Modularity

Value and Measure

Alar Raabe

Content

- Introduction
  - Definitions
  - Goals
- Software Metrics
  - Measures
  - Scales
- Value of Modularity
  - Soft Value
  - Economic Values
- Measure of Modularity
- Discussion
  - What to Measure?
  - How to Proceed?

Introduction – Definitions 1

System
- a collection of interacting components organized to accomplish a specific function or set of functions within a specific environment

Interface (Connection)
- a shared boundary between two functional units, defined by various characteristics of the functions
- component that connects two or more other components for the purpose of passing information from one to the other

Module (Component)
- a logically separable part of a system

Encapsulation
- isolating some parts of the system from the rest of the system
- a module has an outside that is distinct from its inside (an external interface and an internal implementation)

Introduction – Definitions 2

Modularity

- the degree to which a system is composed of discrete components such that a change to one component has minimal impact on other components
- the extent to which a module is like a black box

Coupling

- the manner and degree of interdependence between modules
- the strength of the relationships between modules
- a measure of how closely connected two modules are

Cohesion

- the manner and degree to which the tasks performed by a single module are related to one another
- a measure of the strength of association of the elements within a module

Introduction – Goals

Design Principles [Meyer]

- **Few Interfaces** – every module should communicate with as few other modules as possible
- **Small Interfaces** – if two modules communicate at all, then they should exchange as little information as possible
- **Explicit Interfaces** – when two modules communicate, this fact must be obvious
- **Information Hiding** – all information about a module should be private to that module unless it is specifically declared public

Goals

- For systems
  - High modularity
  - Balanced complexity of modules and overall system
  - Low coupling of modules
- For modules
  - High cohesion
  - (High) Encapsulation

Software Metrics – Measures 1

Line-of-Code (LOC)

LOC is a count of all the code lines (source, blank, and comment)
SLOC is the number of executable code lines

Cyclomatic Complexity [McCabe]

a measure of the complexity of a module's decision structure
the number of linearly independent paths through a module
(indicates of how much effort is required to test a module)

$$v(G) = N_{edges} - N_{nodes} + N_{connected\ components}$$

Variations

Essential complexity $ev(G)$, measures the degree to which a module contains unstructured constructs

Design complexity $iv(G)$, measures module's decision structure as it relates to calls to other modules

Software Metrics – Measures 2

Software Science [Halstead]

Measurable properties

n_1 – number of unique or distinct operators
n_2 – number of unique or distinct operands
N_1 – number of total usage of all the operators
N_2 – number of total usage of all the operands

Calculated properties

$n = n_1 + n_2$ – vocabulary
$N = N_1 + N_2$ – length ($N' = n_1 log_2 n_1 + n_2 log_2 n_2$)
$V = N\ log_2 n$ – volume

$I = (2Vn_2)/(n_1 N_2)$ – intelligence content (complexity of algorithm)

Software Metrics – Measures 3

OO Metrics [Chidamber & Kemerer]

Lack of Cohesion in Methods (LCOM)

$$LCOM = P - Q \text{ if } P > Q$$
$$= 0 \quad \text{otherwise}$$

where

P is the number of pairs of methods that do not share instance variables

Q is the number of pairs of methods that share instance variables

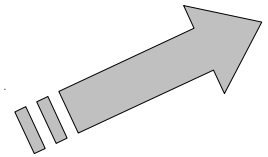
Others

- Coupling Between Object Classes (CBO)
  - number of other classes to which it is coupled
- Weighted methods per Class (WMC)
  - sum of the complexities of the methods of a class
- Depth of Inheritance Tree of a class (DIT)
- Number Of Children of a class (NOC)
- Response Set for a class (RFC)
  - cardinality of the set of methods that can be executed (directly or indirectly) in response to a message received by an object of that class

Software Metrics – Measures 4

Function Points [Albrecht]

- Counting
  - external inputs
  - external outputs
  - logical internal files
  - external interface file
  - external inquiry



	Simple	Average	Complex
External Input	3	4	6
External Output	4	5	7
Logical Internal File	7	10	15
External Interface File	5	7	10
External Inquiry	3	4	6

- Adjusting with
  - type
  - complexity
  - system characteristics (degree of influence 0..5)
    - data communication, distributed functions, performance, heavily used configuration, transaction rate, on-line data entry, end user efficiency, online update, complex processing, reusability, installation ease, operational ease, multiple sites, facilitate change

$$FP = \left[\sum_{i=1}^{n} Weight_i\right] \times \left[0.65 + 0.01 \times \sum_{j=1}^{14} DegreeOfInfluenceOfGSC_j\right]$$

Software Metrics – Measures 5

Entropy [Allen]

What is the average number of bits needed to describe the dependencies a program unit has on the rest of the system?

Entropy (Average bits per node)

$$H(S) = \sum 1/n \; (-log_2 \; P_L)$$

where S is a CDG (Code Dependency Graph) and n is the no. of nodes in S

Total amount of information

$$I(S) = \sum -log_2 \; P_L$$

Coupling

$$C(S) = \sum I(S_i) - I(S)$$

where i ranges on number of nodes

Software Metrics – Scales 1

Coupling (0..7) [Constantine & Yourdon]

- **Content** (high) – one module modifies or relies on the internal workings of another module
- **Common** – two modules share the same global resource (e.g. data)
- **External** – two modules share an externally imposed data format or communication protocol
- **Control** – one module is controlling the logic of another, by passing it information on what to do
- **Stamp** (Data-structured coupling) – modules share a composite data structure and use only a part of it, possibly a different part (e.g. passing a whole record to a function which only needs one field of it)
- **Data** – modules share data through, for example, parameters
- **Message** (low) – modules use a public interface to exchange parameter-less messages (or events)
- **No coupling** – modules do not communicate at all

Software Metrics – Scales 2

Cohesion (0..6) [Constantine & Yourdon]

- **Functional** (best) – parts of a module are grouped because they all contribute to a single well-defined task of the module
- **Sequential** – parts of a module are grouped because the output from one part is the input to another part like an assembly line (e.g. a function which reads data from a file and processes the data)
- **Communicational** – parts of a module are grouped because they operate on the same data (e.g. a module which operates on the same record of information)
- **Procedural** – parts of a module are grouped because they always follow a certain sequence of execution (e.g. a function which checks file permissions and then opens the file)
- **Temporal** – parts of a module are grouped by when they are perfromed (e.g. functions performed after an exception)
- **Logical** – parts of a module are grouped because they logically are categorised to do the same thing (e.g. I/O routines)
- **Coincidental** (worst) – parts of a module are grouped arbitrarily (e.g. frequently used mathematical functions)

Value of Modularity – Soft Value

Maintainability

the ease with which a software system or component can be modified to change (flexibility) or add (extendability) capabilities, correct faults or defects, improve performance or other attributes, or adapt to a changed environment

Portability

the ease with which a system or component can be transferred from one hardware or software environment to another

Reliability

the ability of a system or component to perform its required functions under stated conditions for a specified period of time

Testability

...

Value of Modularity – Economic Value 1

Real Option Theory [Baldwin & Clark]

- Modularity
  - is a financial force
  - accommodates future uncertainty
  - creates choices that can be exercised in future
- Design Structure Matrix – dependency matrix of design parameters
- Modular Operators – correspond to options
  - split, substitute, exclude, augment (add), inversion, porting
- Valuation of modularity as Real Options (American call)
  - Decision trees with probabilities (Markov Processes)
  - Dynamic programming algorithms
  - Monte Carlo simulations
  - ...
- Value

$$\mathrm{NPV}_{\text{strategic}} = \mathrm{NPV}_{\text{traditional}} + \mathrm{Value}_{\text{real options}}$$

Value of Modularity – Economic Value 2

Real Option Theory

Qualitative Design Principles [Sullivan]

If at any time, the expected value of future profits discounted to given time is at least by value of investment opportunity more than the direct costs, then commit to the design decision, otherwise do not

If the expected present value of the future profits that would flow from choice exceeds the direct cost of implementing it, then go ahead and implement the choice, otherwise implement other choice

If the expected present value of future profits that would flow from restructuring exceeds the direct cost of restructuring, then go ahead and restructure, otherwise do not

If the cost to effect a software decision is sufficiently low, then the benefit of investing to effect it immediately outweighs the benefit of waiting, so the decision should be made immediately

With other factors, including the static NPV, remaining the same, the incentive to wait for better information before effecting a design decision increases with risk (ie, with the spread, in possible benefits)

The incentive to wait before investing increases with the likelihood of unfavourable future events occurring

All else being equal, the value of the option to delay increases with variance in future costs

Measure of Modularity

Augmented Constraint Networks (constraint network)
- Design dimensions – variables
- Possible choices – values
- Logical constraints – relations among decisions
- PWDR (pair-wise dependence relation) – two variables are PWD, if change of one causes change of other

ACN → non-deterministic automaton (Design Automaton) representing change dynamics within a design space

Modularity Properties
- Complexity – #Variables (# of involved design dimensions)
- Dependency Density – Density = #PWDR / #Variables (coupling)
- Net Options Value –

$$V = S_0 + NOV_1 + NOV_2 + \ldots + NOV_m$$

$$NOV_i = max_k\{\sigma_i n_i^{1/2} Q(k_i) - C_i(n_i)k_i - Z_i\}$$

where for module i
- $\sigma_i n_i^{1/2} Q(k_i)$ – expected benefit (σ being technical potential, and n complexity)
- $C_i(n_i)k_i$ – cost to run k_i experiments
- Z_i – cost of changing the module

Modular vector

$$\Delta(D' - D) = <\Delta size, \Delta density, modifications, \Delta now>$$

Discussion

What to Measure?

- Complexity of overall system – complexity of interconnections of modules
- Complexity of modules
- Option value
- ...

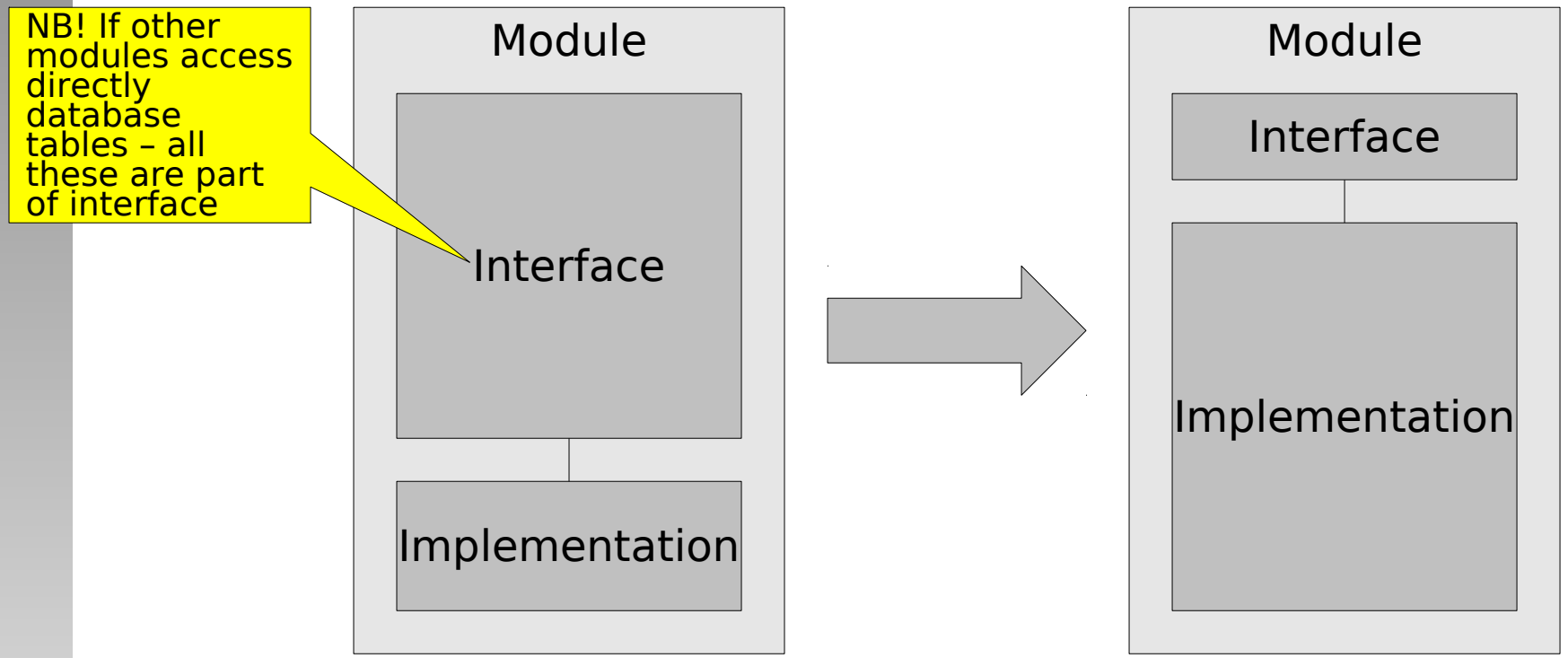
How to Proceed?

- Measure should be easily calculated
- Measure shoul behave intuitively
- ...

Discussion

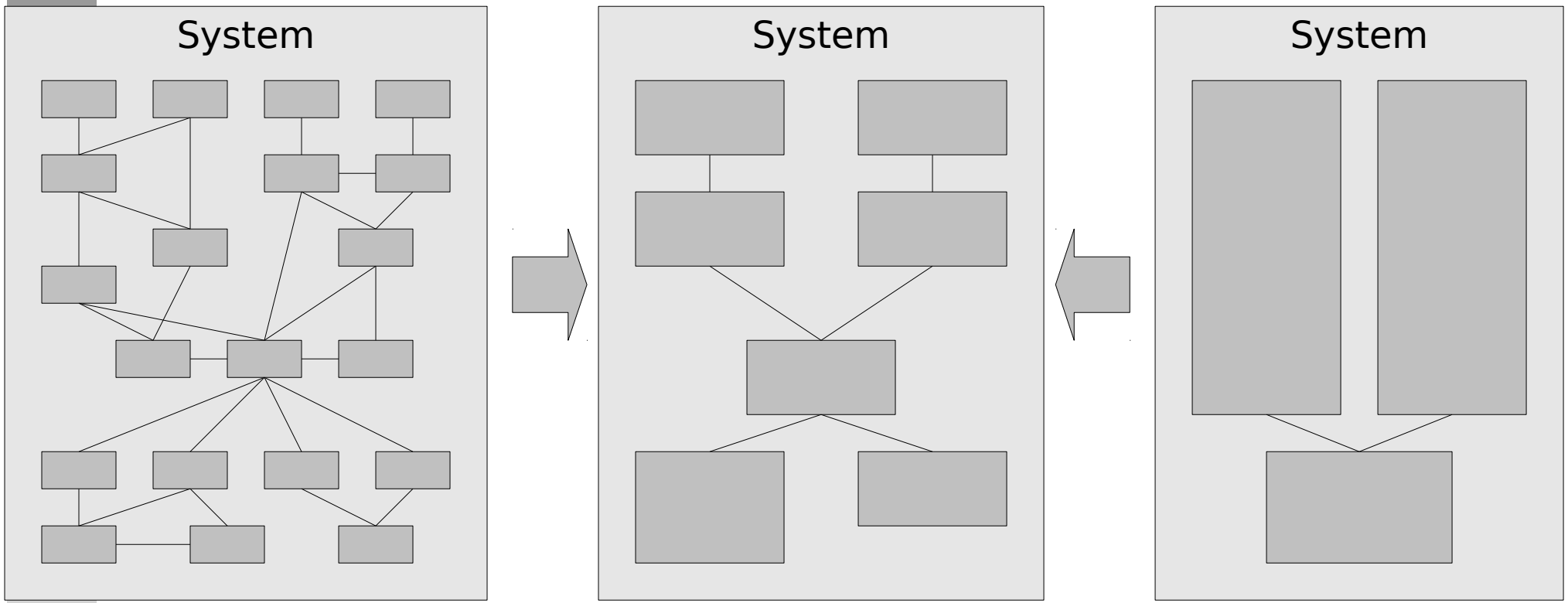
Example: Interface Complexity

Small/Simple Interface & Large/Complex Implementation



Example: Overall Complexity

Balance between system and modules volume/complexity



Measuring Modularity

System Modular Complexity

$$C_{modular} = \sqrt{N^2_{modules} + N^4_{moduleconnections}}$$

Module Complexity

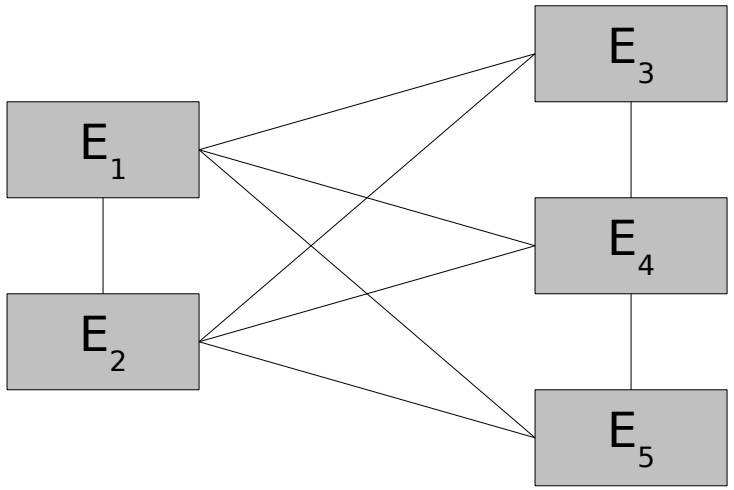
$$C_{module} = \sqrt{N^2_{elements} + N^2_{connections}}$$

System Overall Complexity

$$C_{system} = \sqrt{N^2_{elements} + N^2_{connections}}$$

System Modularity

$$M = \frac{1}{\sqrt{C^2_{modular} + C^2_{largest\,module} + C^2_{system}}}$$

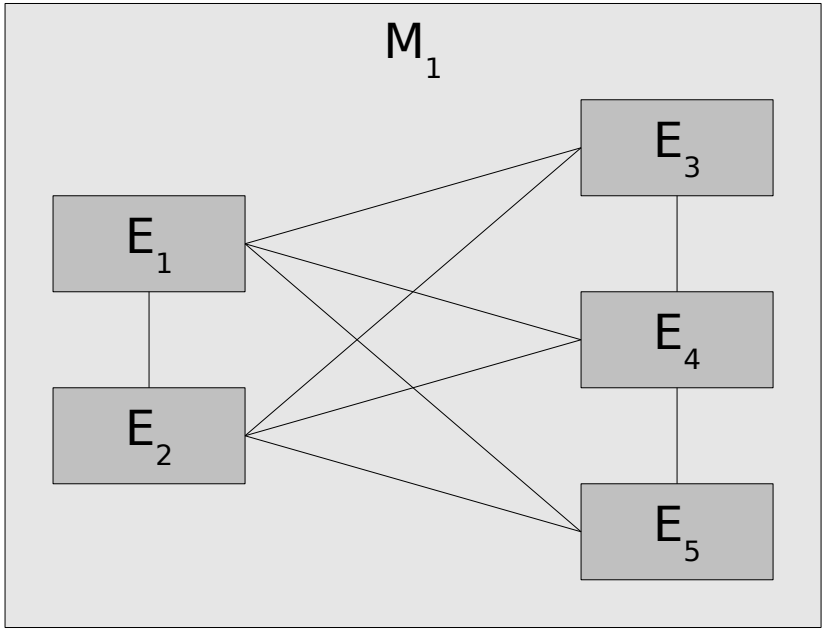


Example 1

System Modularity = 0.07 (!?!)

Modularity is lowest
Single too complex module

Rank 0



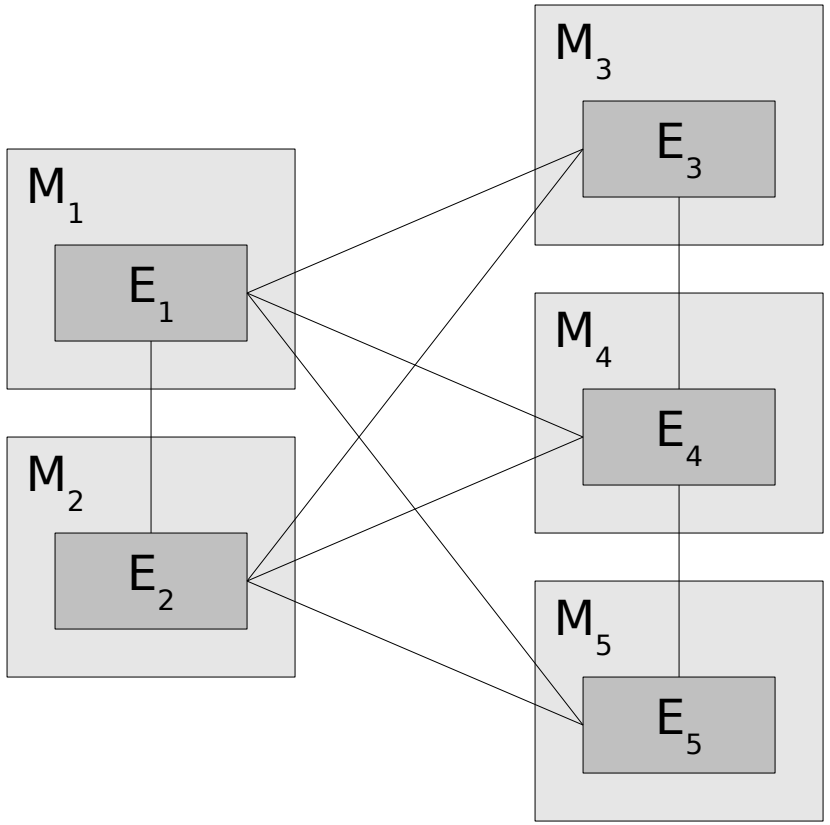
Example 2

System Modularity = 0.01

Modularity is lowest

Too many very simple modules

Rank 0

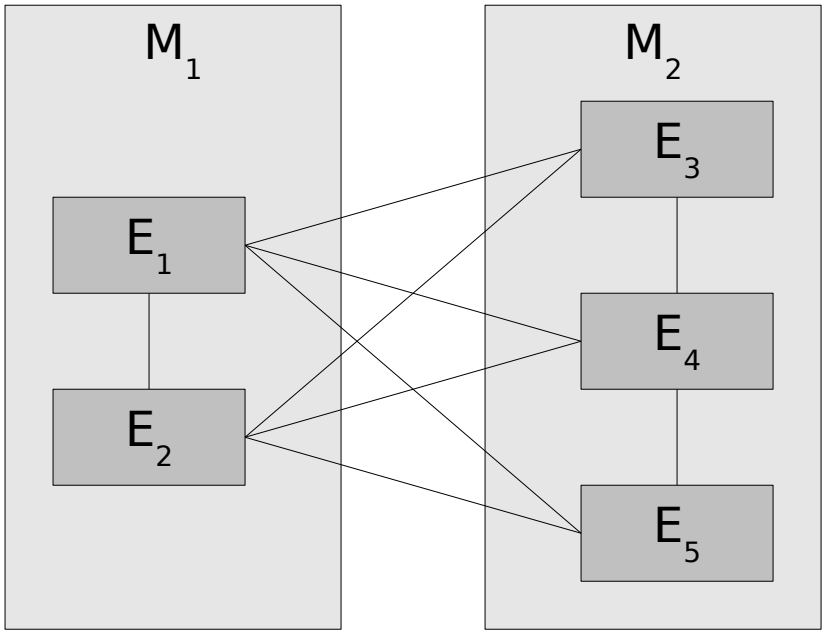


Example 3

System Modulrity = 0.03

Modularity is low

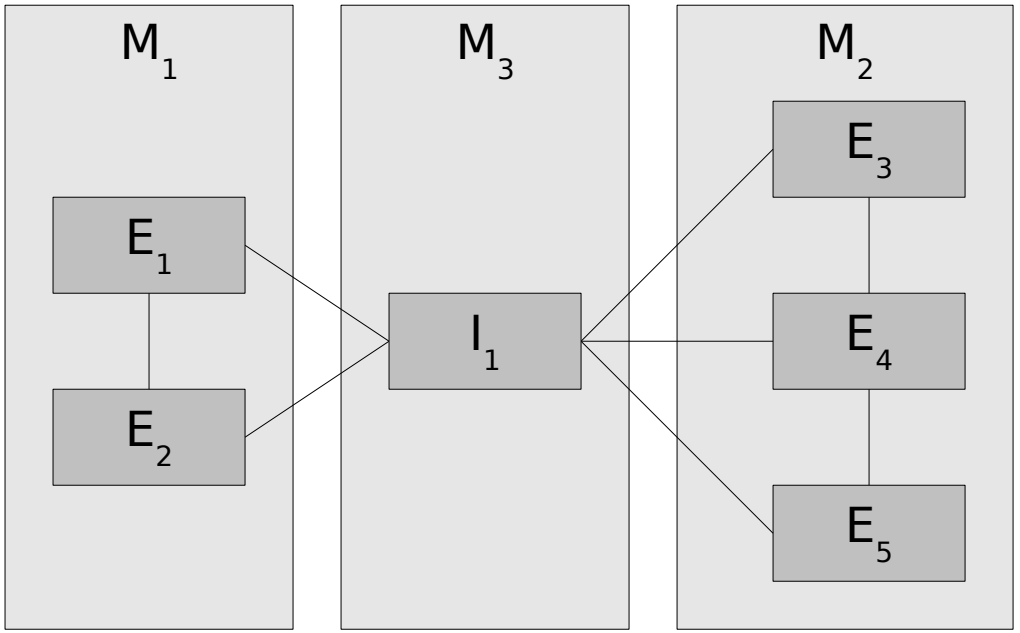
Too many interconnections between modules



Rank 1

Example 4

System Modularity = 0.04



Modularity is low

Too many modules (interface in own module) and inter-connections

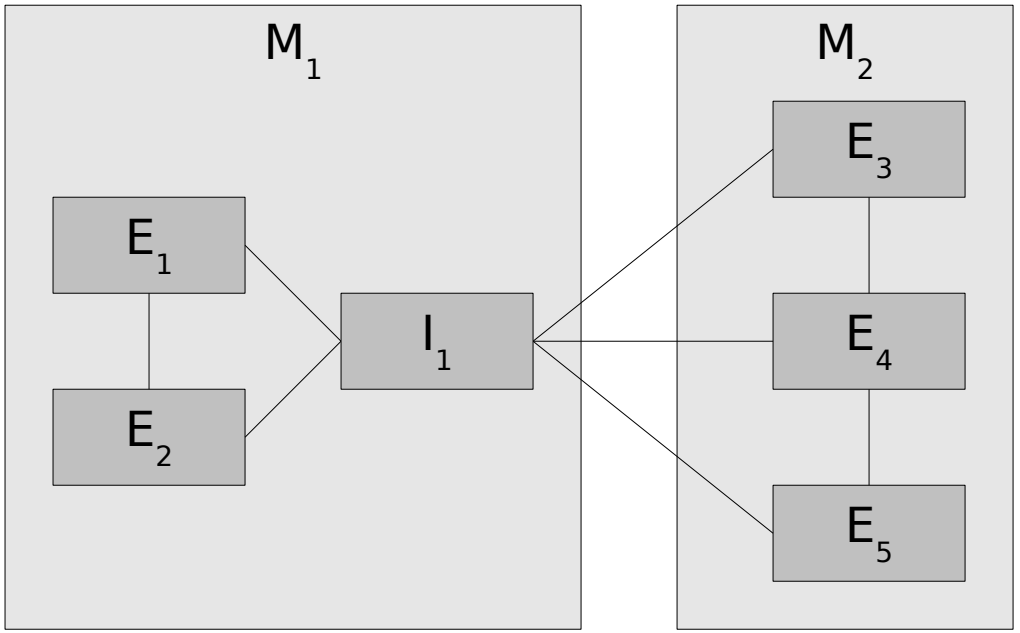
Rank 1

Example 5

System Modularity = 0.08

Modularity is normal

Still too many inter-connections between modules



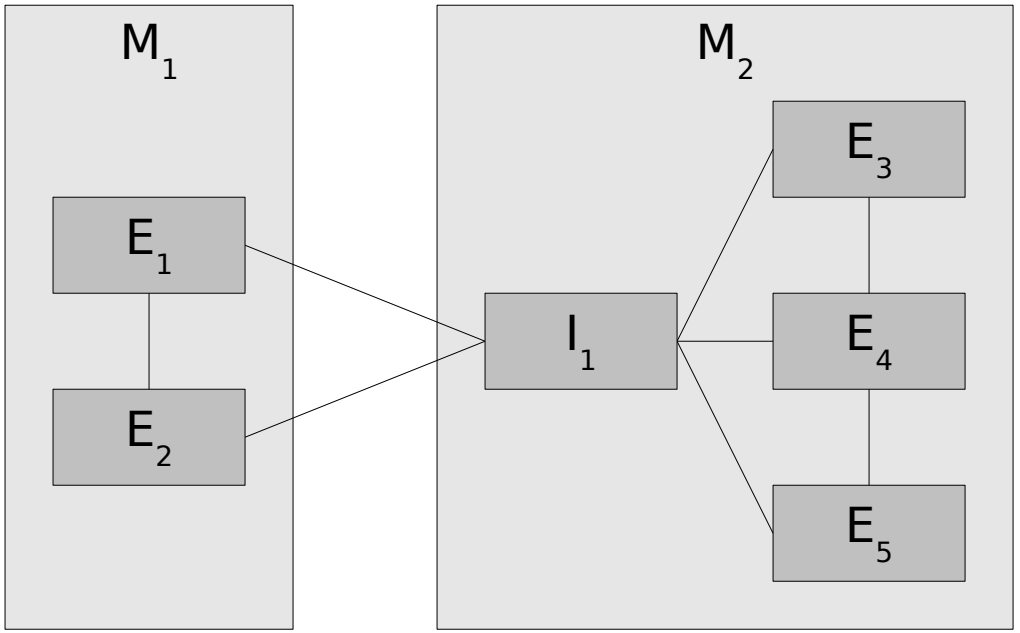
Rank 2

Example 6

System Modularity = 0.09

Modularity is highest

Optimal amount of interconnections between modules



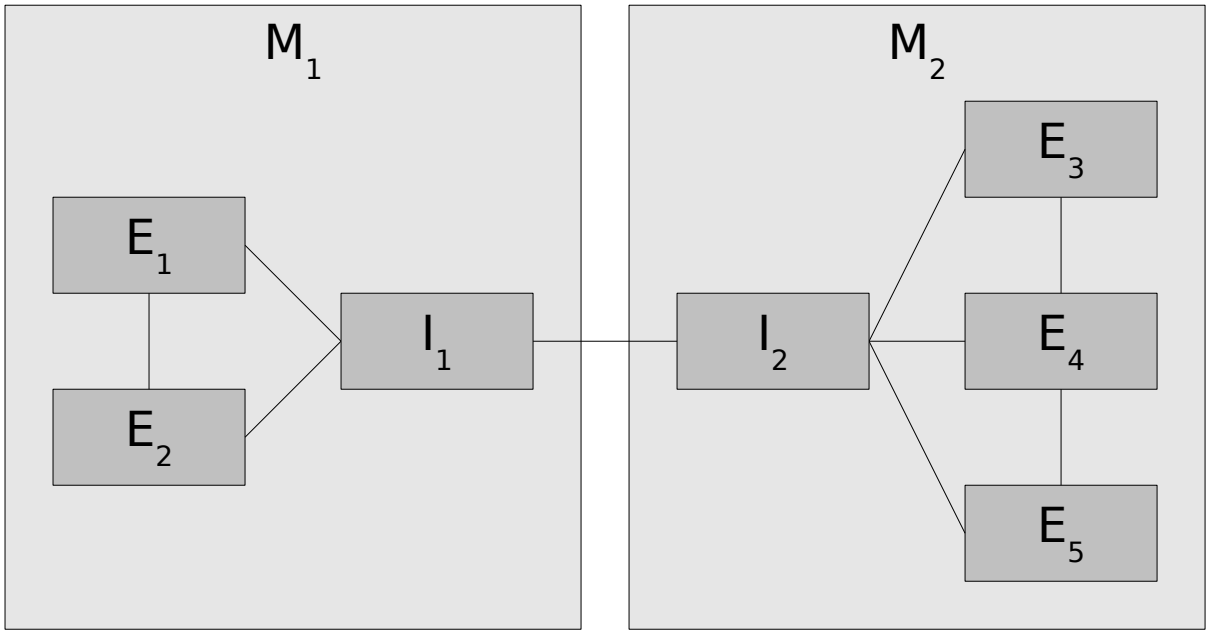
Rank 3

Example 7

System Modularity = 0.08

Modularity is low

Too many inter-face elements



Rank 2

Calculations

	1	2	3	4	5	**6**	7
Number of Modules	1	5	2	3	2	**2**	2
Number of Module Interconnections	0	9	6	5	3	**2**	1
Number of Elements	5	5	5	6	6	**6**	7
Number of Element Interconnections	9	0	3	3	5	**6**	8
Elements in Largest Module	5	1	3	3	3	**4**	4
Connections in Largest Module	9	0	2	2	3	**5**	5
Elements in Smallest Module	5	1	2	1	3	**2**	3
Connections in Smallest Module	9	0	1	0	2	**1**	3
Modularity (info) ?	23.26	23.26	13.51	14	10.75	**14.78**	13.19
Modularity of System (info) ?	0	23.26	13.51	14	10.75	**14.78**	13.19
Modularity of Largest Module (info) ?	23.26	0	4	4	6	**11.61**	11.61
Complexity (as vector)	14.59	81.31	36.7	26.31	12.81	**11.53**	12.61
Modularity (as vector)	0.07	0.01	0.03	0.04	0.08	**0.09**	0.08