

# PV260 QUALITY IN SOFTWARE ENGINEERING

Lean Architecture and Best Practices



Ondřej Krajíček

[ondrej.krajicek@ysoft.com](mailto:ondrej.krajicek@ysoft.com)

<http://linked.in/in/ondrejkrjicek>

@hedragon @ysoftdevs







Kryštof 007

681 007-1





SoftBank

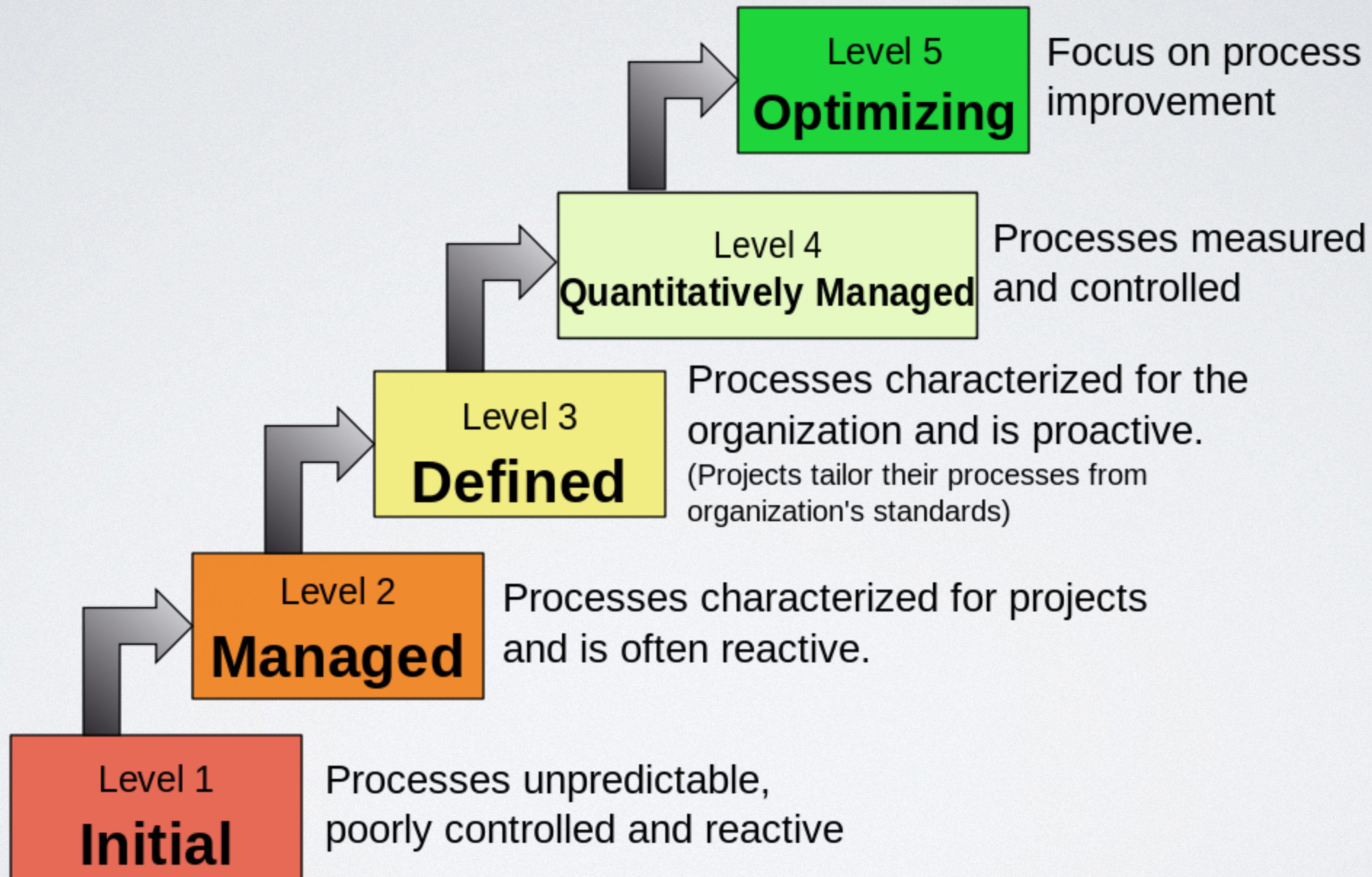
11:24

📶 ✖️ 68

306

km/h

# Characteristics of the Maturity levels



*Organisations which design systems are constrained to produce designs which are copies of the communication structures of these organisations.*

—M. CONWAY



**Architecture**

**Design**

**Code**

# ARCHITECTURE AND DESIGN

**SOFTWARE ARCHITECTURE**  
GIVES ANSWERS TO THE MOST  
EXPENSIVE QUESTIONS.

# SOFTWARE ARCHITECTURE

IS THE SERVANT OF HIGH-PRIORITY STAKEHOLDER VALUES.

IS AS **SIMPLE** AS POSSIBLE, BUT **NOT SIMPLER** AND

IS DESIGNED TO BE **REPLACEABLE**.

(TOM GILB)

- Software Architecture
- Servant to Stakeholder Values
- Simple, but not Simpler
- Replaceable

**SOFTWARE DESIGN** IS A DECOMPOSITION  
OF A SYSTEM  
INTO A SET OF **CONTRACTS**  
BASED ON  
ASSOCIATED QUALITIES,  
SO THAT CONTRACTS RESPECT SYSTEM  
BOUNDARIES.

**SOFTWARE DESIGN** IS ABOUT ANSWERS TO  
QUESTIONS WHERE **CHANGES** GO  
**BEYOND A SINGLE UNIT**, SUCH AS A CLASS OR A  
MODULE.

CODE...



...PROVIDES  
**(SOMETIMES INTENDED)**  
BEHAVIOR.

QUALITY OF THE DESIGN

- **Extreme Programming**
- **SOLID GRASP Principles**
- Design Patterns
- MVC / ECB Patterns
- Refactoring

# EXTREME PROGRAMMING

- KISS (*Keep it Simple and Stupid*)
- YAGNI (*You Ain't Gonna Need it*)

# XP SIMPLICITY RULES

- Kent Beck's Four Rules of Simple Design
  - Tests Pass
  - Expresses Intent
  - No Duplication
  - Minimalistic Code

# TESTS PASS

- **Tests tell you when you are done.**
  - Verification and Validation Tests
  - Automatic or Manual
  - Unit Testing

# UNIT TESTING

- What is a Unit?
- Basic Structure of a Unit Test (Arrange - Act - Assert)
- Best practices?
  - Single assert per test.
- Test Driven Development (Chicago vs. London School of Unit Testing)
  - Black-box vs. White-box Approach



# EXPRESSES INTENT

- Code comments...
- ...are not enough!
- Self-documenting Code  
<http://c2.com/cgi/wiki?SelfDocumentingCode>

# EXPRESSES INTENT (2)

- Methods vs. Code Fragments

<http://c2.com/cgi/wiki?MethodsVsCodeFragments>

- Separate Interfaces from Implementation

<http://c2.com/cgi/wiki?SeparateInterfacesFromImplementation>

- Handle Errors in Context

<http://c2.com/cgi/wiki?HandleErrorsInContext>

# HANDLING ERRORS

- Handle errors in the same place where you detected them.
- Handle errors as late as you possibly can.

# NO DUPLICATION

- What can be duplicated?
  - Code Blocks
  - Methods
  - Classes
  - Functions
  - Components
  - Exceptions
  - ?

# MINIMALISTIC CODE

- What minimalism mean? What do we want to minimize?
- What can be minimized?

# COUPLING AND COHESION

- Cohesion...
  - How different parts of an interface / contract are related to each other and cooperate together.
  - Classes with high cohesion have a *split personality*.
- Coupling...
  - The interaction dependency between different parts of the system.
  - Coupling is directly related to decomposition and you need to keep it in mind when you decompose.

SOLID GRASP

- **S**ingle Responsibility Principle
- **O**pen / Closed Principle
- **L**iskov Substitution Principle
- **I**nterface Segregation Principle
- **D**ependency Inversion



# GRASP

- General Responsibility Assignment Software Patterns
- A set of design patterns or aspects emphasising good coding practices.
- Might be useful, but only after you have good understanding of **SOLID** principles.

# DESIGN PATTERNS

- GRASP (to some extent)
- GoF (Creational, Behavioral and Structural)
- Model-View-Controller
- Entity-Boundary-Control

FOOD FOR THOUGHT

- Example of just-in-time-design : Refactor to Open/Closed  
<http://blog.goyello.com/2014/02/11/example-of-just-in-time-design-refactor-to-open-closed/>
- Extreme Programming Simplicity Rules  
<http://www.c2.com/cgi/wiki?XpSimplicityRules>
- Distributed Big Balls of Mud  
[http://www.codingthearchitecture.com/2014/07/06/distributed\\_big\\_balls\\_of\\_mud.html](http://www.codingthearchitecture.com/2014/07/06/distributed_big_balls_of_mud.html)
- Microservices until Macrocomplexity  
<https://michaelfeathers.silvrback.com/microservices-until-macro-complexity>
- Microservices and the failure of Encapsulation  
<https://michaelfeathers.silvrback.com/microservices-and-the-failure-of-encapsulaton>
- CodeSmell  
<http://c2.com/cgi/wiki?CodeSmell>
- Classic TDD or London School  
<http://codemanship.co.uk/parlezuml/blog/?postid=987>
- Model-View-Controller - a terrific MVC diagram  
<http://alvinalexander.com/uml/uml-model-view-controller-mvc-diagram>