Week 5: Unified Process   
What is a risk? How doe we manage risks?

A risk is a potentially adverse circumstance不利情况 which may impair the development process or the quality of products.

In risk management we identify risks and respond to them before

they endanger the whole project.

1. identify the risk

2. assess their likelihood and potential impact

3. address the risk by devising a plan to deal with it

4. eliminate the risk by implementing the plan

How is the unified process risk-driven?

Risk driven

Major risks \mitigated减缓" during inception.

All risks mitigated by end of elaboration

Why does the unified process emphasize 强调 risks?

Attack risks early and continuously before they will attack you

Why is the unified process architecture-centric?

Elaboration constructs a working architecture

How is the unified process use case driven?

Use cases capture 俘获 the functional requirements

use cases define the contents 内容 of the iterations

Each iteration realizes one or many use cases

Describe the phases and disciplines of the unified process.

Inception 开始

Business case, vision, identify high risks & 10% of key reqs in

detail, estimate elaboration 估计加工 effort

Elaboration 加工

Core & architecturally significant parts coded/tested, key risks

identified/mitigated减缓, 80% of major reqs evolved 逐步形成/defined 确定

Construction 结构

Builds remaining剩余的 system in short iterations, efficient有效的 and

predictable 可预言的 due to solid elaboration 实体加工

Transition

Exposes release candidate 候选者 for review/feedback, then deployment分配

Some guidelines

I Attack risks early and continuously before they will attack you

I Stay focused on developing executable 可执行的 software in early iterations

I Prefer component-oriented architectures and reuse of existing components

I Baseline an executable 可执行的 architecture early

What is visual modeling? Why do UML and unified process emphasize 强调 visual modeling?

Visual modeling

Prior to programming, do at least some visual modeling to explore creative design ideas在编程之前,至少做一些可视化建模探索创造性的设计理念

Why do we need models in software development?

Week 5: Principles   
Why and how do people play such a major role in software development?

People make mistakes

Human intellect理智 can handle only limited amounts of information

(unless it is structured!)

Change is inevitable必然的

People are ... Main Obstacles障碍 to Software Engineering

Communication ... with people

Re-work .. due to mistakes

Complexity ... because we push the envelope

Change ... because we like it

Explain the terms: model, abstraction, generality, modularity, incrementality, separation of concerns, rigour, formality.

Modularity模块的

Separation into parts

Separate interface from implementation

Separate responsibilities

Nested modules = hierarchical decomposition 层次结构分解

Modularity is when a product or process is composed 组成 of

components/modules

greatly aids 改善了 separation of concerns

better understandability

easier to change

modules reusable

high cohesion 结合 within a module when all its elements are

strongly related 密切相关

that is, there is a very good reason why they are together

low coupling 耦合 between modules

coupling measures 测量 the interdependence相互依赖 between modules

e.g. module A and module B share a variable x is high coupling

Incrementality 渐近的

Separate process into increments

... by subsets of features or use cases

is separation into parts and separation in time

increment = a small step/change

Incremental prototyping 原型

identify a useful, but small, subset of the problem and implement it

build a sequence of prototypes by adding features one at a time

\_nally have a complete system

Advantages

early delivery of a (sub)system

early feedback from users

later (more complicated) features may not be necessary

more time to think about complicated features

separation of concerns

Separation of Concerns | Examples

I Specification (what) vs implementation (how).

I Correctness vs e\_ciency. Get it working \_rst, then attend to

performance.

I Functional vs non-functional requirements.

I Spreadsheet design: cell manipulation vs display.

I Application code vs user interface code.

I In-memory processing vs disk access.

A typical sequence is logical data ! block data ! disk

bu\_ers ! disk controller.

In DOS: \_le name (user level) ! \_le descriptor (DOS level)

! disk address (BIOS level) ! driver routines.

I Form vs content in word processing.

Example: CSS, Latex style \_les.

Abstraction

concentrate 集中精力 on **general aspects** of the problem while carefully

removing **detailed aspects**

focus on **important aspects** and downplay低估 the **unimportant**

NB what is important or general varies

eg \person" record in medical database

vs \person" record in payroll database 工资单

vs \person" record in sports performance analysis  
Describe at least 10 examples of separation of concerns that arise during software development.

Generality

more general = able to handle more cases

use of general purpose tools, eg awk

rather than write speci\_c pattern matcher software

more general program may be simpler and reusable

I fewer special cases in specification

I able to handle a broader range of inputs/uses

Rigour and Formality

rigour is achieved by conforming 符合 to given constraints约束

Examples of constraints

I using a given format for the requirements document

I documenting input and output of each procedure/function

I limits on number and size of modules

I following company standards for the development process

rigour is a necessary complement to creativity in SE

| it provides shape, direction, precision, and the ability to compare and measure

There are many levels of rigour | a spectrum

Formality

formality is highest degree of rigour: it requires the process to be driven and evaluated by mathematical laws

What is the most common use of formality in software development?

it requires the process to be driven and evaluated by mathematical laws

For everything you have learned so far, how does it relate to the major problem areas for SE: requirments, architecture, change, complexity?

Requirements

Too often we build the \wrong" system

Communication between technical and business people is not easy

Need precise 精确的, accurate description 准确的描述 of customer needs

Customer does not know what they need

Architecture

Where all the demands of requirements \hit home 打中要害" at once

Most non-functional requirements are \holistic整体的"

Foundation基础 for (managed) change, maintenance, and evolution演变

Change

The world keeps changing, and we like it that way

Software is not easy to change, despite what managers & customers think

Complexity

Always trying to build bigger, better, faster, smaller, cheaper

Challenge挑战 to do things that we have never done before