor uses the address logic to access the information transmitted on the backplanes. The failure mode is assumed as a loss of the address logic, so that the microprocessor cannot access the intended information upon loss of the address logic.

## Example Failure Modes of Generic Components for the DFWCS System (3)

- MUX and DEMUX: Failure modes of MUXs and DEMUXs are defined in terms of the analog signals they process, which include a loss of one or all signals. No other failure modes of MUXs or DEMUXs were defined and therefore, a loss of signal is modeled in this study as signal fails low.
- A/D and D/A converters: Both A/D and D/A converters are linear integrated circuits (ICs), i.e., the inputs and outputs are proportional to each other; all analog inputs and outputs of the same module share them. The failure modes of an A/D converter include all bits of the A/D stuck at zeros, all bits stuck at ones, and a random bit-failure of the A/D converter. The failure modes of a D/A converter include output fails (drifts) high or low
- It is assumed that if the D/A converter output starts drifting, it will eventually reach the high or low detection threshold.

## Example Failure Modes of Generic Components for the DFWCS System (4)

- Current input and output modules (devices): The major components of the current input/output modules are current loops that essentially are linear transmitters/receivers. They also are linear ICs and their failure modes are current signal fails (drifts) high or low. It is assumed that if the current starts drifting, it will eventually reach the high or low detection threshold.
- Digital input/output modules (devices): Digital input or output is implemented via a solid-state switch. The status of a digital signal is controlled by opening or closing the switch. The solid-state switch may fail to operate (fail as is) and spuriously operate (fails to the opposite state).

## Module Level and System Level Failure Modes for the DFWCS System

- Failure effects of the generic components become failure modes of individual modules of the DFWCS and failure effects of modules become the failure modes of the system
- Failure effects of generic components were propagated to the module level and system level using an automated FMEA supporting tool that was developed based on the source code of the DFWCS
- Module level is an intermediate level and failure modes of modules do not have to be defined.
- The system level failure modes depend on functionalities of the system and therefore, are system specific. For the DFWCS, the system failure mode was defined as a loss of automatic control.