NOTICE!

- These materials are prepared only for the students enrolled in the course Distributed Software Development (DSD) at the Department of Computer Science and Engineering, University of Mälardalen, Västerås, Sweden and at the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia (year 2010/2011).
- For all other purposes, authors' written permission is needed!
- The purpose of these materials is to help students in better understanding of lectures in DSD and not their replacement!

Distributed Software Development

Alpha Prototype

Software Patterns Team

Overview

- Project schedule
  - Current state
  - Working hours
  - Results
- Process
- Experiences in project work
  - Problems
  - Experience
  - Work distribution
- Demo

Project Schedule

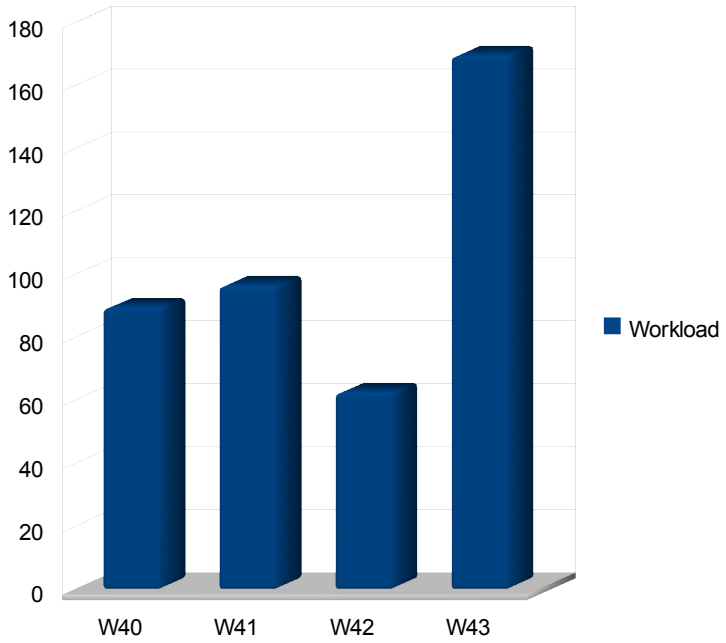
- Current State
  - Submit deliverables on time
  - Studied new technologies (EMF, Eclipse plug-ins, etc)
  - Use horizontal work distribution for alpha version
  - Set goal for alpha version
  - Revisions of existing documents if needed

General Project Status	Fulfillment of Next Milestone
On track	On track

Project Schedule Cont.

- Hours invested

Member	W40	W41	W42	W43	Total
SB	16	22	12	24	74
MV	19	15	13	43	90
AMB	0	14	9	16	39
JC	10	18	7	12	47
JR	15	14	6	15	50
IP	13	0	7	21	41
SG	16	13	8	38	75
Total	89	96	62	169	416

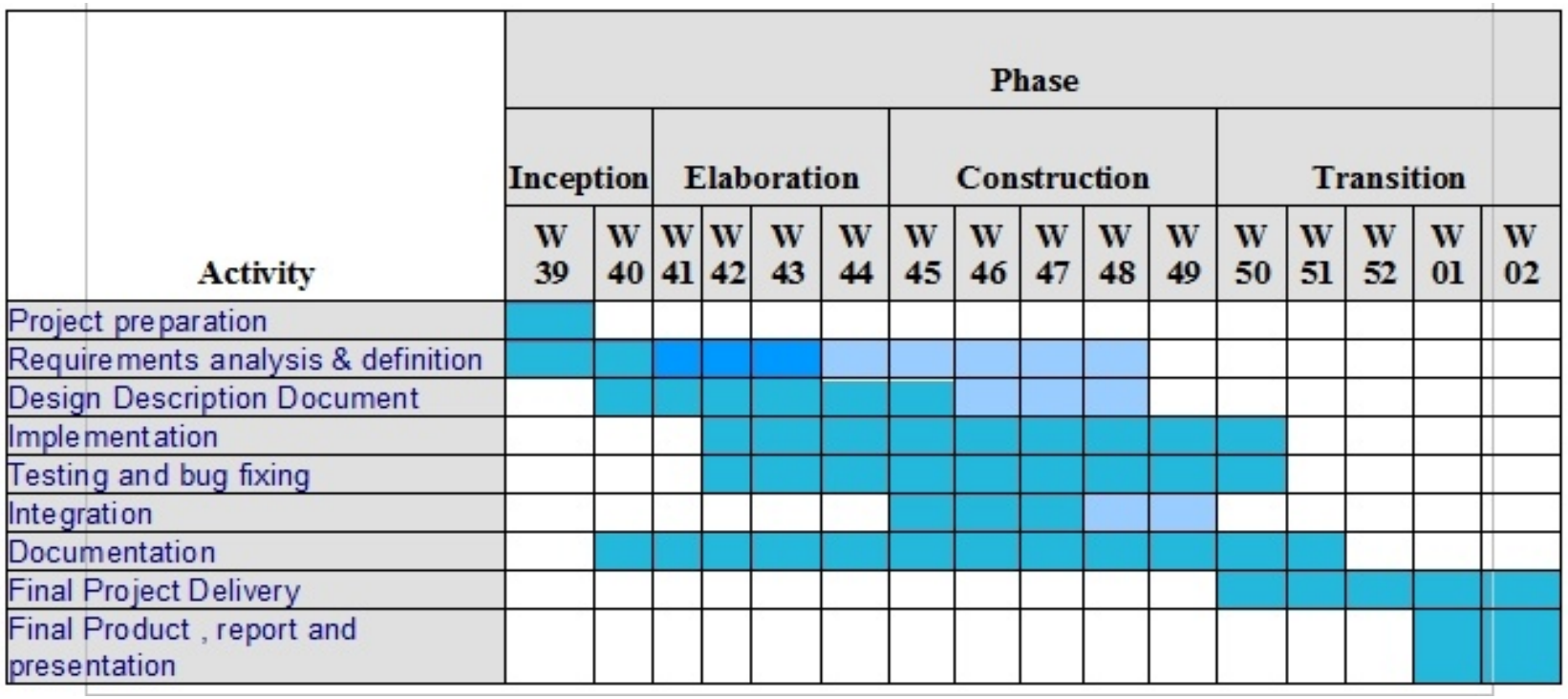


Project Schedule Cont.

- Results
  - Knowledge of plugin development
  - Knowledge of EMF
  - GUI sketches documentation
  - Defined presistence for EMF
  - Installed project management tool (Redmine)
  - Implemented main application UI

Process

- Project activity plan



Activity	Phase															
	Inception		Elaboration				Construction					Transition				
	W 39	W 40	W 41	W 42	W 43	W 44	W 45	W 46	W 47	W 48	W 49	W 50	W 51	W 52	W 01	W 02
Project preparation																
Requirements analysis & definition																
Design Description Document																
Implementation																
Testing and bug fixing																
Integration																
Documentation																
Final Project Delivery																
Final Product , report and presentation																

- Submited requirement and design description documentation

Process Cont.

- Start Implementation at week 42
- Also start unit testing with implementation
- Every member made documentation on what they studied
- All documentation and sample code are uploaded on SVN

Problems

- Lack of knowledge on tools and technologies
  - EMF
  - Eclipse Plugins
  - LaTeX ... etc
- Much time spent documenting
  - Tutorials
  - Eclipse documentation
  - Workshops
- Absent or busy members
  - Ivica and Joanne tripped to Turkey and Finland
  - Exams period in Croatia

Experience

- Working together
- Learning new technologies and work methods

- Learning from past experience
- Learning from each other

Work Distribution

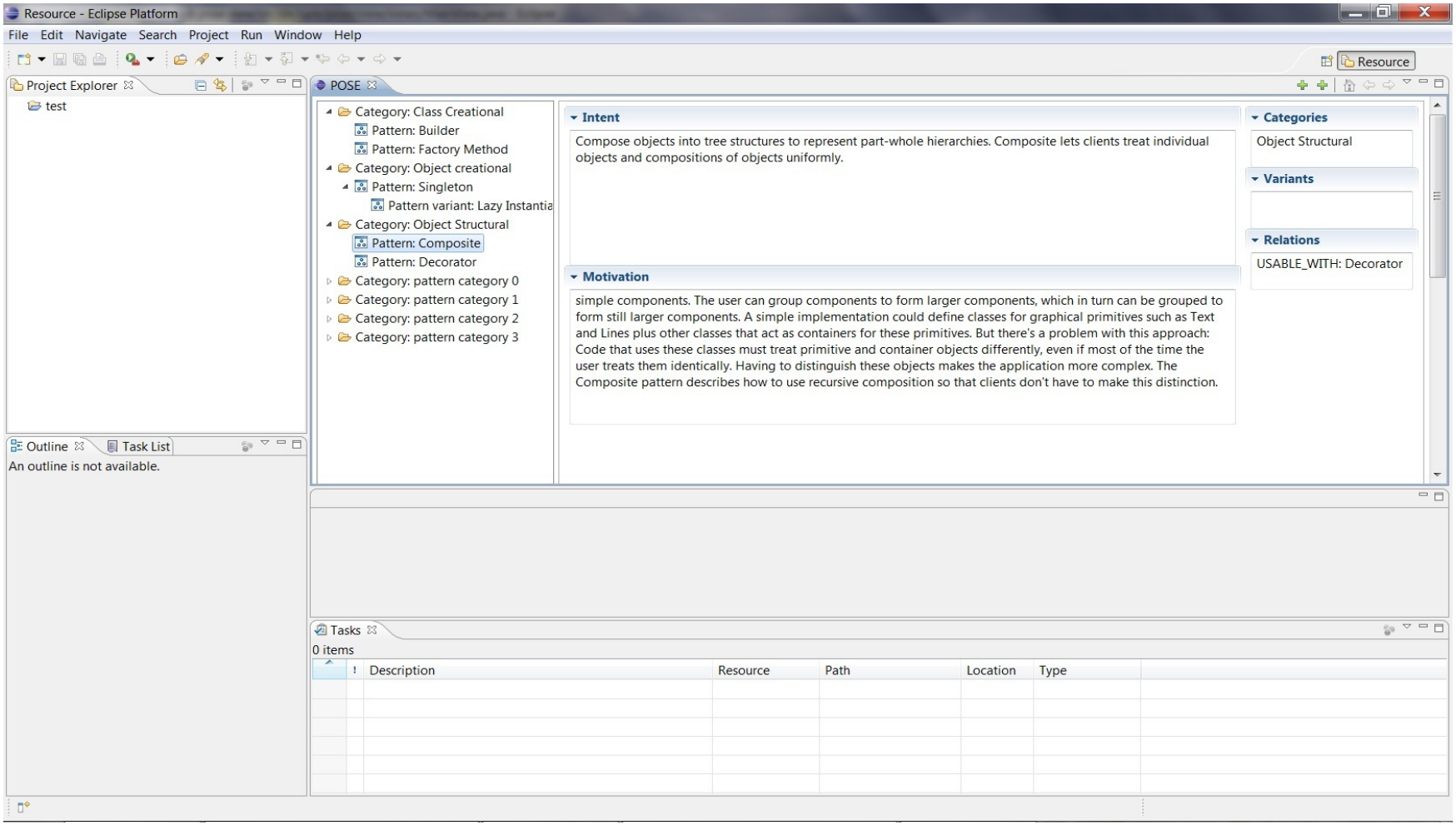
- Swedish team
  - Graphical User Interface (Design)
- Croatian team
  - Persistence
  - MVC architecture
  - Views
  - Documentation tools and coding conventions
- German team
  - Domain model
  - Support

Communication - Collaboration

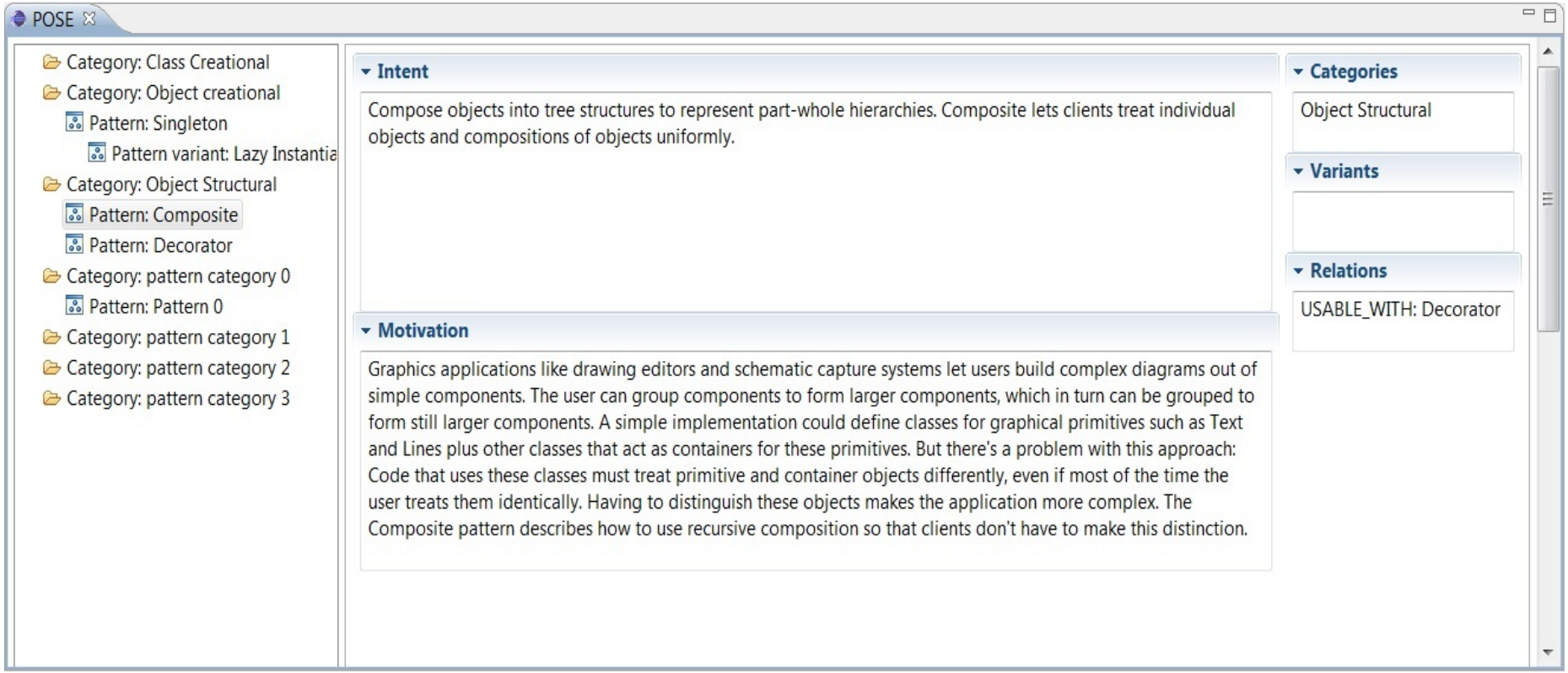
- Nine official meetings, including seven with the Paderborn part of the team.
- Unofficial local meetings in each part (Sweden, Croatia, Germany).
- Discussions (e-mail, Skype, Wiki, IRC channel, Adobe Connect, Doodle, Redmine, etc.)
- EMF and Eclipse workshop arranged by Paderborn.

Demonstration

Eclipse using our plug-in



Main view


POSE
Category: Class Creational
Category: Object creational
Pattern: Singleton
Pattern variant: Lazy Instantia
Category: Object Structural
Pattern: Composite
Pattern: Decorator
Category: pattern category 0
Pattern: Pattern 0
Category: pattern category 1
Category: pattern category 2
Category: pattern category 3
Intent
Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.
Motivation
Graphics applications like drawing editors and schematic capture systems let users build complex diagrams out of simple components. The user can group components to form larger components, which in turn can be grouped to form still larger components. A simple implementation could define classes for graphical primitives such as Text and Lines plus other classes that act as containers for these primitives. But there's a problem with this approach: Code that uses these classes must treat primitive and container objects differently, even if most of the time the user treats them identically. Having to distinguish these objects makes the application more complex. The Composite pattern describes how to use recursive composition so that clients don't have to make this distinction.
Categories
Object Structural
Variants
Relations
USABLE_WITH: Decorator

Future development

- Vertical distribution of work
- Eclipse editor
- Eclipse views
- Define own eclipse perspectives:
  - Catalog manager
  - Modelling

Q&A