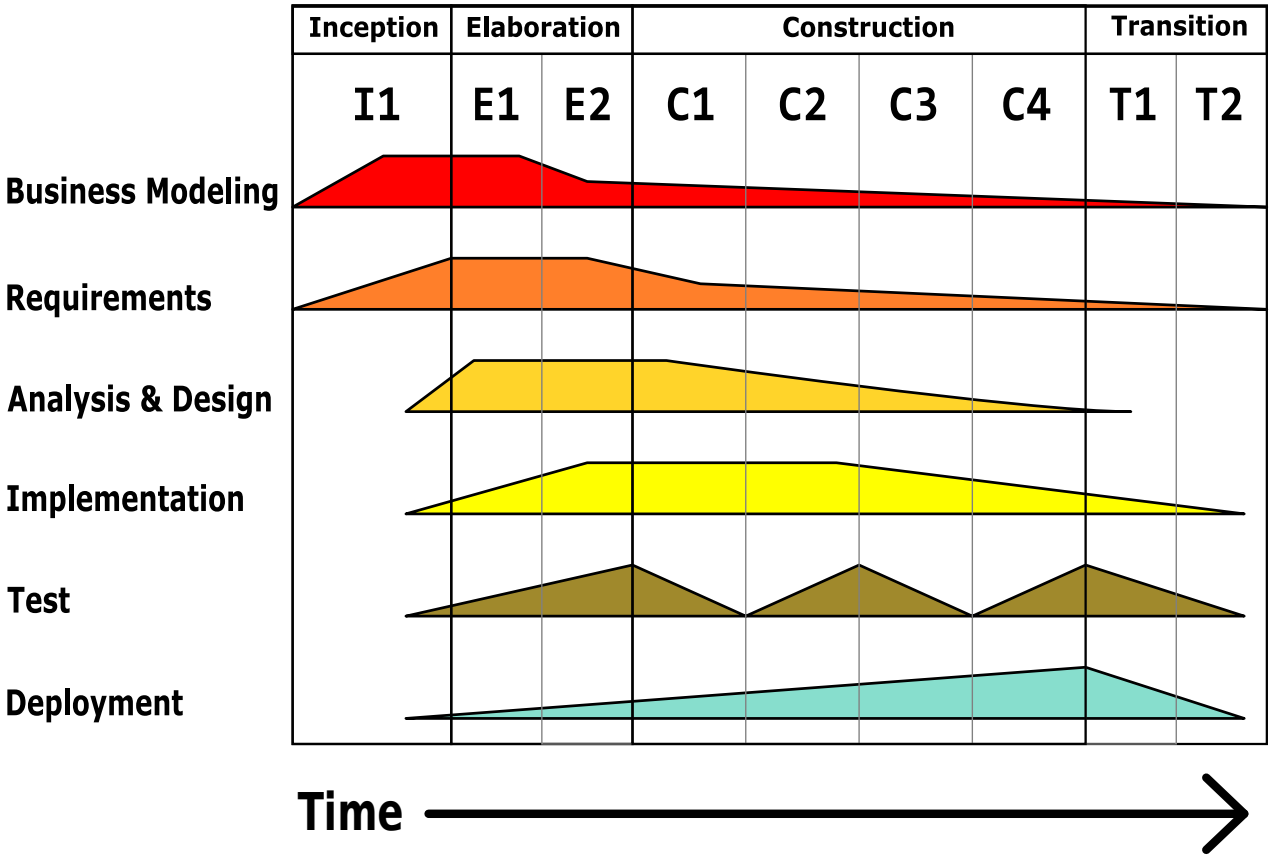
Introduction to Software Requirements

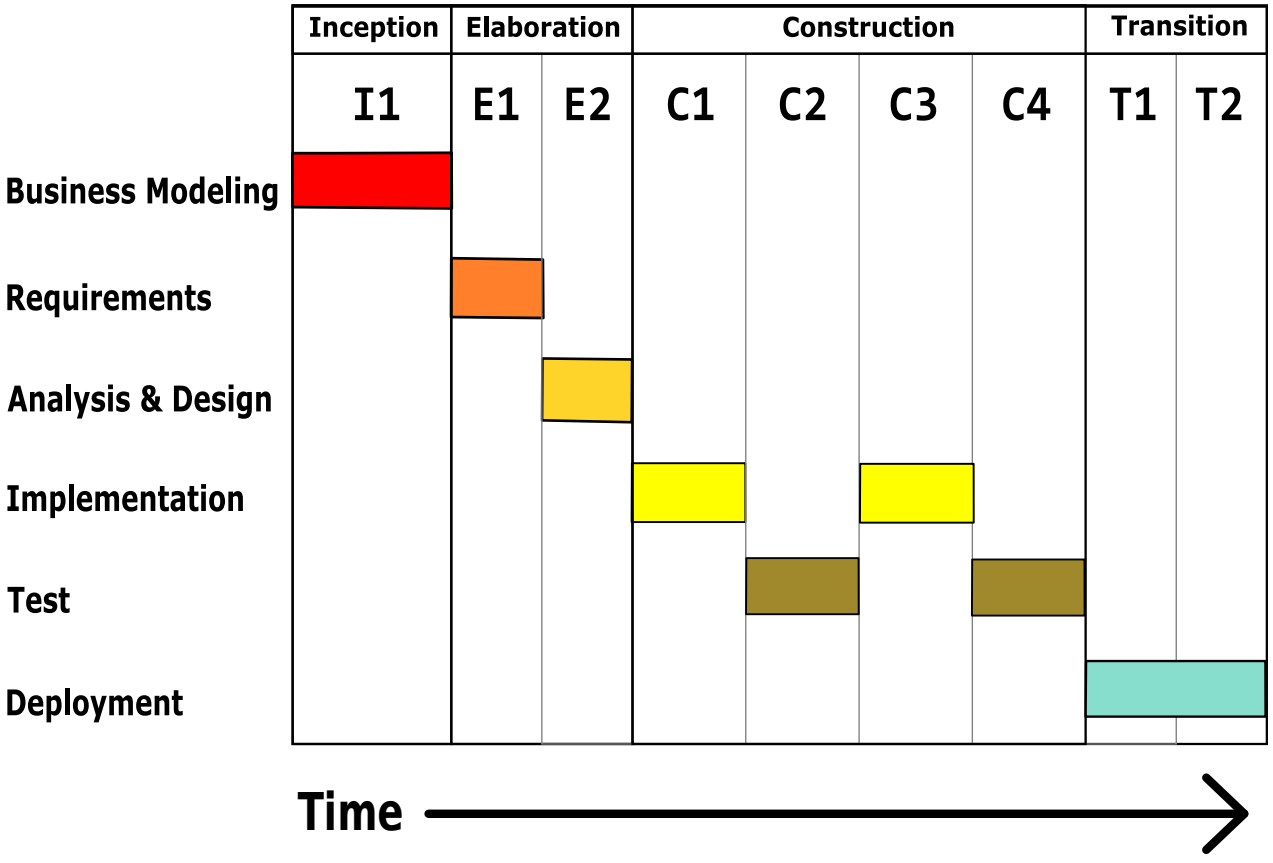
- Modern Iterative Development. Pitfall: Agile Waterfall™
- `define:Software Requirements`
- Software Requirements: Levels & Types
  - Functional/Non-Functional (esp. Quality Attributes)
  - External/Internal
  - Product/Project
  - Today/Tomorrow
- Design Metamodel. Design Tradeoffs
- Requirements Management
- System Vision
- User Stories

Iterative Development

Business value is delivered incrementally in time-boxed crossdiscipline iterations.



Agile Waterfall. So iterative, very agile!



Software Requirements

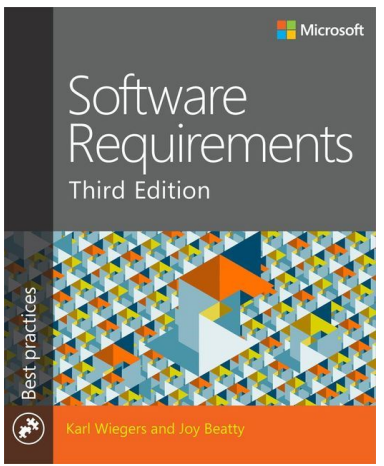
…are a definition of **what should be implemented**.

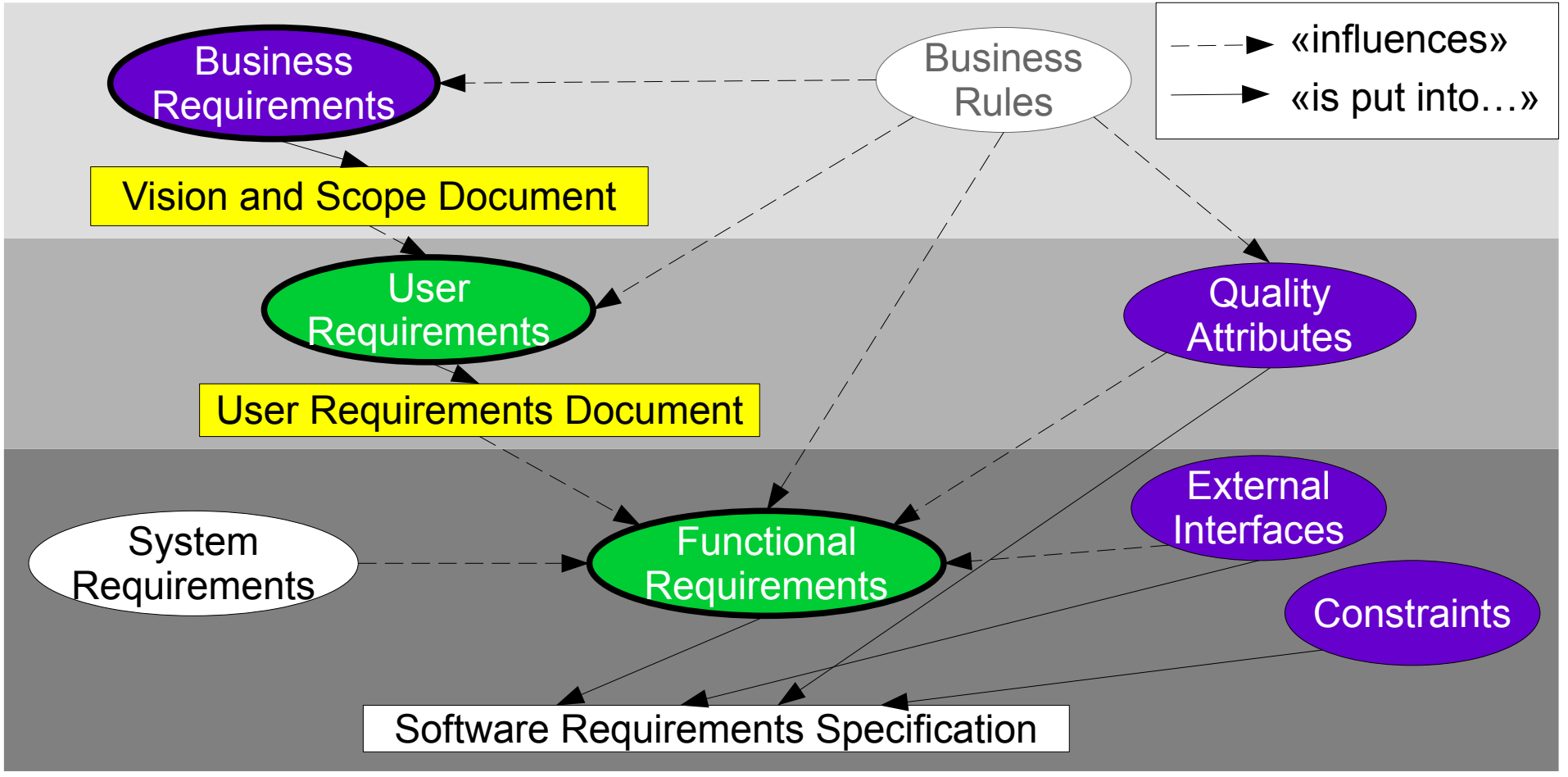
They are descriptions of **how a system should behave**, or of a **system property or attribute**.

They may be a **constraint on the development process** of the system.

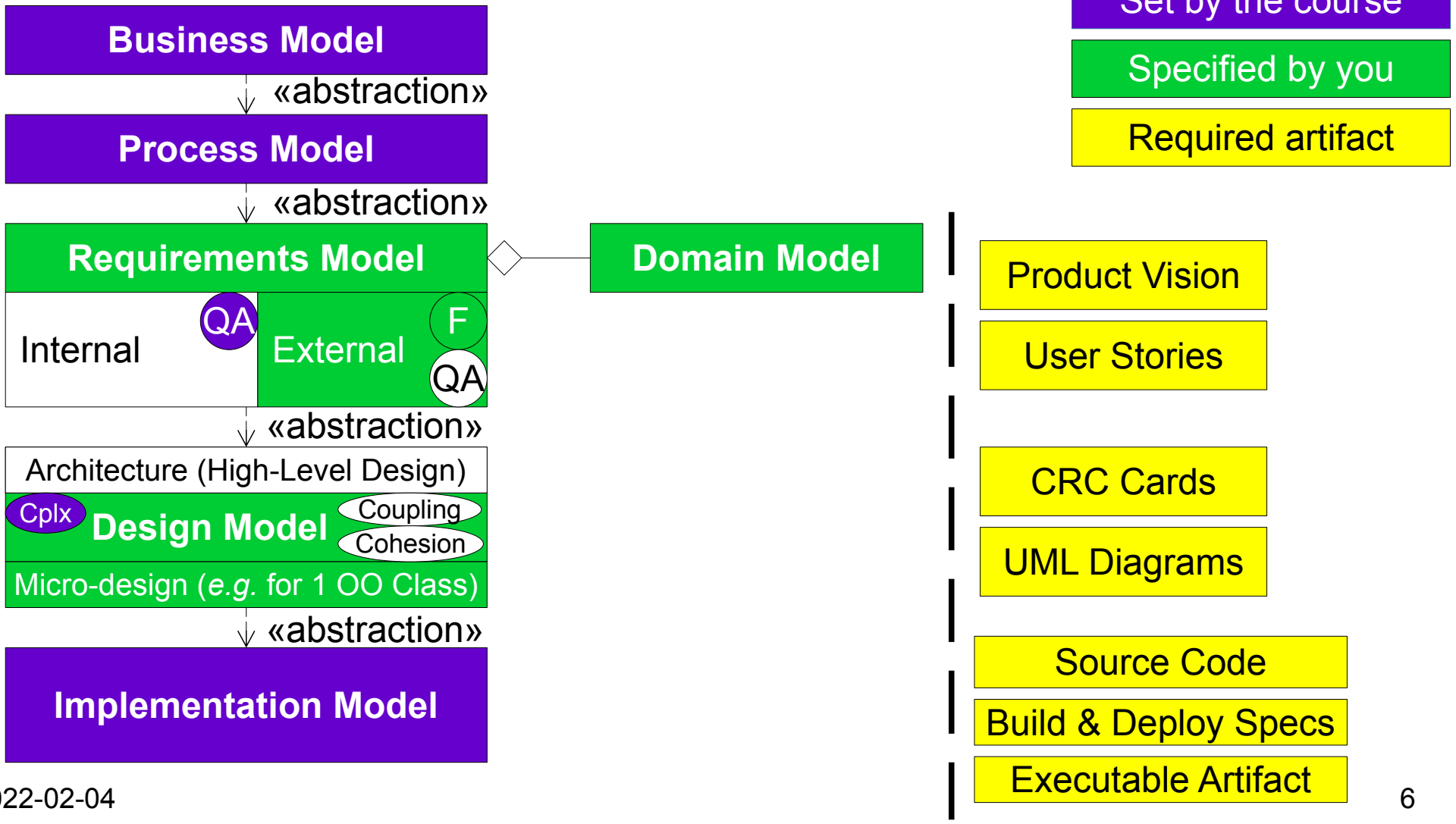
Ian Sommerville, Pete Sawyer. *Requirements Engineering: A Good Practice Guide.* (Wiley, 1997)

via: Karl Wiegers, Joy Beatty. *Software Requirements, 3rd Ed.* (Microsoft Press, 2013)





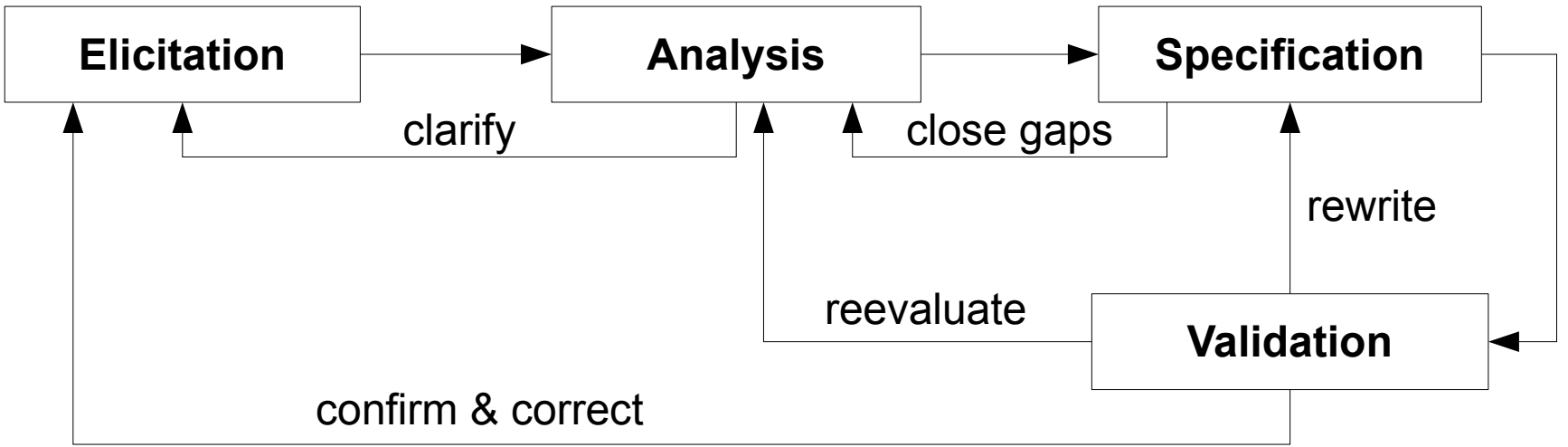
Karl Wiegers, Joy Beatty. *Software Requirements, 3rd Ed.* Ch. 1, p. 8, Fig. 1-1 (Microsoft Press, 2013)


Set by the course
Specified by you
Required artifact
Business Model
«abstraction»
Process Model
«abstraction»
Requirements Model
Internal
QA
External
F
QA
Domain Model
«abstraction»
Architecture (High-Level Design)
Cplx
Design Model
Coupling
Cohesion
Micro-design (e.g. for 1 OO Class)
«abstraction»
Implementation Model
Product Vision
User Stories
CRC Cards
UML Diagrams
Source Code
Build & Deploy Specs
Executable Artifact

Design Tradeoffs

- {Testability, Flexibility} → Complexity
  - Conflicts with Simplicity
  - Too much Flexibility is Overengineering
- {Scalability, Resilience, Independent Evolution} → Consistency
  - Microservices. Now you have lots of stores instead of one, and Transactions are gone. There are only Sagas
- Availability → Consistency (CAP Theorem)

Requirements Management



Karl Wiegers, Joy Beatty. *Software Requirements, 3rd Ed.* Fig. 3-1 (Microsoft Press, 2013)

System Vision

- **Birds' Eye View** of the System
- For **Stakeholders**
  - Business (management, financial etc.)
  - Prominent Users
- Outlines **System Scope** (what it IS and IS NOT)
- Main Questions
  - Why should the Product **exist**? (What is the **need** for it?)
  - Why would it **be successful**?
- Elevator Test

User Stories

As a **<role>** I can **<capability>**, so that **<receive benefit>**
As **<who>** **<when>** **<where>**, I want **<what>** because **<why>**

As an **amateur skier**,

I can **rent ski equipment**

So that I **post a cool skiing photo on Instagram** to impress my pretentious friends

As a **professional skier**,

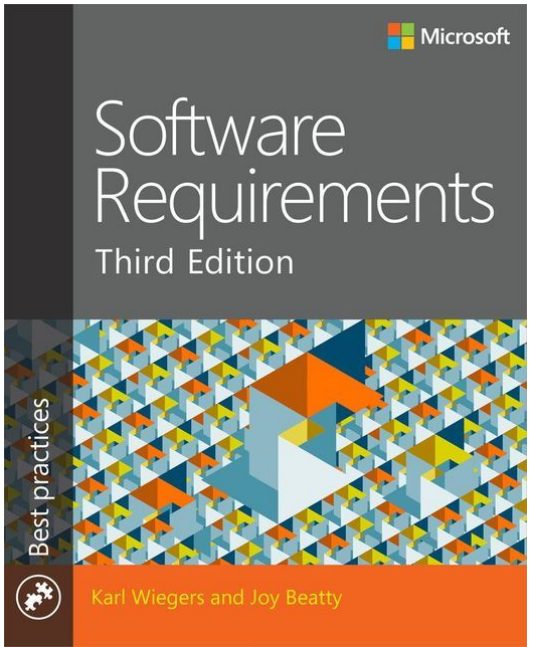
I can **rent high-end ski equipment**

So that I **regularly train for Winter Olympics**

https://en.wikipedia.org/wiki/User_story

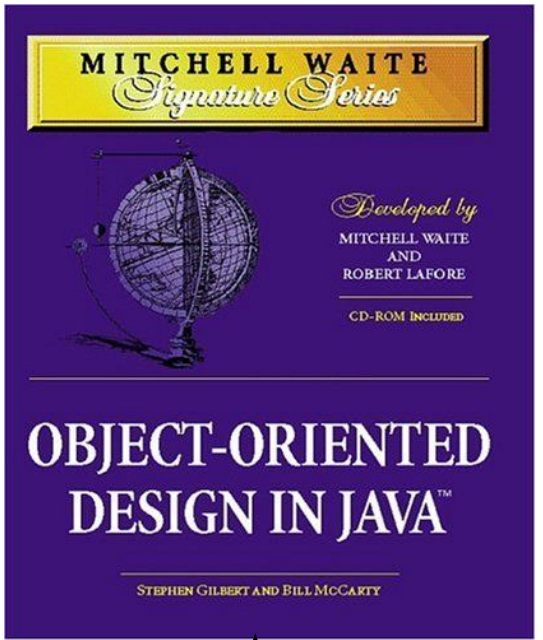
https://pmclub.pro/articles/user-story-pora-primenyat-pravilno

Recommended Reading

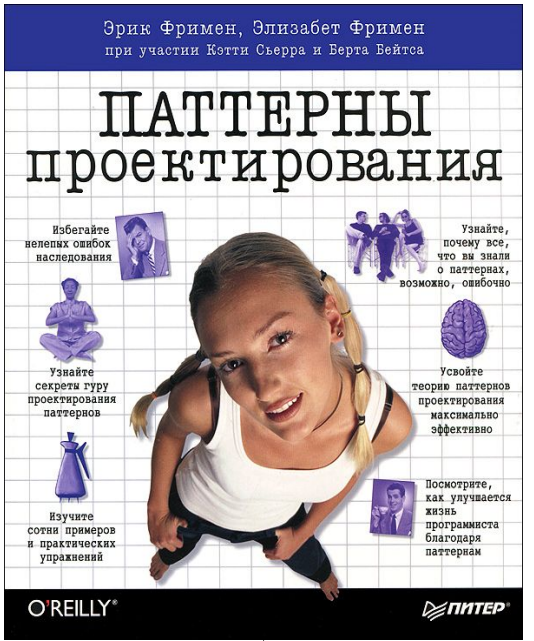


Chapter 1

Chapter 3



OOD: (This Seminar + 1)
Chapters 5..8



Patterns:
(This Seminar + 2)

Recommended Reading (*Contd.*)

- *Head First Patterns* by Eric & Elizabeth Freeman
- *Code Complete* by Steve McConnell
- *Object-Oriented Design in Java*
  by Gilbert & McCarty (https://www.amazon.com/MWSS-Object-Oriented-Design-Mitchell-Signature/dp/1571691340)
  @*see* Chapters 5..8