Towards Generic Design Patterns for Evolvable Modular Architectures

Prof. dr. Herwig Mannaert
Normalized Systems Institute
University of Antwerp
Table of Contents

- Introduction
- Laws of Modularity Combinatorics
- Combinatorics of Aggregation Dimensions
- Conclusions
Table of Contents

• Introduction
  - Sciences of the Artificial
  - Modular Software Design
  - The Power of Modularity

• Laws of Modularity Combinatorics

• Combinatorics of Aggregation Dimensions

• Conclusions
SCIENCES OF THE ARTIFICIAL

Herbert Simon
Sciences of the Artificial

- The architecture of the artificial: **HIERARCHY**
The Architecture: Hierarchy

- Interrelated subsystems, hierarchic in structure, until some lowest level of elementary subsystem.
Separation in space
Separation in time

\[(1-p)^{1000}\]

\[(1-p)^{10}\]

\[(1-p)^{10}\]

\[(1-p)^{10}\]
MODULAR SOFTWARE DESIGN

Dave Parnas – Double Dictum
• **Coupling** is a measure of the dependencies between modules
• **Cohesion** is a measure of how strongly the elements in a module are related

• Good design = **Low** coupling and **high** cohesion!

---

**Universiteit Antwerpen**
Modules – advantages

Complexity Reduction

Reuse

Evolvability
THE POWER OF MODULARITY

Carliss Baldwin & Kim Clark
The Modularity Conundrum

- Powerful systems are built of many elements
- Power comes from elements’ interplay
- This interplay results in essential interdependences and reduces ability to
  - reuse element w/o others
  - change element w/o changing others

Edges mean “knows about”
Design Rules

• Idea: have components depend on design parameters guaranteed not to change, called Design Rules
• Design rule may be a convention, interface, representation, programming language, etc.
• Design rules are the global, unhidden assumptions; they are the architecture
• Chosen properly, elements depend on design rules rather than on each other, becoming true modules
The Value of Modularity

• Modularity not only accommodates change
• It encourages innovation by decentralizing decision making on hidden modules
• Technically, it creates the option for third parties to innovate on a module
  - Parties compete to create a better module
  - A few “experiments” likely to create superior module whose value to users exceeds cost of experiments; downside minimal because can keep old
• Cluster of innovators emerge around architecture, resulting in new industry
Table of Contents

• Introduction
• Laws of Modularity Combinatorics
  - Separating Modules: Cohesive Variation Gains
  - Aggregating Modules: Coupling Ripple Costs
• Combinatorics of Aggregation Dimensions
• Conclusions
SEPARATING MODULES

On Exponential Variation Gains
Systems and Processes

- a software program
- a chair manufacturing
- a car assembly process
- a study/training program
- a business process flow
- a cough syrup production
- ...

Universiteit Antwerpen
Emerging Units and Variants
Emerging Units and Variants

- a software program
  - logic units of work
  - computation variants
- a chair manufacturing
  - chair parts
  - variants of parts
- a car assembly process
  - mounting parts
  - versions and options
- a study/training program
  - course modules
  - majors/electives
- a business process flow
  - tasks or activities
  - various rules
- a cough syrup production
  - ingredients syrup
  - various fever relievers
- ...
Global Module: Config Glue

Unit versions:
\[ \sum_{i=1}^{N} k_i \]

Glue logic
Global Module: Config Glue

- a complex program
  - all variants of all units
  - selection logic
- a chair manufacturing
  - all part versions
  - selection in process
- a car assembly process
  - all variants of all parts
  - selection and steering
- a study/training program
  - all course part versions
  - choosing in process
- a business process flow
  - all versions of all tasks
  - evaluation and selection
- a cough syrup production
  - ingredient variants
  - selection and steering
- ...

Unit versions:

\[ \sum_{i=1}^{N} k_i \]

Glue logic
Global Module: Config Variants

Unit versions: $\sum_{i=1}^{N} k_i$

Module versions: $\prod_{i=1}^{N} k_i$
Global Module: Config Variants

- a dedicated program
  - for every input set
  - input configurator
- a chair manufacturing
  - for every chair type
  - Routing configurator
- a car assembly process
  - for every possible car
  - routing configurator
- a study/training program
  - for every unique program
  - input configurator
- a business process flow
  - for every possible path
  - input configurator
- a cough syrup production
  - for every composition
  - routing configurator
- ...

Unit versions:
\[ \sum_{i=1}^{N} k_i \]

Module versions:
\[ \prod_{i=1}^{N} k_i \]
Emerging Units and Variants

- a software program:
  - logic units of work
  - computation variants
- a chair manufacturing
  - chair parts
  - variants of parts
- a car assembly process
  - mounting parts
  - versions and options
- a study/training program
  - course modules
  - majors/electives
- a business process flow
  - tasks or activities
  - various rules
- a cough syrup production
  - ingredients syrup
  - various fever relievers
- ...

Universiteit Antwerpen
Emerging Cohesive Modules

Module versions: \( \sum_{i=1}^{N} k_i \)

System variants: \( \prod_{i=1}^{N} k_i \)
Cohesive Modules

- a set of modules
  - for every unit of work
  - with various versions
- chair manufacturing units
  - for every chair part
  - with various options
- car assembly process units
  - for every car part
  - with various part variants
- study/training modules
  - for every course module
  - with various options
- business activity modules
  - for every possible task
  - with various options
- syrup ingredient units
  - for every ingredient
  - with various flavours
- ...

Module versions: \[ \sum_{i=1}^{N} k_i \]
System variants: \[ \prod_{i=1}^{N} k_i \]
Combinatorics of Modularity

Module versions: \( \prod_{i=1}^{N} k_i \)

System variants: \( \prod_{i=1}^{N} k_i \)

Module versions: \( \sum_{i=1}^{N} k_i \)

System variants: \( \prod_{i=1}^{N} k_i \)

Exponential Variation Gains

Universiteit Antwerpen
“expect families of routines to be constructed on rational principles so that families fit together as building blocks. In short, [the user] should be able safely to regard components as black boxes.”

Framework Combinatorics

<table>
<thead>
<tr>
<th>Concern type</th>
<th>Multiplicity</th>
<th>Implementations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>4</td>
<td>Postgres, HSQL, SQLServer, MySQL</td>
</tr>
<tr>
<td>Persistency</td>
<td>2</td>
<td>OpenJPA, Hibernate</td>
</tr>
<tr>
<td>Transaction</td>
<td>2</td>
<td>EJB2, EJB3</td>
</tr>
<tr>
<td>Remoting</td>
<td>2</td>
<td>RMI, WS</td>
</tr>
<tr>
<td>Controller</td>
<td>3</td>
<td>Cocoon, Struts2, Struts2-Knockout</td>
</tr>
<tr>
<td>Styling</td>
<td>2</td>
<td>Bootstrap, Plain</td>
</tr>
<tr>
<td>Access</td>
<td>2</td>
<td>JavaEE, NS</td>
</tr>
</tbody>
</table>

Module versions: \[ \sum_{i=1}^{N} k_i \]  
System variants: \[ \prod_{i=1}^{N} k_i \]
AGGREGATING MODULES

On Exponential Ripple Costs
Combinatorics of Coupling

Potential 1st impact: \[ \sum_{j=0}^{d_1} k_{j} \]

Potential 2nd impact: \[ \sum_{j=0}^{d_2} \left[ \sum_{l=0}^{d_3} k_{l} \right] \]

Change: Exponential Ripple Costs
Combinatorics of Coupling

- change a module
  - version of computation
  - with impact on interface
- change a chair leg
  - specific type of leg
  - impact on leg connection
- change a car part
  - other implementation
  - impact on fixation
- change a study module
  - other content
  - impact on prerequisites
- change an activity module
  - other implementation
  - impact on dependencies
- change a fever ingredient
  - other substance
  - interaction complication
- ...

Potential 1st impact: \( \sum_{j=0}^{d_1} k_{j1} \)

Potential 2nd impact: \( \sum_{j=0}^{d_1} \left[ \sum_{l=0}^{d_1} k_{l1} \right] \)
Coupling has Many Faces

- change a module
  - requires new version jdk
  - answer comes back async
- change a chair leg
  - hardness is damaging
  - must be mounted earlier
- change a car part
  - cooling liquid too viscous
  - should be fixated earlier
- change a study module
  - exam facilities limited
  - requires other module first
- change an activity module
  - other authorization scheme
  - other task prerequisite
- change a fever ingredient
  - impact on solubility
  - needs direct pressurization
- ...

Potential 1st impact:
\[ \sum_{j=0}^{d_i} k_{ij} \]

Potential 2nd impact:
\[ \sum_{j=0}^{d_i} \left[ \sum_{l=0}^{d_j} k_{ij} \right] \]
Combinatorics of Re-Use

Change: Exponential Ripple Costs
Combinatorics of Re-Use

• re-use a module
  • in another program
  • changed the interface
• re-use a chair leg
  • in another chair or table
  • changed the leg connection
• re-use a car part
  • In second car or motorbike
  • changed the part fixation
• re-use study modules
  • in another program
  • changed the outcomes
• re-use activity module
  • in other processes
  • changed the dependencies
• re-use fever ingredient
  • in another medicine
  • changed the strength
• ...

Universiteit Antwerpen
Modularity Combinatorics

Unleashing the exponential variation gains, without mastering the coupling that entails the exponential ripple costs, may destroy the evolvability.
The Reality: Manny Lehman

The Law of Increasing Complexity
Manny Lehman

“As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it.”

Modularity is static
Table of Contents

• Introduction
• Laws of Modularity Combinatorics
• Combinatorics of Aggregation Dimensions
  - Multiple dimensions in modular structures
  - Architectures Connecting Modular Dimensions
  - Integrated Elements that Exhibit Evolvability
• Conclusions
MULTIPLE DIMENSIONS IN MODULAR STRUCTURES
Hierarchical Aggregation
The Integration Problem

- Making a software application highly robust and secure may imply redevelopment
- Adding water and electricity to a construction plan may imply several rollbacks
- Taking into account weight/vibration/emission constraints may imply redesign of an engine
- ...

Universiteit Antwerpen
Hierarchical Aggregation
Introducing Other Concerns

- Adding persistency or access control to all modules containing data
- Adding electrical power and control to the majority of car parts
- Asking internationalization and case studies from course modules
- Demanding secure registration or logging from process activities
Ripple Costs due to Concerns

Potential 1st impact: \[ \sum_{i=0}^{N} k_i \]

Potential 2nd impact: \[ \sum_{i=0}^{N} \sum_{j=0}^{d_i} k_{ij} \]

Potential 3rd impact: \[ \sum_{i=0}^{N} \sum_{j=0}^{d_i} \sum_{l=0}^{d_j} k_{il} \]
ARCHITECTURES CONNECTING MODULAR DIMENSIONS
Embedded Dedicated Implementations

- Adding dedicated persistency module to every functional module
- Adding dedicated generator or battery to the majority of car parts
- Develop separate case study module for every course module
- Introduce dedicated secure registration module for every activity
Embedded Standardized Implementations

- Adding standardized persistency module to every functional module
- Adding standardized battery unit to the majority of car parts
- Select standard case study module for every course module
- Introduce standardized secure registration module for every activity
Relay Modules to Dedicated Frameworks

- Adding relay modules to connect to integrated persistency framework
- Adding power units connecting to dedicated electrical network
- Retrieving case study module from dedicated unit to develop cases
- Connecting to a dedicated framework for secure registration of activities
Architectures: 1, 2, and 3
Relay Modules to Standardized Frameworks

- Adding relay modules to tap into immanent standard solution like JPA
- Adding standard connectors to tap into standardized electrical network
- Retrieving case study modules from public provider through keywords
- Adding relay modules to connect to provider of secure logging services
Relay Modules to Framework Gateways

- Relaying persistency authorization to dedicated gateway shielding provider
- Power connectors relaying to central power unit shielding actual provider
- Relaying to dedicated semantic gateway shielding public case provider
- Relaying to central gateway shielding actual secure logging service provider
Architectures: 3, 4, and 5
Aggregating Concerns: Case
INTEGRATED ELEMENTS
THAT EXHIBIT EVOLVABILITY
Some Attempts Towards Evolvability and Scalability
Some Attempts Towards Evolvability and Scalability

Google Modular Phones

Universiteit Antwerpen
Some Attempts Towards Evolvability and Scalability
The Emergence of Elements

• Additional concerns should be systematically distributed throughout the functional structure
• The integration of such a concern should be done in a properly encapsulated/isolated way
• For every concern, an architecture needs to be designed, as standardized as possible
• The integration skeletons need to be provided as deep down as possible in modular structure
The Emergence of Elements
Concept Element: Software

Anthropomorphism

Invoice Element

- Struct Invoice
  - Nr
  - Date
  - ...

Encapsulate

Access control

Persistency

Remote access
Concept Element: Construction

- Structure
- Support core
- Protection
- Isolation
- Water
- Electricity
- Socket
Normalized Systems Elements

• The ultimate goal is to *normalize systems*:
  → to have a normalized transformation:
    *between function and construction*

    \[
    \begin{pmatrix}
    C_1 \\
    \vdots \\
    C_n
    \end{pmatrix}
    =
    \begin{pmatrix}
    0 \\
    \vdots \\
    0
    \end{pmatrix}
    \xrightarrow{x}
    \begin{pmatrix}
    F_1 \\
    \vdots \\
    F_1
    \end{pmatrix}
    \]

  → based on elementary building blocks:
    *that solve the m:n design problem*

Universiteit Antwerpen
Imagine …

- Houses that could be reconfigured or extended without additional plumbing or electricity works …
- Roads and footpaths that could be moved entirely without additional drilling for sewer or cable networks …
- Software systems that could be maintained and evolved without increasing costs and ripple effects …
- Cars, airplanes, and rockets that could be scaled up and down to carry more or less passengers or freight …
- …
Imagine ...

The sciences of the evolving and growing artificial
Table of Contents

• Introduction
• Laws of Modularity Combinatorics
• Combinatorics of Aggregation Dimensions
• Conclusions
Conclusions

- We should study more modularity in general, and characteristics of modular patterns
- There is a tendency to introduce modules for flexibility, and to reap the variation gains
- Unleashing variation gains without mastering coupling, may be very harmful to evolvability
- We should address the multi-dimensional nature of architectural design problems
- The concept of multi-concern elements may contribute to growing and evolving artefacts
Questions