apart FRAMEWORK

Separation of Use Cases and GUI in Business Applications

EFRE.NRW
Investitionen in Wachstum und Beschäftigung
EUROPÄISCHE UNION
Investition in unsere Zukunft
Europäischer Fonds für regionale Entwicklung

Agenda

Motivation

Framework target

Basic concepts

Current state of development

Schedule

Motivation

Existing application: Document manangement

User area: Construction engineers

Government-sponsored research project:

„Cloud Collaboration Software“

Framework: Less than 10% of total work in project

EFRE.NRW
Investitionen in Wachstum und Beschäftigung

EUROPÄISCHE UNION
Investition in unsere Zukunft
Europäischer Fonds für regionale Entwicklung

Framework Target: Business Applications

Application area:

Domain-specific solutions

Architecture:

Database-centric (two-tier)

Framework focus:

Fat client with platform-specific GUI

Perspective of Application Specs

Main business objects

General business rules

Functional requirements defining:

- Use Cases
- Database design
- User interface

Patterns in Business Applications

Class hierarchies for:

- Main components (connected to GUI windows)
- Reusable subcomponents (widgets, tables, models …)
- Database related objects

Access patterns / APIs for:

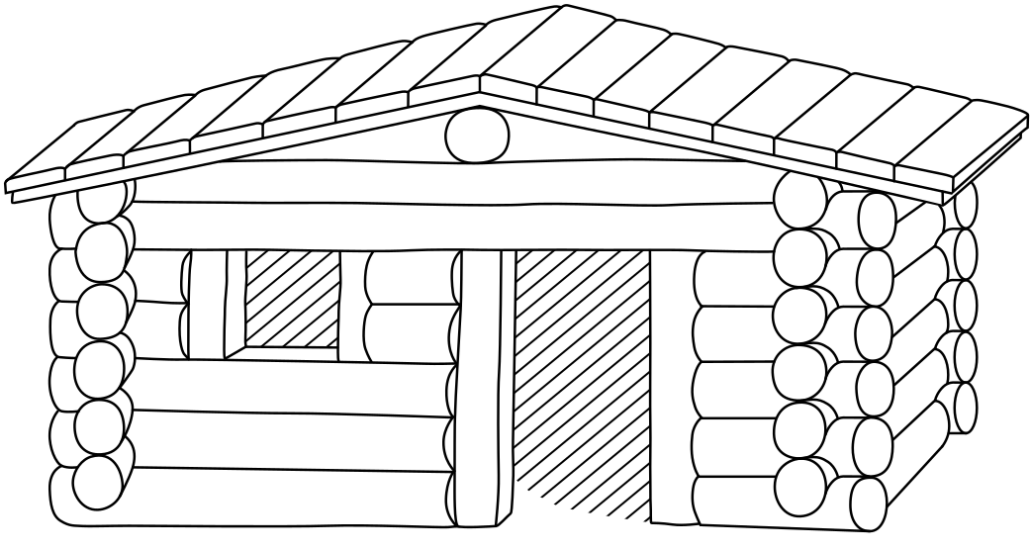
- Opening / interconnecting components
- Providing access to „global“ information
- Dealing with the database (transactions)

Business Rules, Use Cases

Key questions for business rules and use cases:

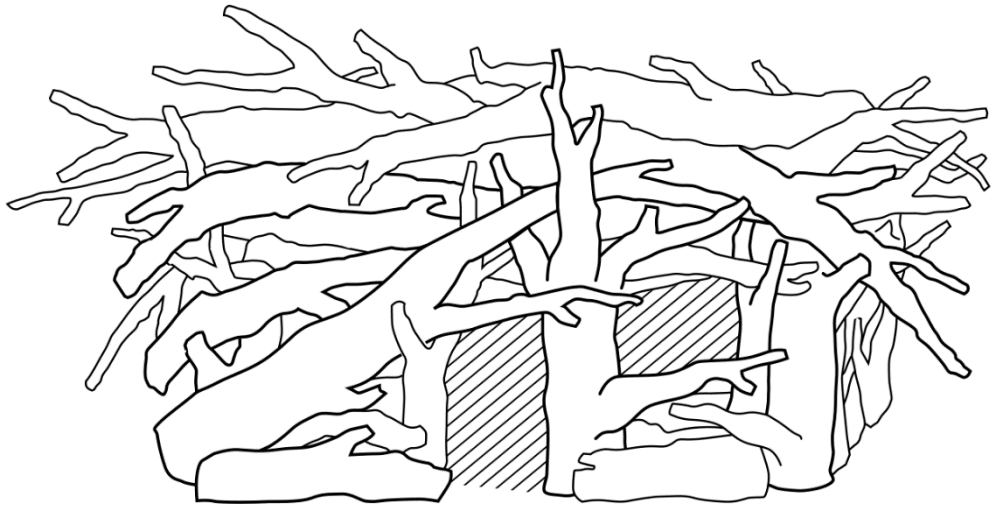
- How strong are they connected to the application environment?
- How easily can they be changed or expanded?
- Where are they located in the code?

How do we construct?



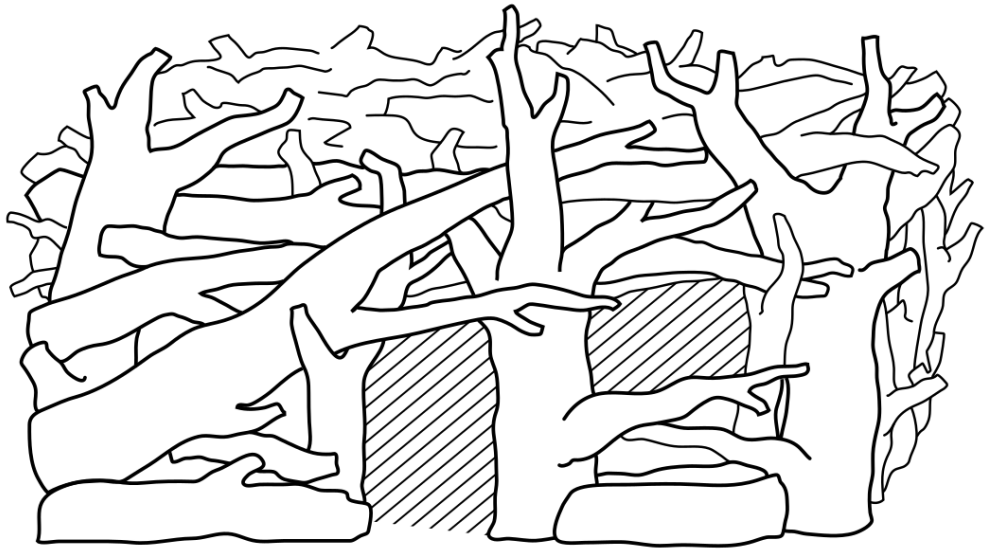
Corner connections, door and window frames, roof tiles are reshaped from raw wood according to our plan.

Or do we rather construct like …



… because we don‘t reshape what we have at hand?

Entangled Construction After Refactoring

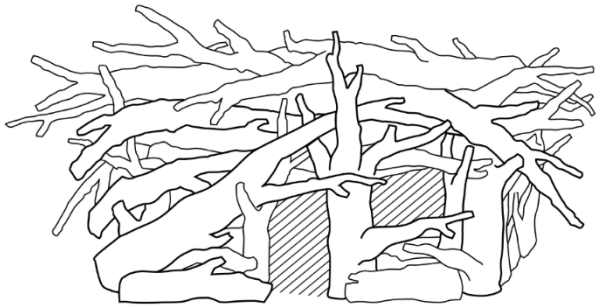


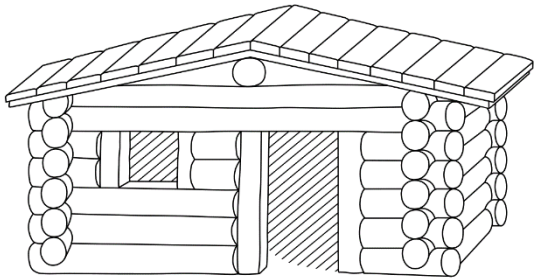
Entangled Constructions

make it harder to:

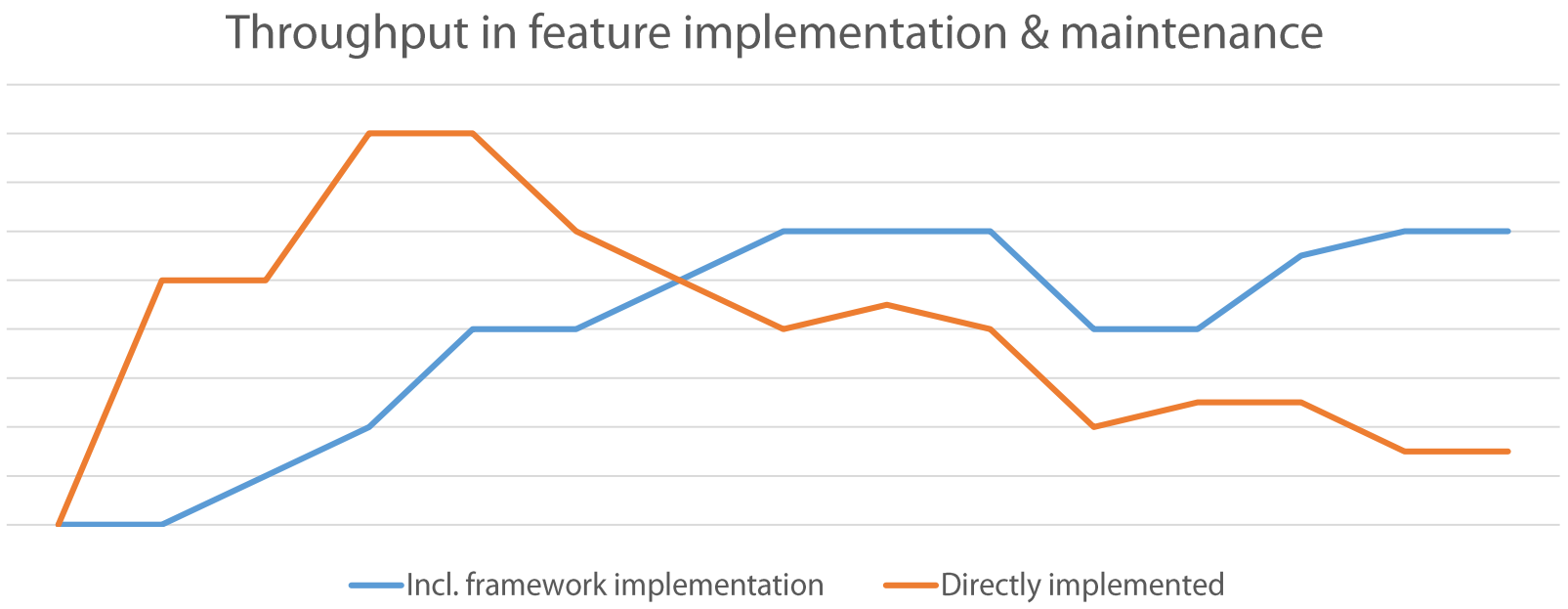
- locate functionality, assess existing dependencies
- estimate costs of changes or extensions

Touching one point might affect everything.
But sophisticated constructions are not for free!

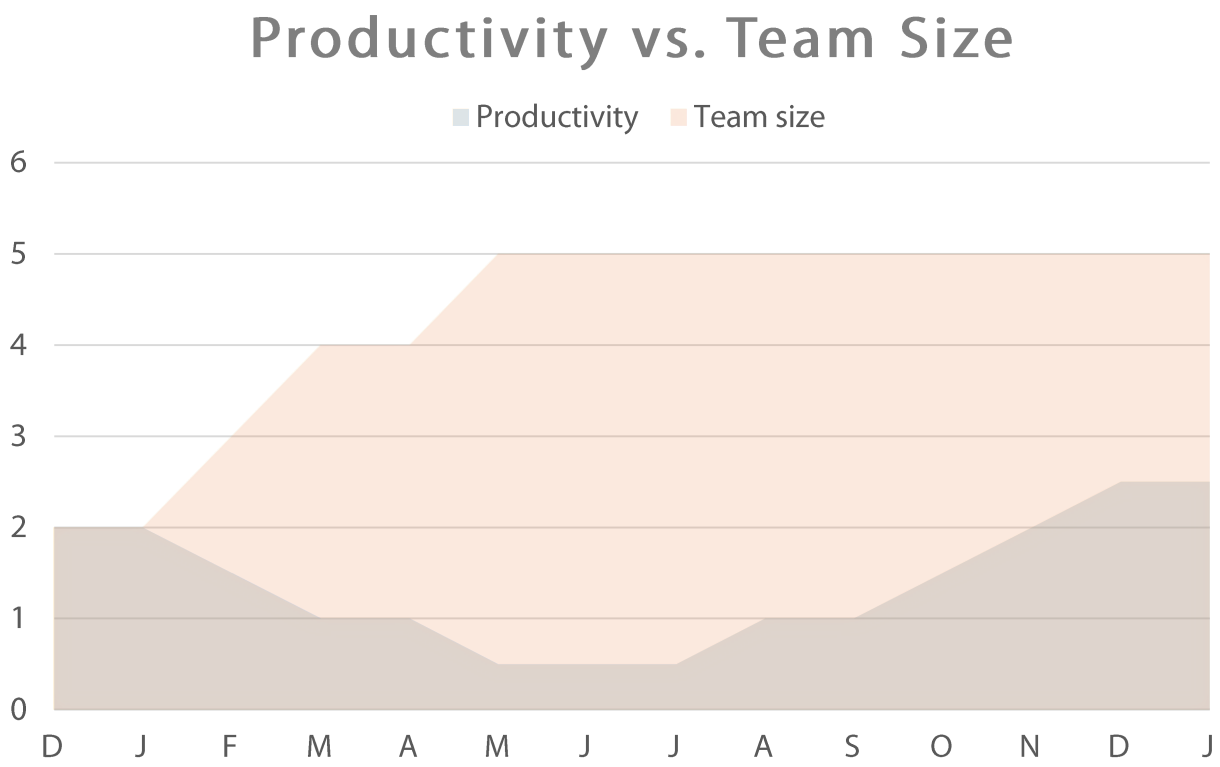




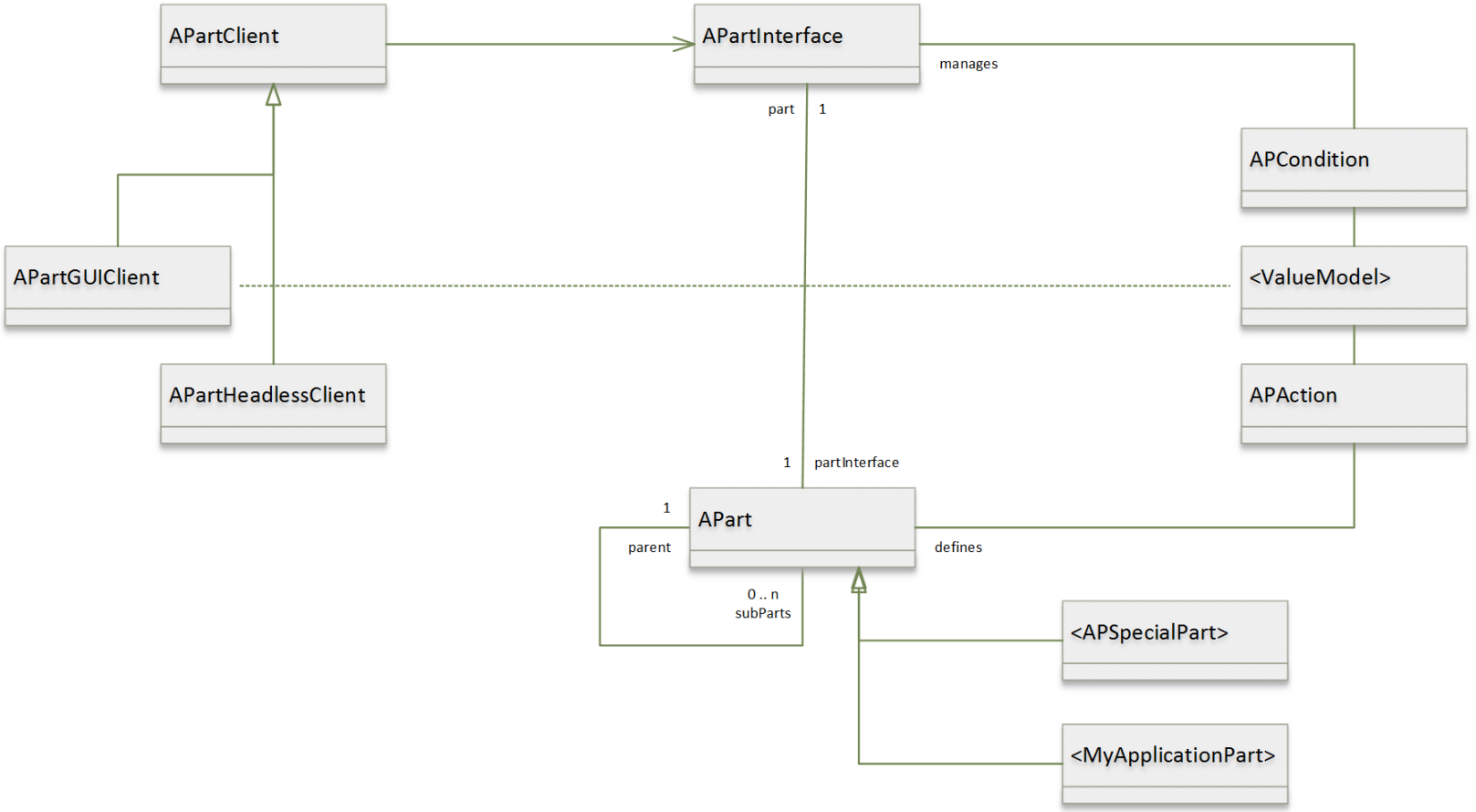
Productivity Compared Over Time



Productivity Loss with Entangled Software



Architecture Overview



Participants and Roles

Modelling level:

- Parts forming the „skeleton“ of the application
- Objects exposed by parts (conditions, values, actions)

UI level:

- Generic GUI clients of the framework
- Application specific objects (widgets, models, ...)

„Glue“ level:

- Arbitrary objects (startup, environment, use cases, ...)

Where Are the Use Cases?

Outside of the framework classes, connected via:

- their creators (typically parts)
- framework condition objects
- part actions
- callbacks to parts maintaining separation from GUI

Typically, a use case does not reference any part. Instead, it is initialised with all objects needed for running. For dynamically needed data it uses named callback requests set up by its creator.

Overview of Advantages

Application feature design:

- Clear definition of what is exposed by parts
- Well-defined setup of parts and subparts
- Best practice: Use cases reference data instead of parts
- Well-defined use cases as anchors of functionality

Internal framework functionality:

- High stability of basic functions (init, shutdown, broadcasts)
- Important functionality based on narrow hooks

Current State of Development

- Implemented under VW 8.2, 8.3 coming soon
- Used solely for new document management app, but with a wide variety of complex scenarios
- Simple examples for all important features
- Reference solution for dealing with Trachel shapes
- Testing support including „recording clients“ that print executable testing methods
- No over-all documentation, Comments in German
- Needs a thorough clean-up to be reusable by others

Schedule

2019:

- Clean-up and refactoring in context of current application, including examples and comments
- Improving support of generated GUIs in VW
- Start with Pharo port and documentation in Pillar

2020:

- Setting up a full-featured reference example including database handling with Glorp
- Working towards a unified portable code base

Visons

What if …

- creating a small business app takes only a few days?
- such a business app is not a prototype but production quality at first try?
- developers prefer to use a framework instead of following their usual habits?