Software Safety

Certification & Safety Case

Prof. Dr.-Ing. Patrick Mäder, M.Sc. Martin Rabe

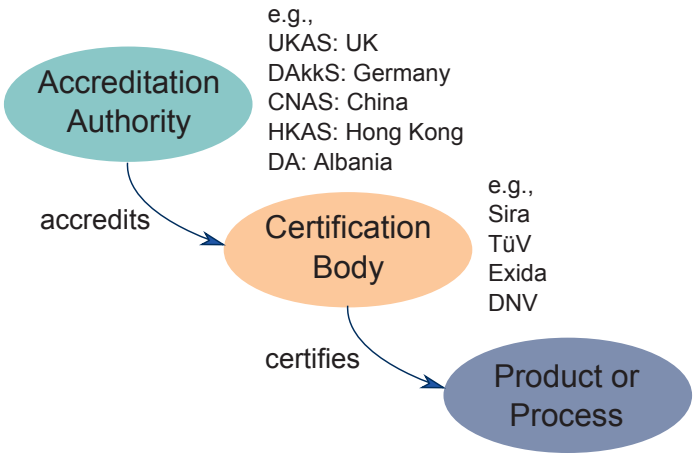
Contents

Certification and Approval

Type Approval vs. Certification

- **Type approval (aka homologation)**
  - is issued by an official authority, e.g., LBA (Luftfahrtbundesamt), KBA, EBA, EASA
  - means that company obtains approval from a government authority that its vehicle types satisfy the legal requirements of the market
  - is mandatory when bringing, e.g., an aircraft or a car to market
  - requires manufacturer responsible for demonstrating compliance through witnessed testing
  - requires proof of safety for hardware and software
- **Certification**
  - is not mandatory nor enforced by law
  - conducted by certification bodies that must be accredited by authorities, e.g., DAkkS
  - simplifies but does not guarantee type approval

Accreditation and Certification



[Hobbs, C. (2019). **Embedded Software Development for Safety-Critical Systems (2nd ed.)**. CRC Press.]

Not Complying to a Relevant Standard?

- Functional safety standards **apply from their official release to all products sold**
- Application of safety standards is **voluntary**
  - **but**: in case of failures or accidents, the compliance to the standards will be checked as they **establish the current state of practice**
  - are **standards violated or not considered**: **serious civil or criminal sanctions** against the company are likely – can be regarded as grossly negligent action or intent → results in increased claims for damages, personal liability and higher insurance premiums for risk hedging (manager liability insurance does not protect)
  - certification according to functional safety standards, on the other hand, **does not relieve product liability**, but it reduces the outstanding claims in case of failure of the system
- The **certification bodies assume no liability**

Certification Is Challenging

[Viagem Rider Expedition Atacama: Cuesta Del Lipan]

- A study of "IEC 61508 and IEC 61511 assessments – some lessons learned" reports on 16 companies' certification attempts [Lloyd and Reeve 2009]:
  - completed: 3
  - failed – given up: 9
  - in progress (at time of study): 4

  → only 19% of projects successfully certified

Gap analysis of failed and incomplete certification attempts

Software Lifecycle Phase	% Compl.
A1 Software Requirements Specification	33
A2 Software Architecture Design	25
A3 Software Design and Development - Support Tools and Programming Language	45
A4 Software Detailed Design	25
A5 Software Module Testing and Integration	31
A6 PE Integration (Hardware + Software)	0
A7 Software Validation	17
A8 Modification	20
A9 Software Verification	20
Overall average compliance	24

[Lloyd, M. H., and P. J. Reeve: IEC 61508 and IEC 61511 assessments-some lessons learned. (2009): 21-21.]

ISO26262: Certification of Items and SEooCs

- Certification of “functions” (safety items)
  - allows certification of functions, e.g., airbag, steering, ESP, including sensors, ECUs, actors
  - function shall be a meaningful functional entity
- Certification of safety elements out of context (SEooC)
  - allows certification of basic elements, e.g., OS, processor, independent of a specific function it realizes
  - these certifications are not sufficient to demonstrate a certified function
  - allows pre-qualification of components for safety critical functions and systems
  - integration of certified elements (SEooC) into functions or systems requires a new certification of the application

ISO 26262: Safety Audits and Assessments

- ISO 26262 **does not prescribe a certification**, but **confirmation measures** for items of ASIL B–D
  - **functional safety audit** evaluates the developed product
  - **functional safety assessment** evaluates the implemented processes
  - depending on item's ASIL: measures shall be conducted by **persons with a certain degree of independence** → the higher the risk the higher the required independence

Confirmation measures	Degree of independency[a] applies to ASIL				Scope
	A	B	C	D	
Functional safety audit in accordance with 6.4.8 Independence with regard to the developers of the item and project management	—	I0	I2	I3	Applies to the highest ASIL among the safety goals of the item
Functional safety assessment in accordance with 6.4.9 Independence with regard to the developers of the item and project management	—	I0	I2	I3	Applies to the highest ASIL among the safety goals of the item

[a] The notations are defined as follows:
— —: no requirement and no recommendation for or against regarding this confirmation measure;
— I0: the confirmation measure should be performed; however, if the confirmation measure is performed, it shall be performed by a different person;
— I1: the confirmation measure shall be performed, by a different person;
— I2: the confirmation measure shall be performed, by a person from a different team, i.e. not reporting to the same direct superior;
— I3: the confirmation measure shall be performed, by a person from a different department or organization, i.e. independent from the department responsible for the considered work product(s) regarding management, resources and release authority.
[b] A software tool development is outside the item's safety lifecycle whereas the qualification of such a tool is an activity of the safety lifecycle.

[ISO 26262]

Tool Classification and Qualification

Development Tools

- Tool Usage
  - pro: potentially improves system safety by automating activities it performs and by performing functions predictably that may be prone to human error
  - con: tool errors may have negative impact if the tool performs its functions inadequately

→ Tool Qualification: processes prescribed in functional safety standards to gain confidence in the utilized tools

- Software tool according ISO 26262: computer program used in the development of an item or element (not limited to SW development, also HW and system)
- Survey on tool landscape: 12(min)–50(typical)–400(max) small helpers, customization scripts, in-house, COTS tools per project [Conrad and Fey 2014]

[Conrad, Mirko, and Ines Fey: Effort and Efficacy of Tool Classification and Qualification. Tagungsband des Dagstuhl-Workshops, 2014]

Tool Qualification

- **Objectives**
  - mitigate errors introduced by development tools or undetected by verification tools
  - ensure integrity of tool functionality
  - obtain certification credits for using software tools within the context of a specific project

- **Level of qualification rigor** depends on tool category and/or necessary confidence
  - the higher the risk of the tool error adversely affecting safety goals, the higher the rigor required for tool qualification

IEC 61508 and EN 50128: Tool Classification

- "The criteria that shall be considered when selecting a suitable modeling or programming language" (and its tooling):
  - T1: "generates no outputs which can directly or indirectly contribute to the executable code or data..." e.g., LATEX used for document preparation
  - T2: "supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors..." e.g., automatic module test generator (Klee)
  - T3: "generates outputs which can **directly or indirectly contribute to the executable code**..." e.g., compiler, Make

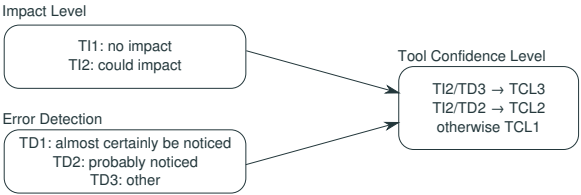
→ "For each tool in class T3, evidence shall be available that the tool conforms to its specification..."

ISO 26262: Tool Classification

- **2-stage process** to determine tool confidence levels (possibly using a FMEA)
- **Example classifications**:

code generator	TCL1…TCL2
guideline checker	TCL2
run-time error detection	TCL2…TCL3
configuration management	TCL2

[Löw, Peter, Roland Pabst, and Erwin Petry: **Funktionale Sicherheit in der Praxis: Anwendung von DIN EN 61508 und ISO/DIS 26262 bei der Entwicklung von Serienprodukten**. dpunkt. verlag, 2011.]



top: [Hobbs, C. (2019). **Embedded Software Development for Safety-Critical Systems (2nd ed.)**. CRC Press.] according [ISO26262], bottom: [Tradeindia: **Iti Plumbing Tool**, 2022]

ISO26262 Tool Qualification Measures

- **TCL1 classified tools**: no need for qualification measures
- **TCL2 and TCL3 classified tools**: demonstrate appropriateness for usage with required level of confidence
- Typically only 2–5% of the utilized tools are TCL2 and TCL3 [Hamann et al. 2011]

Establishing confidence in the use of software tools

Method	TCL 2				TCL 3			
	ASIL A	ASIL B	ASIL C	ASIL D	ASIL A	ASIL B	ASIL C	ASIL D
1a Increased confidence from use	++	++	++	+	++	++	+	+
1b Evaluation of the tool development process	++	++	++	+	++	++	+	+
1c Validation of the software tool	+	+	+	++	+	+	++	++
1d Development in compliance with a safety standard	+	+	+	++	+	+	++	++

derived from [ISO26262-8:11]

[Hamann, Reinhold, et al.: **ISO 26262 release just ahead: remaining problems and proposals for solutions**. No. 2011-01-1000. SAE Technical Paper, 2011.]

Tool Validation: Compiler

- Check generated code and syntax checking.
- Confirm compiler's actions with randomly-generated programs:
  - Generating programs with unknown results and compiling them with different compilers
  - Generating programs with known result

→ "Every compiler we tested was found to crash and also to silently generate wrong code when presented with valid input." [Yang et al. 2011]

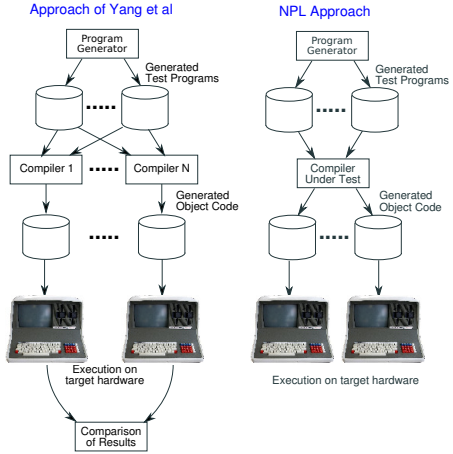
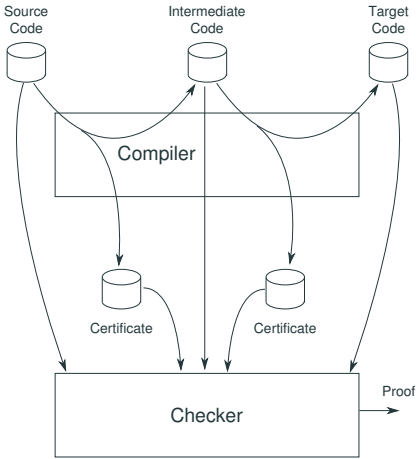


figure: [Hobbs, C. (2019). Embedded Software Development for Safety-Critical Systems (2nd ed.). CRC Press.], approach: [Yang, Xuejun, et al.: Finding and understanding bugs in C compilers. 32nd ACM SIGPLAN, 2011.]

Tool Validation: Compilation

- Confirm at least the compiler and code generator by proof techniques



[Hobbs, C. (2019). Embedded Software Development for Safety-Critical Systems (2nd ed.). CRC Press.]

Replacing a Human Activity by a Tool

- [EN 50128]:
  - "When tools are being used as a replacement for manual operations, the **evidence of the integrity** of tools output can be adduced by the **same process steps as if the output was done in manual operation**."

[KhichdiOnline.com: **World's first humanoid robot staffed hotel set to open in Japan**, 2015]

The Safety Case

The Safety Case

- A Safety Case ...

  ...is a structured argument, supported by a body of evidence that provides a compelling, comprehensive and valid case that a system is safe for a given application in a given operating environment.

[Defence Standard 00-56 Issue 4 (Part 1): **Safety Management Requirements for Defence Systems**. UK Ministry of Defence. p. 17]

Discussion of the Safety Case Definition

- “Structured and Comprehensive”
  - an unstructured pile of paper only works for very small systems
  - structure has to be implicit in the format chosen
- “Evidence”
  - the Safety Case consists of an argument AND the associated evidence (artifacts)
- “Given Application and Given Operating Environment”
  - no system is safe in all environments
  - no system is safe for all applications
  - we must make our claims completely explicit

The Safety Case Must Be Convincing

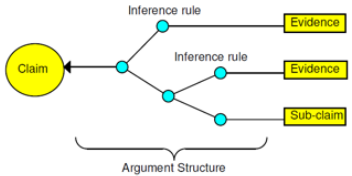
- You need to convince:
  - yourself,
  - your management,
  - your customer, and
  - your auditor

... that you meet the claims that you have made.

[www.bolshoyvopros.ru]

Creating the Safety Case: Method

1. Make explicit claims
   - e.g., if condition *Z* holds, this system will do *X* with a probability *Y*
2. Structure the argument (ignoring evidence for now)
   - e.g., Goal Structuring Notation (GSN) or Bayesian Belief Networks (BBNs)
   - using the same representation for failure analysis and safety case allows integration, e.g., fault tree or BBN
3. Agree the argument structure with readers
   - Would you be convinced by this argument?
4. Add the evidence and confidence in it
5. Perform a sensitivity analysis
   - Where is it worth spending more time increasing our confidence in the evidence?



[Bishop, Peter, and Robin Bloomfield: **A Methodology for Safety Case Development**. Safety-Critical Systems Symposium, Birmingham, UK. 1998.]

EN 50129: Safety Case Structure

- Recommended structure:
  - Part 1: System definition (or sub-system/equipment)

> Precise definition of the system.

EN 50129: Safety Case Structure

- Recommended structure:
  - Part 1: System definition (or sub-system/equipment)
  - Part 2: Quality management report

organizational structure	quality planning and procedures
specification of requirements	design control
design verification and reviews	application engineering
procurement and manufacture	product identification/traceability
handling and storage	inspection and testing
non-conformance and corrective action	packaging and delivery
installation and commissioning	operation and maintenance
quality monitoring and feedback	documentation and records
configuration mgmt./change control	personnel competency/training
quality audits and follow-up	decommissioning and disposal

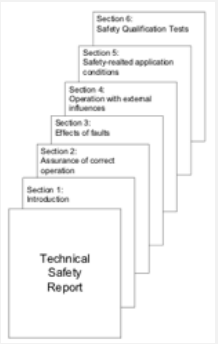
EN 50129: Safety Case Structure

- Recommended structure:
  - **Part 1**: System definition (or sub-system/equipment)
  - **Part 2**: Quality management report
  - **Part 3**: Safety management report

Evidence that a **safety management system** is applied, including **hazard and risk analysis**. Needs to show that **personnel are competent**, have appropriate experience and are correctly trained.

EN 50129: Safety Case Structure

- Recommended structure:
  - Part 1: System definition (or sub-system/equipment)
  - Part 2: Quality management report
  - Part 3: Safety management report
  - Part 4: Technical safety report



[Hobbs, C. (2019). Embedded Software Development for Safety-Critical Systems (2nd ed.). CRC Press.]

EN 50129: Safety Case Structure

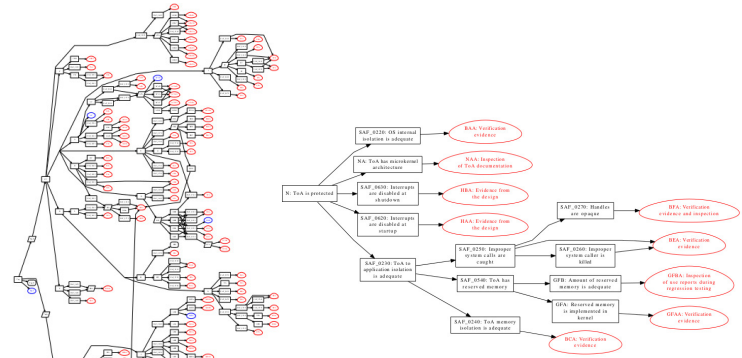
- Recommended structure:
  - Part 1: System definition (or sub-system/equipment)
  - Part 2: Quality management report
  - Part 3: Safety management report
  - Part 4: Technical safety report
  - Part 5: Related safety cases

EN 50129: Safety Case Structure

- **Recommended structure**:
  - **Part 1**: System definition (or sub-system/equipment)
  - **Part 2**: Quality management report
  - **Part 3**: Safety management report
  - **Part 4**: Technical safety report
  - **Part 5**: Related safety cases
  - **Part 6**: Conclusion

> "This shall **summarize the evidence** presented in the previous parts of the safety case and **argue** that the relevant system/ sub-system/ equipment is **adequately safe**, subject to compliance with the specified application conditions."

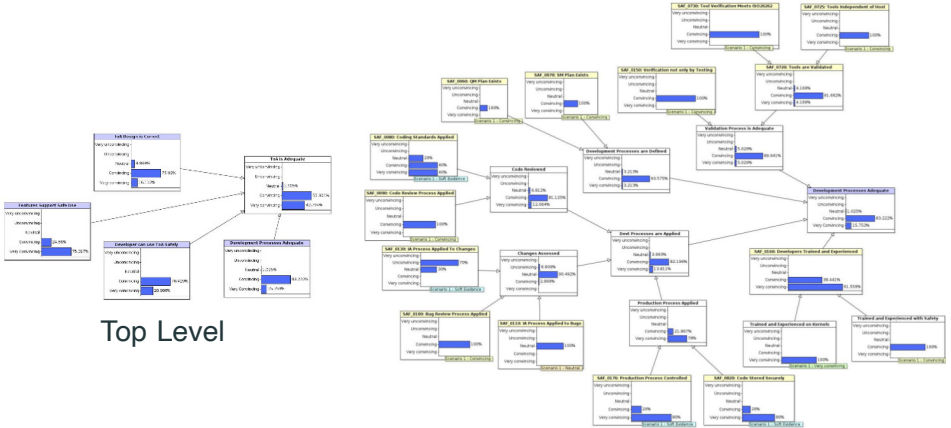
Example 1: Part of a Safety Case using GSN



Detailed subsection of the goal graph

[Hobbs, C. (2019). **Embedded Software Development for Safety-Critical Systems (2nd ed.)**. CRC Press.]

Example 2: Part of a Safety Case using BBN

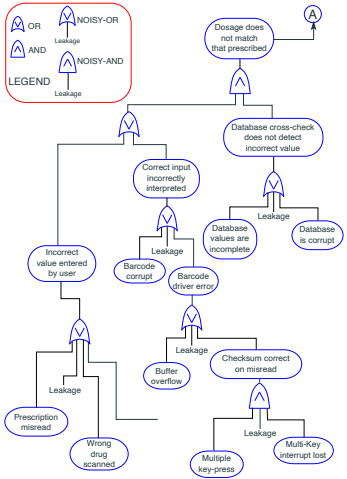


[Hobbs, C. (2019). Embedded Software Development for Safety-Critical Systems (2nd ed.). CRC Press.]

Example 3: Infusion Device Safety Case (1/2)

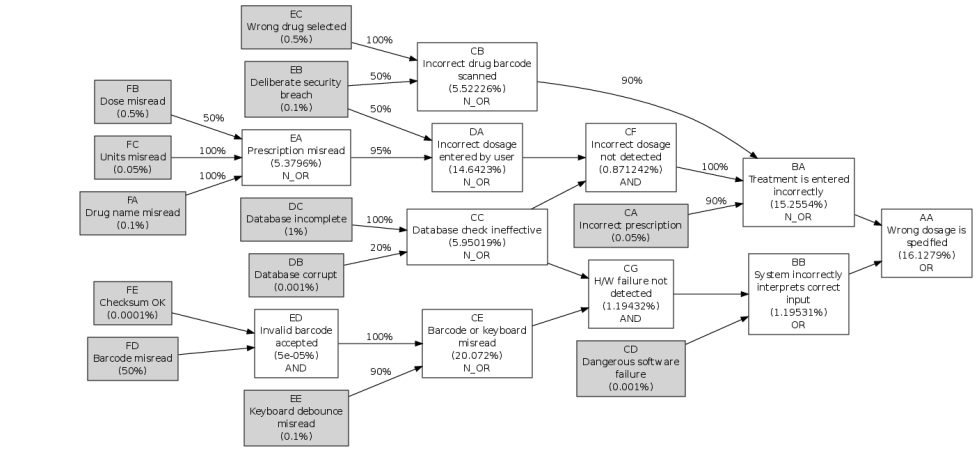
[Ravie Lakshmanan: **B.Braun Infusomat Pumps Could Let Attackers Remotely Alter Medication Dosages**. The Hacker News, 2021.]

- Small part of a safety case for demonstrating the steps taken to ensure that the correct dosage is delivered by an infusion device



[Hobbs, C. (2019). **Embedded Software Development for Safety-Critical Systems (2nd ed.)**. CRC Press.]

Example 3: Infusion Device Safety Case (2/2)



[Hobbs, C. (2019). **Embedded Software Development for Safety-Critical Systems (2nd ed.)**. CRC Press.]

Improper Safety Case: Lessons Learned (1/2)

- Sep 2nd, 2006: a Royal Air Force Hawker Siddeley Nimrod suffered an in-flight fire and subsequently crashed in Kandahar, Afghanistan, killing all fourteen crew members on board [Wikipedia]

- Aug 16th, 2004: the position had clearly become critical. **Only 50 out of 104 hazard** Pro-formas had been done. [...] There were, however, only two weeks to go until the Customer Acceptance Conference. At a review meeting that morning, Chris Lowe said there was to be a “final push” and the aim was for “end of the week completion”. [...] Mech systems was told “to come up with a plan” to **complete the remaining hazards “by the end of the week”**. This was not realistic.

[Charles Haddon-Cave QC: **The Nimrod Review**, 2009.]

Improper Safety Case: Lessons Learned (2/2)

- Review on the 2006 Nimrod crash [Wikipedia]:
  - Unfortunately, the Nimrod Safety Case was a **lamentable job** from start to finish. It was riddled with errors. It **missed the key dangers**. Its production is a story of incompetence, complacency, and cynicism. The best opportunity to prevent the accident to XV230 was, tragically, lost. … the task of drawing up the Safety Case became essentially a paperwork and 'tickbox' exercise.

→ Safety case must be an integral part of development and maintenance

image: [Charles Haddon-Cave QC: **The Nimrod Review**, 2009.],
quote: [**Nimrod Crash**, Wikipedia, 2022]

Confirmation Bias as Problem for Safety Cases (1/2)

- Given the sequence below, what could be the rule for generating the next number:
  - **2, 4, 6, 8, 10, ...**
  - (class activity:) make guesses and we will tell you whether you are right or wrong

- The Rule: **the next number must be larger than the previous one**.

→ Note how guesses were more useful if they did not support your preconceived idea
→ but: it is difficult to break away from confirmation bias

Confirmation Bias as Problem for Safety Cases (2/2)

Nancy Leveson has pointed out (in numerous places) that **humans are subject to confirmation bias**. This means that we are **unable to produce a genuine Safety Case** → believing in a Safety Case is therefore fundamentally flawed.

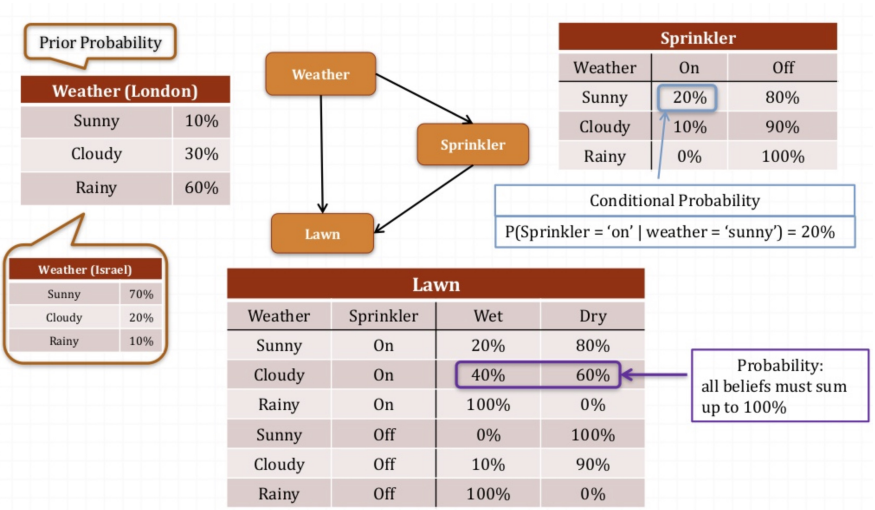
[Nancy Leveson, MIT]

Bayesian Belief Networks

Bayesian Belief Network (BBN)

- A probabilistic graphical model (more specifically DAG)
- **Nodes** are features
  - Each has a set of possible states (also: parameters, values)
  - Example: weather ={sunny, cloudy, rainy}
  - States occur with certain probability
  - Example: a fair coin has two possible values: {heads, tails}, each occurs with 50% probability
- **Edges** relations between features
  - Direction of edges indicates **causality**
- **Belief** probabilities of occurring states
- All beliefs of all possible states of a node are gathered in a single **Conditional Probability Table (CPT)**

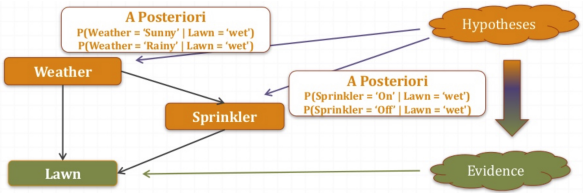
Example: A Simple Bayesian Network



[Gilad Barkan: Bayesian Belief Networks for dummies. Slideshare, 2015.]

BBN Inferencing

- After CPT calculation (training), we can answer questions like:
  - Given a rainy weather, is the lawn wet? (trivial answer – not interesting)
  - Given that the lawn is wet, what could be the reason for that? (the real action begins – cool)
    - → rainy weather or turned-on sprinkler?
- Decision rule example: maximum a posteriori probability (MAP)
  - For P(Weather='rainy' |Lawn='wet') = 0.1; P(Sprinkler='On' |Lawn='wet') = 0.08 choose Weather = 'rainy', i.e., given the lawn is wet it is more probable that a rainy weather caused it rather than a turned-on sprinkler



[Gilad Barkan: Bayesian Belief Networks for dummies. Slideshare, 2015.]

Advanced BBNs: Noisy-OR Conjunction (1/2)

- Why Noisy-OR?
- Because it is rare:
  - ...that each input to an OR gate would, by itself, be 100% certain to cause the output.
  - ...for us to be sure that we have identified every cause of a particular output.

[Roger Hargreaves: **Mr. Noisy**. 1976]

Advanced BBNs: Noisy-OR Conjunction (2/2)

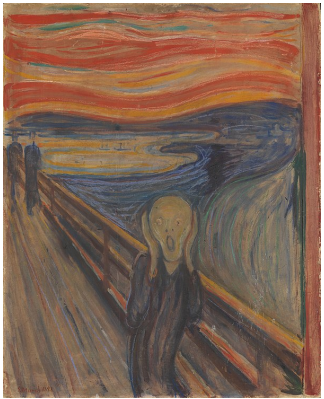
- Given a variable X that depends on $Y_1, Y_2, \ldots, Y_N$, probabilities are defined for each $Y_i : p_i = p(X \mid Y_i, \{\bar{Y}_j\}_{j=1, j \neq i}^{N})$.
- The distribution of X conditional on the Y_i is then be defined as $p(X \mid Y_1, \ldots, Y_N) = 1\Pi_{i:Y_i}(1p_i)$.
- **This does not take into account leakage** – the level at which the Y_i, are the only variables on which X depends.

- Leakage can be incorporated by defining **k** to be the **level of confidence in** X when all the Y_i are false: $k = p(X \mid \bar{Y}_1, \bar{Y}_2, \ldots, \bar{Y}_N)$.

→ The distribution of X on the Y_i is then: $p(X \mid Y_1, \ldots, Y_N) = 1[(1k)\Pi_{i_{Y_i}}(1p_i)]$

Advanced BBNs: Other Noisy Conjunctions

- Noisy-AND: need A and B and C but A, B and C don't weigh equally and, even if they're all true, the result may not be true.
- Noisy-MAX: each of A, B and C contributes an amount and the result is the largest of the contributions.
- Noisy-SUM: each of A, B and C contributes an amount and the result is the sum of the contributions.



[Edvard Munch: The Scream, Wikipedia, 2022.]

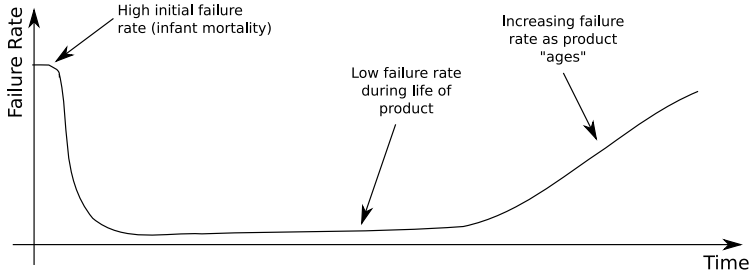
Measuring Software Dependability

Modeling Software Dependability

- ...to create a model we need to know how often the software will fail. How can we assess that?
- For assessing hardware failure we have books of failure rates (HRD5 from British Telecom, SR-322 from Telcordia, CNET 93 from France Telecom, RDF 2000, etc.). Where are these books for software?

Measuring Hardware vs. Software Dependability

- …hardware failure can be discussed statistically; software failure can't
- …hardware failures follow the traditional bathtub curve; software failures don't
- …the techniques for predicting hardware failure are proven and repeatable; such techniques don't exist for software



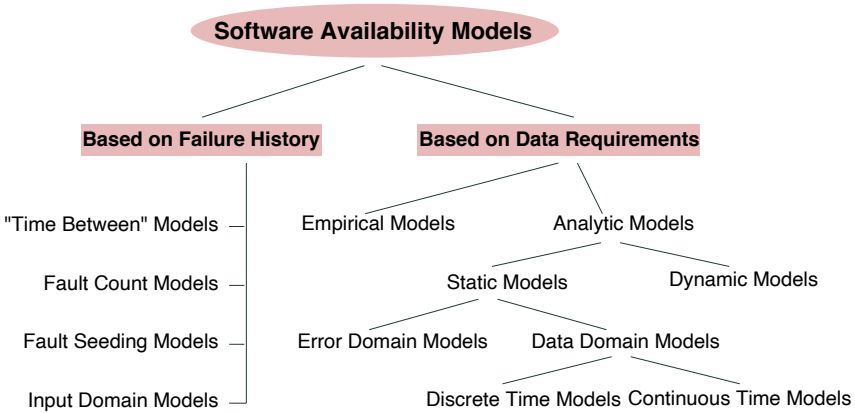
[Hobbs, C. (2019). **Embedded Software Development for Safety-Critical Systems (2nd ed.)**. CRC Press.]

Prediction vs. Estimation of Analytical Models

	Prediction	Estimation
data source	historical	current project
when used	concept phase	verification phase
time frame	future	current progress

[Tookapic, pexels, 2022]

Taxonomy of Software Availability Models



[Shanmugam, Latha, and Lilly Florence: **An overview of software reliability models**. International Journal of Advanced Research in Computer Science and Software Engineering 2.10 (2012): 36-42.]

Estimating Software Dependability

- For estimating the dependability of a software, we need to know:
  - How many faults? (e.g., The system contains 1,324 faults)
  - The failure rate per fault (e.g., On average, a fault leads to a failure every 1,872 hours)
    - Most faults do not lead to failures, e.g., using $\mathrm{char}\ \mathrm{x}[20]$; rather than $\mathrm{char}\ \mathrm{x}[10]$; likely causes no failure
- Where do hardware information come from? → from history!

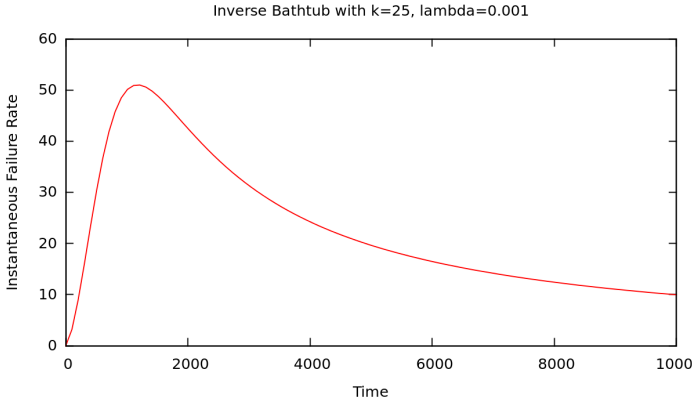
→ Historic fault information for software

- used programming language
- information for used COTS and SOUP
- code metrics, e.g., code complexity
- module history, e.g.,
  - the number of changes
  - the people who made the changes
  - the number of people who have made changes
  - the days (and time of day) on which the changes were made

Software Failure Rates: the Inverse Bathtub

- Failure rate per fault
  e.g., the inverse bathtub:

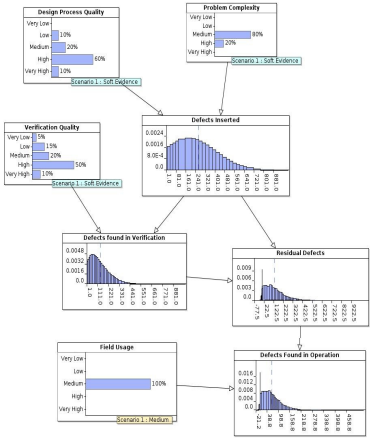
$$N = a\frac{(\lambda t)^{\kappa-1}}{1+(\lambda t)^{\kappa}}$$



[Hobbs, C. (2019). Embedded Software Development for Safety-Critical Systems (2nd ed.). CRC Press.]

The Bayesian Approach

- A problem that remains is that code that is not used appears to have no faults



[Fenton, N., Martin Neil, and D. Marquez: **Using Bayesian networks to predict software defects and reliability**. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability 222.4 (2008): 701-712.]

The Importance of Record Keeping

- All of these estimations depend on excellent record keeping: bug reports, code modification counts and extents, etc.

[Alyson Krueger: **6 Ways Technology Is Improving Healthcare**. Insider, 2010.]

Historic Data for New Developments

- Potential problem: a totally new product has no history
  - Are all of the components (OS, database, middleware, etc.) totally new? → Is there history available for them?
  - Is the product based on other products produced by the same company? → Can some history be brought forward?
  - Is the development similar to an open source project from which information is available?
  - Does an industry standard model apply – lots of commercial companies with models?
- Local “history” must be gathered from the first line of code and be used to augment these.

Summary

Summary

- The Safety Case is designed primarily for you
  - *"The report [Oil Spill Commission] makes the surprisingly candid admission that* ***no amount of government regulation will stop another Deepwater Horizon****. Future safe operating must spring from the* ***safety culture*** *of the organizations doing the work." (The word 'culture' is used 17 times.)*
- A Safety Case needs to be structured: the argument and the evidence should be separated
  - *"We have an ethical duty to come out of our mathematical sandboxes and take more social responsibility for the systems we build – even if this means career threatening conflict with a powerful boss. Knowledge is the traditional currency of engineering, but we must also deal in belief." [Les Chambers, on the Deep Water Horizon disaster]*
- Techniques exist to estimate software dependability. They must be supported by good historical information

Questions?