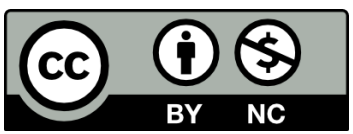
8 ARCHITECTURAL RISKS

course "software requirements and architecture"

Oct 2023



contents

contents

failure

> The concept of failure is central to the design process, and it is by thinking in terms of obviating failure that successful designs are achieved.
>
> Henry Petroski, 1994

- Avoiding failure is central to all engineering.
- One can use architecture techniques to mitigate the risks.
- Developers discard the designs that are predestined to fail and prefer those with low risk of failure.
- Building successful software means anticipating possible failures.
- how much design should developers do?
  - no up-front design: just write code
  - use a yardstick: spend 10% of the time on design
  - build a documentation package
  - ad hoc: decide on the spot what to do

approach

- The risk-driven approach help developers to build quality software quickly and at low cost.
- Architects have more architecture techniques than they can afford to apply.
- The risk-driven approach helps answer two questions:
  - how much software architecture work should be done?
  - which techniques should be used?
- The risk-driven approach guides architects to apply a minimal set of architecture techniques to reduce their most pressing risks.

approach

- The risk-driven approach follows three steps:
  1. identify and prioritize risks
  2. select and apply a set of techniques
  3. evaluate risk reduction
- Architects do NOT want to:
  - waste time on low-impact techniques
  - ignore project-threatening risks
- Architects want to:
  - build quality systems by taking a path that spends their time most effectively
  - address risks by applying design techniques, only when they are motivated by risks

approach

- Different developers perceive risks differently.
- They choose different techniques.
- The risk-driven model has the useful property that it yields arguments that can be evaluated.
- An example argument:

> We identified A, B, and C as risks, with B being primary. We spent time applying techniques X and Y because we believed they would help us reduce the risk of B. We evaluated the resulting design and decided that we had sufficiently mitigated the risk of B, so we proceeded on to coding.

contents

risk = failure x impact

- In engineering, risk is commonly defined as the chance of failure multiplied by the impact of that failure:
  risk = probability of failure x impact
- The probability of failure and the impact are not certain, since they are difficult to measure precisely.
- A risk can exist (i.e., one can perceive it) even if its consequences do not materialize.
- A risk can be stated categorically, often as the lack of a needed quality attribute.
- It is better to describe risks such that one can later test if they have been mitigated.

kinds of risks

- Describe each risk of failure as a testable failure scenario.
- Example: "During peak loads, customers experience user interface latencies greater than five seconds".
- Projects face many different kinds of risks.
- Each one pays attention to the risks of his/her specialty.
- Misunderstood quality attributes are a common risk.

project management risks	**software engineering risks**
lead developer hit by bus	the server may not scale to 800 users
customer needs not understood	parsing of the alert messages are buggy
VP hates the product owner	the system is working now, but if we change anything it may collapse

prototypical risks

After working in a domain for a while, developers notice risks that are common to most projects in that domain.

project domain	**prototypical risks**
IT	complex, poorly understood problem unsure the real problem is being solved integration with existing software domain knowledge scattered modifiability
Systems	performance, reliability, security concurrency composition
Web	security scalability developer productivity / expressibility

risks prioritisation

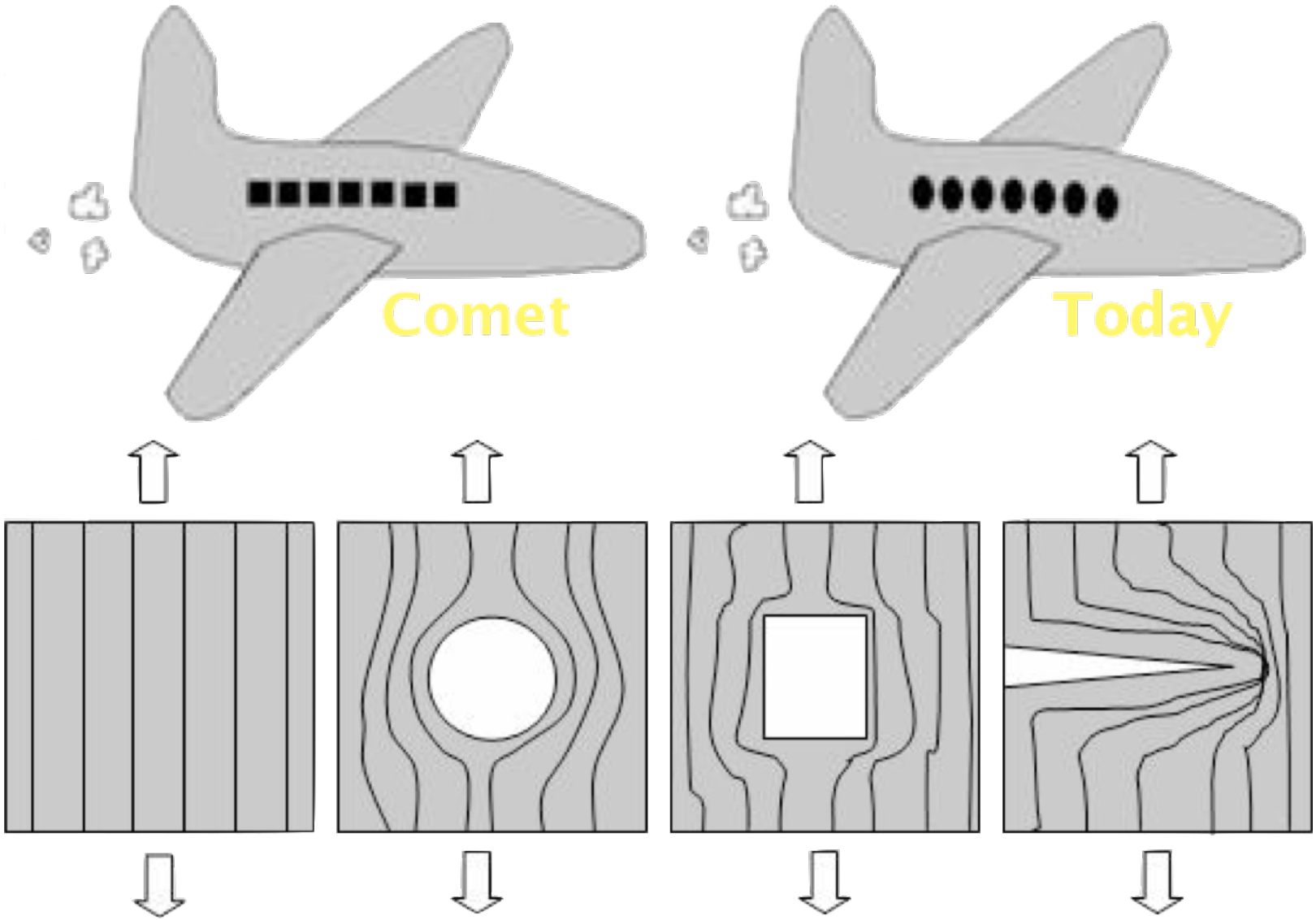
Likelihood/ Impact	Nearly No	Minor	Moderate	Major	Disaster
Will Happen	Medium	High	High	Extreme	Extreme
Most likely	Medium	Medium	High	High	Extreme
Possible	Low	Medium	Medium	High	Extreme
Unlikely	Low	Medium	Medium	Medium	High
Rare	Low	Low	Medium	Medium	High

- Not all risks are equal, so they can be prioritized.
- Most teams prioritize risks by discussing the priorities amongst themselves.
- The team's perception of risks may not be the same as the stakeholders' one.
- Some technical risks, such as platform choices, may not be easily assessed by the stakeholders.
- Risks can be categorized on two dimensions:
  - their priority to stakeholders (business)
  - their perceived difficulty by developers (engineering)

risk mitigation

- Once one identifies the risks, some techniques can be applied to reduce their impact.
- Techniques go from pure analyses (calculate stresses) to pure solutions (flying buttress on churches).

risk mitigation



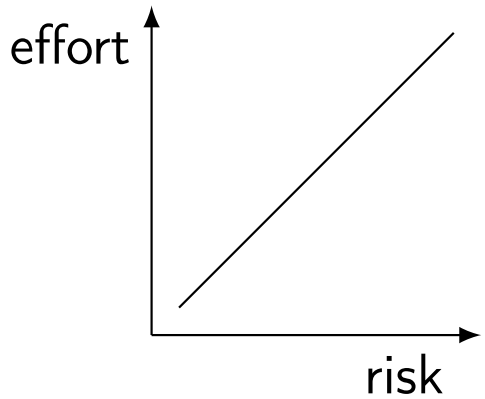
risk techniques

software engineering	**other engineering fields**
design patterns	stress calculation
domain modeling	breaking point testing
throughput modeling	thermal analysis
security analysis	reliability testing
prototyping	prototyping

risk techniques

- One needs to be able to explicitly state how to choose techniques in response to risks.
- Create a handbook with entries:
  if we have <a risk>, consider <a technique> to reduce it
- A particular technique is good at reducing some risks but not others.
- Ideally, there would be a single technique to address every known risk.
- Some risks can be mitigated by multiple techniques, while other risks require techniques to be invented on the fly.

effort proportional to risks



- The risk-driven approach uses this principle: architecture efforts should be proportional with the risk of failure
- Whenever developers are unconcerned about security risks, they spend no time on security design.
- When performance is a critical risk, it should be addressed until an acceptable level is reached.
- When one applies the risk-driven approach, one only designs the areas where failure risks are perceived.
- Often, applying a design technique means building a model of some kind.

effort proportional to risks

The effort spent on designing the architecture should be commensurate with the risks faced by the project.

effort proportional to risks

- The architecture model will be detailed in some areas and sketchy, or even non-existent, in others.
- Models are an intermediate product.
- One can stop working on the models once he is convinced that the architecture is suitable for addressing the risks.
- The risk-driven approach facilitates the decisions, but it cannot make judgments.
- It identifies prioritized risks and corresponding techniques and guides on asking the right questions about the design work.

contents

three styles

- evolutionary design (No DUF - NDUF)
- planned design (Big DUF - BDUF)
- minimal planned design (Enough DUF - EDUF / Little DUF - LDUF)

evolutionary design - NDUF

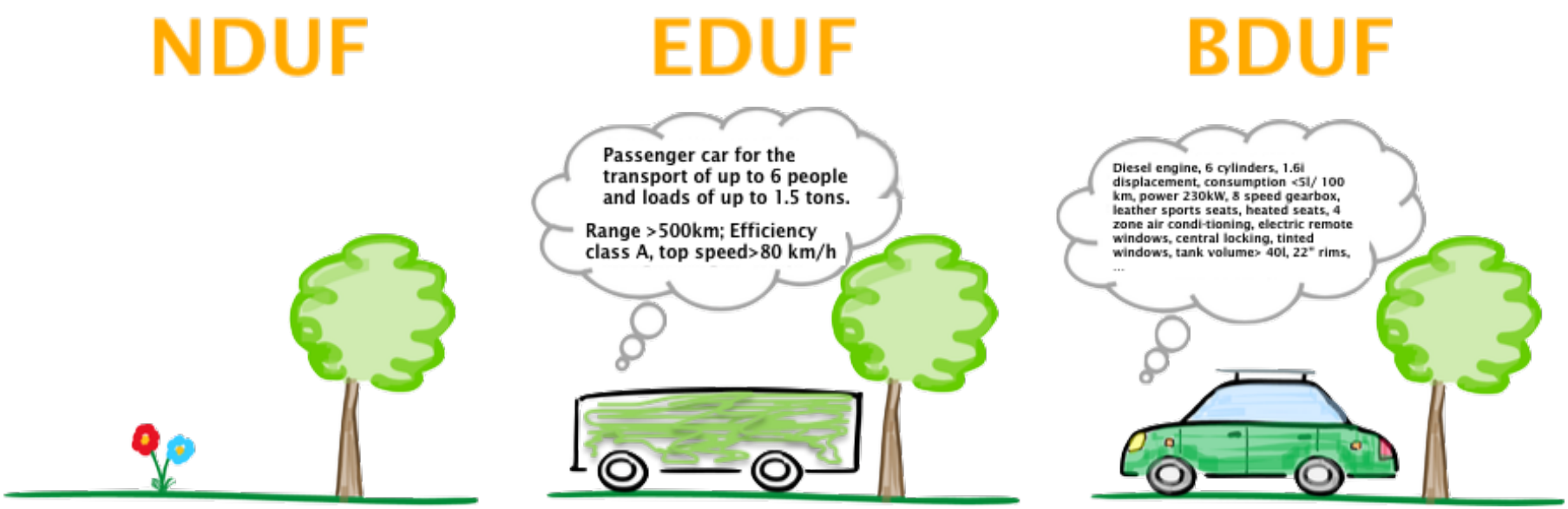
- In evolutionary design, the design of the system grows as the system is implemented.
- Recent trends in software processes address most of the shortcomings of evolutionary design.
- Agile practices work against the chaos:
  - **refactoring** cleans up the uncoordinated local designs
  - **test-driven development** ensures that changes to the system do not cause it to lose or break existing functionality
  - **continuous integration** provides the entire team with the same codebase

planned design - BDUF

- In planned design, architecture is detailed before construction.
- Analogies with bridges are used, since bridge construction rarely begins before the design is ended.
- Few people advocate doing planned design for an entire software system.
- Planned architecture design is useful when many teams working in parallel share an architecture.
- Even in planned design, an architecture should rarely be 100% complete before prototyping or coding.
- The design can be perfected with feedback from code.

planned design - BDUF

BDUF is dumb, but doing NDUF is even dumber. Dave Thomas



minimal planned design - LDUF

- An hybrid style is minimal planned design.
- One can balance planned and evolutionary design.
- One way is to do some initial planned design to ensure that the architecture handles the biggest risks.
- Future changes to requirements can be handled through local design, or with evolutionary design.
- The idea is to perform architecture-focused design to set up an architecture that handles the biggest risks.
- One has more freedom in other design decisions.

styles of design

- The risk-driven approach is compatible with the three design styles.
- All of these design styles agree that design should happen at some point.
- In BDUF, that time is up-front.
- In NDUF, it means doing architecture design during development, whenever a risk looms sufficiently large.
- Applying it to LDUF is a combination of the other two.

Summary

- The risk-driven approach guides developers to apply a minimal set of architecture techniques to reduce their most pressing risks.
- Architecture efforts should be proportional with the risk of failure.
- One can stop working on the models once the architecture is suitable for addressing the risks.
- Few people advocate doing planned design (BDUF) for an entire software system.

bibliography

- Fairbanks G; *Just-enough software architecture: A risk-driven approach*, Marshall & Brainerd, 2010. [chapter 3]