



Using Domain Model For Structuring Pattern Language

Veli-Pekka Eloranta, Marko Leppänen and
Kai Koskimies
{firstname.lastname@tut.fi}

NW-Mode'09 28.8.2009



Patterns

"There are no patterns in Gang of Four book!"



Patterns

"There are no patterns in Gang of Four book!"

By definition:

*A design pattern is a **general reusable** solution to commonly occurring problem in a context*



Pattern languages

- A pattern language is a structured way of describing good design practices within a field of expertise.
- Originally introduced by Christopher Alexander in a book: *A Pattern Language*
- Coplien: Collection of patterns that build on each other to generate a software system



Pattern languages (cont)

- Definitions on the term differ
- No systematic way to structure a pattern language



Ways of structuring pattern languages

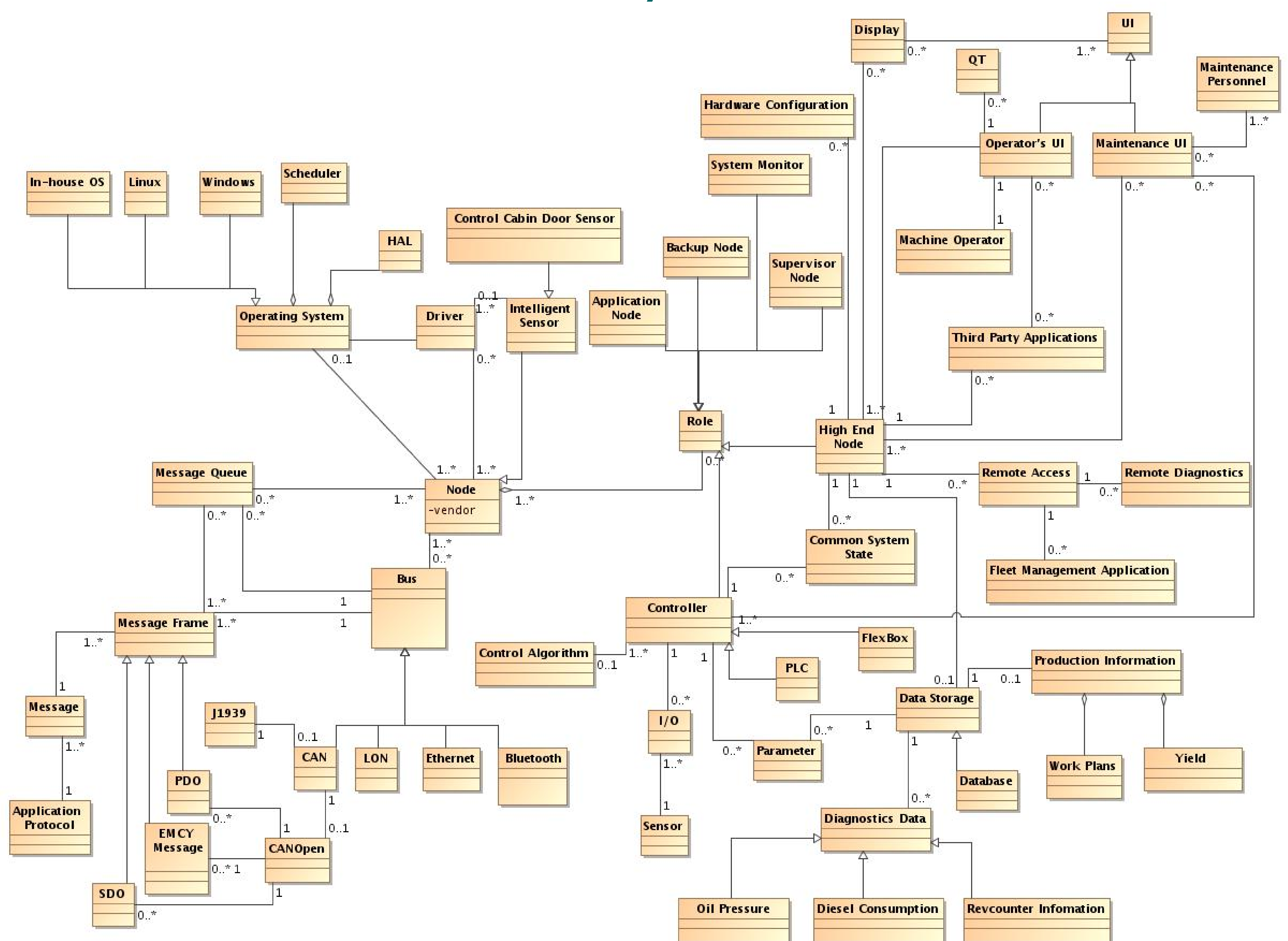
- Different ways of organizing a pattern language can be divided into two categories: Problem-centered and solution-centered
- Problem-centered languages build on the problem that patterns solve whereas solution-centered are based on the solution



Project background

- ATAM evaluations in Finnish machine industry
- 10 evaluations in total
- Collected 38 patterns from evaluations
- Domain model for machine control domain

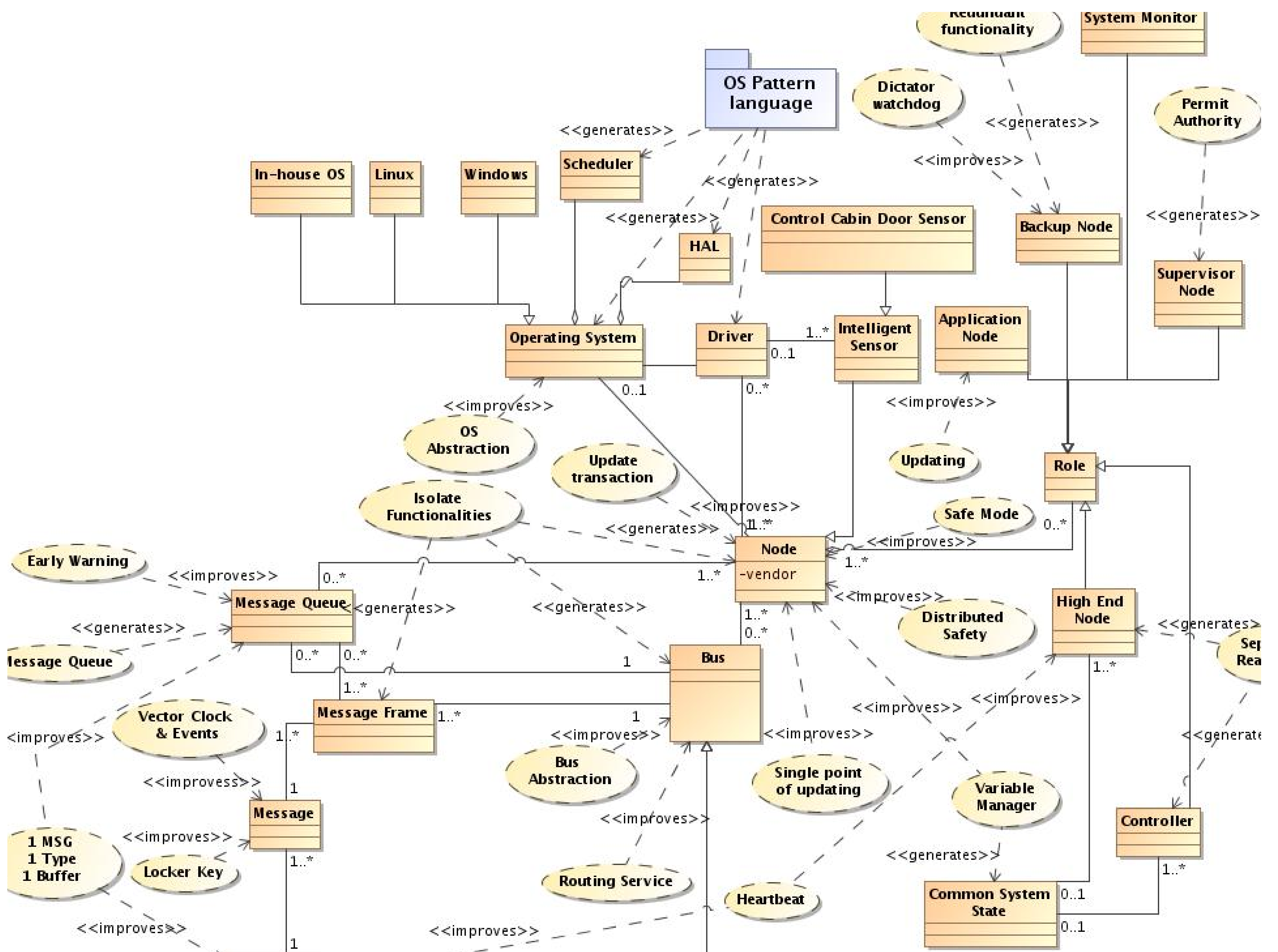
Domain model for distributed embedded control system domain





Annotating model with patterns

- Patterns were attached to entities that are closely related to pattern
- Scenarios that had resulted in patterns was analysed
- Evaluation results of scenarios was analysed
- Domain model entities was identified from evaluation results
- Relationship of a pattern and the domain was refined





Different kinds of relationships

- Generates – solution described in pattern generates the entity in domain model
- Improves – solution described in pattern improves the solution behind the entity
- **Is related** – this relationship was used initially, but it was unnecessary



Domain model and multiple pattern languages

- Pattern languages can intersect, i.e. a part of one language could be a part of another
- For example, pattern language for operating systems
- Drawn in annotated domain model as a package



Structuring pattern language using domain model

Method consists of following steps

1. Domain model must meet certain pre-requisites
2. List of equations are derived from the domain model according to rules
3. Equations are drawn as a pre-pattern language graph
4. Pattern language graph is finished by combining nodes of the pre-language graph



Building pattern language from domain model - prerequisites

1. Each entity E_i must be generated at most once
2. If pattern P generates an entity E which aggregates other entities $\{E_1, E_2, \dots, E_n\}$ all aggregated entities are implicitly generated by pattern P
3. If pattern P generates an entity E which has child entities $\{E_1, E_2, \dots, E_n\}$ then child entities can not be generated by any other pattern
4. If E has subclass entities $\{E_1, E_2, \dots, E_n\}$, either E must be generated by a pattern or all subclass entities $\{E_1, E_2, \dots, E_n\}$ must be generated, but not both.



Building pattern language from domain model – rules

1. If E is generated by P , $E = P$
2. If P improves E then $E > P$
3. If E is subordinately connected to E' then $E < E'$
4. If E is equally connected to E' , then $E = E'$



Building pattern language from domain model – derived facts

For example we can derive following facts

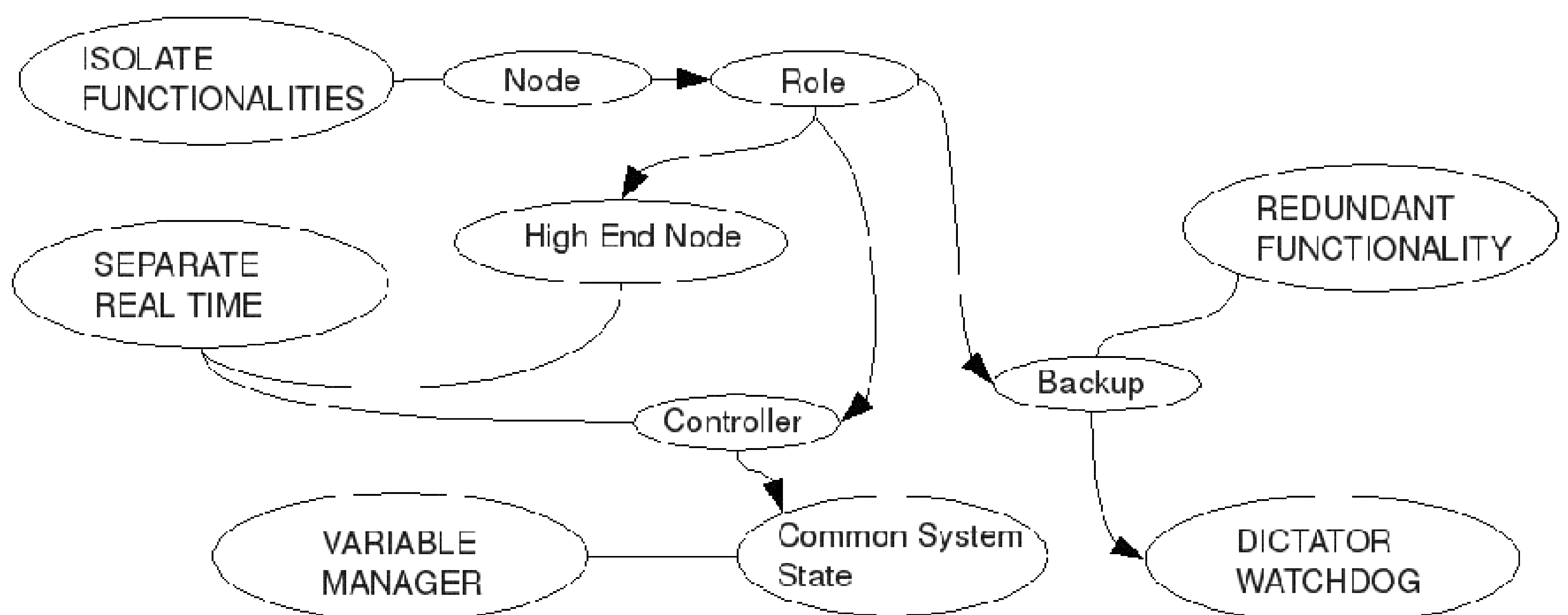
- ISOLATE FUNCTIONALITIES = Node, [by rule #1](#)
- Node > Role, [by rule #3](#)
- High End Node < Node, [by rule #4](#)
- VARIABLE MANAGER < Node, [by rule #2](#)



Building pattern language from domain model – from rules to graph

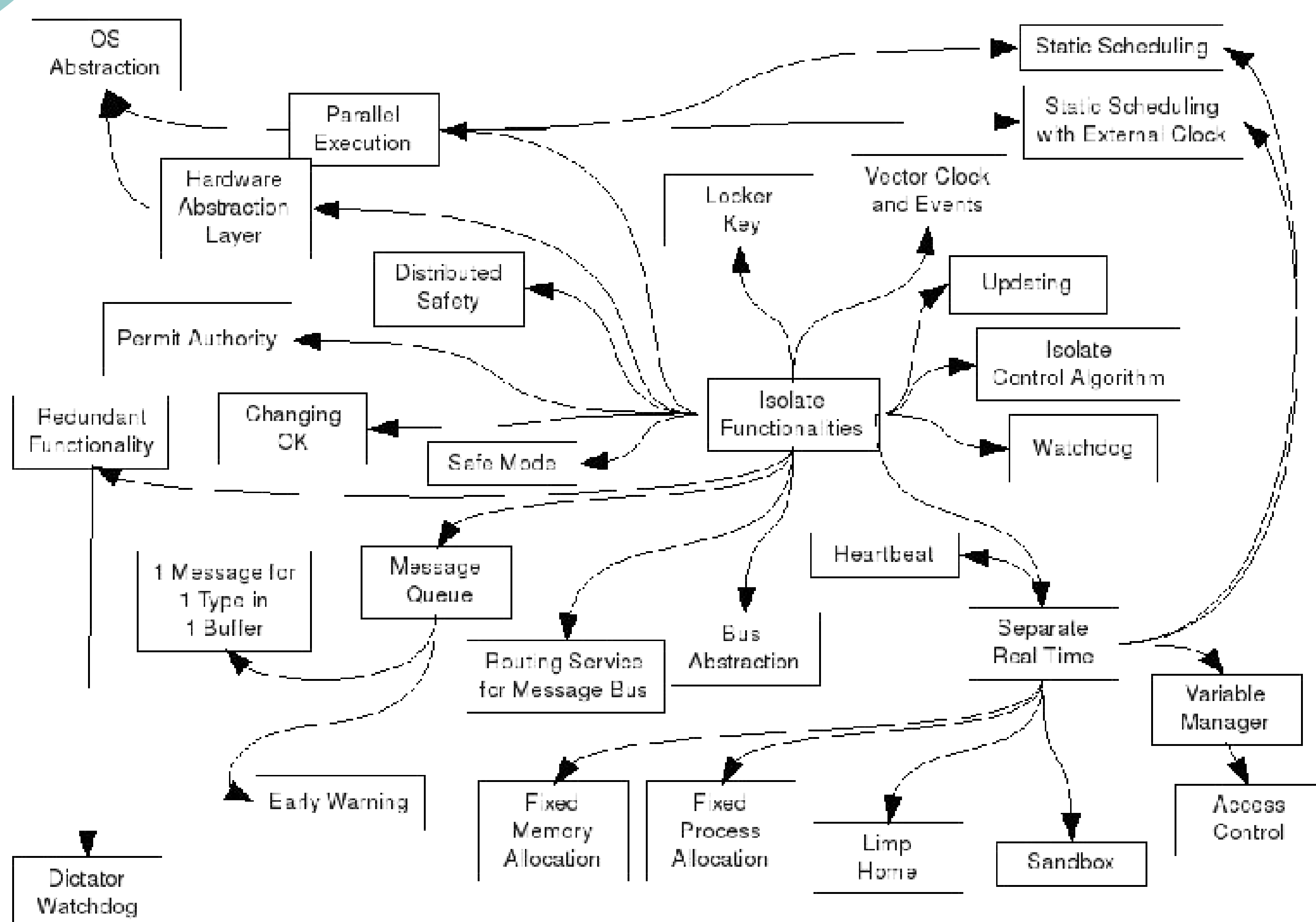
- Order of patterns and entities can be determined from the equations
- The smallest pattern and entities related to it are drawn first
- Equally connected entities and patterns are drawn with a line in between of them
- Subordinately connected are drawn with arrow

Building pattern language from domain model – pattern entity graph

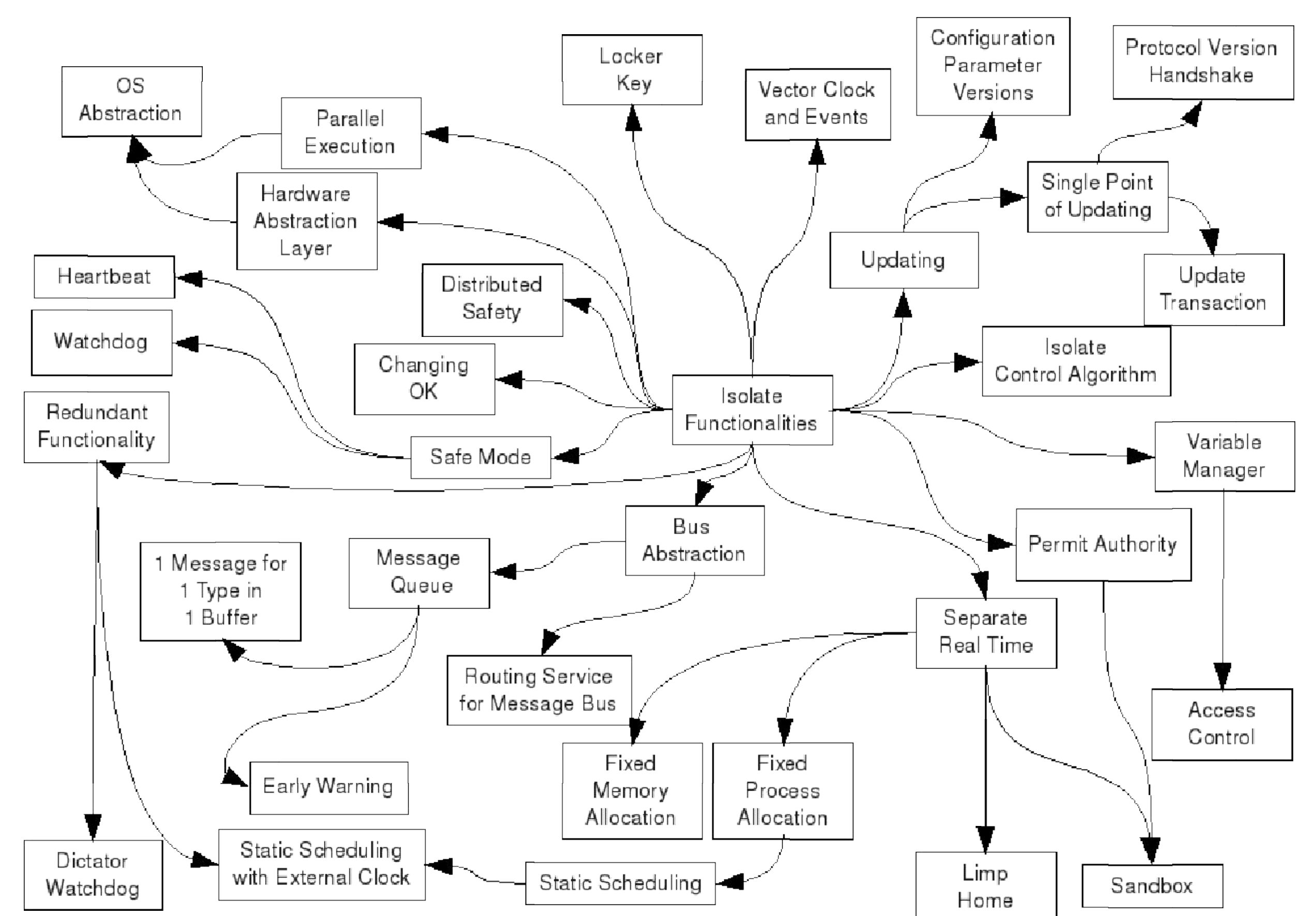


Results

Language structured using the method



Intuitively structured language





Pros and cons of the method

- Gives a systematic way to create pattern language
- Structured language can be used as a starting point for intuitive structuring
- May reveal gaps in the pattern collection
- Requires a domain model, which may be hard to create



Future research topics

- Could the method be automatized with a tool?
- How sensitive is the method for changes in the domain model?
- Applying the method on different domains



Thank you

Questions

