

Software Engineering Essentials

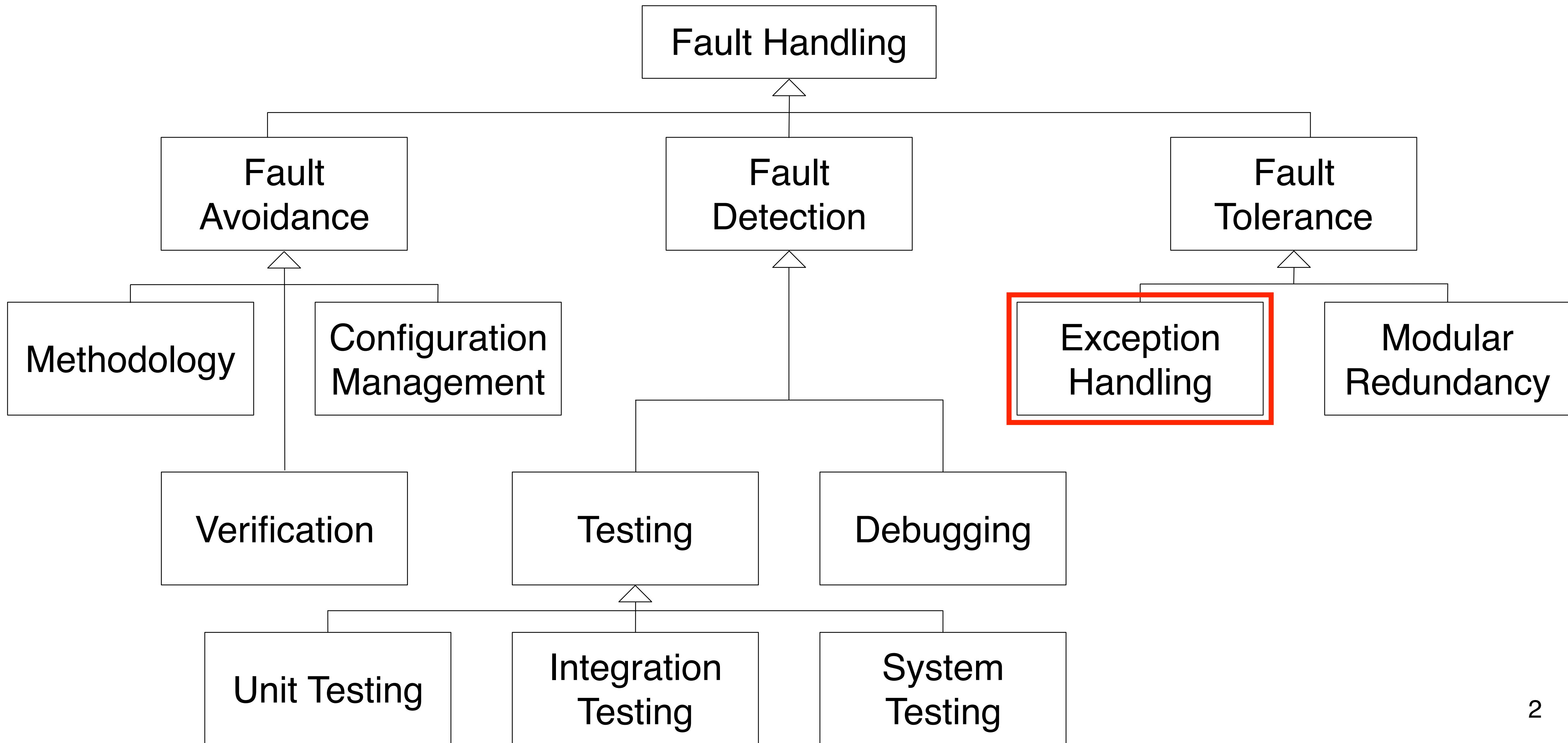


Exception Handling

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics



Taxonomy for Fault Handling Techniques



Exception Handling

Exception handling is a better way to deal with a failure than letting a system crash

- Developers must foresee the failure, they have to write exceptions handling code
- An operating system, virtual machine or a server usually creates only a generic exception handling method:

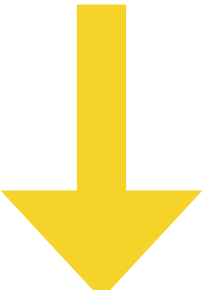
A screenshot of a Java IDE's console window. The window title is "Console" with a maximize icon. The status bar at the top indicates "<terminated> Main (18) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (20 Dec 2016, 16:00:24)". The main text area displays a red error message: "Exception in thread 'main' java.lang.ArithmeticException: / by zero". Below this, the stack trace is shown in red text: "at edu.tum.cs.i1.ease.Student.<init>(Student.java:22)" and "at edu.tum.cs.i1.ease.Main.main(Main.java:70)".

```
<terminated> Main (18) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (20 Dec 2016, 16:00:24)
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at edu.tum.cs.i1.ease.Student.<init>(Student.java:22)
    at edu.tum.cs.i1.ease.Main.main(Main.java:70)
```

Example Exception Handling in Java

```
class Account {
    private int balance;

    public int withdraw(int amount) {
        if (amount > balance)
            return -1;
        else {
            balance -= amount;
            return 0;
        }
    }
}
```



```
class Account {
    private int balance;

    public void withdraw(int amount) throws BalanceException {
        if (amount > balance) throw new BalanceException();
        balance -= amount;
    }
}
```

Bank Customer invoking the
withdraw() method

```
try {
    account.withdraw(123);
} catch (BalanceException e) {
    //Deal with the exception
    System.out.println("Not enough money!");
}
```

Terminology Checked vs. Unchecked Exceptions

Unchecked Exception

- Runtime exceptions that the calling method does not need to handle, e.g. `NullPointerException`
- It is the responsibility of the caller to do any testing

Checked Exceptions

- The calling method has to handle the possible exceptions using a `try-catch` block
- Alternative: Declare that the calling method passes the exception using `throws`
- It is the responsibility of the callee to do any testing

Good Design

- Prefer unchecked exceptions because your code gets more robust (often you cannot expect that the called method does any checking)
- Robustness as design goal

Software Engineering Essentials



Exception Handling

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics

