Problem Analysis for Benefit Delivery

CS4505, Software Architecture, Lecture 2,

September 2024, Arie van Deursen

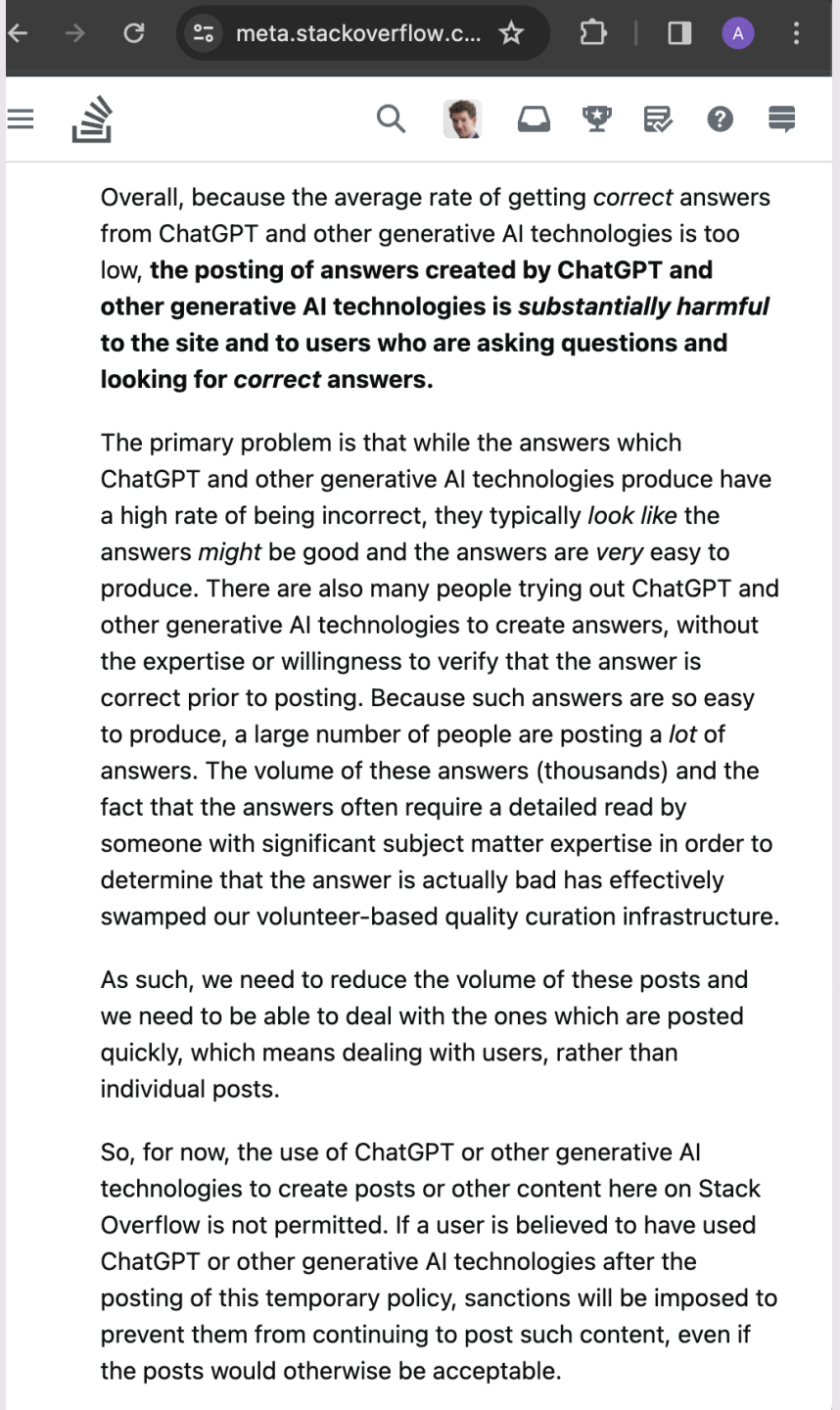
AI-Augmented Architecting?

- Idea generation & creativity
- Writing assistant
- Coding assistant
- Access to common knowledge
  - Analysis of pros / cons, tradeoffs
  - General domain knowledge (verify!!)
- Design space exploration?

StackOverflow Generative AI Policy

All use of generative AI
(e.g., ChatGPT and other LLMs)
is banned when posting content
on Stack Overflow.

This includes "asking" the question
to an AI generator
then copy-pasting its output
as well as using an AI generator
to "reword" your answers.



Overall, because the average rate of getting *correct* answers from ChatGPT and other generative AI technologies is too low, **the posting of answers created by ChatGPT and other generative AI technologies is *substantially harmful* to the site and to users who are asking questions and looking for *correct* answers.**

The primary problem is that while the answers which ChatGPT and other generative AI technologies produce have a high rate of being incorrect, they typically *look like* the answers *might* be good and the answers are *very* easy to produce. There are also many people trying out ChatGPT and other generative AI technologies to create answers, without the expertise or willingness to verify that the answer is correct prior to posting. Because such answers are so easy to produce, a large number of people are posting a *lot* of answers. The volume of these answers (thousands) and the fact that the answers often require a detailed read by someone with significant subject matter expertise in order to determine that the answer is actually bad has effectively swamped our volunteer-based quality curation infrastructure.

As such, we need to reduce the volume of these posts and we need to be able to deal with the ones which are posted quickly, which means dealing with users, rather than individual posts.

So, for now, the use of ChatGPT or other generative AI technologies to create posts or other content here on Stack Overflow is not permitted. If a user is believed to have used ChatGPT or other generative AI technologies after the posting of this temporary policy, sanctions will be imposed to prevent them from continuing to post such content, even if the posts would otherwise be acceptable.

Beware of ChatGPT's Limitations

It will *hallucinate*: It will invent likely answers, with no guarantee any of it is correct.

Answers as useful as the prompt you engineer: This takes effort too.

Answers may bias you, and limit your incentive to look beyond them

Easier to distinguish right from wrong if you're actually an expert

You'll need to turn vague, wordy, and abstract writing into crisp, to the point, and compelling line of reasoning

ChatGPT / Generative AI CS4505 Usage Constraints

- Usage not encouraged, but permitted
  - Idea generation, writing/coding assistant, common knowledge
- You must **verify** any outcomes
  - Never trust ChatGPT, always check answers!
  - Always identify and reference reliable knowledge sources (!)
- All usage must be **documented**
  - Clearly explain usage and verification procedure (in appendix)
- Great if it speeds you up: Different way to use 140h (e.g. PoC)

Responsible Architecting

- The architect is a decision maker
- Making a decision means *being accountable*
- This accountability can never be replaced by AI

Problem Analysis for Benefit Delivery

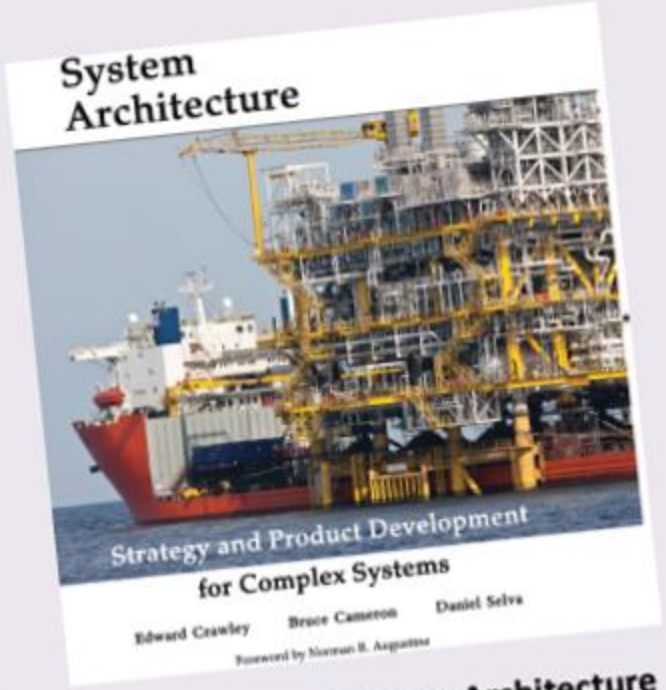
CS4505, Software Architecture, Lecture 2,

September 2024, Arie van Deursen

LO1:
Structure The Problem Space

- Identify the primary externally delivered value-related function
- Identify architecturally relevant properties and scenarios
- Resolve upstream *ambiguity*
- Reconcile conflicting stakeholder needs
- Highlight technical limits *and opportunities*

Principle of Benefit Delivery



26 Principles of System Architecture

Good architectures deliver benefit,
first and foremost,

built on the primary externally delivered
function of the systems

by focusing on
the emergence of functions,
and their delivery

across the system boundary
at an interface.

Today's Focus: *Benefit*

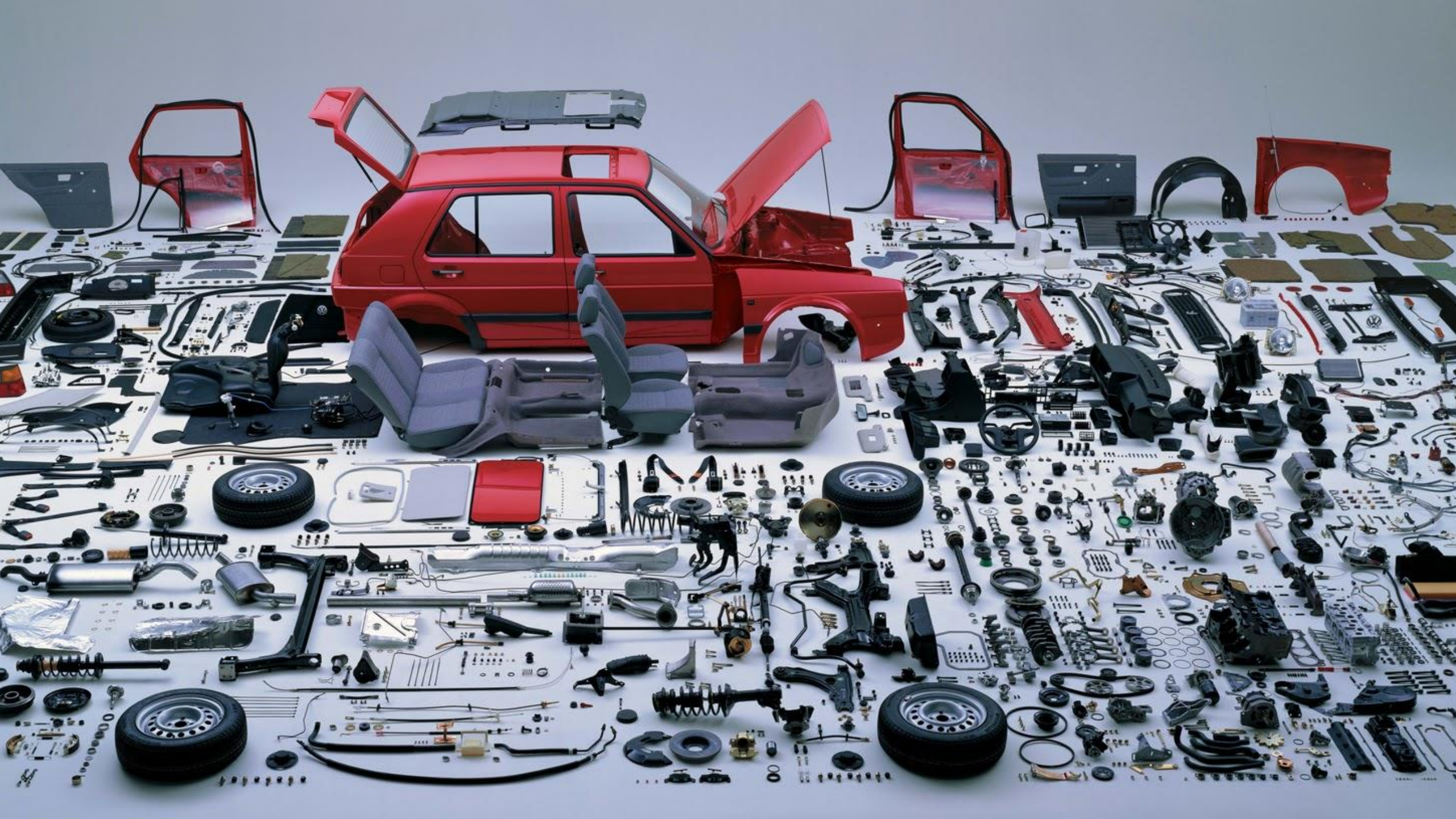
- What is it (the system, its function) that we really need?
- What is the benefit that the intended system should deliver?
  - What is the primary externally delivered function?
  - How does this primary function emerge from "sub functions"?
- Who will benefit from the system?
- How can we sequence the delivery process so that
  - users start receiving benefits early,
  - risk of project failure is reduced through early feedback

Systems Thinking

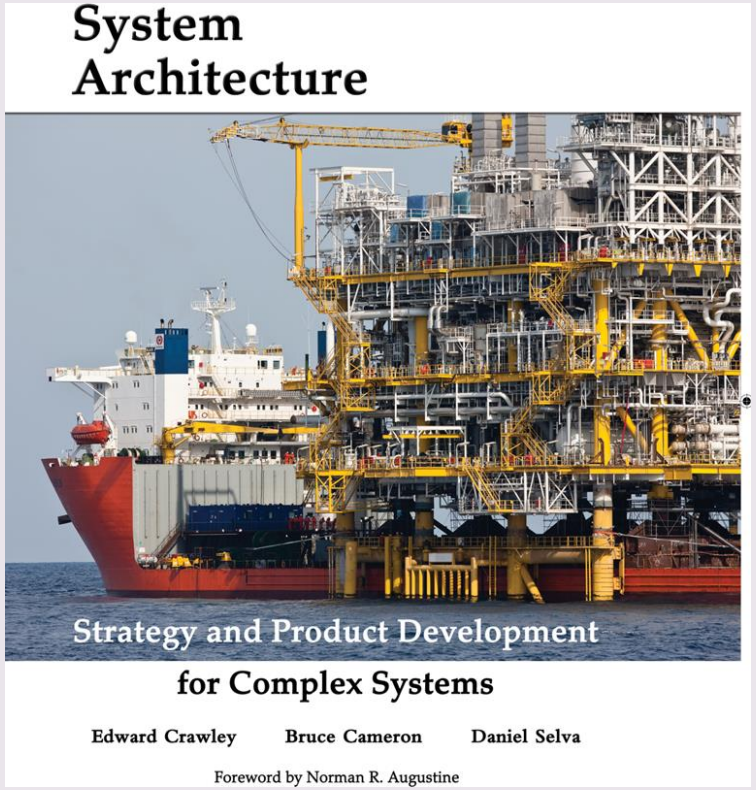
- A system is a whole that consists of parts, each of which can affect its behavior
- A system is a whole that cannot be divided into independent parts: The parts are *interconnected*.
- The essential or defining properties of any system are properties of the whole *that none of its parts have*
- A system is not the sum of the behaviors of its parts, *but the product of their interactions*

https://youtu.be/OqEeIG8aPPk?t=183

Russel Ackoff, 1994



Principle of Emergence



26 Principles of System Architecture

As the entities of a system
are brought together,

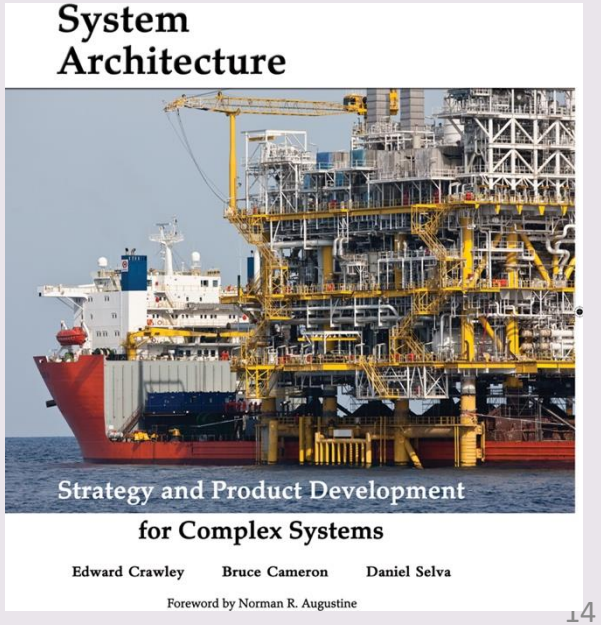
their interaction

will cause function, behavior, performance,
and other
intrinsic properties

to emerge.

Emerging Success or Failure

- Interaction of entities leads to emergence.
  - Emergence: what appears, materializes, or surfaces when the system operates.
  - Emergence gives systems added value.
- Predicting emergent properties is hard
  - Depends on interactions of possibly many entities
  - Changing one entity may have many ripple effects
- System **success**: *anticipated properties emerge*
- System **failure**:
  - Anticipated properties fail to emerge
  - Undesirable emergent properties appear



Articulating Function: *System Vision*

- Clear vision of what the system is and will do
  - Reflects primary externally delivered function
- Simple, compelling, articulated, shared, inspired
- Expressible in terms that are understandable to end users
- Comes with a credible roadmap towards this vision.
- Driven / enabled by sound architectural foundations
- Co-production of product manager and architect

Our Mission

[redacted] is the leading destination for short-form mobile video. Our mission is to inspire creativity and bring joy.

[redacted]

Our mission is to unlock the potential of human creativity—by giving a million creative artists the opportunity to live off their art and billions of fans the opportunity to enjoy and be inspired by it.

About

[redacted] is an open source driver assistance system. [redacted] performs the functions of Automated Lane Centering and Adaptive Cruise Control for over 85 supported car makes and models.

[redacted]

Long before we knew that it would be called [redacted] we knew what we wanted it to be. Instead of teaching the rest of the world cryptography, we wanted to see if we could develop cryptography that worked for the rest of the world. At the time, the industry consensus was largely that encryption and cryptography would remain unusable, but we started [redacted] with the idea that private communication could be simple.

Our Mission

TikTok is the leading destination for short-form mobile video. Our mission is to inspire creativity and bring joy.

For the Record

Our mission is to unlock the potential of human creativity—by giving a million creative artists the opportunity to live off their art and billions of fans the opportunity to enjoy and be inspired by it.

About

openpilot is an open source driver assistance system. openpilot performs the functions of Automated Lane Centering and Adaptive Cruise Control for over 85 supported car makes and models.

Signal Foundation

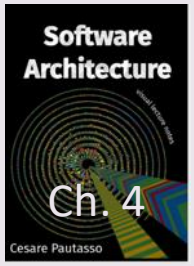
moxie0 on 21 Feb 2018

Long before we knew that it would be called Signal, we knew what we wanted it to be. Instead of teaching the rest of the world cryptography, we wanted to see if we could develop cryptography that worked for the rest of the world. At the time, the industry consensus was largely that encryption and cryptography would remain unusable, but we started Signal with the idea that private communication could be simple.

Articulating Function: *Scenarios*

- Easy, intuitive, way of describing essence of system
- Short, step-by-step descriptions of typical way in which ”actors” use a system
- Purpose: Architectural decision making
  - Scenarios touching multiple components
- Overview first: List of scenarios described by just a sentence
- Details: Selected scenarios worth exploring in depth

Example Music Player Scenarios



1. Browse for new songs
2. Search for interesting songs
3. Play the song sample
4. Pay to hear the entire song
5. Download the purchased song on the device
6. Play the song
7. Play multiple songs on a predefined playlist
8. Play multiple songs in random order
9. Share songs with friends
10. Make a backup of the device's content
11. Suggest related songs
12. Generate a tasteful playlist
13. Display album cover image
14. Show the device's battery status
15. Record sounds with amicrophone

Example: Nestorix Pension System

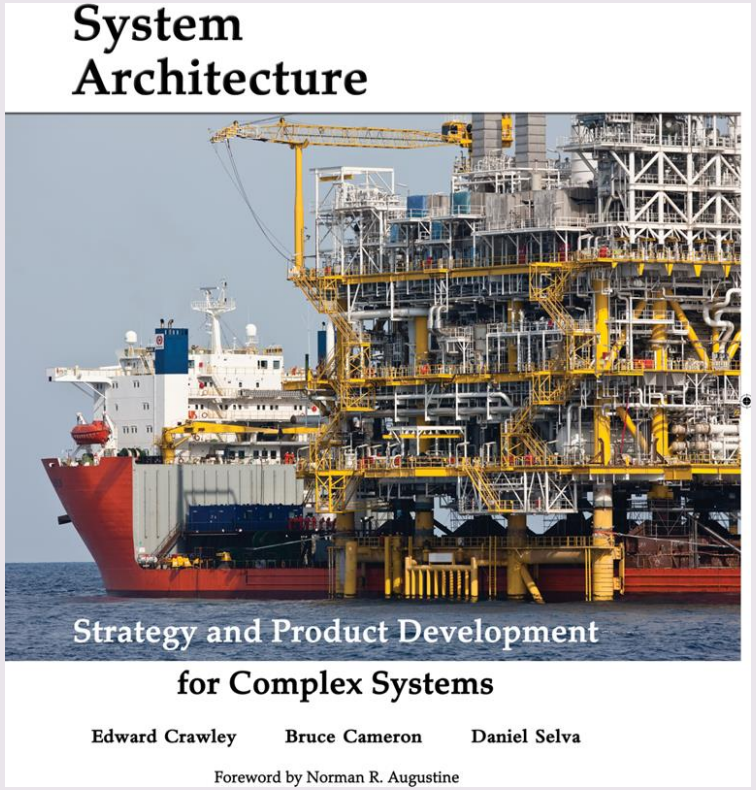
Nestorix: Rethinking Pension Provisioning

In this post, we propose a radical rethinking of IT support for pension systems, through a new system called Nestorix. The primary functionality of Nestorix is giving citizens authoritive information about the status of their pension — today, tomorrow, and many years from now. The system connects to third party pension fund suppliers, to obtain accurate pension data. Furthermore, it allows third party suppliers to specialize the Nestorix user experience so that it includes unique features of the pensions offered. The initial market place for this system is The Netherlands, but the ultimate vision is to make this a European pension system, supporting the free movement of people and their pensions through Europe.

Example Nestorix Scenarios

- **Logon:** A Dutch citizen logs on to Nestorix using their national identity authorization system (presently DigiD)
- **Partner Pension:** A citizen uses Nestorix to discover how much of their pension will transfer to their partner in case of passing away
- **Employment Reduction:** A citizen two years before retirement uses Nestorix to explore how much pension they would receive in case of reducing their work in the last two years before retirement to just two days per week instead of five days per week
- **Immediate Payout:** A citizen not yet retired explores the implications of paying out an insurance-based product immediately instead of paying it out during retirement
- **Change of Job:** A citizen changing jobs seeks to explore the impact of stopping with their current occupational pension fund, replacing it by another, asking for advice on how to mitigate negative consequences

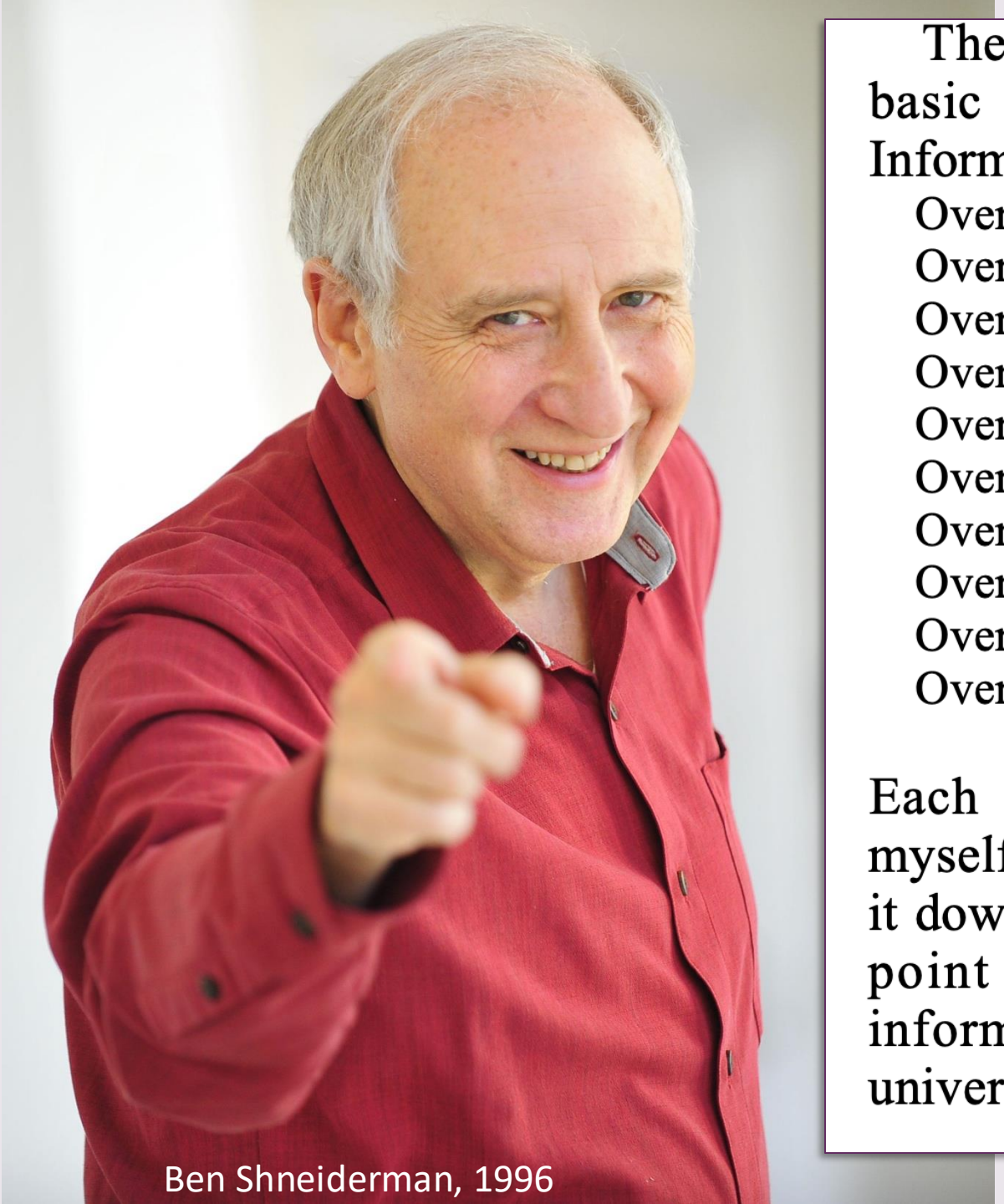
Principle of Focus



26 Principles of System Architecture

The number of identifiable issues that will influence a system at any point is beyond one's ability to understand.

One must identify the most critical and consequential issues, and focus on them.



Ben Shneiderman, 1996

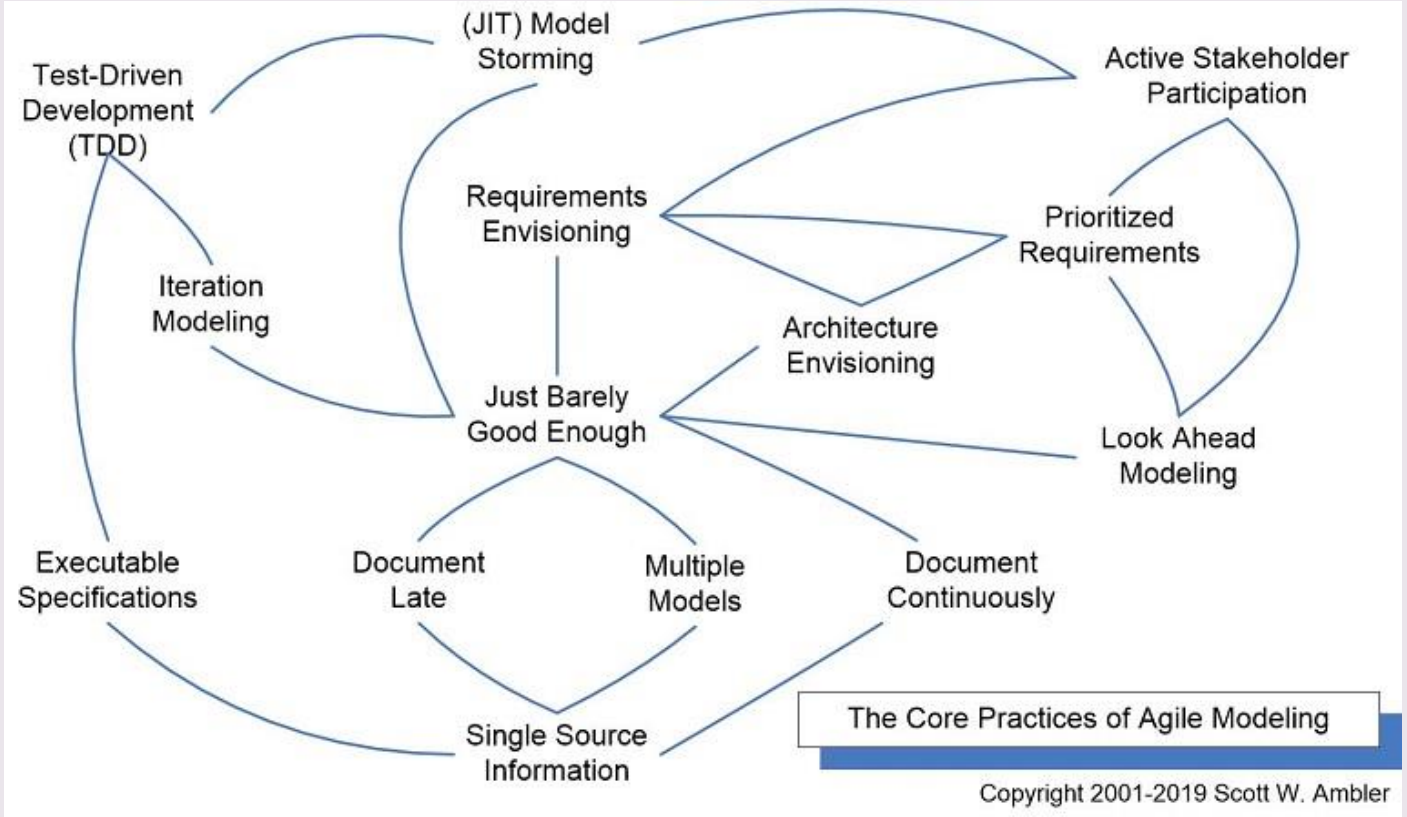
There are many visual design guidelines but the basic principle might be summarized as the Visual Information Seeking Mantra:

Overview first, zoom and filter, then details-on-demand
Overview first, zoom and filter, then details-on-demand
Overview first, zoom and filter, then details-on-demand
Overview first, zoom and filter, then details-on-demand
Overview first, zoom and filter, then details-on-demand
Overview first, zoom and filter, then details-on-demand
Overview first, zoom and filter, then details-on-demand
Overview first, zoom and filter, then details-on-demand
Overview first, zoom and filter, then details-on-demand
Overview first, zoom and filter, then details-on-demand

Each line represents one project in which I found myself rediscovering this principle and therefore wrote it down it as a reminder. It proved to be only a starting point in trying to characterize the multiple information-visualization innovations occurring at university, government, and industry research labs.

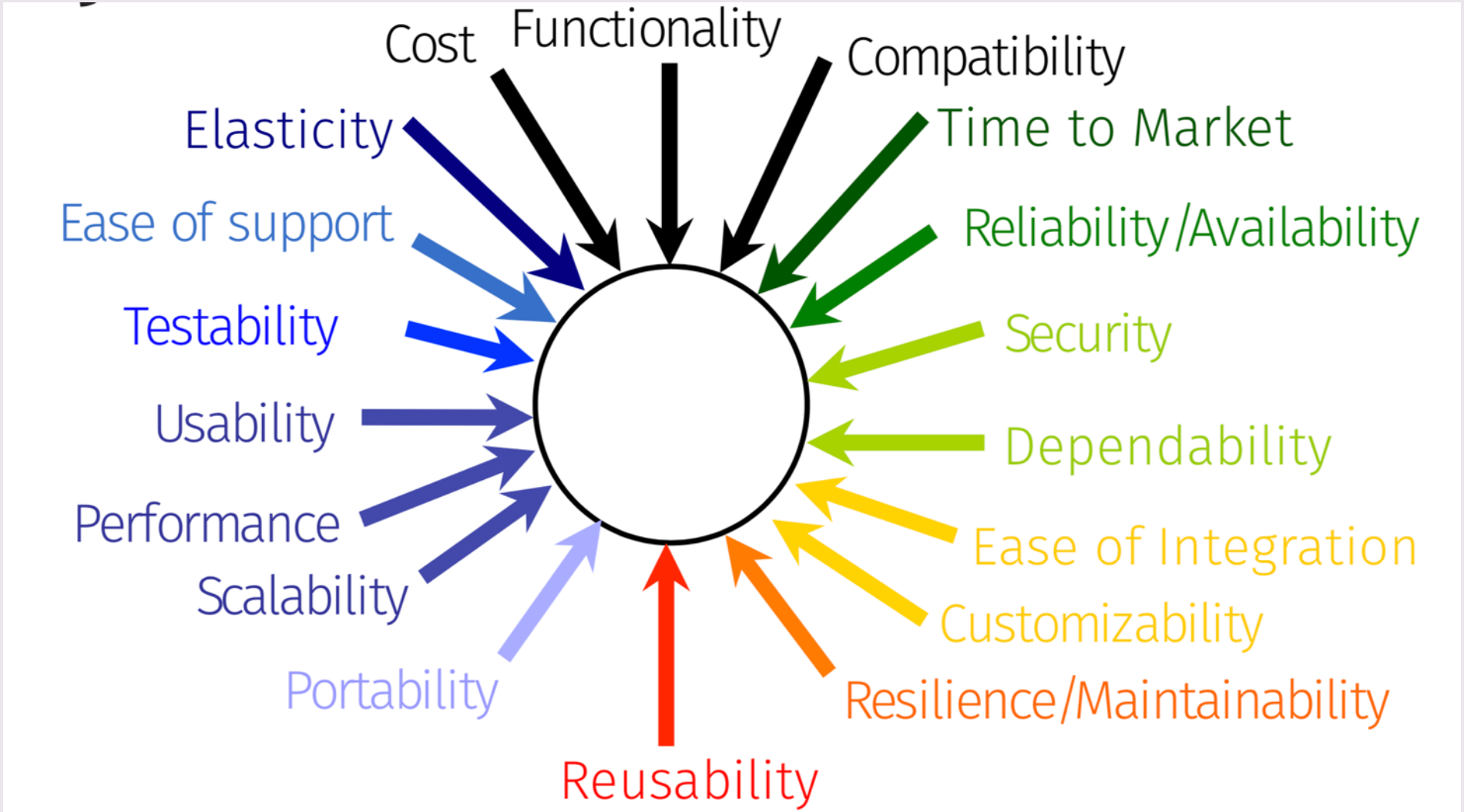
Agile Modeling Use Case Template (for details-on-demand)

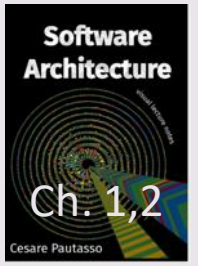
- Name
- Identifier
- Description
- Preconditions
- Postconditions
- Basic Course of Action
- Alternate Courses



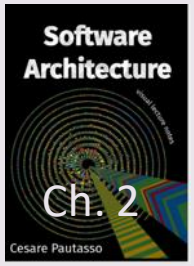
https://agilemodeling.com/artifacts/systemUseCase.htm

Articulating Function: Quality Attributes





Quality Attributes

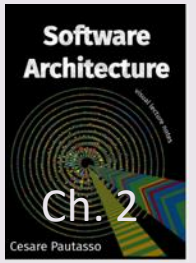


- Desirable properties of a system (under construction)

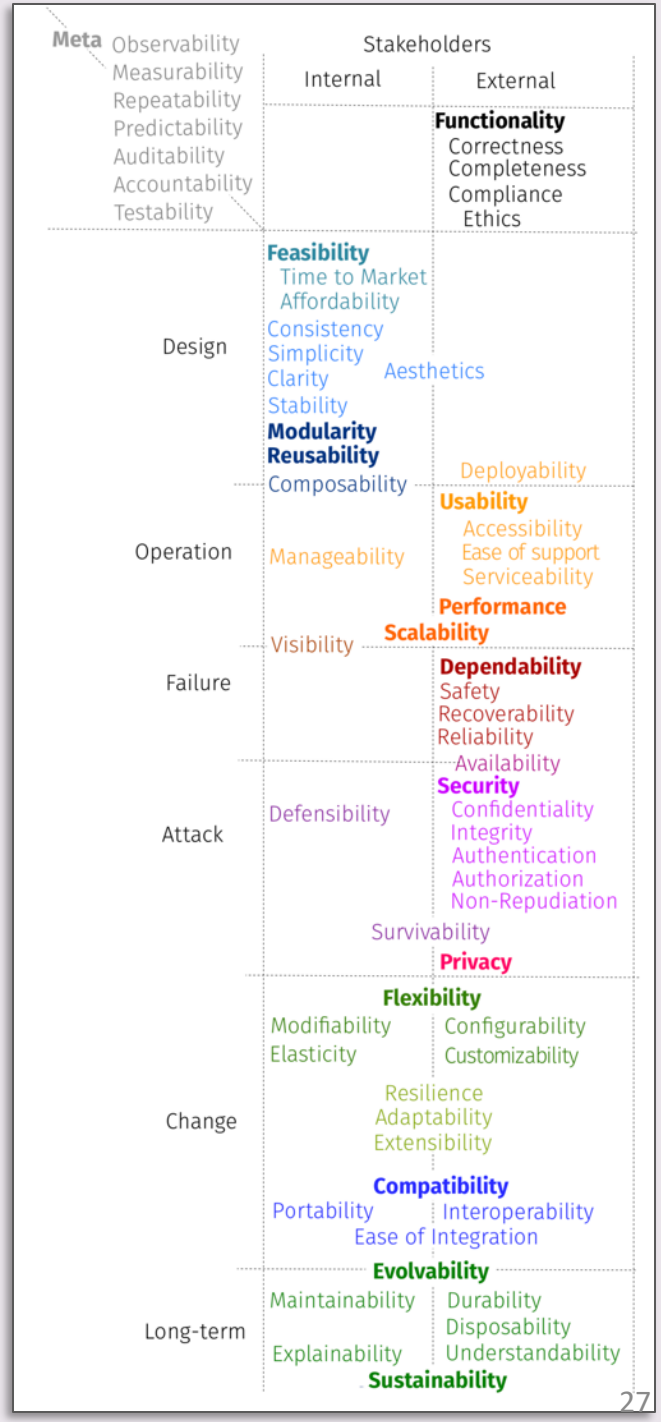
- **External**: fitness for purpose / does it meet stakeholder needs
- **Internal:** fitness for engineering process / can devs work with it
  - Internal attributes indirectly affect many external attributes

- **Static:** structural properties of the design / code (”form”)
- **Dynamic:** run time properties of the system in action (”function”)

A Catalogue of “ilities”



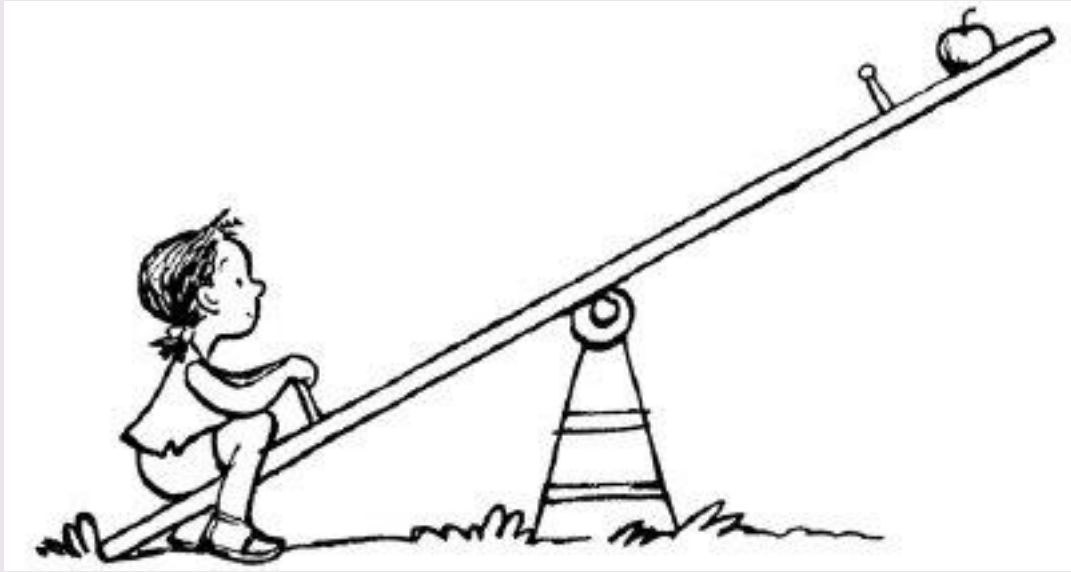
- **Meta** Qualities of qualities: measurability
- **Functionality** Correctness, completeness
- **Design** Modularity, reusability
- **Operation** Usability, performance, scalability
- **Failure** Recoverability, reliability, availability
- **Attack** Privacy, confidentiality, integrity
- **Change** Flexibility, extensibility, configurability
- **Long-term** Maintainability, explainability



Managing “ility” tradeoffs

- Privacy vs usability
- Modularity vs time-to-market
- Availability vs configurability
- Extensibility vs integrity
- Performance vs interoperability
- Performance vs confidentiality
- …

The architect needs to understand which attributes must be optimized, and which ones can be sacrificed



Example Nestorix Quality Attributes

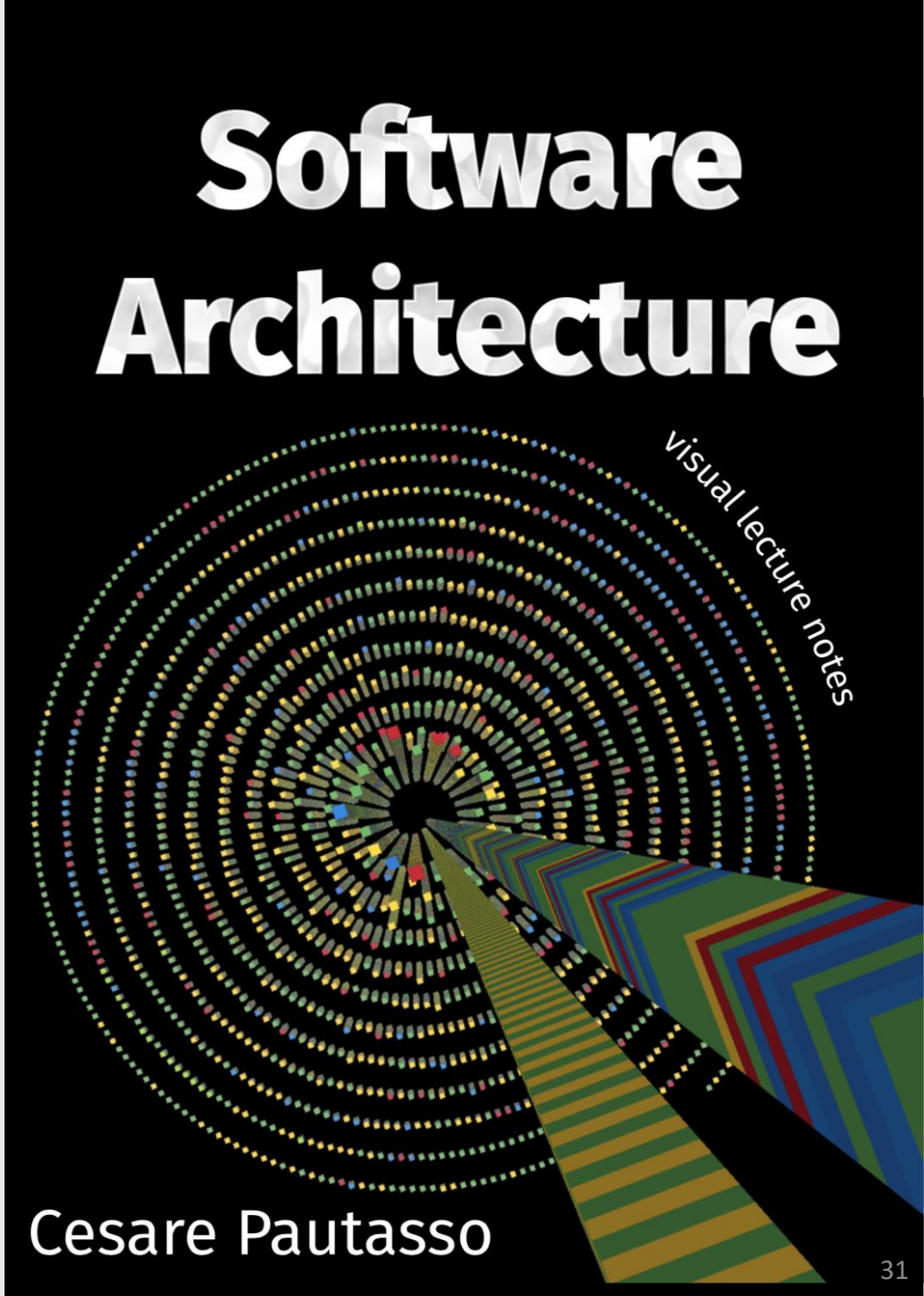
- **Flexibility:** Pension laws and the market players are changing constantly which the system should easily adjust to
- **Extensibility:** New market players may come and go
- **Configurability:** Different market players will want to bring in different aspects of their products
- **Usability:** The system should be easy to both for digitally natives and people with little digital affinity (possibly with varying user interfaces).
- **Scalability:** The number of users may be up to 10 million. This may affect storage as well as performance
- **Security:** All user data is sensitive and should be well-protected against cyber-attacks.
- **Privacy:** Data should be made available by need. Unnecessary data collection is to be avoided.

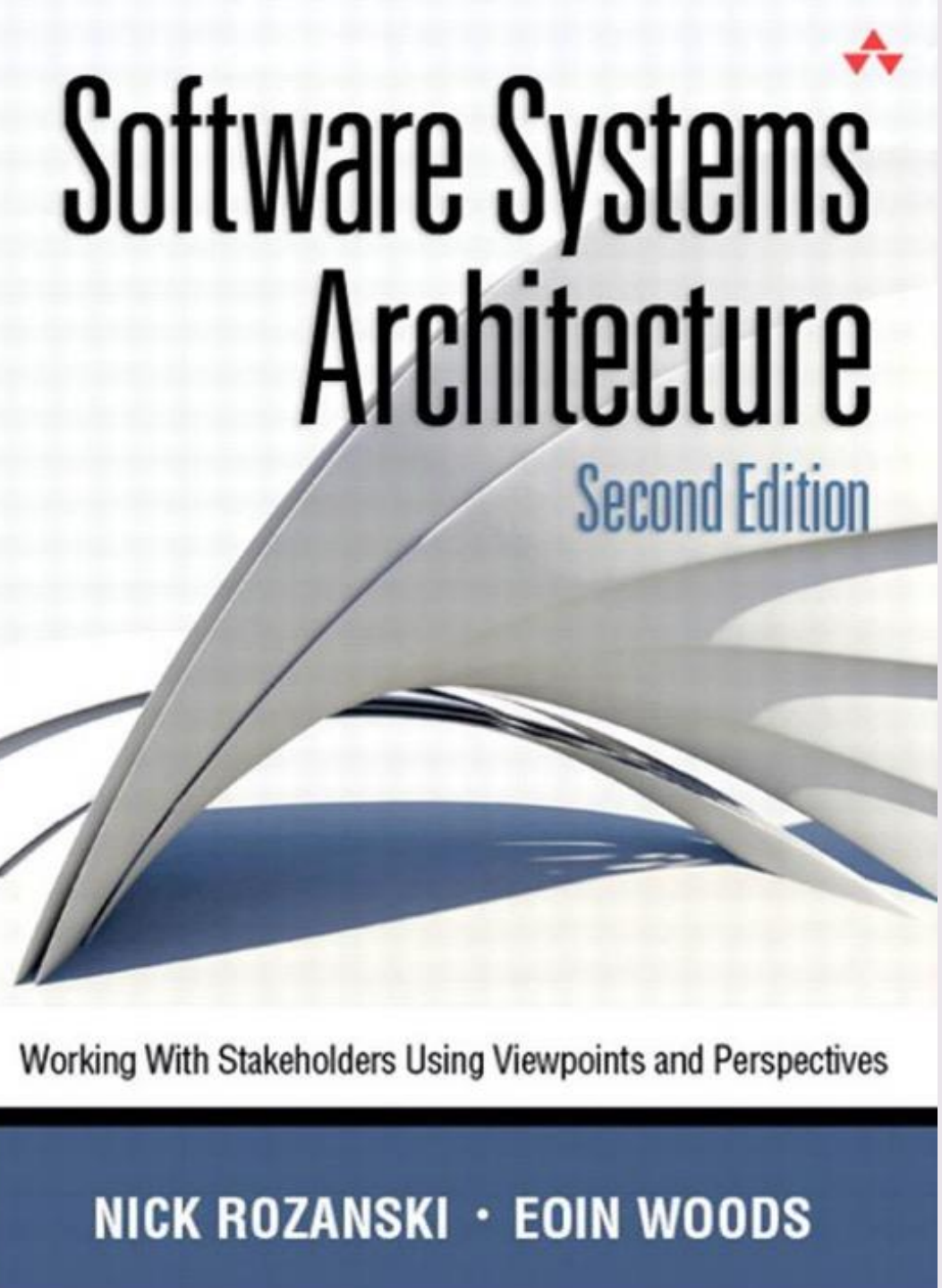
https://how.complexsystems.fail

16. **Safety is a characteristic of systems and not of their components**

    Safety is an emergent property of systems; it does not reside in a person, device or department of an organization or system. Safety cannot be purchased or manufactured; it is not a feature that is separate from the other components of the system. This means that safety cannot be manipulated like a feedstock or raw material. The state of safety in any system is always dynamic; continuous systemic change insures that hazard and its management are constantly changing.

1. Introduction
2. Quality Attributes
3. Definitions
4. Modeling Software Architecture
5. Modularity and Components
6. Reusability and Interfaces
7. Composability and Connectors
8. Compatibility and Coupling
9. Deployability, Portability and Containers
10. Scalability
11. Availability and Services
12. Flexibility and Microservices





Benefits for Whom?

Stakeholder Principle

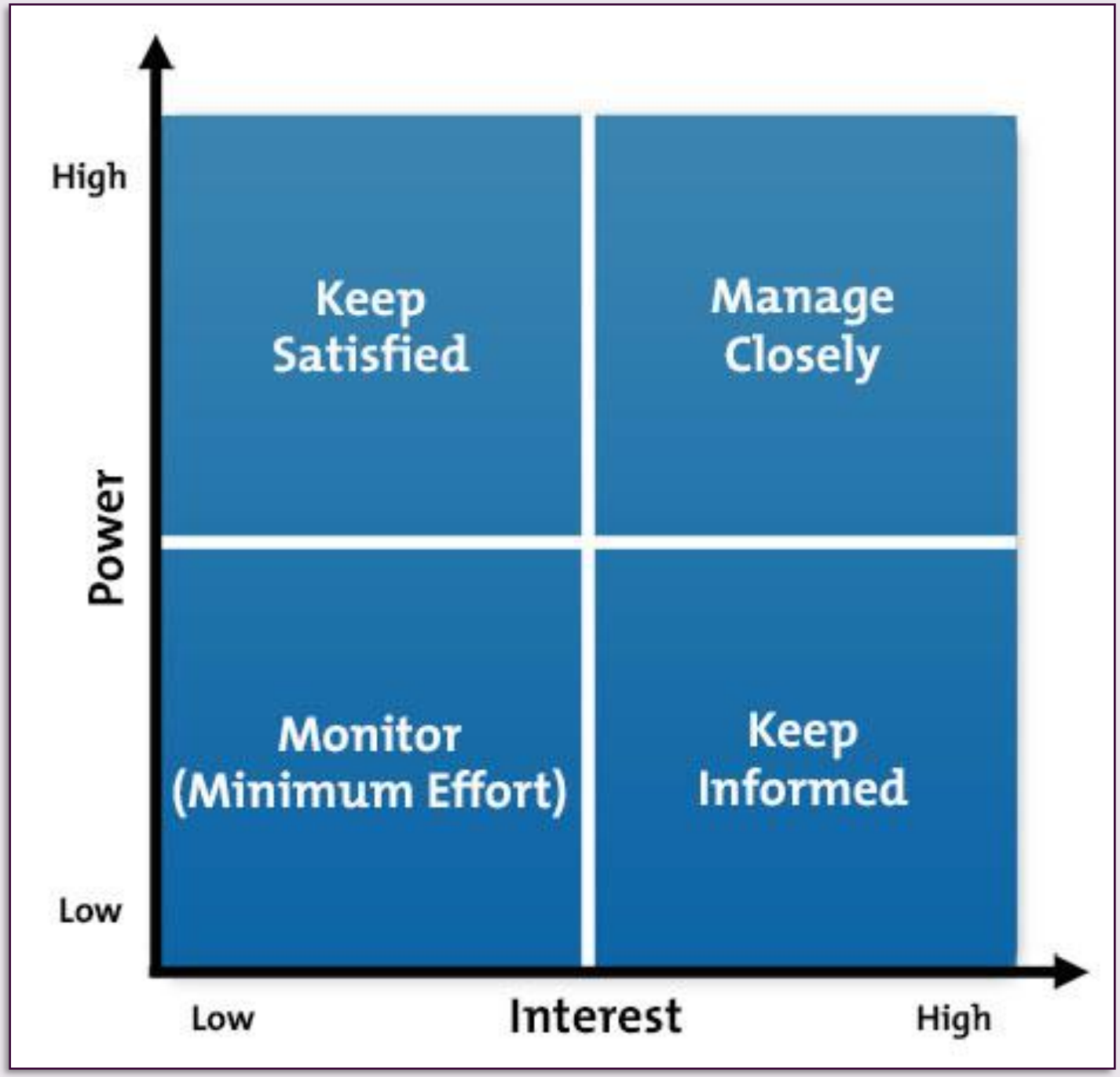
Architectures are created solely to meet stakeholder needs

Common Stakeholder Categories

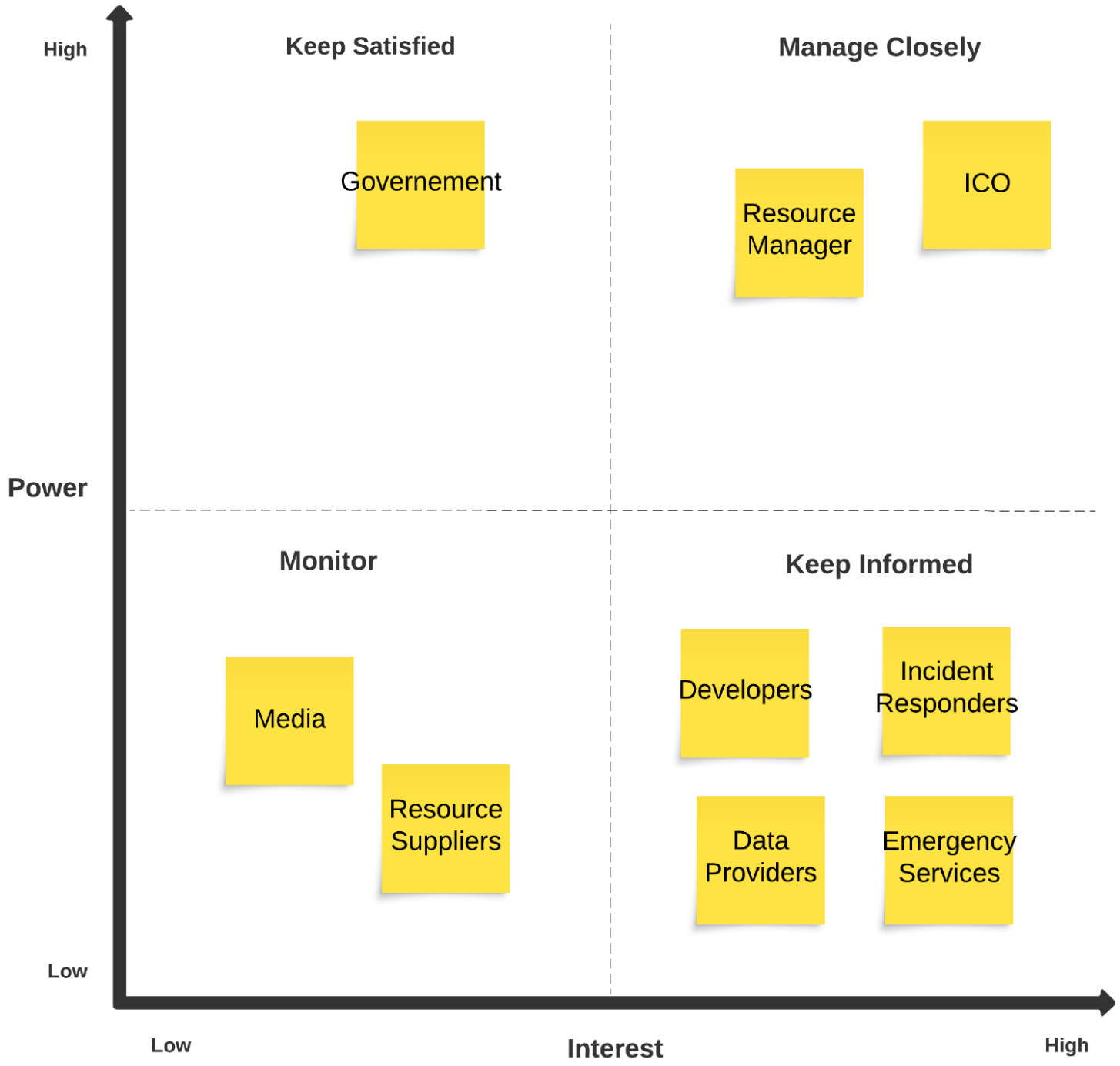
- Users (end-users, admins, moderators, …)
  - Anyone directly working with the system in operation
- Customers (companies, individuals, …)
  - Organizations deciding to buy / deploy / use the given system
- Engineers (developers, testers, …)
  - Anyone involved in the creation or operation of the system
- Governance / “The Business” (management, budget holders)
  - Decision makers on direction / termination / continuation
- External parties (outsiders you may need to influence)
  - Libraries / services the system will depend on

Managing stakeholder expectations:

The Power / Interest Grid



DESOSA 2024:
BlazeGuard: A Wildfire Resource Management System



Pricing Models and Architectural Implications

- Distribution channel:
  - Web, app store, downloads
- Free/premium:
  - Enable/disable functionality
- Monthly / yearly subscription
  - Release and license management
- In app purchases:
  - Extension mechanism (plugins)
- Advertisements: user / click tracking?
- Pay per use: usage tracking?

Habits of Mind

> Always design a thing by considering it in its next larger context – a chair in a room, a room in a house, a house in an environment, an environment in a city plan.

— ELIEL SAARINEN

https://quotesondesign.com/eliel-saarinen/

26

Systems in Context

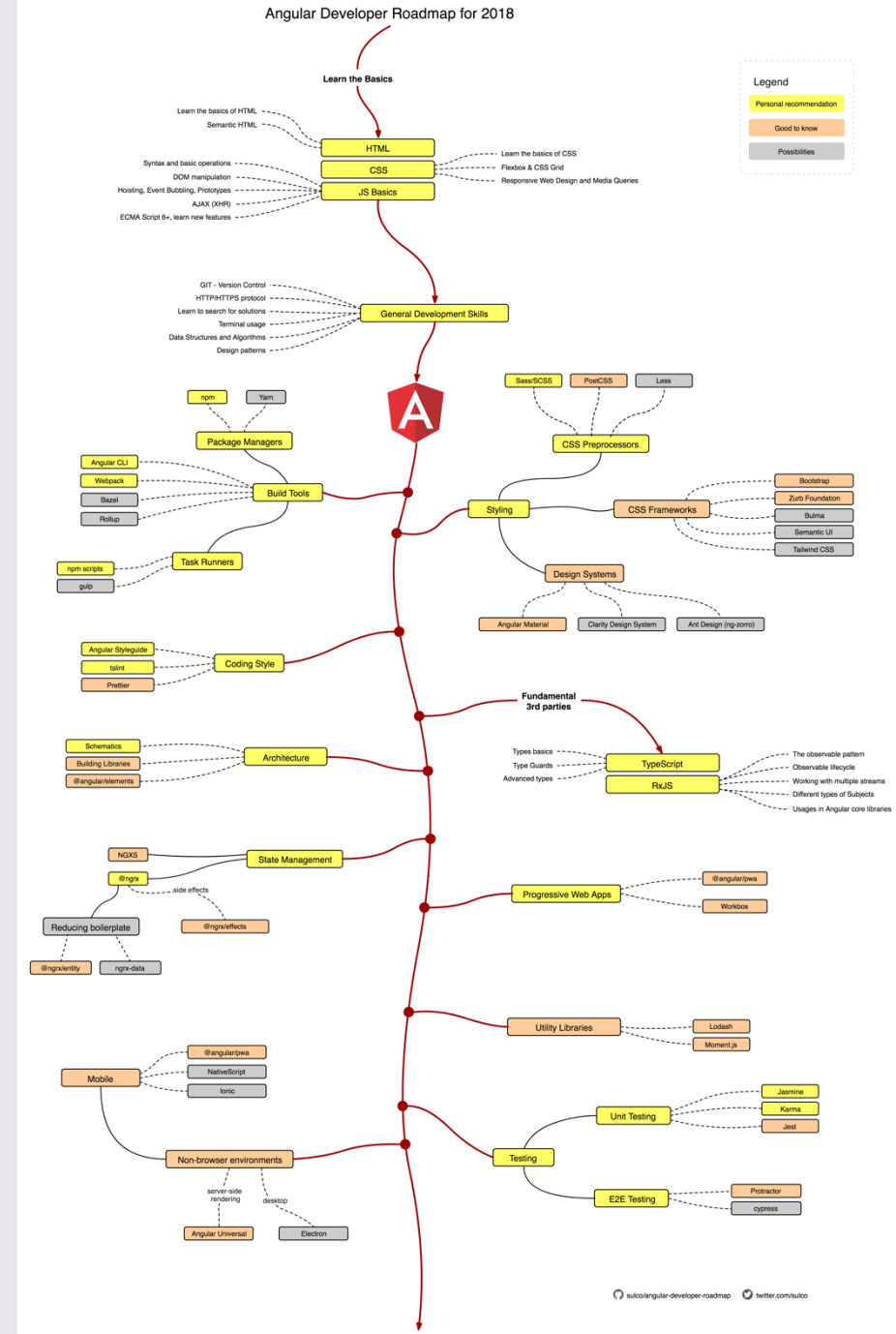
- How will it be used
  - Provided APIs, market place, …
  - How will context gain value from it
- What does the system need from its environment
  - External services, data dependencies, …

Sequencing the Delivery Process

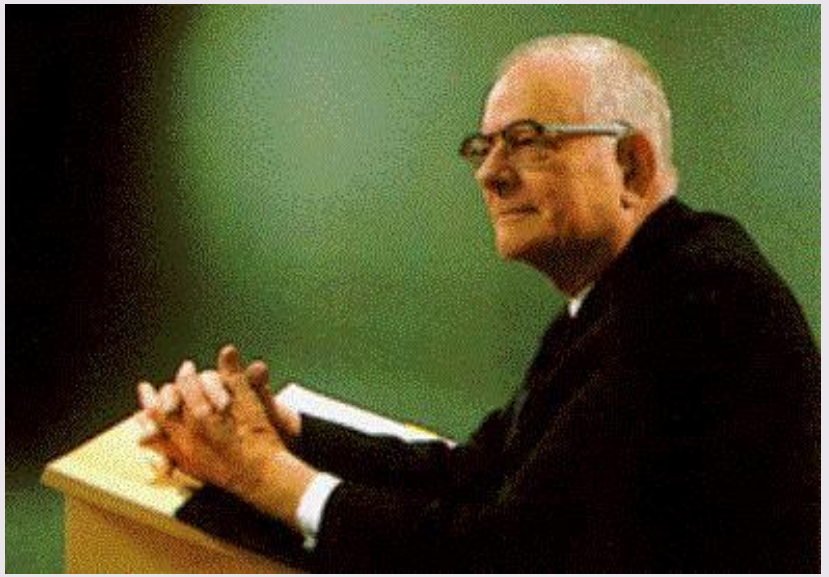
- What is the minimal viable product that already delivers (some) value?
- “Next feature” should be chosen based on
  - Maximizing value returned at that moment
  - Maximizing ability to learn from feature as it is in operation (is it indeed “valuable”?)
- This requires mind shift for ”problem owner”!

Roadmapping

- Obtain overview of future architecturally significant functionality
- Identify dependencies / interactions
  - Might impact ordering
  - Might increase / decrease costs, or even block future features
  - Keep options open
- Estimate effort (costs, due dates)
- Enable business to create roadmaps informed by architecture



Scaled Agile Framework Principles



> *The impression that 'our problems are different' is a common disease that afflicts management the world over. They are different, to be sure, but the principles that will help to improve the quality of product and service are universal in nature.*
>
> —W. Edwards Deming, *Out of the Crisis* [1]

https://scaledagileframework.com/safe-lean-agile-principles/

Scaled Agile Framework Principles

#1 Take an economic view

#2 Apply systems thinking

#3 Assume variability; preserve options

#4 Build incrementally with fast, integrated learning cycles

#5 Base milestones on objective evaluation of working systems

#6 Make value flow without interruptions

#7 Apply cadence, synchronize with cross-domain planning

#8 Unlock the intrinsic motivation of knowledge workers

#9 Decentralize decision-making

#10 Organize around value

https://scaledagileframework.com/safe-lean-agile-principles/

BUSINESS AGILITY

Organizational Agility

Enterprise | Government

Operational Value Streams

Lean Portfolio Management

Epic Owners | Enterprise Architect

Agile Product Delivery

Business Owners

Product Mgmt | System Architect

RTE

Team and Technical Agility

Agile Teams

Product Owner

Scrum Master / Team Coach

Business & Technology

Portfolio Flow

Strategic Themes → Portfolio Vision → Portfolio Backlog (NFRs)

Big Data

PB Lean Budgets

Guardrails

Value Stream Management: Epic | Enabler | Epic — Coordination

Development Value Streams

Solutions

KPIs

PORTFOLIO

ART Flow

Customer Centricity

Design Thinking

Lean UX

ART Backlog (WSJF, NFRs)

Team Flow

SAFe Scrum

Built-In Quality

Team Backlogs (NFRs)

SAFe Team Kanban

Continuous Delivery Pipeline

AGILE RELEASE TRAIN

Continuous Exploration | Continuous Integration | Continuous Deployment

Release on Demand

Solution

Solution Context

Cloud

DevOps

System Demos

PI Planning | PI Planning | CD | CI | CE | IP Iteration | PI Planning | PI Planning

Enabler | Feature | Story | Enabler

Iterations

PI

PI Objectives

Architectural Runway

ESSENTIAL

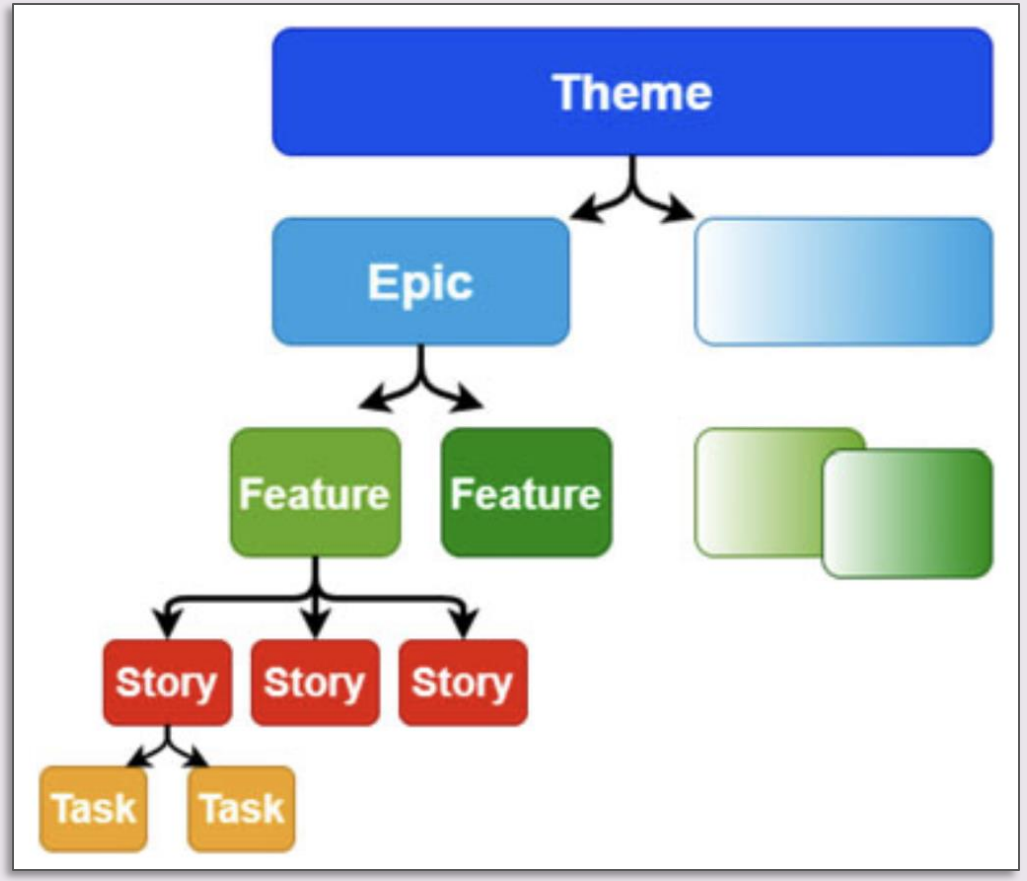
Vision | OKRs | Roadmap | AI | Shared Services | CoP | System Team | Measure & Grow

Lean-Agile Leadership | Lean-Agile Mindset | Core Values | SAFe Principles | Implementation Roadmap | SPC | Continuous Learning Culture

Elvan Kula

Agile at Scale

- ING Bank, with 15,000 IT staff
- 100s of self-organizing teams (5-9 developers)
- Short iterations (1-4 weeks)
- User stories, features, epics
- Delivered in releases (2-6 months)
- Quarterly planning of all releases



Ethical Benefits?

Grady Booch

@Grady_Booch

Every line of code has a moral and ethical implication.

Ethics

Well-founded standards of right and wrong

that prescribe what humans ought to do,

usually in terms of rights, obligations, benefits to society, fairness, or specific virtues.

The continuous effort of studying

our own moral beliefs and our moral conduct,

and striving to ensure that we, and the institutions we help to shape,

live up to standards that are reasonable and solidly-based

https://www.scu.edu/ethics/ethics-resources/ethical-decision-making/what-is-ethics/

ACM Code of Ethics and Professional Conduct

Preamble

Computing professionals' actions change the world. To act responsibly, they should reflect upon the wider impacts of their work, consistently supporting the public good. The ACM Code of Ethics and Professional Conduct ("the Code") expresses the conscience of the profession.

The Code is designed to inspire and guide the ethical conduct of all computing professionals, including current and aspiring practitioners, instructors, students, influencers, and anyone who uses computing technology in an impactful way. Additionally, the Code serves as a basis for remediation when violations occur. The Code includes principles formulated as statements of responsibility, based on the understanding that the public good is always the primary consideration. Each principle is supplemented by guidelines, which provide explanations to assist computing professionals in understanding and applying the principle.

Section 1 outlines fundamental ethical principles that form the basis for the remainder of the Code. Section 2 addresses additional, more specific considerations of professional responsibility. Section 3 guides individuals who have a leadership role, whether in the workplace or in a volunteer professional capacity. Commitment to ethical conduct is required of every ACM member, and principles involving compliance with the Code are given in Section 4.

On This Page

The Ethical Software Architect?

Ethics of the Product

- Avoid harm
  - Human control;
  - Dual use
- Respect human dignity
  - Fairness
  - Privacy
  - Tracking
- Consider impact on society
  - Behavior amplification?

Ethics of the Construction

- Actively design an ethical product
- Avoid waste: Resource usage, energy consumption, development effort
- Respect licenses / constraints of reused libraries or services
- Ensure robustness, so that the product can be depended on
- Avoid engineering shortcuts
- Embrace code of conduct

Summary as Possible R1 Content

- The system vision
- Defining function
- Scenarios
- Quality attributes / tradeoffs
- Stakeholder analysis
- Pricing models
- Context and dependencies

- Feature (value) roadmap
- Ethical considerations
- …

But what about the application domain?
(… to be continued …)