# Flexible Applications
# &
# Ampersand

## Agenda

- Goals & Principles
  - Variation points
  - Architecture of prototypes
  - New features of Ampersand

## Goals & Objectives of Ampersand

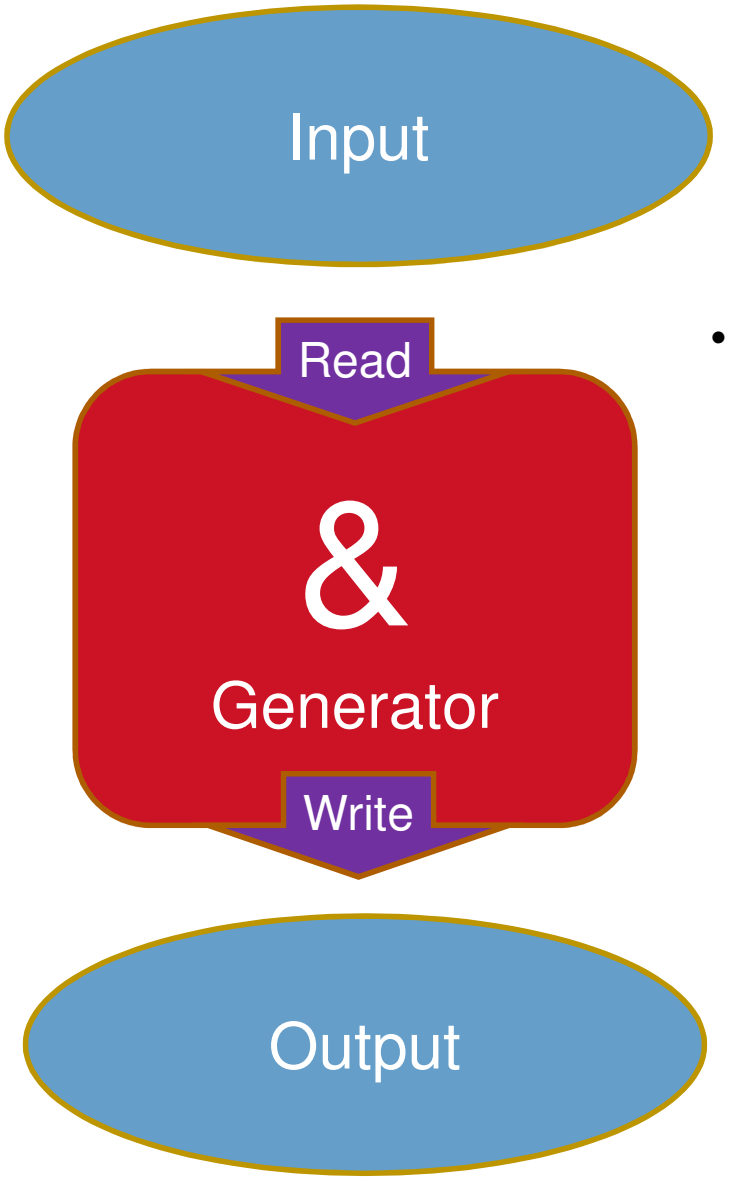
- Support businesses with systems that behave in desired ways
  - Transform business rules into compliant software
    - System behaviour mimics rules correctly
    - Business stakeholders can validate correctness
    - Software is designed for change

## Agenda

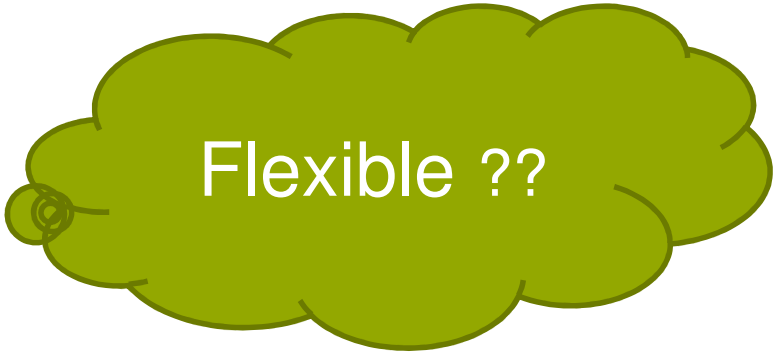
- Goals & Principles
- **Variation points**
- Architecture of prototypes
- New features of Ampersand

## Change ahead!

- Business domains
- Business rules
- Business data
- (User) Interfaces
- …




- Formalized Specification:
  - Concepts
  - Relations
  - Rules
  - Interfaces
  - ...
- Configuration (optional)

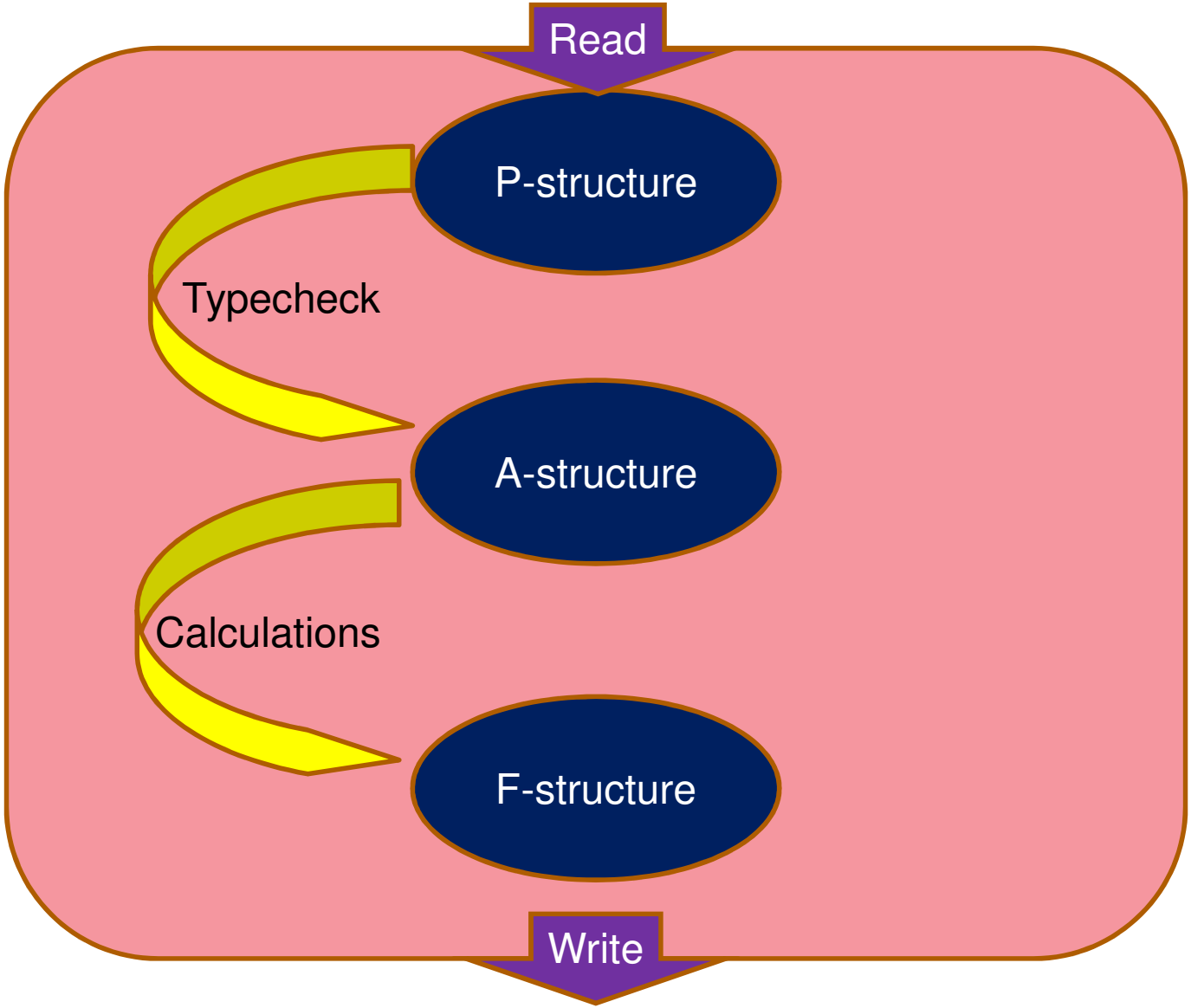



- Functional prototype
- Documentation
  - Domain language (for business)
  - Diagnosis (for &-analist)
  - Functional specification (for IT)
- Technical artefacts (mainly for QA)
- ‘Meatgrinder’ (experimental)

# Flexibility in input

| Variation point | Approach | Flexible? |
|---|---|---|
| Business specific (concepts, relations, rules) | Isolation (In &-model) | ++ |
| Rule types | Behavioural rules (~~Decision rules~~) | - |
| Interfaces | In model:<br>• Which interfaces<br>• What information<br>• What can be modified<br>• Who is it for<br>In configuration:<br>• Look & feel | + |
| Business data | Both in model (design time) and RDBMS (runtime) | ++ |
| Formal definitions | Model must be correct | -- |
| Format of definitions | Currently plain text files. Different formats could be supported | + |
| | | |

# Flexibility inside &-generator



# Flexibility in output

| Variation point | Approach | Flexible? |
|---|---|---|
| Documentation of specific model | Based on natural language annotations (meanings and purposes) in specification. | ++ |
| Documentation (generic) | Different types of documents (chapters) are generated. | + |
| Documentation format | Pandoc (library) | +/- |
| Documentation language | Currently two: English and Dutch | +/- |
| Prototype | (more later) | ++ |
| Reasoning about rules | ‘Meatgrinder’. Follows the formal specification of & itself. | + |
| Other output | Extend the generator | ++ |
| | | |

## Design Principles

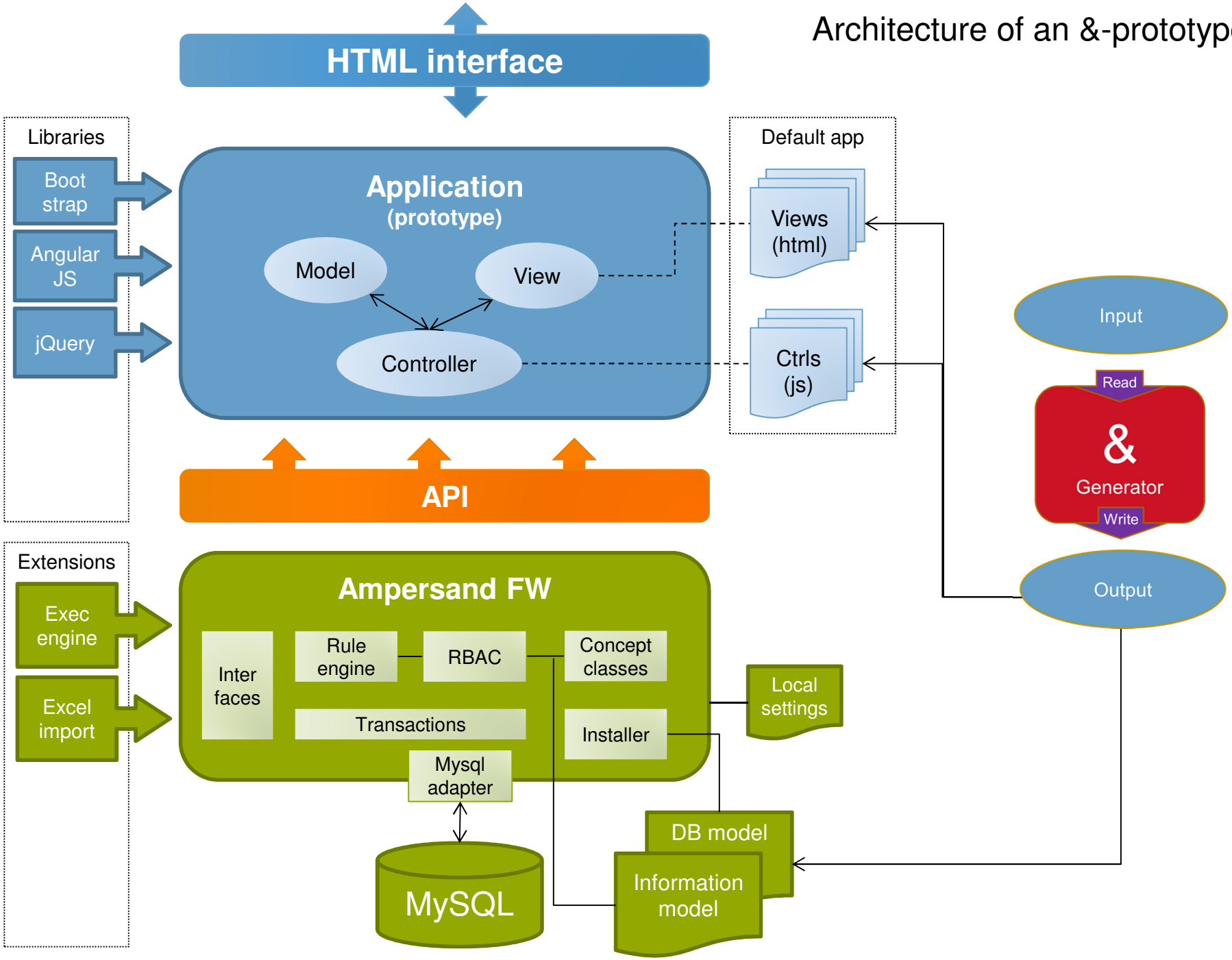
- Flexibility
  - Isolate whatever is known to change
  - Loose coupling of components
  - Single responibility of components
  - Reuse existing software where it makes sense

- Validation & verification
  - Document the ‘why’ (purpose, source, for tracability)
  - Automate when possible
  - Throw errors quickly
  - …

## Agenda

- Goals & Principles
- Variation points
- **Architecture of prototypes**
- New features of Ampersand

# Architecture of an &-prototype



# Flexibility in prototype

| Variation point | Approach | Flexible? |
|---|---|---|
| Persistence of data | Relational database | ++ |
| Method of changing data | Transactional | -- |
| Control of changes | API is generated according to &-model | ++ |
| Look & feel | Html templates<br>Support for web-frameworks | ++ |
| Behaviour | Controlers | + |
| | | |

## Agenda

- Goals & Principles
- Variation points
- Architecture of prototypes
- New features of Ampersand

## New features

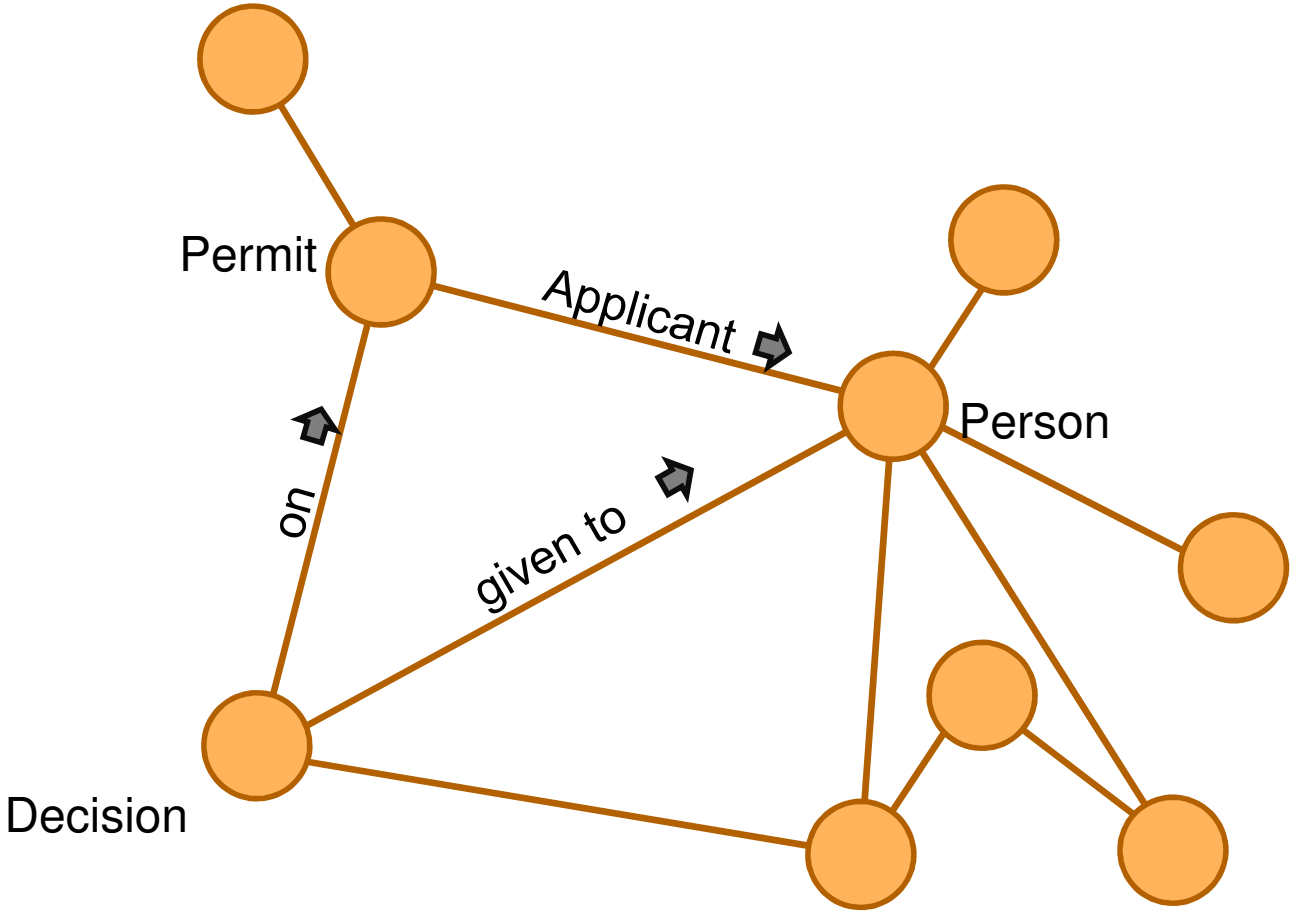
- Support for Classifications
- Architecture of frontend + backend-API
- Extended features in interface specification
- Bulk import / export
- Improved performance of prototype
- Improved error messages for modeller
- Execution engine (rulebased execution of actions)
- Improved software process
- Improved QA
- Enhancements in document generation

**Flexible Applications?**

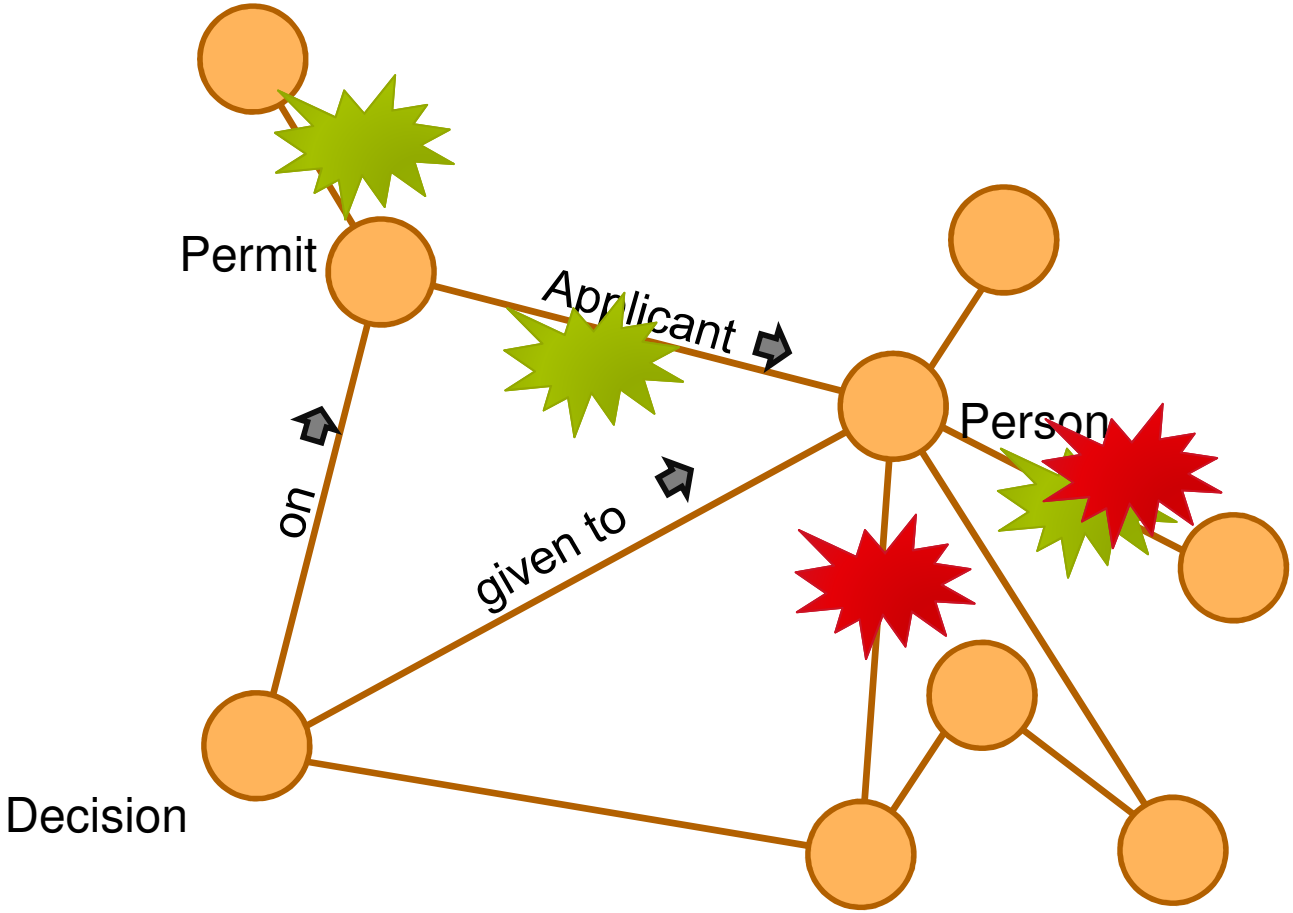
**Ampersand!**

# Reserve dia’s

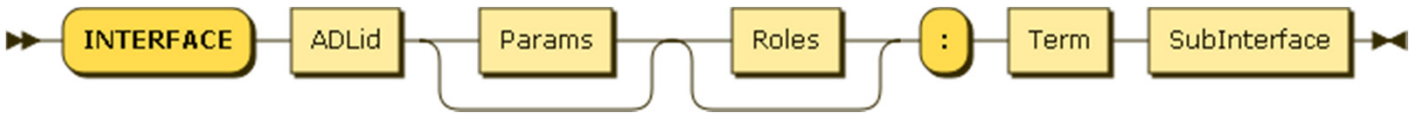
# How does this work?

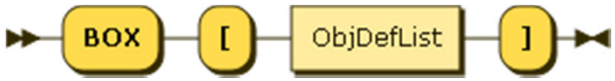


# How does this work?



# Interface specification : Old



# Interface specification: New