**Silabus Rekasaya Perangkat Lunak (PTI Semester 4)**

Dosen: Priyanto

## Tujuan Mata Kuliah

Tujuan kuliah ini adalah agar mahasiswa memahami teori dasar dan tahapan rekayasa perangkat lunak, dan menerapkan prinsip-prinsip teori dasar ini pada proyek pengembangan perangkat lunak.

| No | Topik | Subtopik | Acuan |
|---|---|---|---|
| 1 | Pendahuluan | • Silabus dan Peraturan perkuliahan<br>• Meluruskan keslahan dalam RPL | Pressman (2010) |
| 2 | Software Engineering | The nature of software, the unique nature of WebApps, the software process (communication, planning, modeling; construction, deployment) | 3-14 |
| 3 | Process Model 1 | The waterfall model, incremental process, RAD model, evolutionary process models | 39-50 |
| 4 | Process Model 2 (Agile Development) | Extreme programming (XP); Adaptive Software Development (ASD); Dynamic System Development Model (DSDM), Feature Driven Development (FDD) | 67-87 Optional |
| 5 | Software Engineering Practice | • Core principles.<br>• Principles: communication, planning, modeling; construction, deployment | 98-115 |
| 6 | Requirements Modeling | • Requirements analysis, data modeling, class-based modeling<br>• Flow-oriented modeling, behavior modeling | 186-198 |
| 7 | Design Concept | • Design concept: Abstraction, modularity, information hiding, fuctional independence (coupling and cohesion)<br>• The design model: data design elements, architectural design elements, interface design elements, component-level design elements, deployment-level design elements. | 216-240 |
| 8 | Architectural Design Concept | • Architectural style<br>• Architectural mapping using data flow: transform flow, transaction flow, transform mapping, transaction mapping. | 249-272 |
| 9 | Component-Level Design | • Component: an object-oriented view, the traditional view<br>• Designing class-based components<br>• Designing traditional components | 277-302 |
| 10 | User Interface Design | The golden rules, interface design steps | 312-321 |
| 11 | Software Testing Strategies | • A strategic approach to software testing<br>• Test strategies: unit testing, Integration testing<br>• Validation testing: Alpha and Beta testing<br>• System testing: recovery testing, security testing, stress testing, performance testing, deployment testing. | |
| 12 | Testing Conventional Application | • Software testing fundamentals<br>• Whitebox testing: basis path testing; control structure testing, blackbox testing<br>• Blackbox testing | 482-501 |

**Buku Acuan Utama:**

Pressman, R.S. (2010). *Software Engineering, A Practitioner'a Approach*. Seventh Edition. Singapore: McGraw-Hill Education.