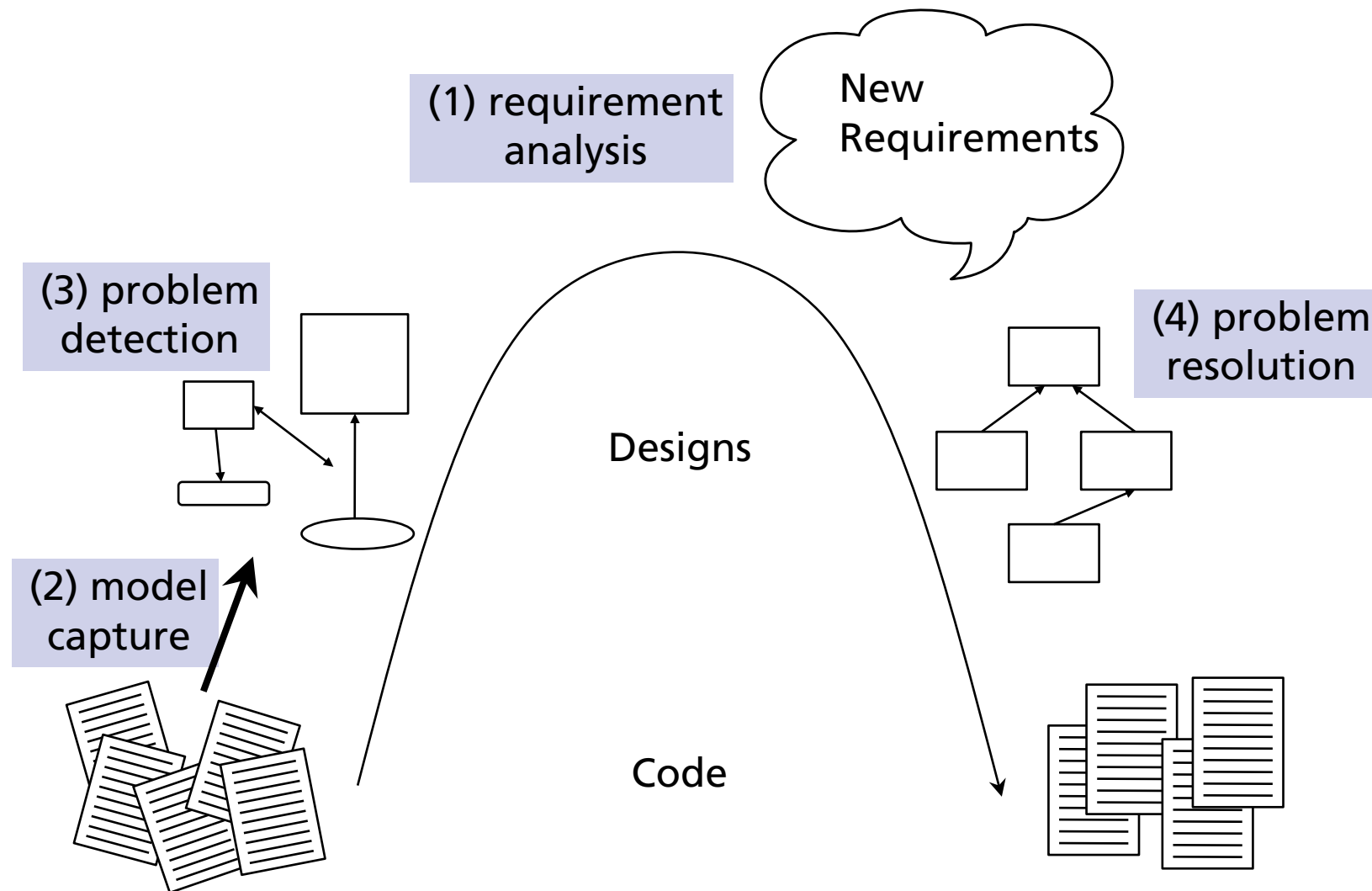


Software Reengineering

P2: Setting Direction

Martin Pinzger
Delft University of Technology

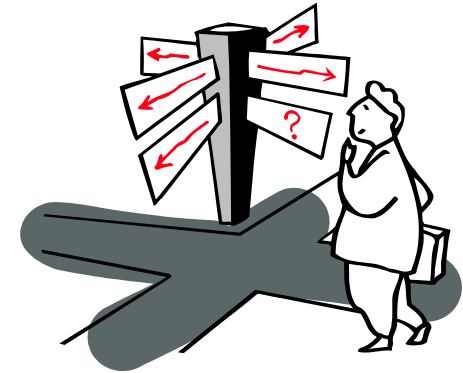
Reengineering Life-Cycle



Outline

Setting direction

First contact



The book

Object-Oriented Reengineering Patterns Demeyer, Ducasse & Nierstrasz



Present simple, lightweight techniques on how to cope with large-scale, complex legacy systems in need of reengineering

Uses patterns, descriptions of generic problem-situations, possible solutions and trade-offs

Download a free copy from: <http://scg.unibe.ch/download/oorp/>

A few of these patterns...

Most valuable first

Chat with the maintainers

Read all the code in one hour

Skim the documentation

Interview during demo

Do a mock installation

Speculate about the design

Study the exceptional entities

Refactor to understand

Write tests to enable evolution

...

Useful during...

Setting direction

- What are the goals of the project?

- Find Go/No-Go decision

First contact

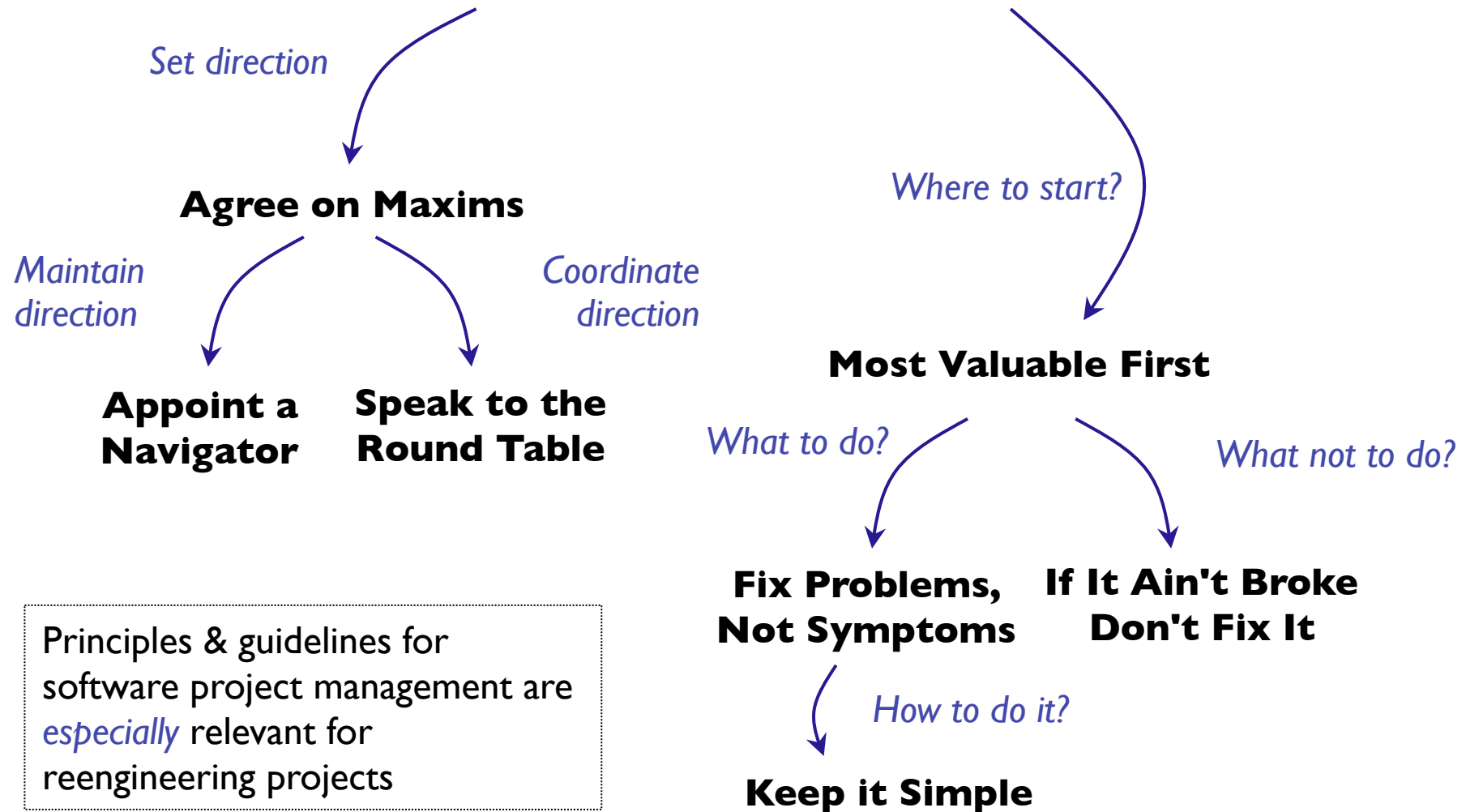
- You are facing a system that is completely new to you and within hours/days you should determine:

 - Whether the software is still viable

 - A plan of work

 - A cost-estimation

Setting direction patterns



Pattern: Most Valuable First

Problem: Which problems should you address first?



Most valuable first (2)

Solution: Work on aspects that are most valuable to your customer

- Maximize commitment

- Deliver results early

- Build confidence

Most valuable first (3)



Copyright © 2003 United Feature Syndicate, Inc.

Most valuable first (4)

How do you tell what is valuable?

- Identify your customer

- Understand the customer's business model

- Determine measurable goals

- Consult change logs for high activity

- Play the Planning Game

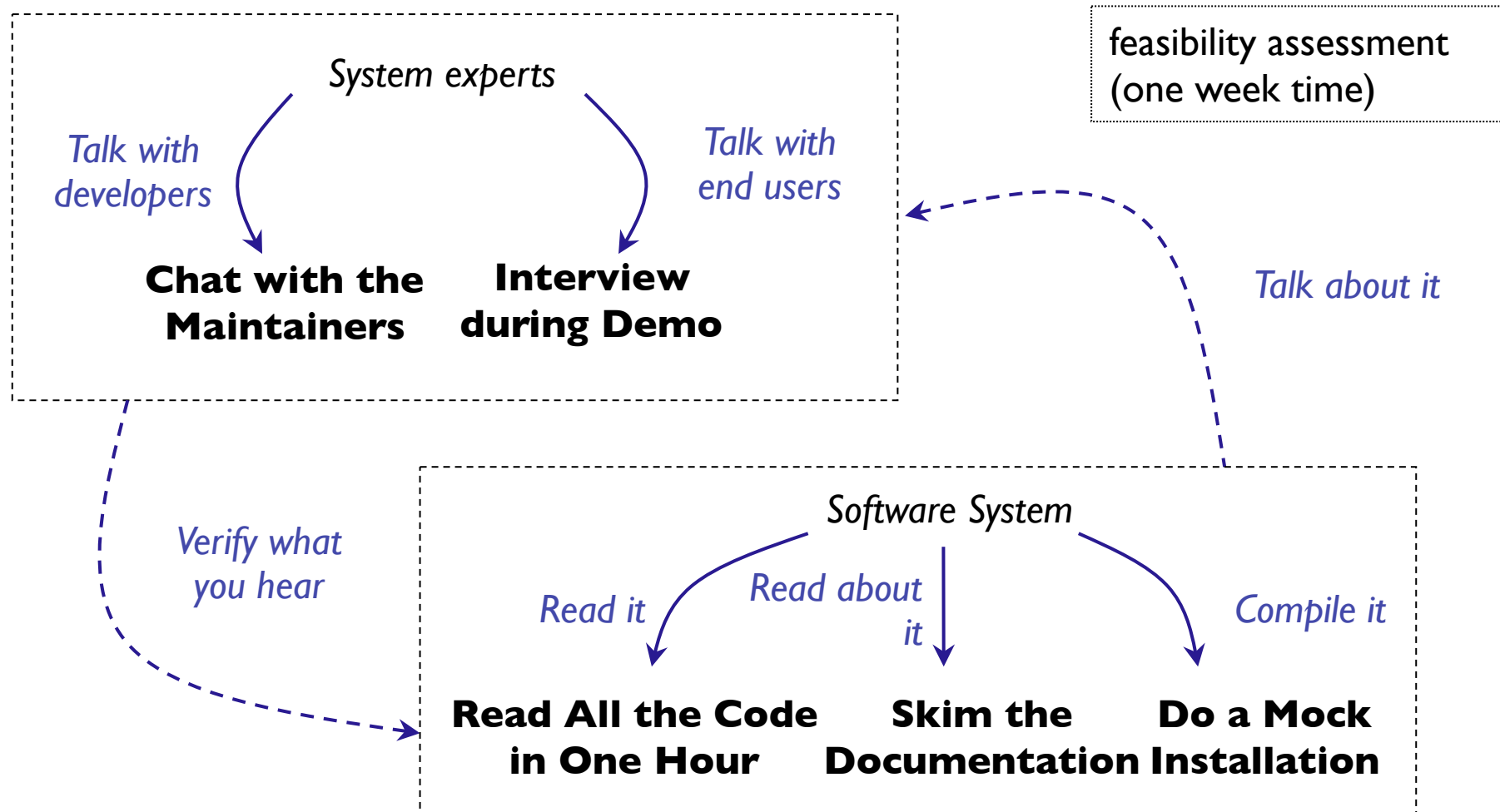
- Fix Problems, not Symptoms

Most valuable first (5)

Planning Game



First contact patterns



Pattern: Chat with the Maintainers

Problem: How to get insights into the legacy system?

Difficult because:

- Documentation records decisions about the solution, not the historical context

- Often people-related (political) issues are at the bottom of the legacy problem

- People working with the system may mislead you to cover up their own mistakes

Chat with the maintainers (2)

Solution:

Treat the maintainers as “brothers in arms”

Make sure they are on your side.

Possible questions include

What was the easiest/hardest bug you had to fix over the last month?

How long did it take you?

Why was it easy/difficult to fix?

How are priorities given?

Is there a version control system in place?

...

Chat with the maintainers (3)

Trade-offs

Pros

- Obtain information effectively

- Get acquainted with your colleagues

Cons

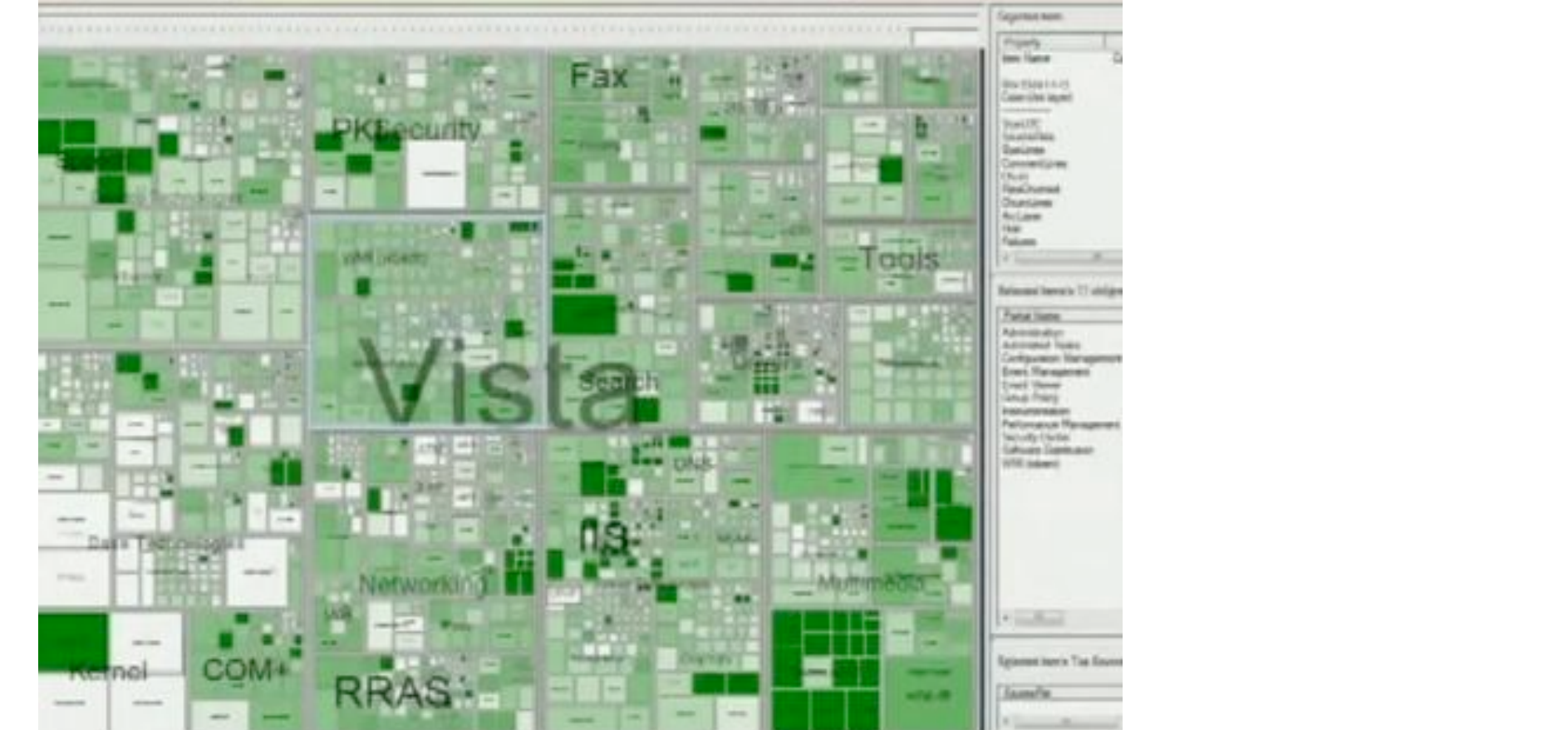
- You only get anecdotal evidence, no hard facts

Difficulties

- People protect their jobs

- Teams may be unstable

Address	Amount	Date	Description	Due Date	City	State	Postal
1234 Main St	1000	1/1/2020	John Doe	12/31/2020	New York	10001	



Read all the code in one hour (2)

Solution: Read the code in one hour

Focus on:

- Functional tests and unit tests

- Abstract classes and methods and classes high in the hierarchy

- Surprisingly large structures

- Comments

- Check classes with high fan-out

- Study the build process

In Java programs focus on

```
public abstract class Example {  
    ...  
}
```

```
/**  
 * Block comment  
 */  
public class Example {  
    public void foo() {  
        int x = 1;  
        for (int x=1; i<100; i++) {  
            // do something comment  
        }  
    }  
}
```

```
public interface IExample {  
    ...  
}
```

```
public class Test {  
    ...  
    @Test  
    public void testExample() {  
        ...  
    }  
}
```

Read all the code in 5 mins session

Form teams of 2

Read all the code in 5 minutes

Write down your findings/questions

Follow-up discussion

Discussion of results

What is the Java application doing?

Are there abstract classes and interfaces?

Are there comments?

Are there complicated structures?

Are there tests?

Where are shortcomings/problems?

Pattern: Skim the documentation

Problem: What about documentation?



Skim the documentation (2)

Solution: Skim the available documentation

Do a general assessment of the documentation (will it be of use or not?)

Is there a table of contents, searchable, ...?

Are there figures? Formal specs?

Make a list of the useful parts of the documentation

Check whether it is up to date

Look at version numbers!

Pattern: Interview during demo

Problem: What are the main features?



Interview during demo (2)

Solution: Do an interview during a demo

Let an end-user show you around in the functionality of the system

It will give you some usage scenarios

Could be useful for dynamic analysis!

The main features of the system

And whether they are appreciated or not

Consider different demos with different persons

Managers, sales-person, help desk, maintainer, etc.

Pattern: Do a mock installation

Problem: Can you (re)build the system?

Do a mock installation (2)

Solution: Do a mock installation of the available system in a clean environment

Check whether you have all the necessary artifacts available by installing the system, compiling the code and running the tests.

Gives insight into:

- Dependencies

- Version numbers of libraries

- Problems

Attention: easy to get carried away and loose time

First project plan

Project scope (1/2 page)

Description, context, goals, verification criteria

Opportunities

Identify factors to achieve project goals

Skilled maintainers, readable source-code, documentation, etc.

Risks

Identify risks that may cause problems

Absent test-suites, missing libraries, etc.

Record likelihood & impact for each risk

Go/no-go decision, activities (fish-eye view)

Summary

Setting direction patterns to

- Set the goals

- Find the Go/No-Go decision

- Increase commitment of clients and developers

First contact patterns to

- Obtain an overview of the system

 - Design, implementation, documentation

- Grasp the main issues

- Assess the feasibility of the project

Homework

Read the chapters about

- Initial understanding patterns

- Detailed model capture patterns

Form teams of two students

- Mail team composition (name, student number) to me

Install

- DA4Java

- Findbugs, PMD, Metrics

- CCFinderX