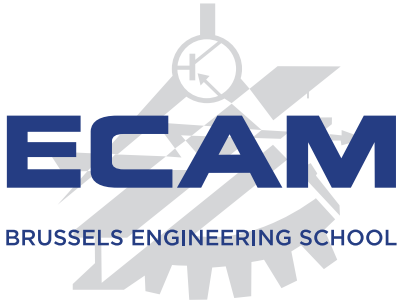


I402A Software Architecture
and Quality Assessment

Session 6

Software Design Patterns

Sébastien Combéfis

Fall 2019

Objectives

- Implementation pattern and excellence in programming

  Communication, simplicity and flexibility

- Other categories of software design patterns
  - Concurrency patterns for synchronisation, communication, etc.
  - Architectural patterns in relation with enterprise solutions
  - Green patterns for sustainable development


Implementation Pattern

Implementation Pattern

- One should aim for excellence in programming

  Best program offerts best extension option,
  has no unrelated elements and is easy to understand

- Three values that are important to strive to excellence
  - **Communication**: read can understand, modify and use
  - **Simplicity**: no unnecessary complexity for the program
  - **Flexibility**: allows for easy evolution and changes

Principle (1)

- Structure the code to ensure local consequences

  A change "here" can not result in a problem "there"

- Minimise repetitions of code, structure, algorithm, etc.

  Duplication, parallel class hierarchies, similar code

- Keep together data and the logic manipulating them

  Following the object oriented programming paradigm

Principle (2)

- Make explicit symmetry at different levels

  The presence of an `add` should suggest a `remove`

- Express intentions as declaratively as possible

  Use annotations in imperative programming

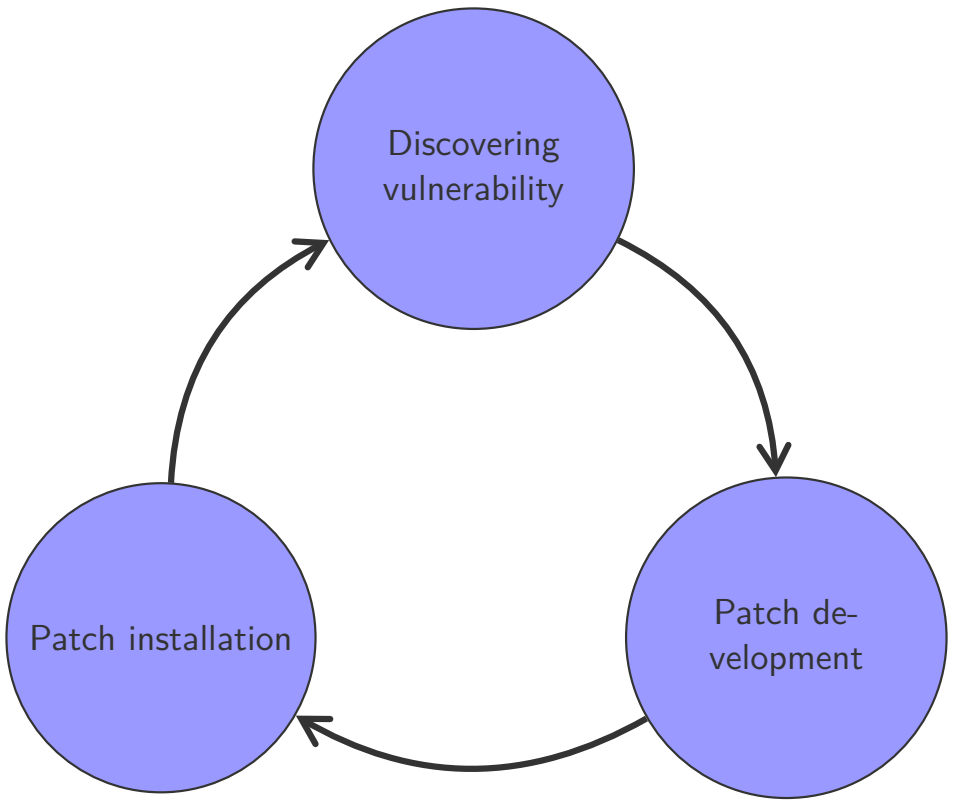
- Gather code and data by change rate

  Allows changes to the same place at the same time

Security Aspect

- Software development vulnerability cycle

  Important to take security into account in the development cycle



Security versus Usability

- Never possible to achieve high security and high usability

  Two opposites software quality criteria

- Important to choose the right compromise
  - Depending on a risk analysis on the system
  - Depending on the category of targeted users

- It is not always necessary to look for a trade-off

  Security by design taking into account UX aspects

KISS

- **Keep it Simple, Stupid**

  Avoid making problems more complicated

- Guiding principle in software engineering

  Avoid over-engineering, favour pragmatic solutions

> "Many great problem solvers were not great coders, but yet they produced great code!"

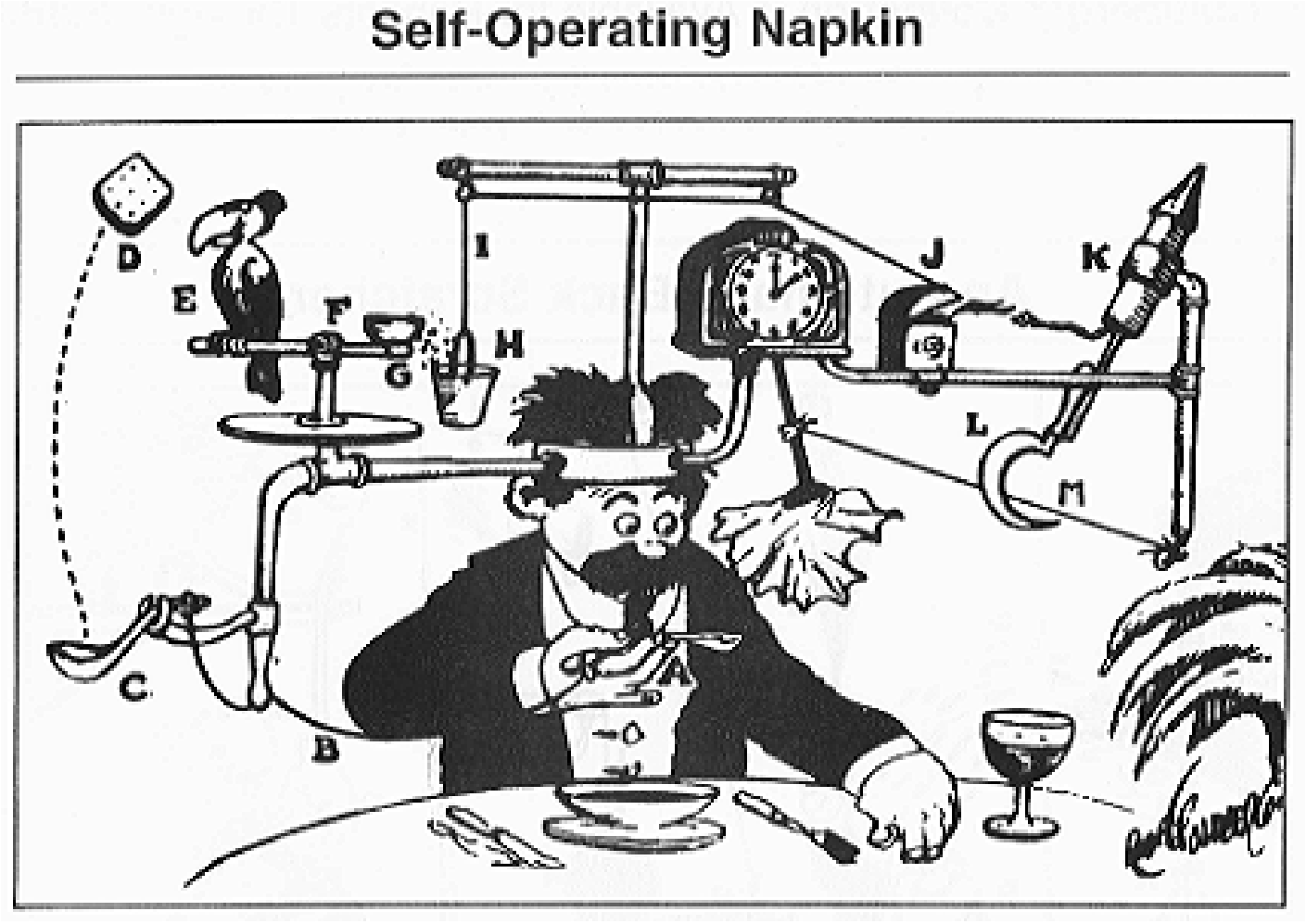
Some Citations

"Simplicity is the ultimate sophistication" – Leonardo Da Vinci

"Everything should be made as simple as possible, but no simpler" — Albert Einstein

"Perfection is achieved not when there is nothing more to add, but when there is nothing left to take away" — Antoine de Saint Exupéry

Rube Goldberg Machine



Other Pattern

- Concurrency patterns

  Patterns used with multithreaded programming

  (Double-checking locking, Lock, Monitor, Thread pool, etc.)

- Architectural patterns

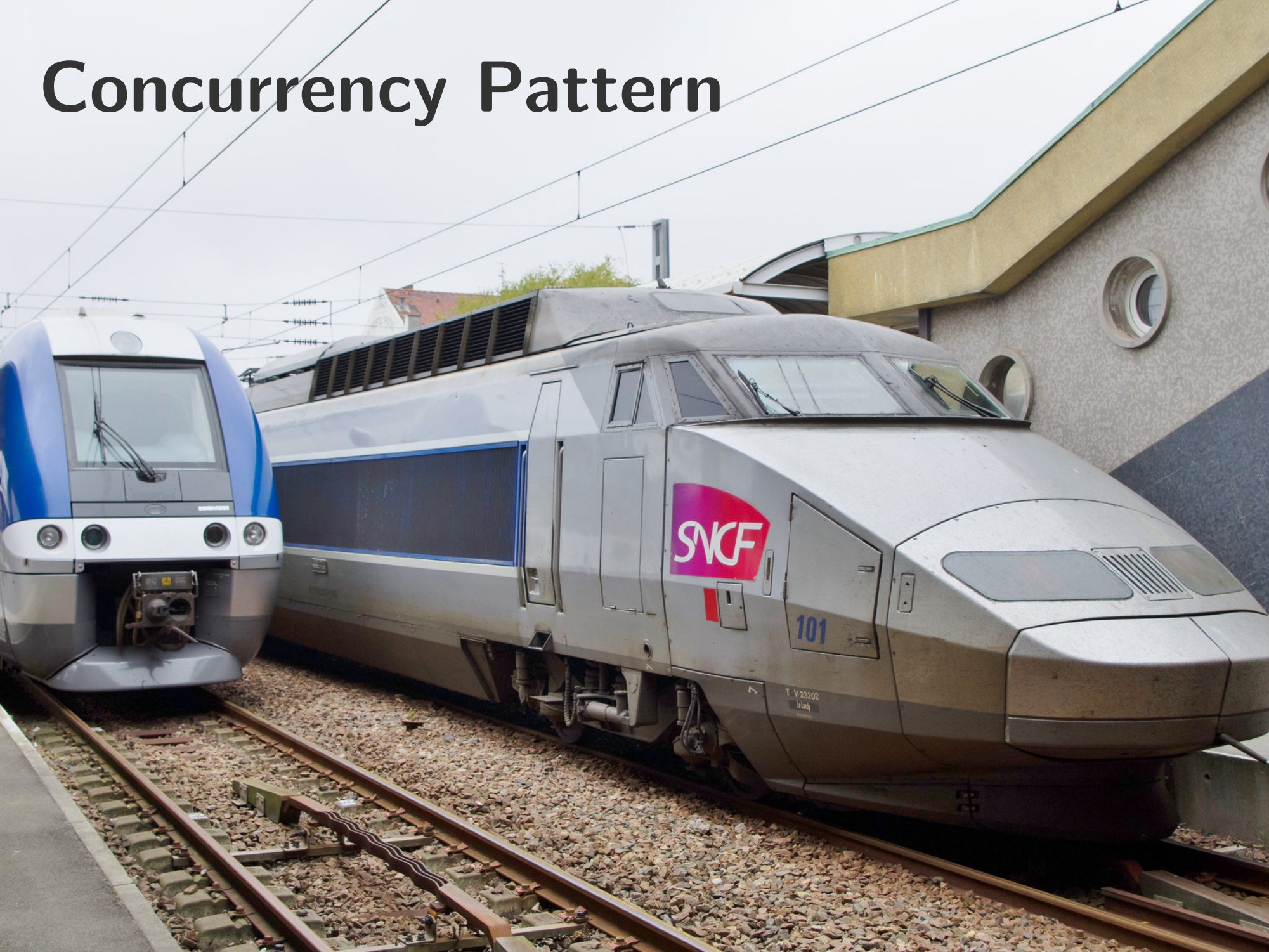
  Patterns with wider scope than design patterns

  (Interceptor, MVC, n-tier, Active record, etc.)

- Patterns specific to programming languages

  Depend on the programming paradigm specific to the language

Concurrency Pattern



Concurrency Pattern

- Patterns applicables for writing concurrent programs

  Synchronisation, communication, data storage, cache, etc.

- Several examples covering different aspects
  - **Messaging Design Pattern** (MDP)

    Message exchange between components
  - **Read-write lock**

    Concurrent read access and exclusive write access
  - **Thread pool**

    Set of threads to execute a set of tasks

Data Access Pattern

- Patterns to control/regulate access to data

  Concurrent access, efficient storage, resources management, etc.

- Several examples covering different aspects

  - **Update Factory**

    Automatic update with an ORM framework

  - **Optimistic/Pessimistic Lock**

    Lock regulating concurrent access to data

  - **Cache Accessor**

    Data access through a cache

Architectural Pattern

Enterprise Pattern

- Patterns used in enterprise frameworks like J2EE or .NET

  Problems similar to all users with similar solution

- Several examples covering different aspects
  - **Business Delegate**

    Access to business data thanks to presentation-tier client
  - **Service Activator**

    Activating or not a service asynchronously
  - **Composite View**

    Unified presentation of information from several sources


Green Pattern

Green Pattern (1)

- Make software development more sustainable

  Software eco-design and set of good practices

- The IT sector represents $\sim 2\%$ of CO_2 emissions

  As much as the field of aviation

- *"Software is getting slower more rapidly than hardware becomes faster"* (Wirth's law)

https://www.greencodelab.fr

Green Pattern (2)

- Good practices for social and environmental impact
  - Avoid obsolescence
  - Reduce consumption (performance, storage, etc.)
  - Limit exclusions
- Do we really need to...
  - ...know the precise number of results of a search?
  - ...leave all its tabs always loaded in a browser?
  - ...transfer lots of small images from a server to a client?


Other Patterns

Real-Time Pattern

- Development of real-time or embedded application

  Concurrency, memory, resource, distribution, security, safety, etc.

- Several examples covering different aspects

  - **Rendezvous Pattern**

    Synchronising a set of threads

  - **Data Bus Pattern**

    Communication between components that ignore each other

  - **Watchdog Pattern**

    Automatic failure identification and correction

Reactive Design Pattern

- Patterns for distributed systems driven by messages

  Resilient, responsive and elastic system

- Several examples covering different aspects
  - **Let-it-crash Pattern**

    Let a service crash and restart
  - **Active-Passive Replication Pattern**

    Data replication management
  - **Request-Response Pattern**

    Message exchange between components

User-Interface Pattern

- Recurring user interface design issues

  Request inputs, display data, navigation, social, etc.

- Several examples covering different aspects

  - **Morphing Controls Pattern**

    Only show the available controls according to the mode

  - **Slideshow Pattern**

    Displaying a media collection as a sequence of images

  - **Accordion Menu Pattern**

    Navigation in the sections by jumping on subsections

Cloud Design Pattern

- Solving big challenges in cloud computing

  Availability, data management, resilience, performance, etc.

- Several examples covering different aspects

  - **Bulkhead Pattern**

    Isolate elements in pools, to survive failures

  - **CQRS Pattern**

    Separation of operations that read and write data

  - **Retry Pattern**

    Transparent recall of a service that failed

Many Other Patterns

- Responsive web design pattern

  https://developers.google.com/web/fundamentals/design-and-ux/responsive/patterns

- Patterns by programming languages

  JavaScript, PHP, etc.

- Patterns specific to programming paradigms

  Functional programming, object oriented programming, etc.

- Game Programming Patterns

  http://gameprogrammingpatterns.com

References

- Kent Beck, *Implementation Patterns*, Addison-Wesley, 2007. (ISBN: 978-0-321-41309-3)
- Susan Morrow, *Security versus usability a conundrum in modern identity management?*, July 3, 2018. https://www.csoonline.com/article/3286609/security-versus-usability-a-conundrum-in-modern-identity-management.html
- Monique Magalhaes, *Security vs. Usability: Does there have to be a compromise?*, December 20, 2018. http://techgenix.com/security-vs-usability
- *What does KISS stand for?*, retrieved on September 25, 2019. http://people.apache.org/~fhanik/kiss.html
- *Web Energy Archive: Mesurez l'impact environnemental d'Internet!*, retrieved on September 25, 2019. https://wea.greencodelab.org
- Roland Kuhn, Brian Hanafee, & Jamie Allen, *Reactive Design Patterns*, Manning Publications, 2017. (ISBN: 978-1-617-29180-7)
- Anders Toxboe, *User Interface Design patterns*, retrieved on September 25, 2019. http://ui-patterns.com
- Microsoft, *Cloud Design Patterns*, January 3, 2018. https://docs.microsoft.com/en-us/azure/architecture/patterns

Credits

- David, June 15, 2011, https://www.flickr.com/photos/djgw/17340332605.
- Goldenluigi, March 31, 2010, https://fr.wikipedia.org/wiki/Fichier:Rube_Goldberg%27s_%22Self-Operating_Napkin%22_(cropped).gif.
- Joshua Brown, April 25, 2015, https://www.flickr.com/photos/joshtechfission/17281610121.
- mehmet bilgin, October 13, 2017, https://www.flickr.com/photos/sonyalpha330lynxpardus/37845748181.
- Bernard Spragg. NZ, December 6, 2013, https://www.flickr.com/photos/volvob12b/11263534936.
- Vishweshwar Saran Singh Deo, September 5, 2008, https://www.flickr.com/photos/vssdeo/2910865923.