CS4125
SYSTEMS ANALYSIS
SPRING SEMESTER 2010-2011

J.J. Collins
Dept of CSIS
University of Limerick

Lecture A (W1L1)

Module Overview

1. Module Background: Lecturer

- Name: J.J. Collins
- Room: CS128
- Email: j.j.collins@ul.ie with subject as "CS4125: ....."
- Research Interest:
  - Machine Learning paradigms for robot control.
  - Performance Engineering.

- Tutorials: start week 3
- Labs: start week 3.

1. Module Background: Objectives

- To understand and apply:
  - Modelling notation such as the Unified Modelling Language (UML) to the development of models of software.
  - A generic Object-Oriented Analysis and Design (OOAD) methodology.
  - Architectural and design patterns.

  within an enterprise-level context
- To familiarise oneself with an industrial strength modelling environment

1. Module Background: Assessment

- ATTENDANCE MONITORING WILL SOMETIMES BE USED.
- Assessment Instruments
  - 40%: OOAD project in teams of 2 or 3.
  - 10%: Midterm week 7.
  - 50%: Final Exam
- RESULTS:
  - 2009-2010: five As, eight Ds, and three Fs (class size 42)
  - 2008-2009: seven As, six Ds, and one F (class size 23)

1. Module Background: Assessment

Please note:

- An F or NG grade for the project will automatically result in an F for the module.
- A compulsory question in the final exam will examine project related material. Failing this will result in project marks being omitted in the calculation of one's final grade.

1. Module Background: Texts

- Bennett, S., McRobb, S., and Farmer, R. Object-Oriented Systems Analysis and Design, FOURTH Edition. McGraw-Hill. 2010.
- Stevens, P., and Pooley, R. Using UML: Software Engineering with Objects and Components, Second Edition. Addison-Wesley. 2005.

2. Lecture Plan

Week 1	Introduction
Week 2	Requirements Engineering with Use Cases
Week 3	The Object-Oriented paradigm
Week 4	Analysis: classes
Week 5	Analysis: behaviour using sequence and communication diagrams
Week 6	Analysis: state charts

2. Lecture Plan

Week 7	Software Lifecycles
Week 8	System Design
Week 9	Detailed Design
Week 10	Patterns
Week 11	Component and Deployment Diagrams
Week 12	Patterns

3. Problems & Solutions: Greenfield et al.

- **Greenfield et al. Software Factories. Wiley. 2004.**

- "We see a capacity crisis looming. The industry continues to hand-stitch applications distributed over multiple platforms housed by multiple businesses located around the planet, automating business processes like health insurance claim processing and international currency arbitrage, using strings, integers and line by line conditional logic. Most developers build every application as though it is the first of its kind anywhere."

3. Problems and Solutions: Greenfield et al.

- "Scaling up to much higher levels of productivity will require the ability to rapidly configure, adapt and assemble independently developed, self describing, location independent components to produce families of similar but distinct systems. It will require a transition from craftsmanship to manufacturing like the ones we have seen in other industries …."

3. Problems and Solutions: Greenfield et al.

- "To accomplish this, the industry must capitalize on some key innovations in software development. We must synthesize ideas such as domain specific languages, software product lines, component specification and assembly, patterns, framework completion, domain analysis and feature variability analysis into a cohesive approach to software development that can learn from the best patterns of industrialized manufacturing. "
- Many of these approaches are predicated on the ability to model at the correct level of abstraction.
- Why?

3. Problems and Solutions: Economic Realities

According to the Standish Group (very old figures):

- 250$ Billion spent annually in USA on software development.
- 16% of projects completed on time and within budget.
- 31% cancelled, primarily because of quality.
- 53% cost more than planned, with average excess running to 189%, creating losses of 59$ Billion approximately.
- Projects that reach completion deliver only 42% of specified features.

- For how long more will this be tolerated?

3. Problems and Solutions: Failure to Reuse

- "Japanese managers did not adopt factory models and pursue scope economies simply out of faith. Detailed studies concluded that as much as 90 percent of the programs they developed in any given year, especially in business applications, appeared similar to work they have done in the past, with designs of product components falling into a limited number of patterns"

  Michael Cusumano. Japan's Software Factories. Oxford University Press. 1991.

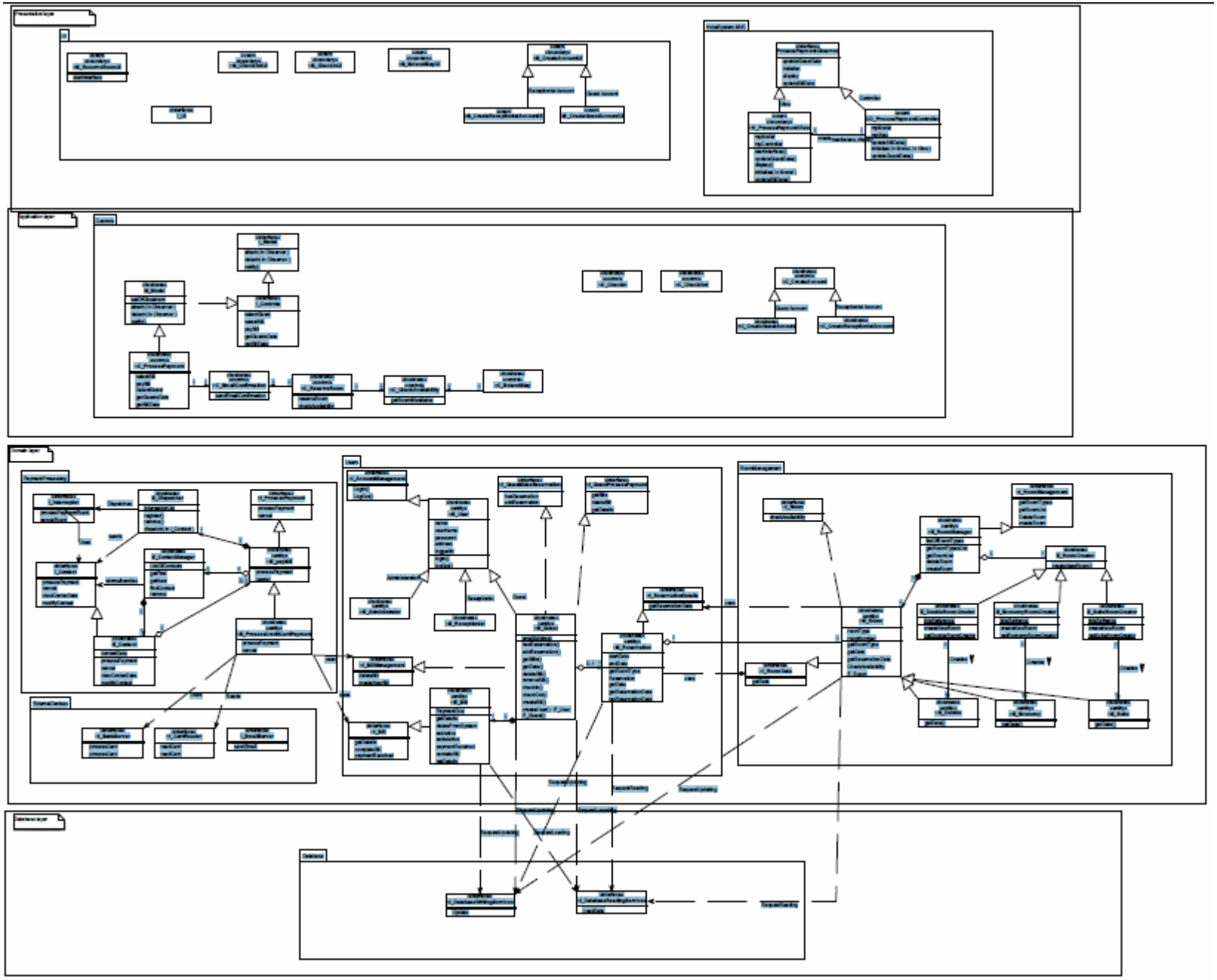
4. Solution Summarised

Understand

- Characteristics of good software
- Modelling languages
- Methodology i.e OOAD method
- Architecture centric development
- Patterns, and more patterns
- Appropriate processes and tools.
- Technology and standards awareness.
- Etc.

5. Note on XP

- CS4125 is based on upfront commitment to design.
- Contrasts with Agile Manifesto as typified by Extreme Programming (XP).
- CS4125 does not imply an implicit criticism of XP on the downside!
- “Horses for Courses”.



An example of a design time class diagram submitted by a project team

Reading

- Chapters 1 and 2 from Bennett, McRobb, and Farmer (4th edition).