

3

Materi Pembelajaran

Matakuliah :

REKAYASA PERANGKAT LUNAK

Kode Matakuliah : **SKO25412**

Dosen Pengampu Matakuliah:
Bayu Nugroho, S.Kom., M.Eng



Program Studi SISTEM KOMPUTER

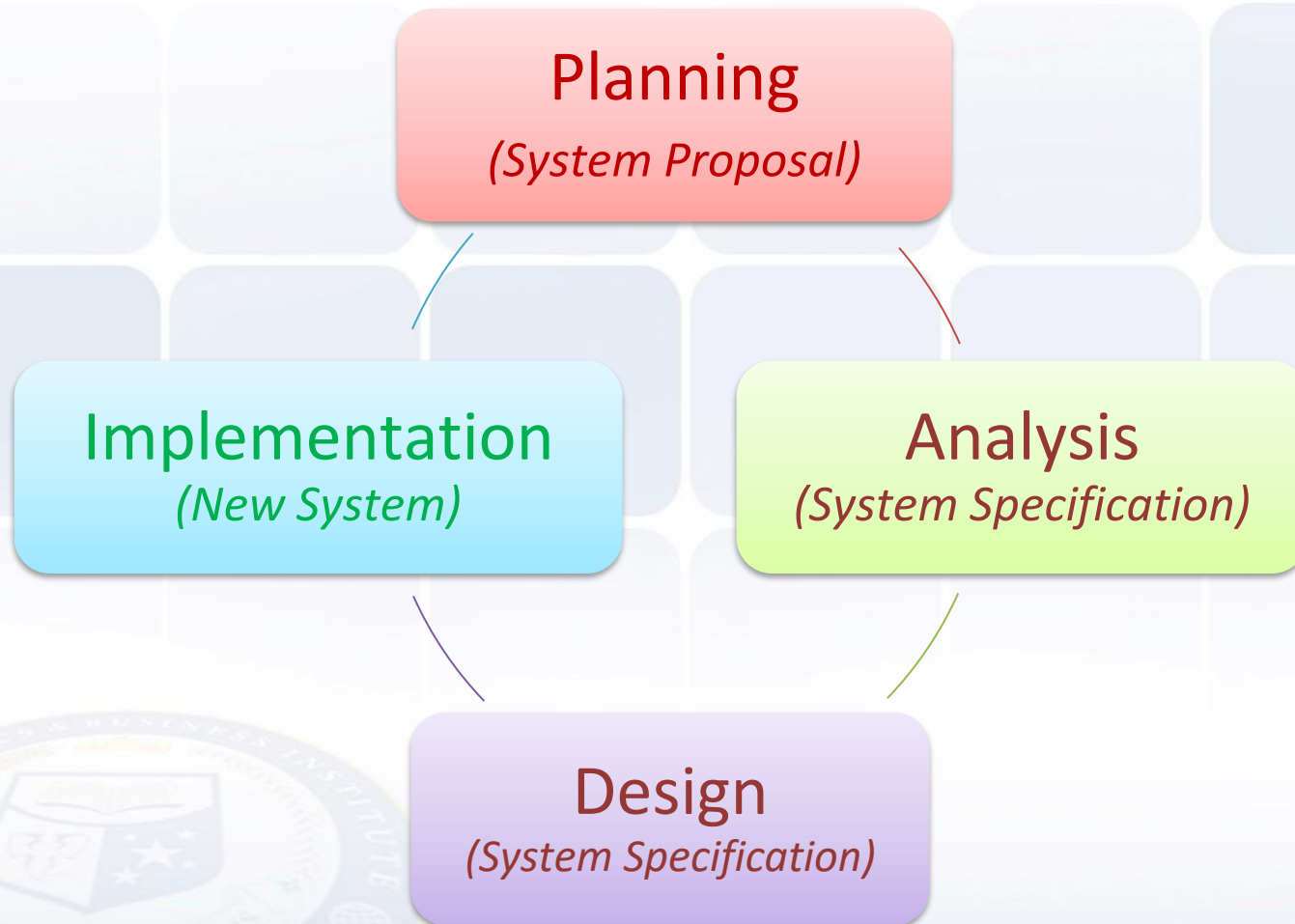
IIB DARMAJAYA

2026

Systems Development Life Cycle **(SDLC)**

3.1 Siklus Pengembangan Software

Siklus Pengembangan Software



(Tilley, 2012)

(Dennis, 2016)

(Valacich, 2017)

Systems Development Life Cycle (SDLC)

1. **Planning:** Mengapa Software harus Dikembangkan?
 - System request, feasibility analysis, project size estimation
2. **Analysis:** Siapa, Kapan Digunakan dan Alur Kerja Software?
 - Requirement gathering dan business process modeling
3. **Design:** Bagaimana Bekerja dan Komposisi dari Software?
 - Program design, user interface design, data design
4. **Implementation:** Konstruksi dan Penyerahan Software
 - System construction, testing, documentation, installation

Metodologi Pengembangan Software (Model Process)

- A formalized **approach to implementing** the Software Development Life Cycle (SDLC) (*Dennis, 2012*)
- A **simplified representation** of a software process (*Sommerville, 2015*)
- A distinct **set of activities**, actions, tasks, milestones, and work products required to engineer high quality software (*Pressman, 2015*)

Major Software Development Methodologies

1. Structured Design

(Prescriptive) (1967-)

- Waterfall method
- Parallel development

2. Rapid Application Development

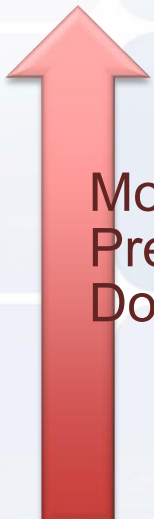
(Iterative) (1985-)

- Phased Development
- Prototyping

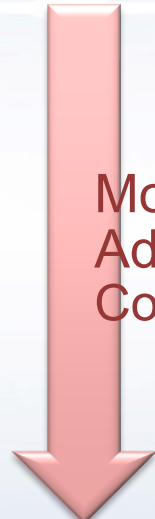
3. Agile Development

(Adaptive) (1995-)

- Extreme Programming (XP)
- Scrum



More
Prescriptive/
Documentation

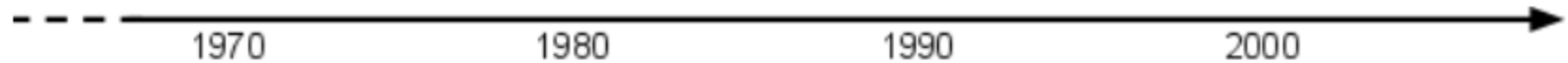


More
Adaptive/
Communication

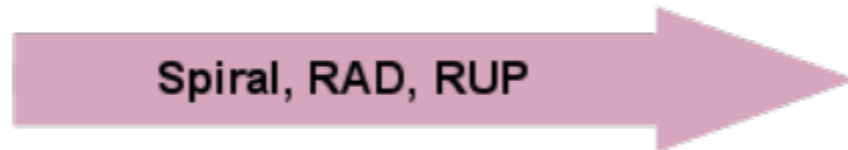
Compiled from (Dennis, Wixom and Tegarden, 2016)

Software Development Methodology

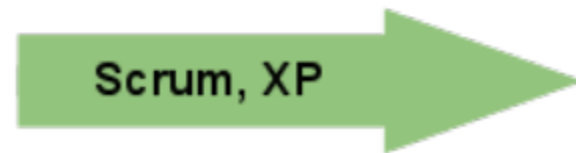
Timeline



Predictive: phases, documentation-centric, functional handoffs, get it right the first time



Iterative: process framework, phases, tool driven, artifact heavy

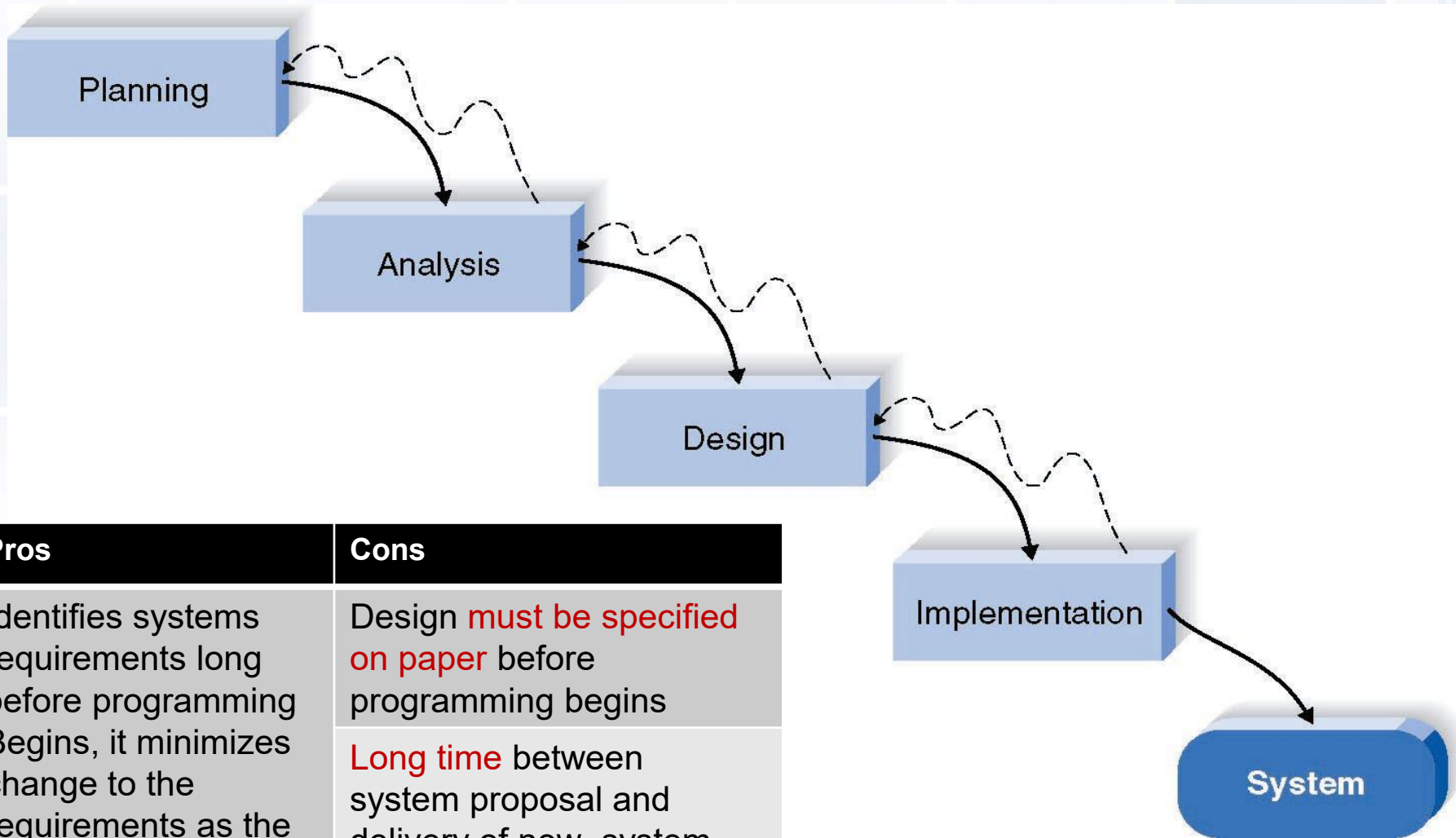


Adaptive: iterative, self-organizing teams, value driven, transparent

Structured Design

- Tahapan SDLC bergerak secara metodis dan sistematis
 - Setelah satu tahap selesai baru menuju ke tahap berikutnya
- Contoh Structured Design:
 1. Waterfall Method
 2. Parallel Development

Waterfall Method



Pros

Identifies systems requirements long before programming Begins, it minimizes change to the requirements as the project proceed (**mature**)

Cons

Design **must be specified on paper** before programming begins

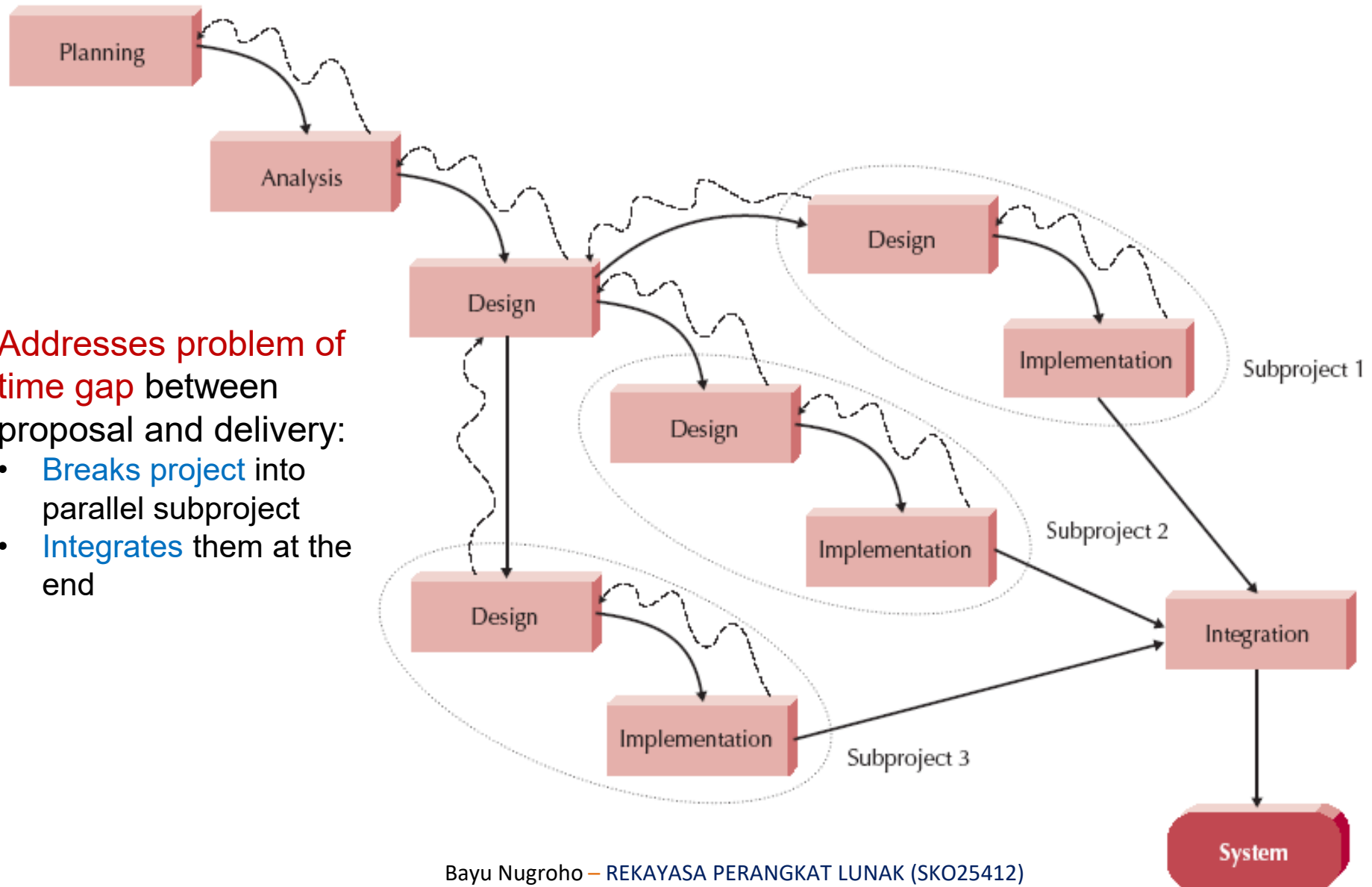
Long time between system proposal and delivery of new system

Rework is very hard

Parallel Development

Addresses problem of time gap between proposal and delivery:

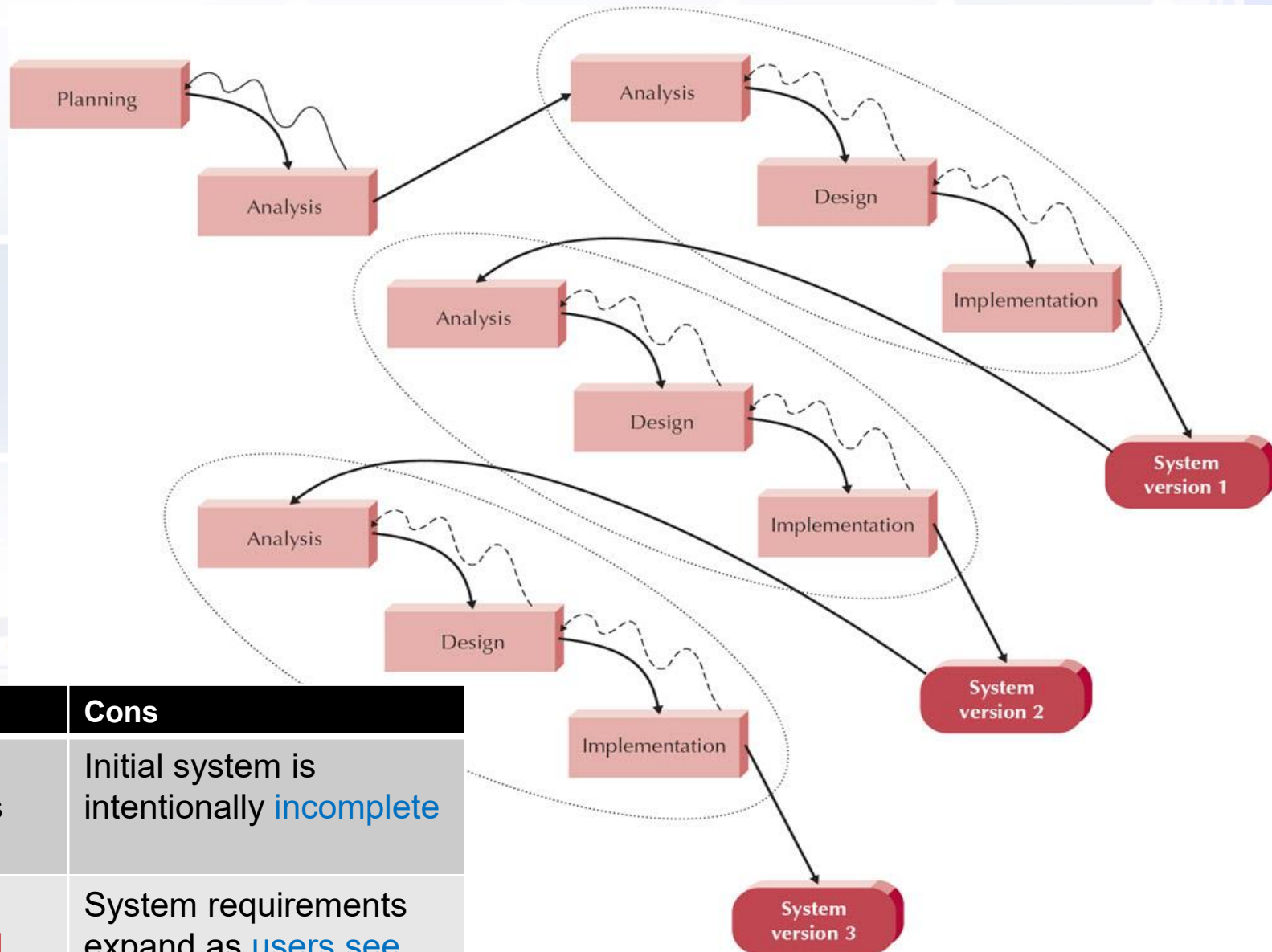
- Breaks project into parallel subproject
- Integrates them at the end



Rapid Application Development (RAD)

- Contoh **RAD**:
 1. **Phased development**: alur seri dari versi
 2. **Prototyping**: system prototyping
 3. **Throw-away prototyping**: design prototyping
- Teknik untuk **mempercepat SDLC**:
 - **CASE tools**
 - **Visual programming languages**
 - **Code generators**

RAD: Phased Development



Pros	Cons
Gets useful system to users quickly	Initial system is intentionally incomplete
Most important functions tested most	System requirements expand as users see versions

System Request: Sistem Penjualan Musik Online

Project Sponsor: Margaret Mooney, Vice President of Marketing

Business Needs: Project ini dibangun untuk:

1. Mendapatkan pelanggan baru lewat Internet

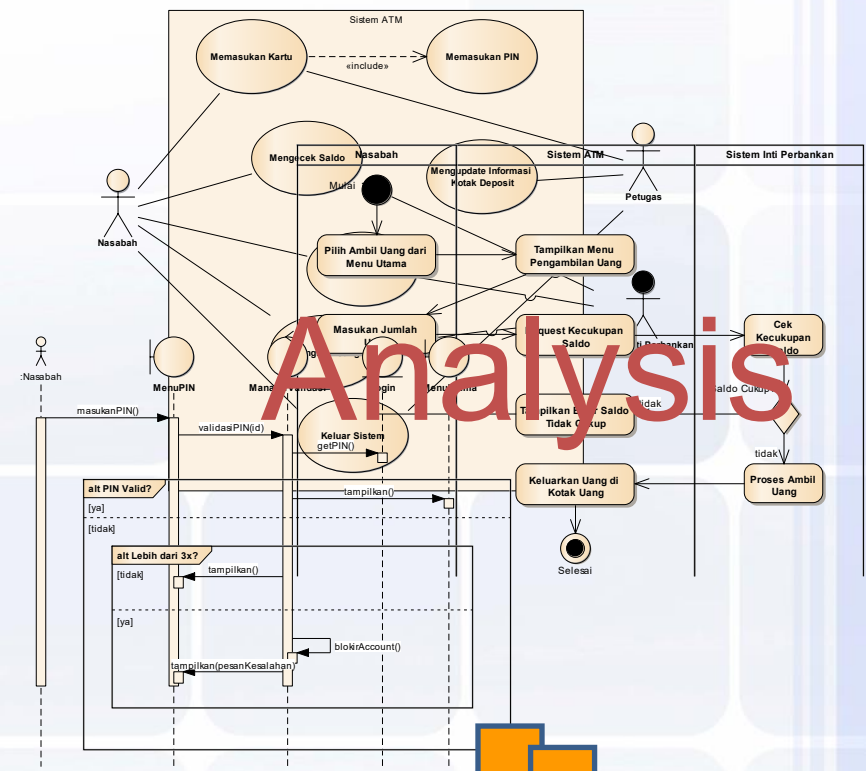
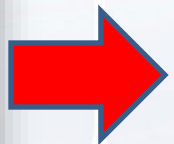
Studi Kelayakan Sistem Penjualan Musik Online

Business Requirement: Margaret Mooney dan Alec Adams membuat studi kelayakan untuk pengembangan Sistem yang mendukung Sistem Penjualan Musik Online

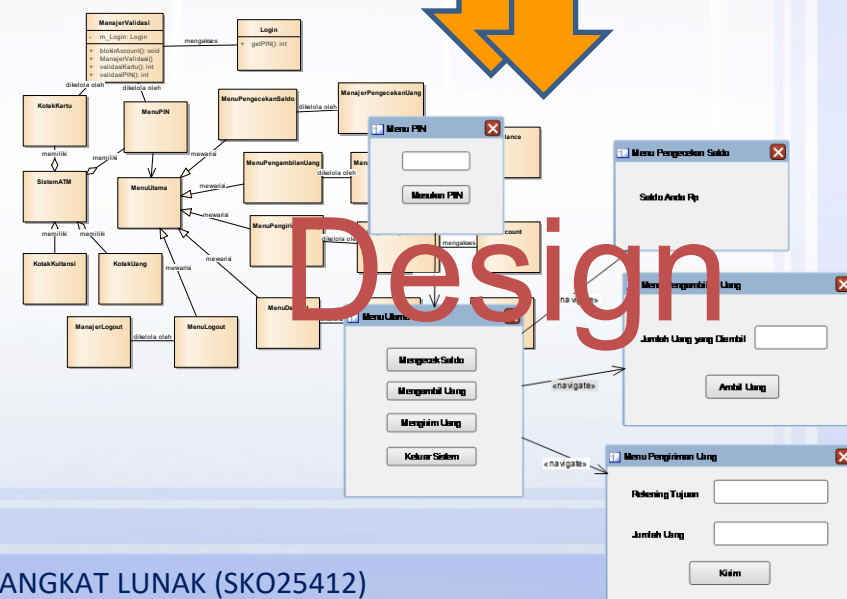
Kelayakan Teknis

	2016	2017	2018
Peningkatan penjualan dari pelanggan baru	0	400,000,000	500,000,000
Peningkatan penjualan dari pelanggan lama	0	600,000,000	700,000,000
Pengurangan biaya operasional dan telepon	0	100,000,000	100,000,000
Total Benefits:	0	1,100,000,000	1,300,000,000
PV of Benefits:	0	978,996,084	1,091,505,068
PV of All Benefits (Business Value):	0	978,996,084	2,070,501,152
Honor Tim (Planning, Analysis, Design and Implementation)	0	0	0
Honor Konsultan Struktur Internet	0	0	0
Total Development Costs:	0	0	0
Honor Pengelolaan Web	0	70,000,000	80,000,000
Biaya Lisensi Software	50,000,000	60,000,000	70,000,000
Hardware upgrades	100,000,000	100,000,000	100,000,000
Biaya Komunikasi	20,000,000	30,000,000	40,000,000
Biaya Marketing	100,000,000	200,000,000	300,000,000
Total Operational Costs:	330,000,000	460,000,000	590,000,000
Total Costs:	780,000,000	460,000,000	590,000,000
PV of Costs:	735,849,057	409,398,362	495,375,377
PV of all Costs:	735,849,057	1,145,247,419	1,640,622,796
Total Project Costs Less Benefits:	-780,000,000	640,000,000	710,000,000
Yearly NPV:	-735,849,057	569,597,722	596,129,691
Cumulative NPV:	-735,849,057	-166,251,435	429,878,356
Return on Investment (ROI) di Tahun 3:	429,878,356/1,640,622,796		0.262021445
Break-even Point (BEP):	2 + (596,129,691/429,878,356)		2.278884507

Planning



Analysis

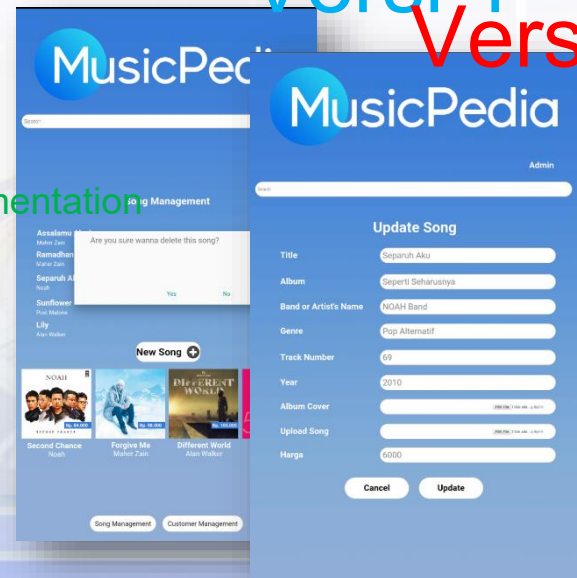


Design

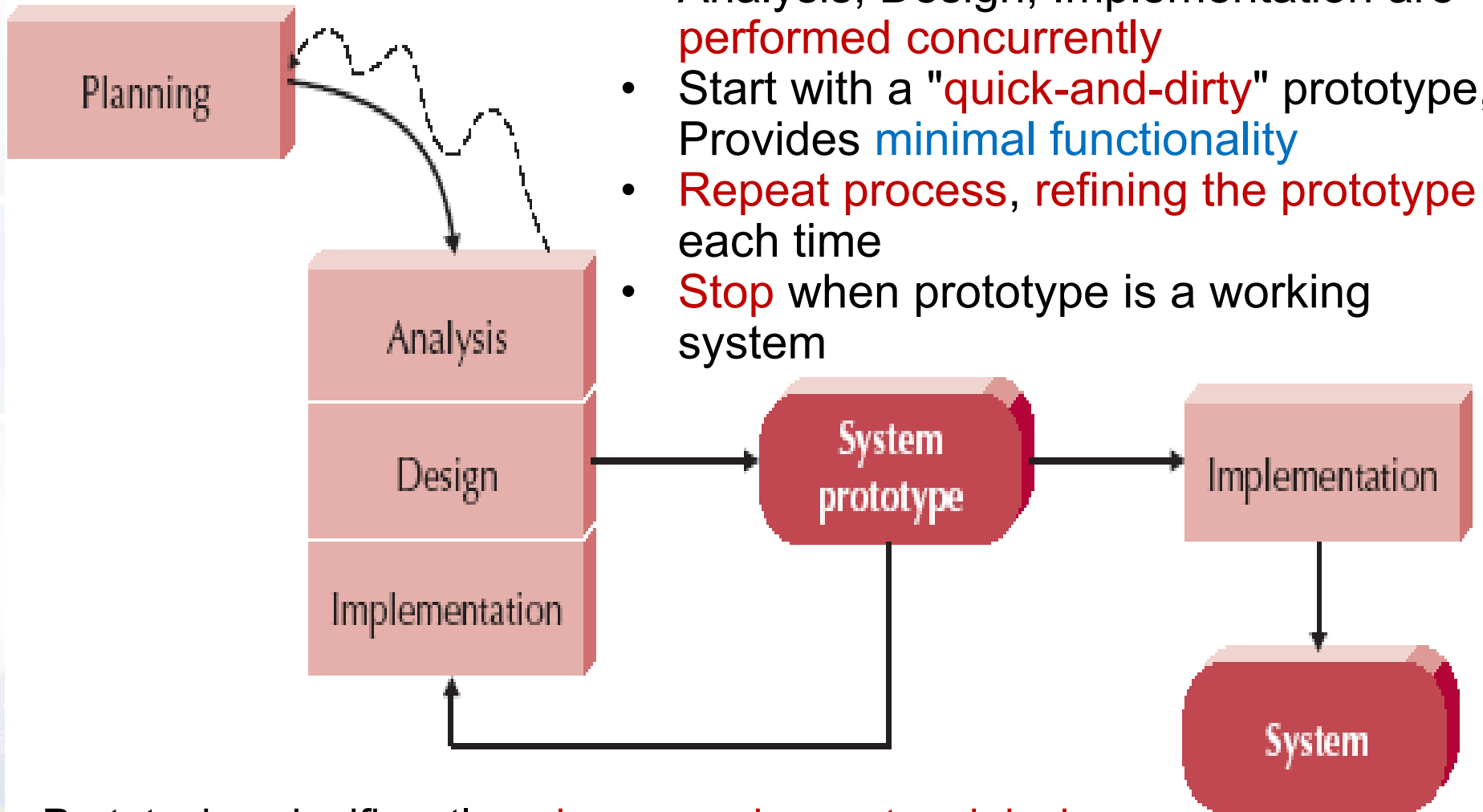


Versi 1
Versi 2

Implementation



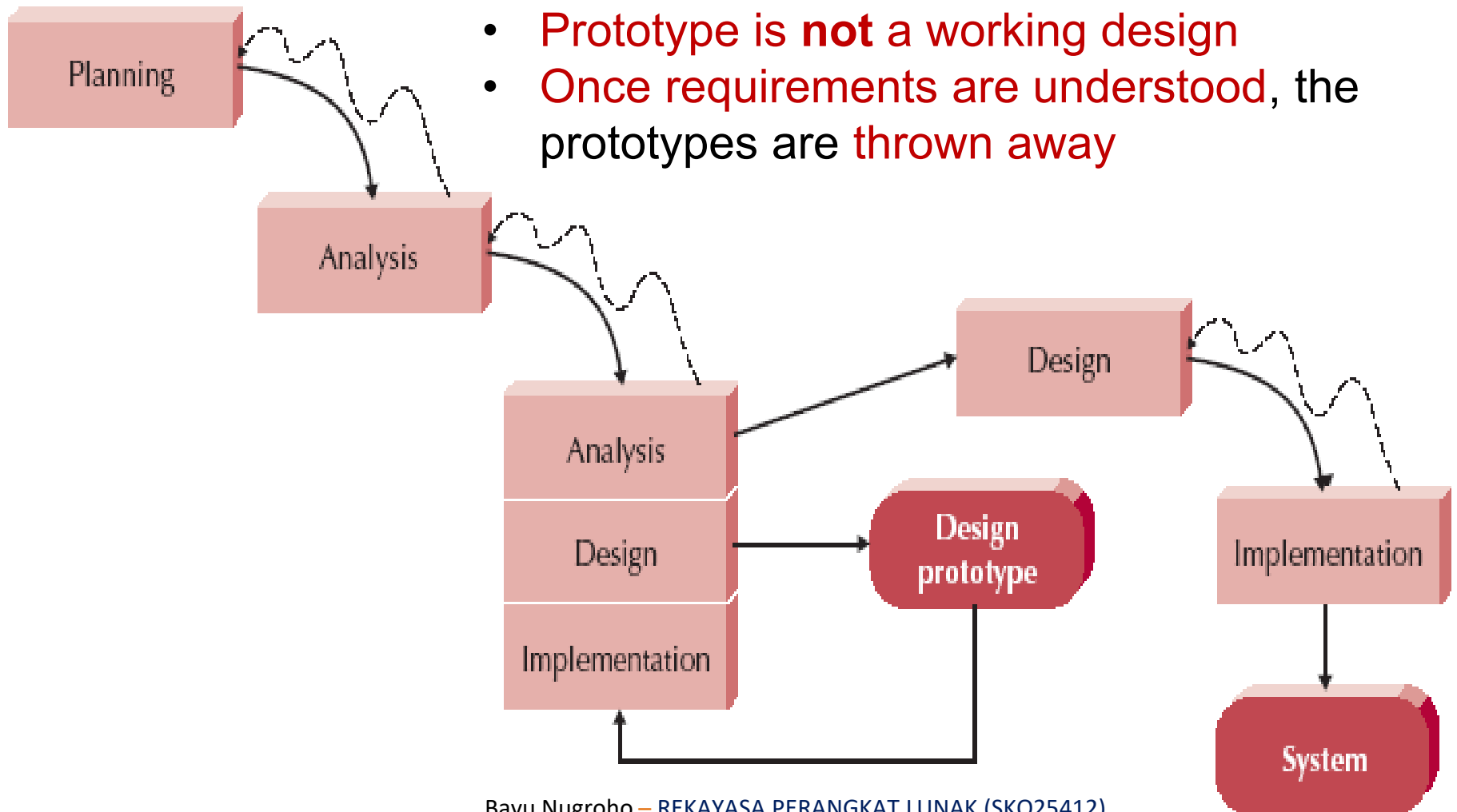
RAD: Prototyping



Prototyping significantly reduces requirement and design errors, especially for user interfaces (*Boehm's First Law, Endres, 2013*) [L3]

RAD: Throw-Away Prototyping

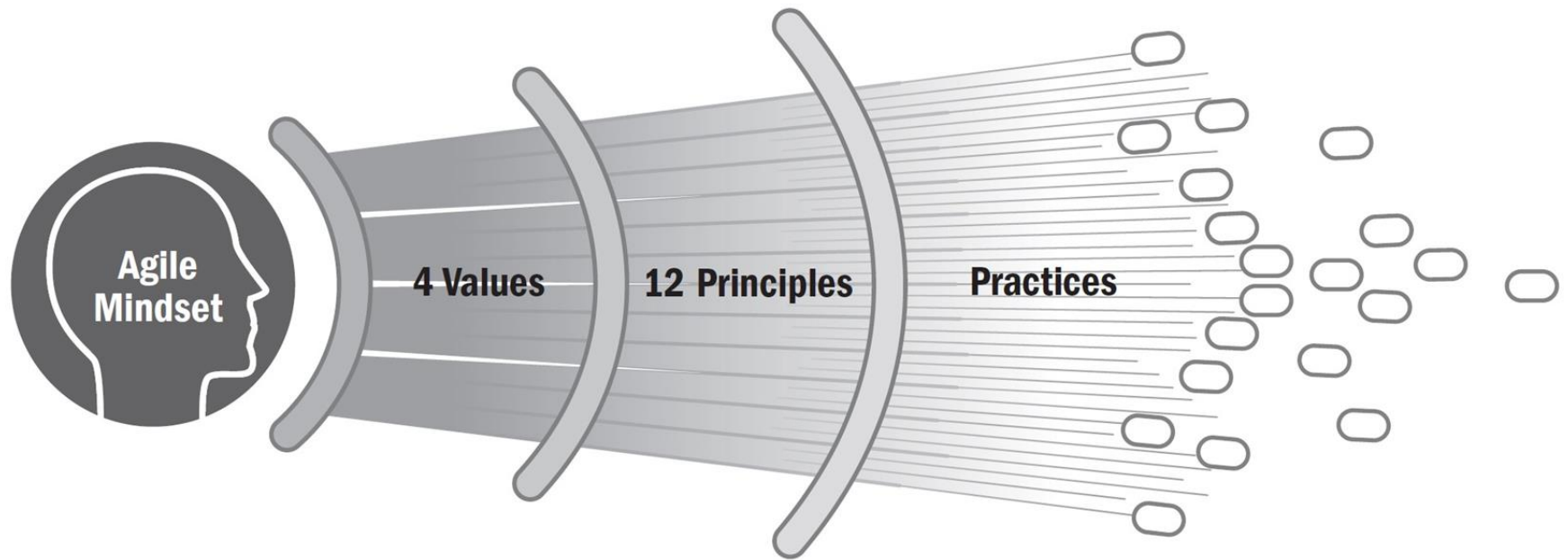
- Use prototypes only to understand requirements
 - Example: use html to show UI
- Prototype is **not** a working design
- Once requirements are understood, the prototypes are **thrown away**



Agile Development

- Menggunakan beberapa aturan yang mudah dipahami dan diikuti
- **Mempercepat proses SDLC**
 - Mengurangi pemodelan dan dokumentasi
 - Mengembangkan software dengan simple dan iteratif
- **Agile Approach:**
 1. Agile Values
 2. Agile Principles
 3. Agile Practices

Agile Values, Principles and Practices



4 Agile Values



Individuals and interactions over
processes and tools



Working software over
comprehensive documentation



Customer collaboration over
contract negotiation



Responding to change over
following a plan

Agile Principles

February 2001, **Jeff Sutherland** and **Ken Schwaber** were amongst the 17 software development leaders creating the **Manifesto for Agile Software Development**



Jeff Sutherland



Ken Schwaber

12 Agile Principles (Manifesto)

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

The best architectures, requirements, and designs emerge from self-organizing teams.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Simplicity, the art of maximizing the amount of work not done, is essential.

Deliver working software frequently, from every couple of weeks to couple of months, with a preference to the shorter timescale.

Continuous attention to technical excellence and good design enhances agility.

Business people and developers must work together daily throughout the project.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Working software is the primary measure of progress.

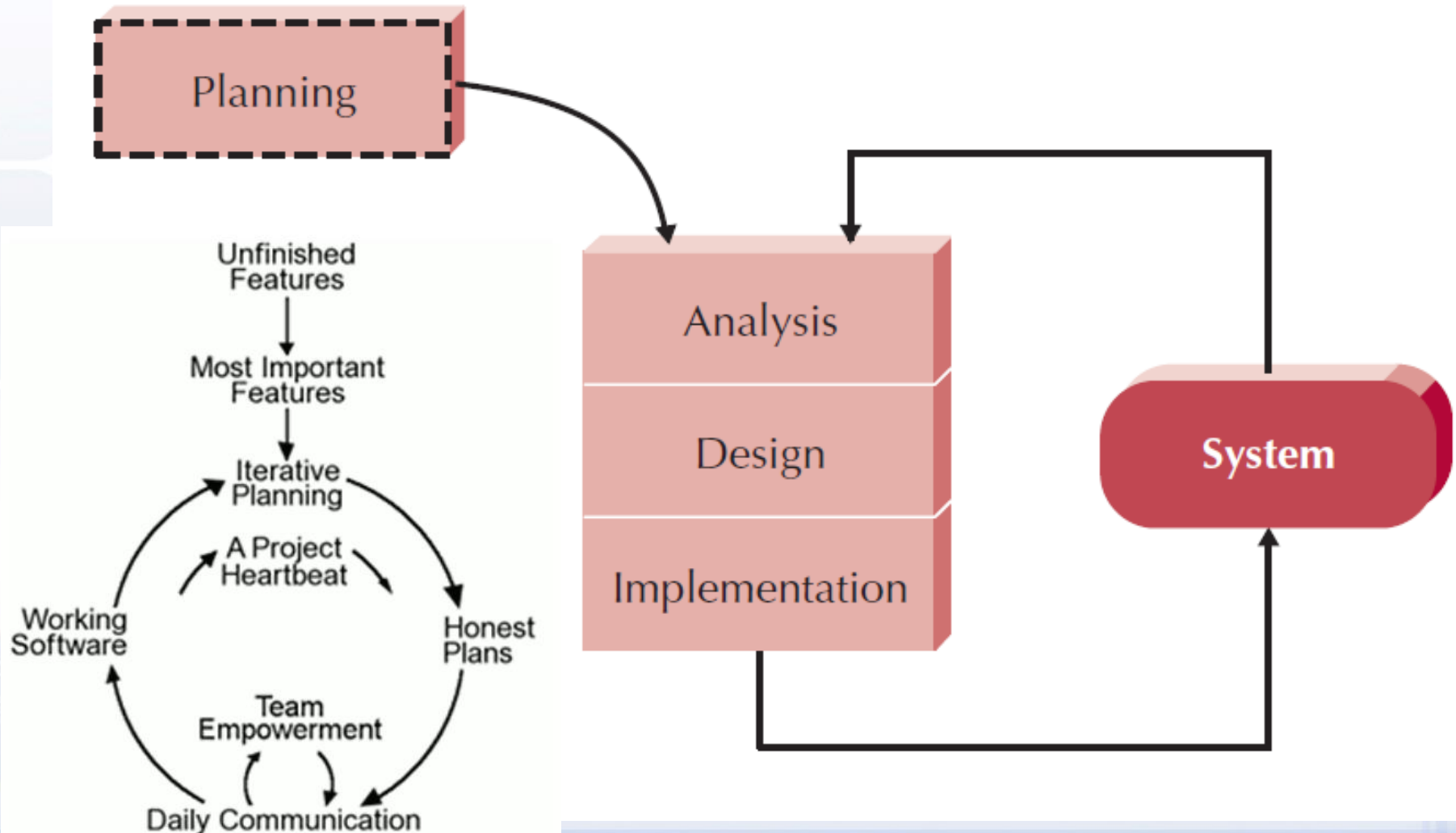
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Agile Practices

- **Extreme Programming** (XP) (Kent Beck)
- **Scrum** (Ken Schwaber and Jeff Sutherland)
- Lean Development (Mary Poppendieck and Tom Poppendieck)
- Dynamic Systems Development Model (DSDM) (Dane Faulkner)



Extreme Programming (XP)

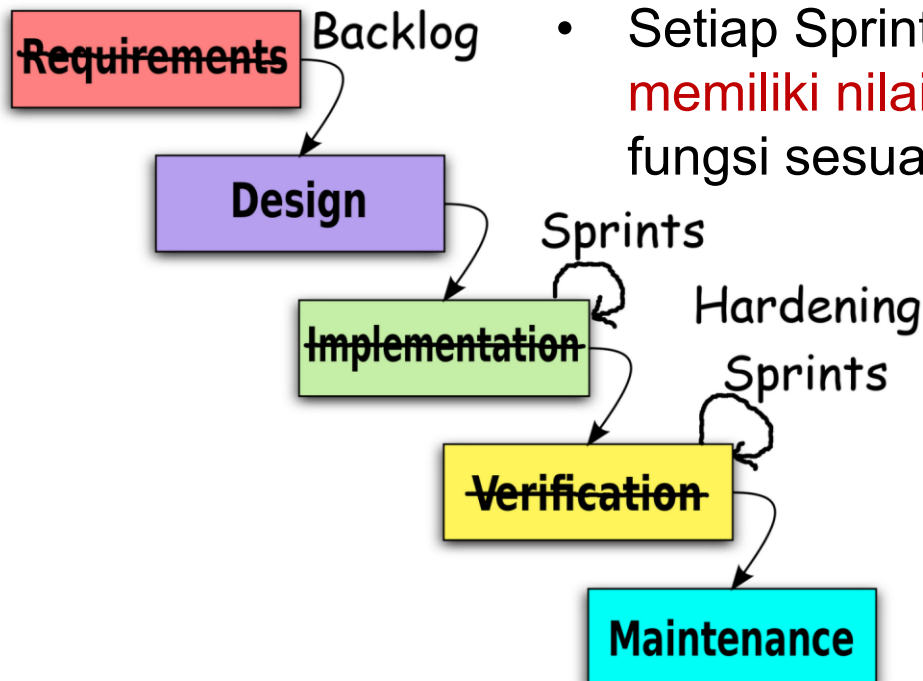


XP Values

1. **Communication**: Building software requires communicating requirements to the developers
 1. Pair Programming
 2. Communication replace documentation
2. **Simplicity**: Encourages starting with the simplest solution, extra functionality can then be added later
3. **Feedback**:
 1. Feedback from the system: by writing unit tests, or running periodic integration tests, the programmers have direct feedback from the state of the system after implementing changes
 2. Feedback from the customer: The acceptance tests are planned once in every two or three weeks so the customer can easily steer the development
 3. Feedback from the team: When customers come up with new requirements in the planning game the team directly gives an estimation of the time that it will take to implement
4. **Courage**: Several practices embody courage. One is the commandment to always design and code for today and not for tomorrow

Scrum

- Tim terdiri dari **3-9 pengembang**
- Tujuan dari **Sprint** adalah **menentukan**, **memprioritaskan**, dan **memecah** pekerjaan menjadi tugas yang lebih detail
- Tim memiliki karakter **self-organized**, bisa mengelola diri sendiri, memahami tugas, dan bertanggungjawab terhadap deadline
- Setiap Sprint menghasilkan fitur atau **produk yang memiliki nilai** dan menambahkan/memperbaiki fungsi sesuai kebutuhan user/product owner



Origin of Scrum



- Introduced “**Scrum**”
- Projects using **small, cross-functional** teams historically produce the best results
- **Takeuchi and Nonaka** relate these high-performance teams to “Scrum” **formations in Rugby**

野中 郁次郎 氏
一橋大学大学院
国際企業戦略研究科 名誉教授
株式会社富士通総研
経済研究所 理事長



- Flatter World in Software Engineering
- Introduced **Scrum for software development** in 1993

Jeff Sutherland 氏
The Scrum Training Institute
Chairman
Ph.D., Biometrics, Radiology,
Preventive Medicine

Waterfall vs Scrum

Rather than doing all of one thing at a time...

Waterfall (Defined)

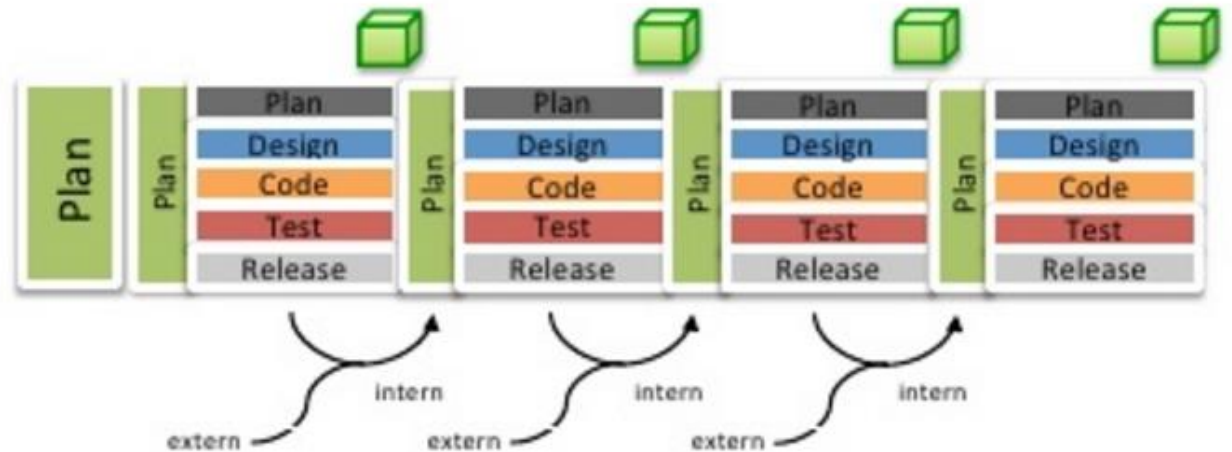
Plan for the entire project up-front



...Scrum teams do a little of everything all the time

Scrum (Empirical)

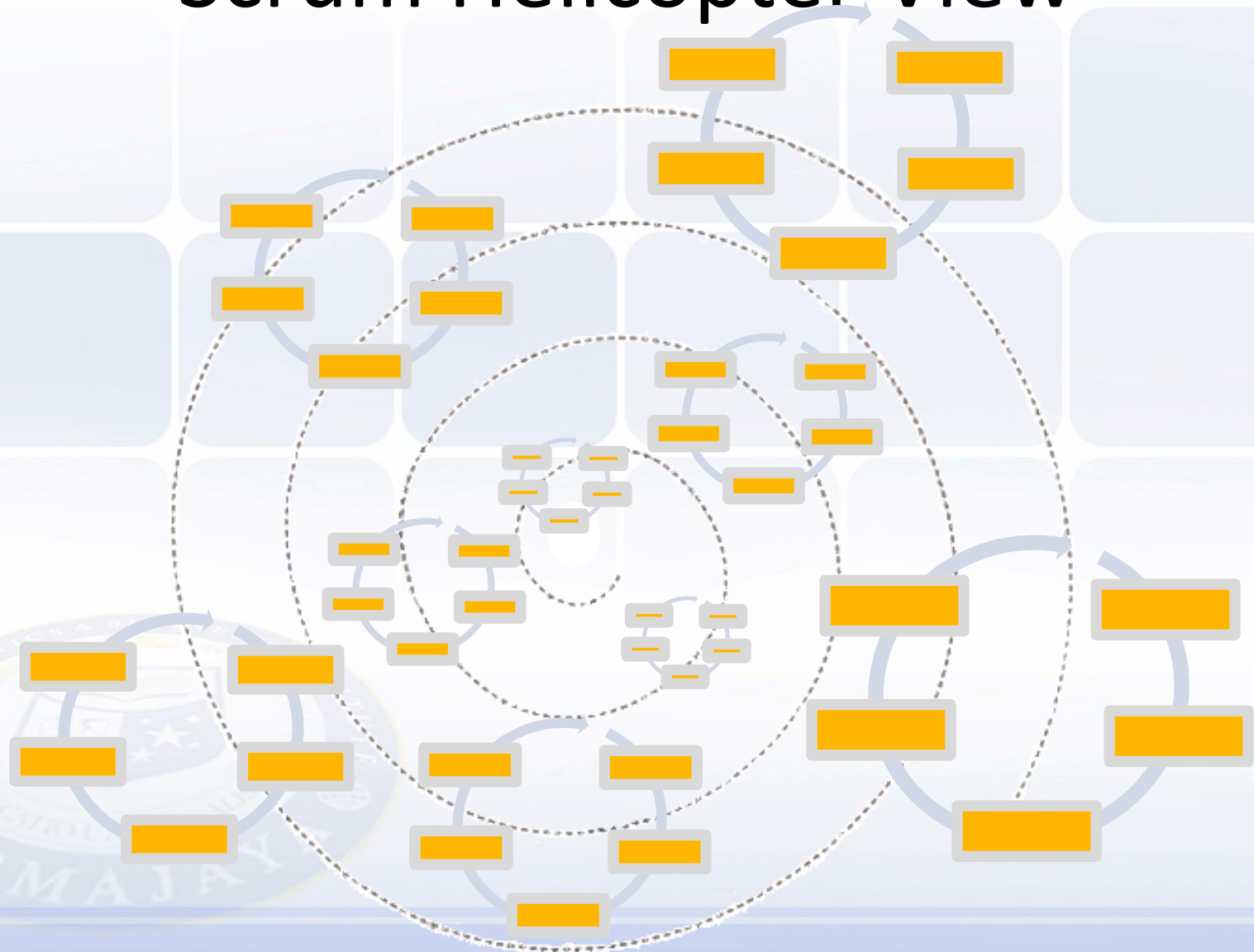
Plan a little for the entire project and then a little for each Sprint



(source: ADM)

“The New New Product Development Game” by Takeuchi and Nonaka. Harvard Business Review, January 1986.

Scrum Helicopter View



Product Backlog



Hasil Product Backlog Grooming (Sparx EA)

LOW IMPORTANT

Menambah Data Pelanggan

Deskripsi PBI:

[See more](#)

Melihat Data Pelanggan

Deskripsi PBI

[See more](#)

Mengubah Data Pelanggan

Deskripsi PBI

[See more](#)

Menghapus Data Pelanggan

Deskripsi PBI

[See more](#)

Mengeksport Laporan Penjualan Lagu

Deskripsi PBI:

[See more](#)

MEDIUM

Melakukan Pembelian Lagu

Deskripsi PBI:

[See more](#)

Mengubah Data Pribadi

Deskripsi PBI:

[See more](#)

Mengubah Data Lagu

Deskripsi PBI:

[See more](#)

Menghapus Data Lagu

Deskripsi PBI:

[See more](#)

Melakukan Logout

Deskripsi PBI:

[See more](#)

HIGH IMPORTANT

Melakukan Registrasi

Deskripsi PBI:

[See more](#)

Melakukan Login

Deskripsi PBI:

[See more](#)

Menambah Data Lagu

Deskripsi PBI:

[See more](#)

Melihat Daftar Lagu

Deskripsi PBI:

[See more](#)

Memutar Trial Lagu

Deskripsi PBI:

[See more](#)

Melakukan Registrasi

Deskripsi PBI:

As a <type of user> I want to <perform some task> so that I can <achieve some goal>

Sebagai Pengunjung, saya ingin dapat melakukan pendaftaran sebagai pelanggan agar saya dapat nantinya dapat melakukan pembelian lagu

Catatan PBI:

- Membutuhkan dokumentasi Sequence Diagram, BPMN, User Interface Design yang didalamnya terdapat field description, action control dan business rules
- Data yang disimpan adalah Nama Pelanggan, Alamat Email, Password, Jenis Kelamin, Tanggal Lahir, Negara Asal, dan Nomor Ponsel

Cara Demo dan Pengujian PBI:

1. Memilih tombol sign up di Menu Login
2. Mengisi Username dan Password Baru
3. Mengisi Data Profil Pribadi
4. Mengklik tombol save

[See more](#)

Melakukan Login

Deskripsi PBI:

As a <type of user> I want to <perform some task> so that I can <achieve some goal>

Sebagai Pengunjung dan Admin, saya ingin dapat melakukan login agar saya dapat masuk ke aplikasi MusicPedia dan bisa menggunakan fitur-fitur yang terdapat pada aplikasi tersebut

Catatan PBI:

- Membutuhkan dokumentasi Sequence Diagram, BPMN, User Interface Design yang didalamnya terdapat field description, action control dan business rules
- Jika gagal login 3x maka akan muncul notifikasi "Ingin mereset password?"
- Proses Login parameter yang dimasukan adalah Username dan Password

Cara Demo dan Pengujian PBI:

1. Memasukan Email dan Password
2. Apabila salah memasukan username & password maka menampilkan pesan kesalahan yaitu "Username & Password yang dimasukan salah"

[See more](#)

Menambah Data Lagu

Deskripsi PBI:

As a <type of user> I want to <perform some task> so that I can <achieve some goal>

Sebagai Admin, saya ingin dapat menambahkan data lagu agar daftar lagu bisa diakses dan diputar oleh pengguna

Catatan PBI:

- Membutuhkan dokumentasi Sequence Diagram, BPMN, User Interface Design yang didalamnya terdapat field description, action control dan business rules
- Data yang disimpan judul lagu, lirik lagu, suara dari lagu, pencipta, penyanyi, dan label musik dari lagu tersebut

Cara Demo dan Pengujian PBI:

1. Memilih Menu Pengelolaan Data Lagu
2. Memilih Tambah Data Lagu
3. Menambah Lagu
4. Konfirmasi Tambah

Hasil Sprint Planning (Sprint 1)

SPRINT GOAL

Berhasil digunakan oleh
Pengunjung untuk melakukan
Registrasi dan login ke aplikasi
Musik pedia, kemudian berhasil
Memutar trial lagu.

PBI Prioritas Pertama

PBI Prioritas Kedua

Sprint Goal

Product Backlog Item (PBI)	
Nama PBI	Poin PBI
Menambahkan Data lagu	
Deskripsi PBI	
Sebagai Admin, saya ingin dapat menambahkan data lagu agar daftar lagu baru bisa diakses dan diputar oleh pengunjung	
Catatan PBI	
1. Membutuhkan dokumentasi Sequence Diagram, BPMN, User Interface Design yang didalamnya terdapat field description, action control dan business rules	
Estimasi PBI (Hari)	
Cara Demo dan Pengujian PBI	
1. Memilih pengelolaan data lagu 2. Memilih tombol data lagu 3. Menambah lagu 4. Konfirmasi tambah	

Product Backlog Item (PBI)	
Nama PBI	Poin PBI
Melihat Daftar Lagu	
Deskripsi PBI	
Sebagai pelanggan, saya ingin dapat melihat daftar lagu agar saya dapat melihat lagu yang akan diputar	
Catatan PBI	
1. Membutuhkan dokumentasi Sequence Diagram, BPMN, User Interface Design yang didalamnya terdapat field Description, Action Control dan Business Rules 2. Data yang ditampilkan adalah Id Lagu, Judul Lagu, Id Penyanyi, Nama Album, Id Pencipta, Id Label Rekaman, Release Time, Img cover lagu dan Id Link	
Estimasi PBI (Hari)	
Cara Demo dan Pengujian PBI	
1. Memilih tombol mencari daftar lagu di menu utama 2. Memasukkan keyword	

Product Backlog Item (PBI)	
Nama PBI	Poin PBI
Memutar Trial Lagu	
Deskripsi PBI	
Sebagai pelanggan, saya ingin dapat memutar trial lagu agar saya dapat memutar lagu tanpa membayar dan mendengarkan sekilas lirik dan nadanya, serta menyediakan ruangan untuk saya melakukan pembelian terhadap lagu tersebut	
Catatan PBI	
1. Membutuhkan dokumentasi Sequence Diagram, BPMN, User Interface Design yang di dalamnya terdapat field description, action control dan business rules 2. Lagu yang diputar hanya ref pertama saya 3. Disetiap lagu bisa ditampilkan tombol untuk melakukan pembelian lagu	
Estimasi PBI (Hari)	
Cara Demo dan Pengujian PBI	
1. Memilih lagu yang akan dimainkan 2. Memilih tombol start audio	

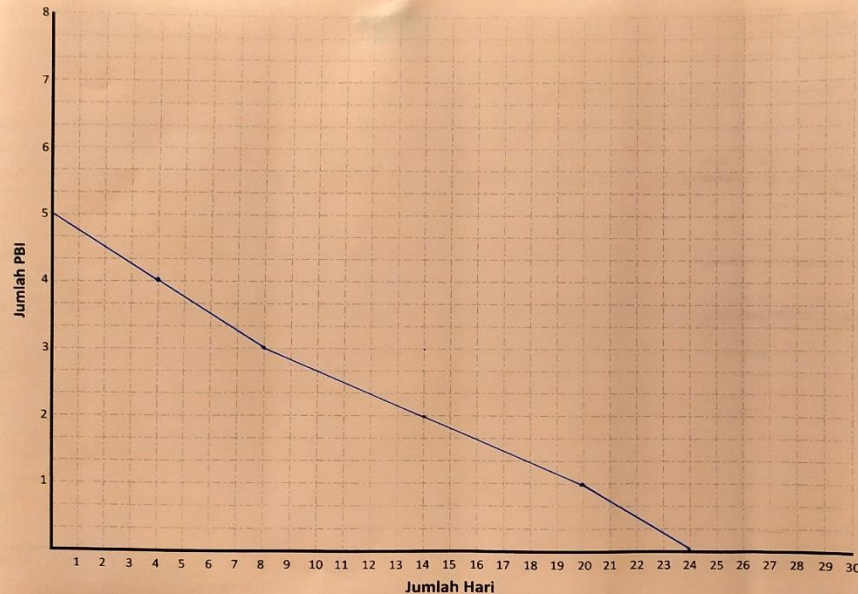
PBI Prioritas Ketiga

PBI Prioritas Keempat

PBI Prioritas Kelima

Hasil Sprint Planning (Sprint 1)

BURNDOWN CHART



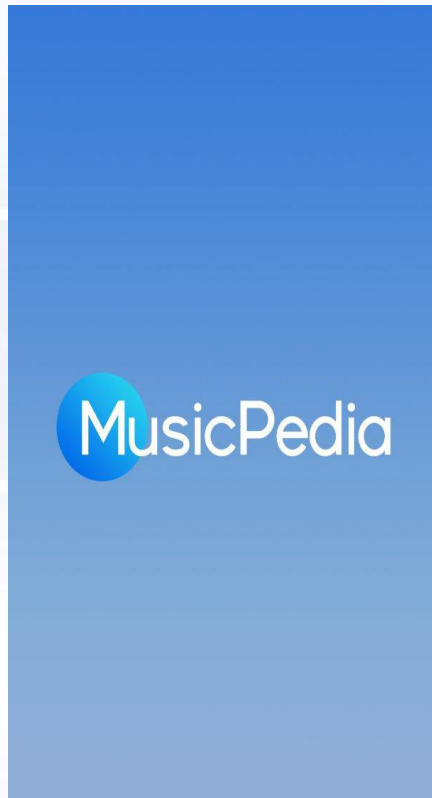
DEFINITION OF DONE

- ☐ Melakukan *review* terhadap dokumentasi Business Process Model and Notation (BPMN)
- ☐ Melakukan *review* terhadap dokumentasi Sequence Diagram
- ☐ Dibuatkan rancangan User Interface Design (UID), beserta penjelasan Field Description dan Business Rules
- ☐ Kode program diberikan komentar penjelasan pada alur logika atau pada setiap *function*
- ☐ Kode program mengikuti aturan standar dari *developer*
- ☐ Melakukan Unit Testing
- ☐ Melakukan Integration Testing
- ☐ Melakukan System Testing
- ☐ Melakukan User Acceptance Testing
- ☐ Melakukan *Go Live* di *Production Server*

Hasil Sprint Planning (Sprint 1)

BACKLOG	TODO	DOING	DONE
Melakukan Registrasi Story Point = 6 Deskripsi PBI: See more	Analysis - Business Process Model and Notation Analysis - Sequence Diagram Design - User Interface Design (UID) Design - Data Model Implementation - Program Code Proses Query Logic Registrasi Pengguna Implementation - Program Code Tampilan Halaman Registrasi Pengguna Implementation - Testing Plan Implementation - Unit Test & Refactor kode program Analysis - Business Process Model and Notation		Sprint Goal (Sprint 1) Berhasil Digunakan oleh Pengunjung untuk registrasi dan login ke aplikasi MusicPedia, kemudian berhasil memutar trial lagu Burndown Chart (Sprint 1) Definition of Done (Sprint 1) ☐ Melakukan review terhadap dokumentasi BPMN ☐ Melakukan review terhadap dokumentasi Sequence Diagram ☐ Dibuatkan Rancangan User Interface Design ☐ Kode program diberikan komentar penjelasan pada alur logika/pada setiap function ☐ Kode program mengikuti aturan standar dari developer Brainmatics ☐ Melakukan Unit Testing ☐ Melakukan Integration Testing ☐ Melakukan System Testing ☐ Melakukan User Acceptance Testing ☐ Melakukan Go Live di Production Server Checklist = <memo>
Melakukan Login Story Point = 6 Deskripsi PBI: See more			
Menambah Data Lagu Story Point = 8 Deskripsi PBI: See more			
Melihat Daftar Lagu Story Point = 8 Deskripsi PBI: See more			
Memutar Trial Lagu Story Point = 6			

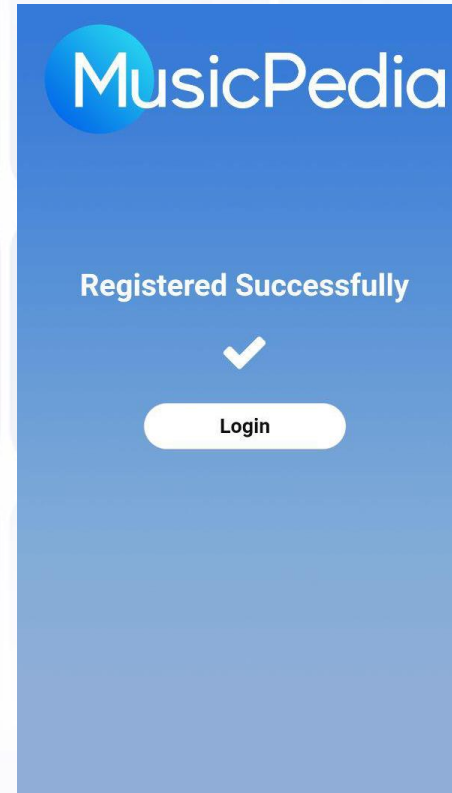
Hasil Sprint Execution (Sprint 1)



Splash Screen

The registration form is titled 'Register Your Account' in white text. It includes input fields for 'Username', 'Email', 'Password', and 'Confirm Password'. Below these are three dropdown menus for 'Day', 'Month', and 'Years'. At the bottom, there are radio buttons for 'Pria' (Male) and 'Wanita' (Female), followed by a 'Register' button.

Halaman
Registrasi



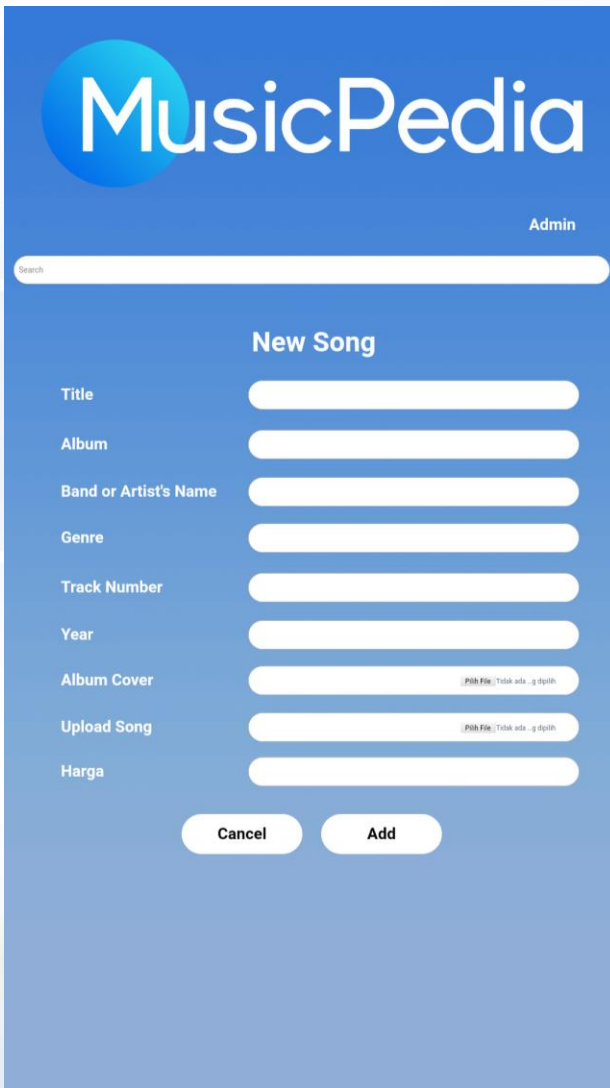
Notifikasi Berhasil
Registrasi

The login form features the MusicPedia logo at the top. It includes input fields for 'username' and 'password', a 'Remember Me' checkbox, and a 'Login' button. At the bottom, there are links for 'Problem when Login? [help here](#)' and 'Don't Have an Account? [Sign Up](#)'. The background is a solid blue color.

Halaman Login

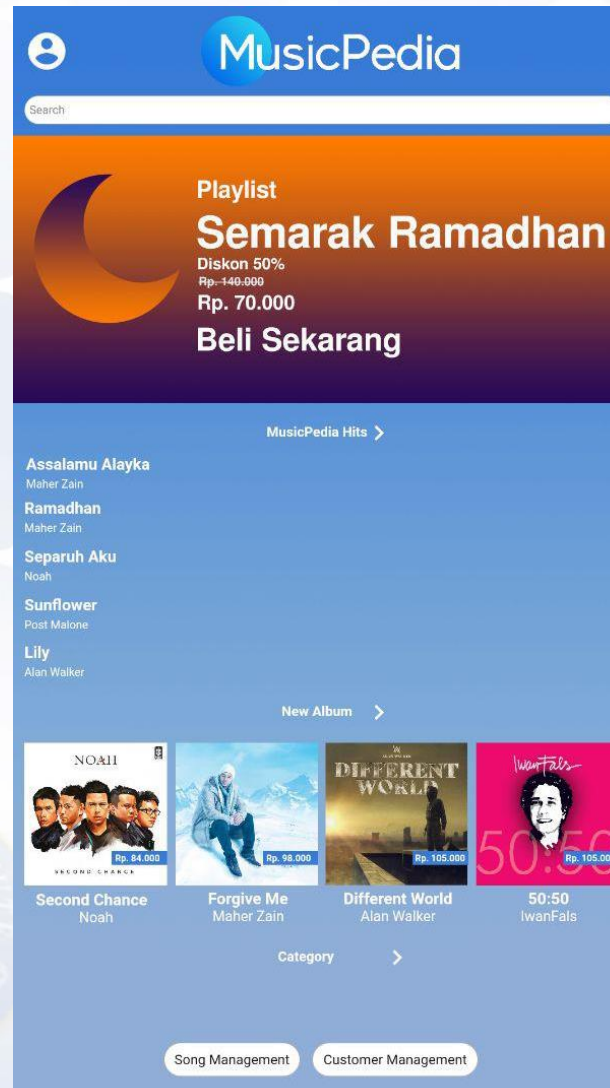
Sprint 1 Goal: Pengunjung dapat melakukan Registrasi, login, dan memutar trial lagu

Hasil Sprint Execution (Sprint 1)

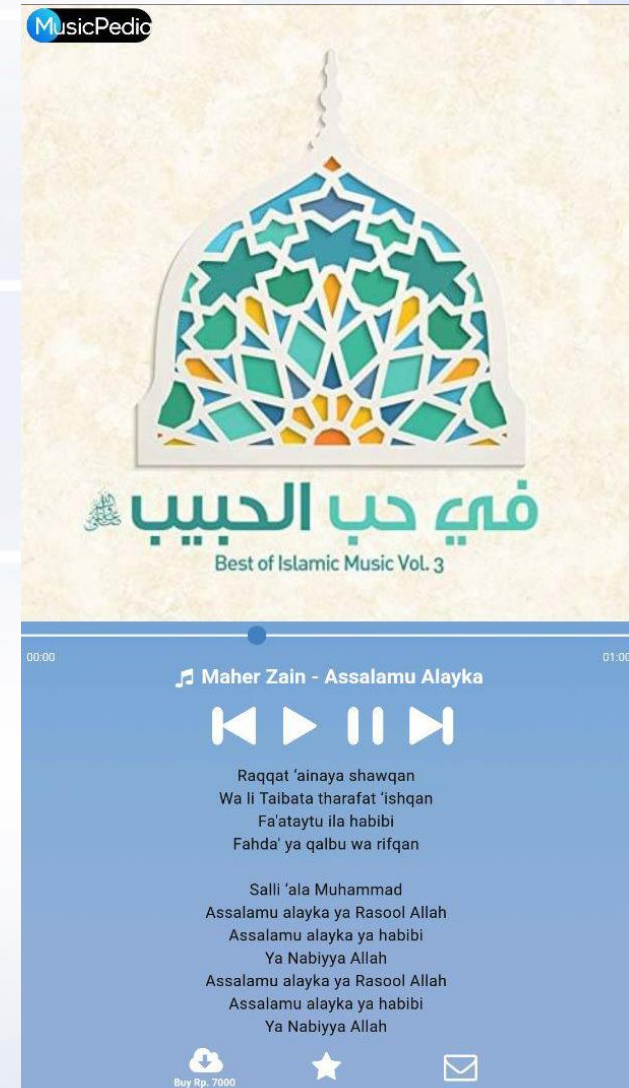


The screenshot shows the 'New Song' form in the MusicPedia app. The form has a blue header with the 'MusicPedia' logo and an 'Admin' link. Below the header is a search bar. The form fields are: Title, Album, Band or Artist's Name, Genre, Track Number, Year, Album Cover (with a file upload icon), Upload Song (with a file upload icon), and Harga. At the bottom are 'Cancel' and 'Add' buttons.

Halaman
Penambahan Data
Lagu



Halaman Melihat Data Lagu



Memutar Trial Lagu

XP vs Scrum vs Lean

- **XP** deals with **how to work** with programming
- **Scrum** deals with **how the project is organized** and planned
- **Lean Development** deals with which comprehensive principles should **apply for the entire development organization**

Strategi Pemilihan Metodologi - Faktor

1. **Clarity** of User **Requirements**
2. Familiarity with **Technology**
3. System **Complexity**
4. System **Reliability**
5. Short **Time Schedules**
6. Schedule **Visibility**

Strategi Pemilihan Metodologi - Matriks

Ability to Develop Systems	Structured Methodologies		RAD Methodologies			Agile Methodologies	
	Waterfall	Parallel	Phased	Prototyping	Throwaway Prototyping	XP	SCRUM
With Unclear User Requirements	Poor	Poor	Good	Excellent	Excellent	Excellent	Excellent
With Unfamiliar Technology	Poor	Poor	Good	Poor	Excellent	Good	Good
That Are Complex	Good	Good	Good	Poor	Poor	Good	Good
That Are Reliable	Good	Good	Good	Poor	Poor	Excellent	Excellent
With a Short Time Schedule	Poor	Good	Excellent	Excellent	Good	Excellent	Excellent
With Schedule Visibility	Poor	Poor	Excellent	Excellent	Good	Excellent	Excellent

Who Is The WIN ??????

Pemilihan Metodologi: Sistem SDM

- Seandainya, anda adalah seorang software engineer di perusahaan PT BlackSoft, sebuah perusahaan IT yang memiliki **kantor cabang di berbagai tempat di dunia**
- Divisi Sumber Daya Manusia pada PT BlackSoft ingin membangun sebuah sistem yang bisa **merekam, mengubah, menghapus dan menampilkan data pegawai** yang dimiliki, baik itu lokasi saat ini, latar belakang pendidikan, jadwal pekerjaan dan pengalaman kerja yang dimiliki
- PT BlackSoft memiliki jaringan internasional dimana kantor cabang di berbagai negara menggunakan **hardware dan software yang berbeda**
- Divisi Sumber Daya Manusia berkantor di Jakarta, sedangkan Divisi Teknologi Informasi yang mendapatkan tugas mengembangkan software berkantor di Denpasar, pertemuan untuk requirement gathering harus dischedulekan sejak lama
- Manajemen ingin agar sistem dapat selesai dikerjakan dan mulai bisa **berjalan dalam satu tahun**

Pemilihan Metodologi: Sistem DSS

- Seandainya, anda adalah seorang software engineer di perusahaan PT BlackSoft, sebuah perusahaan IT yang memiliki **kantor cabang di berbagai tempat di dunia**
- Divisi Business Development pada PT BlackSoft ingin membangun sebuah sistem Decision Support System (DSS). DSS akan **mengolah berbagai data perusahaan**, kemudian **mengubahnya menjadi pola-pola pengetahuan**, yang akhirnya bisa digunakan untuk pengambilan keputusan manajemen
- Divisi Business Development paham betapa pentingnya sistem ini, karena itu akan **all out dan bersedia masuk ke tim pengembangan**
- Divisi Business Development **belum bisa menjelaskan secara pasti software yang dibangun** ini akan seperti apa. Oleh karena itu dibutuhkan **diskusi intensif dan mendalam**, serta bila perlu ada pertemuan harian, sehingga software bisa dikembangkan sesuai kebutuhan
- Divisi Business Development berharap bahwa sistem dapat selesai dikerjakan dan mulai bisa **berjalan dalam enam bulan**. Release pertama harus dalam 1 bulan. Untuk itu monitoring progress akan dilakukan secara harian.

Rangkuman

- **Systems Development Lifecycle:**
Planning, Analysis, Design, and Implementation
- **Systems Development methodologies:**
 1. **Structured Design**
 - **Waterfall** method
 - Parallel development
 2. **Rapid Application Development**
 - **Phased** Development
 - **Prototyping**
 3. **Agile Development**
 - Extreme Programming
 - **Scrum**

3.2 Requirement Gathering



Metode Requirement Gathering dan Tantangan



Boehm's First Law

Errors are **most frequent during the requirements and design** activities, and are the more expensive the later they are removed

(Endres, 2003)

[L2]



What is a Requirements

- Business Requirements
 - Statement of what the system must do
 - Focus on what the system must do, not how to do it
- There are 2 kinds of requirements

1. Functional Requirements

2. Nonfunctional Requirements

D. Nonfunctional Requirements

1. Operational Requirements

- 1.1. The system will operate in Windows and Macintosh environments
- 1.2. The system will be able to read and write Word documents, RTF, and HTML
- 1.3. The system will be able to import Gif, Jpeg, and BMP graphics files

2. Performance Requirements

- 2.1. Response times must be less than 7 seconds
- 2.2. The Inventory database must be updated in real time

3. Security Requirements

- 3.1. No special security requirements are anticipated

4. Cultural and Political Requirements

- 4.1. No special cultural and political requirements are anticipated

C. Functional Requirements

1. Printing

- 1.1. The user can select which pages to print
- 1.2. The user can view a preview of the pages before printing
- 1.3. The user can change the margins, paper size (e.g., letter, A4) and orientation on the page

2. Spell Checking

- 2.1. The user can check for spelling mistakes; the system can operate in one of two modes as selected by the users
 - 2.1.1. Mode 1 (Manual): The user will activate the spell checker and it will move the user to the next misspelled word
 - 2.1.2. Mode 2 (Automatic): As the user types, the spell checker will flag misspelled words so the user immediately see the misspelling
- 2.2. The user can add words to the dictionary
- 2.3. The user can mark words as not misspelled but not add them to the dictionary

1. Functional Requirement

- Kebutuhan tentang **fungsi software** secara menyeluruh
- Pemodelan dengan **UML**, ataupun penjelasan fitur-fitur dalam bentuk **problem statements**, adalah termasuk dalam **Functional Requirement**
 - **Diagrams:**
 - **Use Case** Diagrams
 - **Activity** Diagrams
 - **Problem Statements:**
 - Must **search** for inventory
 - Must **perform** these calculations
 - Must **produce** a specific report

2. Nonfunctional Requirements

1. **Operational** – Physical/technical environment
2. **Performance** – Speed and reliability
3. **Security** – Who can use the system
4. **Cultural & Political** – Company policies, legal issues

Nonfunctional Requirements

Jaringan Dokumentasi dan Informasi Hukum (JDIH)

Date: 23 Oktober 2018

Operational Requirements

1. Sistem dapat digunakan oleh semua perangkat lunak yang lainnya.
2. Sistem layanan harus memiliki tingkat ketersediaan 99%.
3. System harus go live sebelum tanggal 25 Januari 2018.

Performance Requirements

1. Waktu respon harus tidak lebih dari 2 detik.
2. Mampu diakses oleh lebih dari 50 pengguna secara bersamaan.
3. Server harus bersih dari virus.

Security Requirements

1. Akses ke dalam system tanpa otorisasi tidak dimungkinkan.
2. Data hanya dapat diubah oleh administrator sistem.
3. Seluruh data harus di-backup setiap 24jam serta hasil backup-nya disimpan di lokasi yang berbeda dengan sistem.
4. Seluruh komunikasi antara client-server harus dipastikan keamanannya.

Deadline Requirements

1. Pengembangan aplikasi tidak boleh melewati jangka waktu yang sudah ditentukan.

Metode Requirement Gathering

1. Document Analysis
2. Interviews
3. Joint Application Design (JAD)
4. Questionnaires
5. Observation



1. Document Analysis

- Provides clues about the "formal" existing **As-Is system**
- Typical **documents**
 - Forms
 - Reports
 - Policy manuals
- Look for user additions to forms
- Look for **unused form** elements
- Do document analysis before interviews

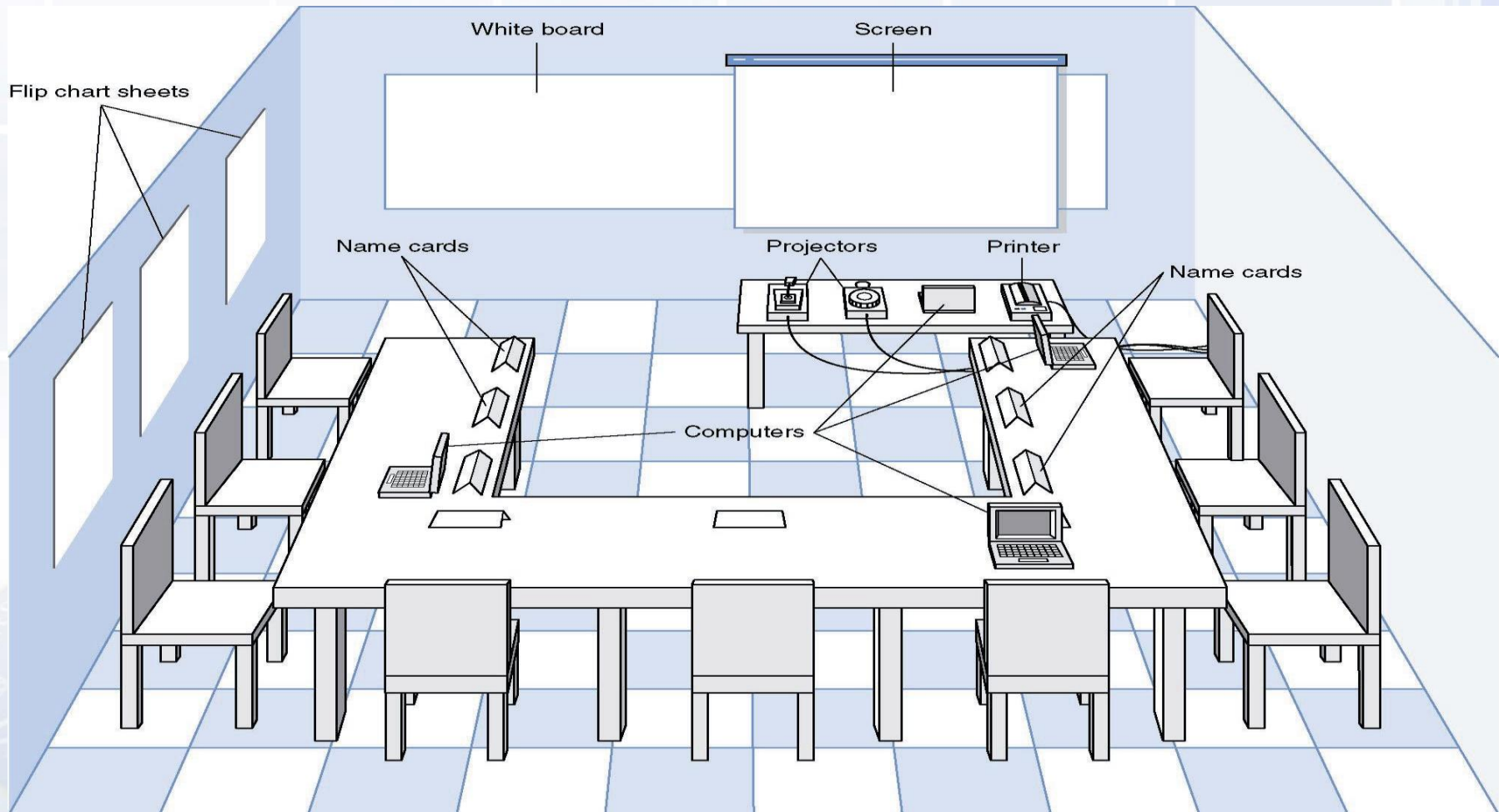
2. Interviews

- Most **commonly used technique** and very natural
- If you need to know something, you **ask someone**
- Five basic **steps**:
 1. Selecting interviewees
 2. Designing interview questions
 3. Preparing for the interview
 4. Conducting the interview
 5. Post-interview follow-up

3. Joint Application Design (JAD)

- Allows project managers, users, and developers to work together
- May reduce scope creep by 50%
- Avoids requirements being too specific or too vague
- Include 10 to 20 users
- Tend to last 5 to 10 days over a three-week period

JAD Meeting Room



4. Questionnaire

- **Selecting** participants
 - Using **samples** of the population
- **Designing the questionnaire**
 - More important than interview questions
 - Prioritize questions to grab attention
 - Distinguish between:
 1. **Fact-oriented questions** (specific answers)
 2. **Opinion questions** (agree – disagree scale)
- **Administering** the questionnaire
 - Explain its **importance** & **how it will be used**
 - Give **expected response date**
 - **Follow up** on late returns and have **supervisors** follow up
 - **Promise to report** results
- **Questionnaire follow-up**
 - Send results to participants

5. Observation

- Users/managers often **don't remember** everything they do
- **Validates info gathered** in other ways
- **Behaviors change** when people are **watched**
- **Keep low profile**, don't change the process
- Careful **not to ignore** periodic activities
 - Weekly ... Monthly ... Annual

Karakteristik Metode Requirement Gathering

	Interviews	JAD	Questionnaires	Document Analysis	Observation
Type of Information	As-Is Improve. To-Be	As-Is Improve. To-Be	As-Is Improve.	As-Is	As-Is
Depth of Information	High	High	Medium	Low	Low
Breadth of Information	Low	Medium	High	High	Low
Integration of Info.	Low	High	Low	Low	Low
User Involvement	Medium	High	Low	Low	Low
Cost	Medium	Low-Medium	Low	Low	Low-Medium

Business Process Analysis Strategies

1. BPA (Business Process Automation)
 - Makes almost no changes to business processes, just makes them more efficient
2. BPI (Business Process Improvement)
 - Change what the users do, not just how efficiently they do it
3. BPR (Business Process Reengineering)
 - Throw away everything, start with a blank page

Business Process Analysis Strategies

Comparison

	Business Process Automation	Business Process Improvement	Business Process Reengineering
Potential Business Value	Low-Moderate	Moderate	High
Project Cost	Low	Low-Moderate	High
Breadth of Analysis	Narrow	Narrow-Moderate	Very Broad
Risk	Low	Low-Moderate	Very High

Tantangan di Requirement Gathering

1. The "Yes, But" syndrome

- Stems from **human nature** and the **users' inability** to experience the software as they might a physical device

2. The "Undiscovered Ruins"

- Searching for requirements is like **searching** for "Undiscovered Ruins"
- The **more you find**, the **more you know** remain

3. The "User and the Developer" syndrome

- Reflects the profound differences between these two, making **communication difficult**

The "Yes, But" Syndrome

For whatever reason, we always see **two** immediate, distinct, and separate **reactions** when the users see the system implementation for the first time

1. "**Wow, this is so cool**; we can really use this, what a neat job" and so on.
2. "Yes, **but, hmmm**, now that I see it, what about this ... ? Wouldn't it be nice if ... ? Whatever happened to ... ?"

The "Undiscovered Ruins" Syndrome

- In many ways, the search for requirements is **like a search for undiscovered ruins**
 - **The more you find, the more you know remain**
 - You never really feel that you have found them all, and perhaps you never will.
- Indeed, software development teams always **struggle to determine when they are done** with requirements elicitation. When have they found
 - **all** the requirements
 - or **at least enough** requirements?

The "User and the Developer" Syndrome

- **Communication gap** between the user and the developer
- **Users and developers** are
 - typically from different worlds,
 - may even speak different languages,
 - have different backgrounds,
 - have different motivations, and
 - have different objectives

The "User and the Developer" Syndrome

Reasons for this problem and some suggested solutions

Problem	Solution
Users do not know what they want, or they know what they want but cannot articulate it.	Recognize and appreciate the user as domain expert; try alternative communication and elicitation techniques.
Users think they know what they want until developers give them what they said they wanted.	Provide alternative elicitation techniques earlier: storyboarding, role playing, throwaway prototypes, and so on.
Analysts think they understand user problems better than users do.	Put the analyst in the user's place. Try role playing for an hour or a day.
Everybody believes everybody else is politically motivated.	Yes, its part of human nature, so let's get on with the program.

Paradigma Analysis dan Design



Evolusi Paradigma Analysis dan Design

	Paradigm	Diagrams
1	Process-oriented Paradigm	Flowchart
2	Data-oriented Paradigm	DFD
3	Object-oriented Paradigm (data + process)	UML



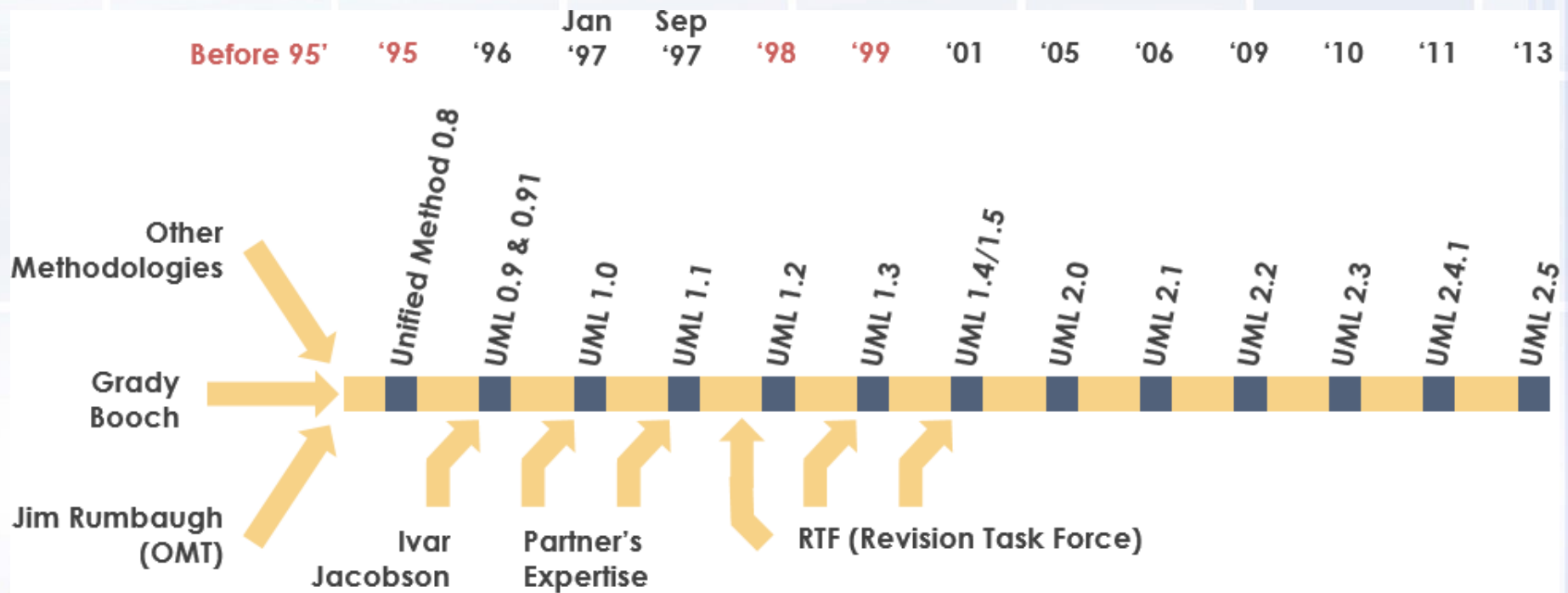
What is the UML?

- UML: **Unified Modeling Language**
- UML can be used for **modeling all processes in the development life cycle** and across different implementation technologies (technology and language independent)
- UML is the **standard language** for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
- UML is a **communication tool** – for the team, and other stakeholders

Sejarah UML



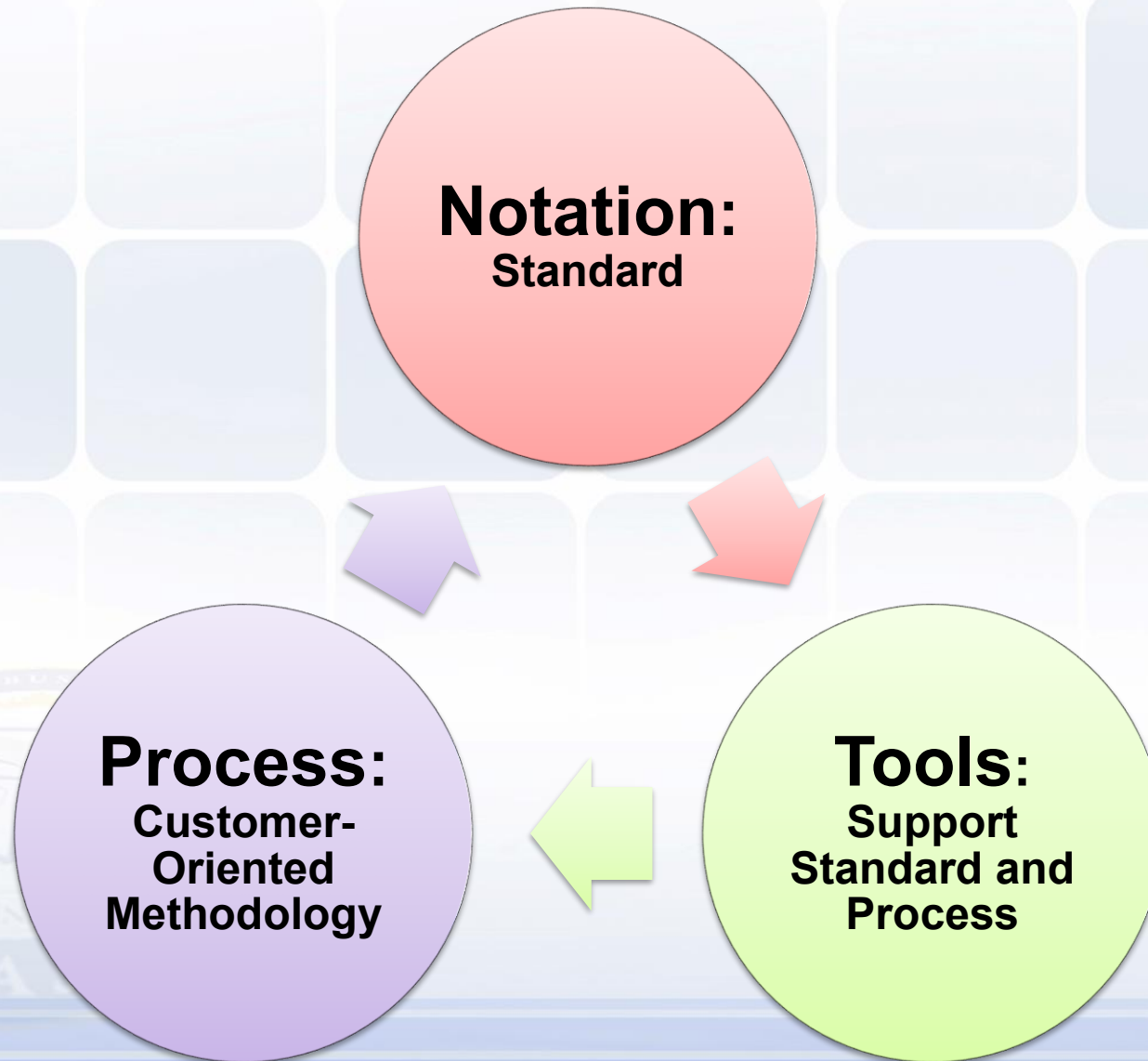
Booch, Jacobson, Rumbaugh



Before 95' - Fragmentation ► 95' - Unification ► 98' - Standardization ► 99' - Industrialization

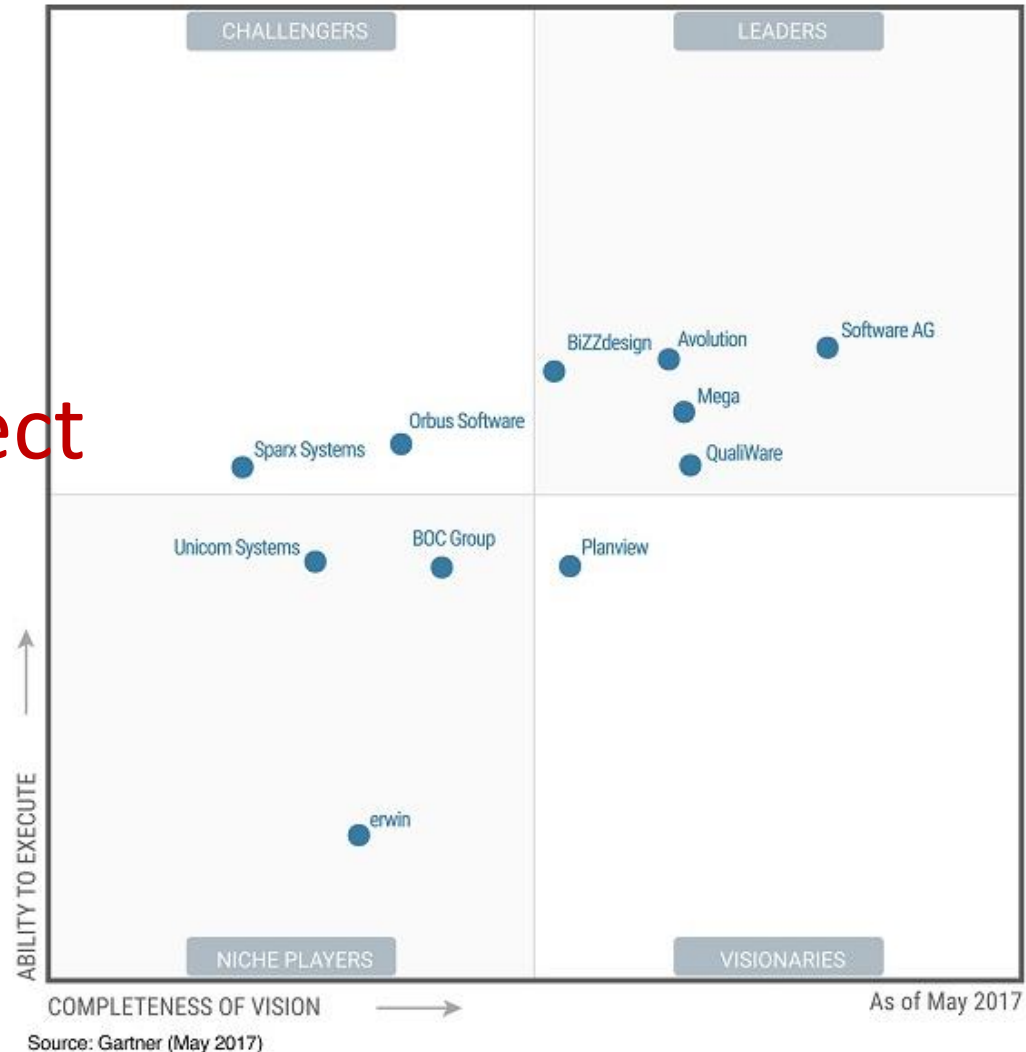
Segitiga Sukses Pengembangan Software

(Rumbaugh, 1999)



State of the Art UML Tools

- Rational Rose
- Visual Paradigm
- Sparx Systems Enterprise Architect
- Microsoft Visio
- Star UML



UML 2.0 Diagrams

Diagram Name	Used to	Primary Phase
Structure Diagrams		
Class	Illustrate the relationships between classes modeled in the system.	Analysis, Design
Object	Illustrate the relationships between objects modeled in the system. Used when actual instances of the classes will better communicate the model.	Analysis, Design
Package	Group other UML elements together to form higher level constructs.	Analysis, Design, Implementation
Deployment	Show the physical architecture of the system. Can also be used to show software components being deployed onto the physical architecture.	Physical Design, Implementation
Component	illustrate the physical relationships among the software components.	Physical Design, Implementation
Composite Structure	Illustrate the internal structure of a class, i.e., the relationships among the parts of a class.	Analysis, Design
Behavioral Diagrams		
Activity	Illustrate business workflows independent of classes, the flow of activities in a use case, or detailed design of a method.	Analysis, Design
Sequence	Model the behavior of objects within a use case. Focuses on the time-based ordering of an activity.	Analysis, Design
Communication	Model the behavior of objects within a use case. Focuses on the communication among a set of collaborating objects of an activity.	Analysis, Design
Interaction Overview	Illustrate an overview of the flow of control of a process.	Analysis, Design
Timing	Illustrate the interaction that takes place among a set of objects and the state changes in which they go through along a time axis.	Analysis, Design
Behavioral State Machine	Examine the behavior of one class.	Analysis, Design
Protocol State Machine	Illustrates the dependencies among the different interfaces of a class.	Analysis, Design
Use-Case	Capture business requirements for the system and to illustrate the interaction between the system and its environment.	Analysis

UML Problems

1. UML is modeling notation, it is **not a development process** or a methodology
 - UML driven development process?
2. UML is **too complex, difficult to understand quickly**
 - Which UML diagrams should we use?

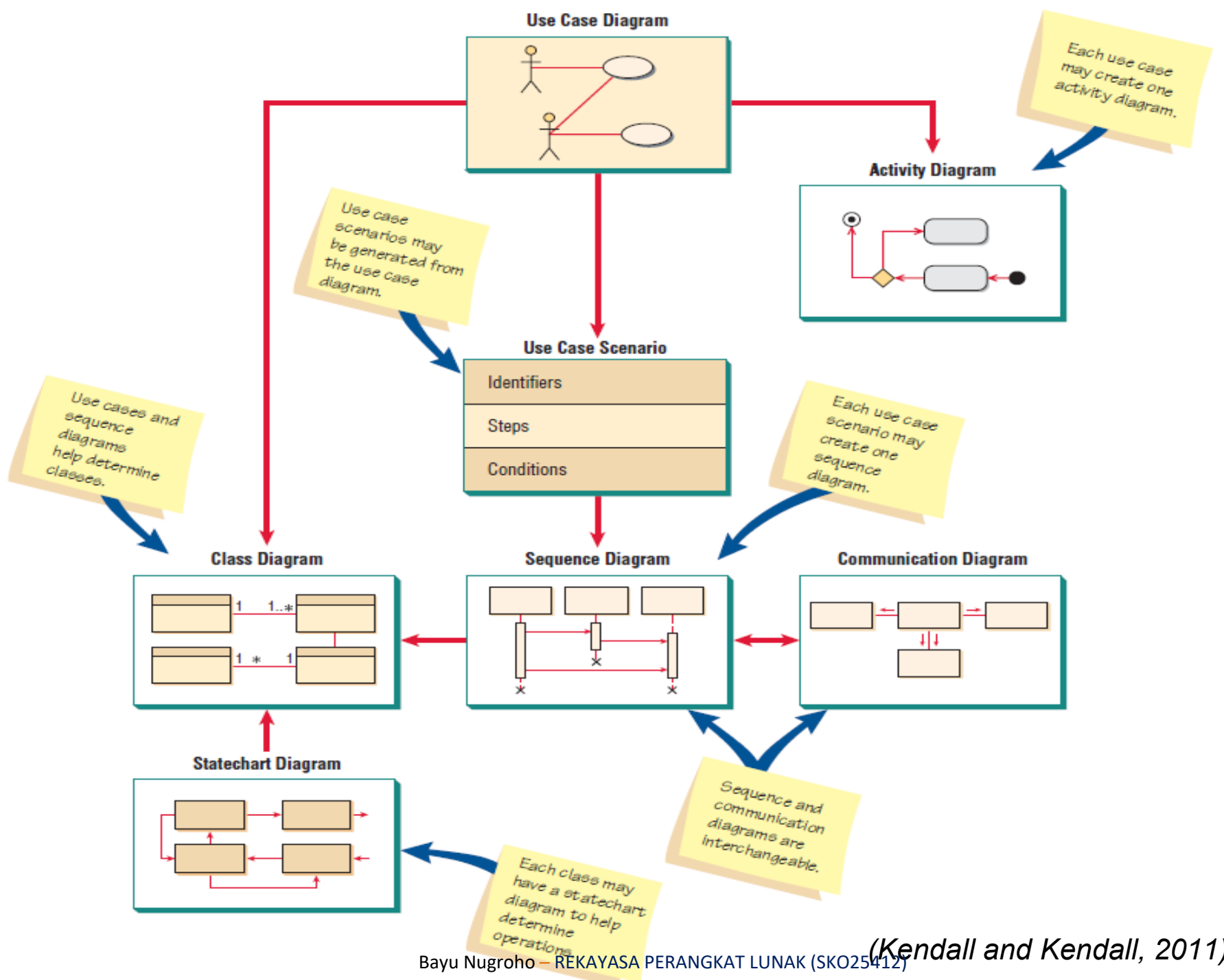
UML based Software Analysis and Design

(Sparx Systems EA)

1. Display the boundary of a system and its major functions using **use cases and actors**
2. Model the organization's business process with **activity diagram**
3. Illustrate use case realizations with **sequence diagrams**
4. Represent a static structure of a system using **class diagrams**
5. Reveal the physical implementation architecture with **deployment diagrams**

UML based Software Analysis and Design (*Kendal, 2011*)

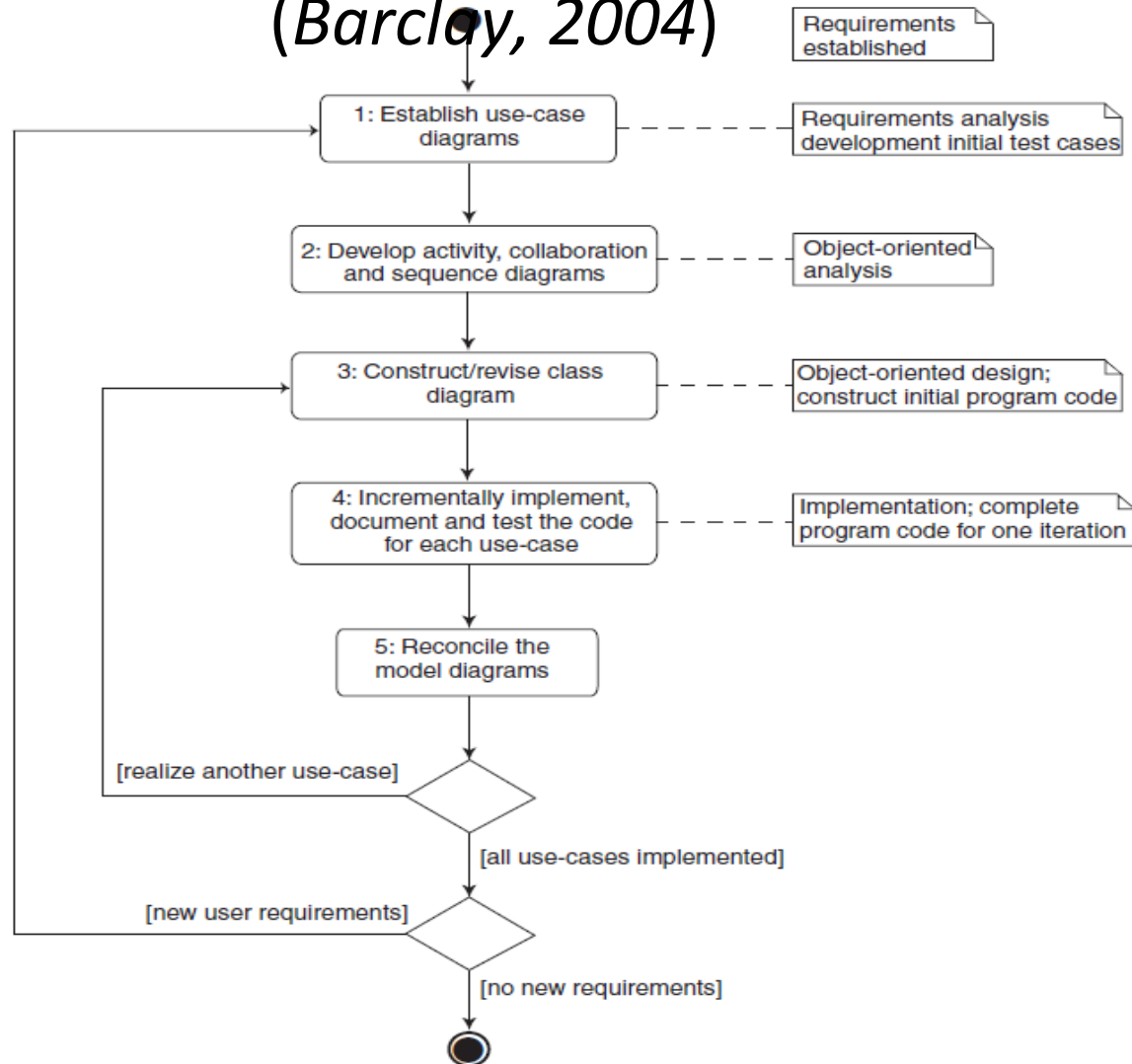
1. A **use case diagram**, describing how the system is used. Analysts start with a use case diagram
2. An **activity diagram**, illustrating the overall flow of activities. Each use case may create one activity diagram
3. **Sequence diagrams**, showing the sequence of activities and class relationships. Each use case may create one or more sequence diagrams
4. **Class diagrams**, showing the classes and relationships. Sequence diagrams are used to determine classes
5. **Statechart diagrams**, showing the state transitions. Each class may create a statechart diagram, which is useful for determining class methods



(Kendall and Kendall, 2011)

UML based Software Analysis and Design

(Barclay, 2004)



Identifikasi Proses Bisnis dengan Use Case Diagram



UML based Software Analysis and Design

1. Systems Analysis

1.1 Identifikasi Proses Bisnis dengan **Use Case Diagram**

1.2 Pemodelan Proses Bisnis dengan **Activity Diagram** atau **BPMN**

1.3 Realisasi Proses Bisnis dengan **Sequence Diagram**

(**Boundary** - **Control** - **Entity**)

2. Systems Design

2.1 Pemodelan **Class Diagram**

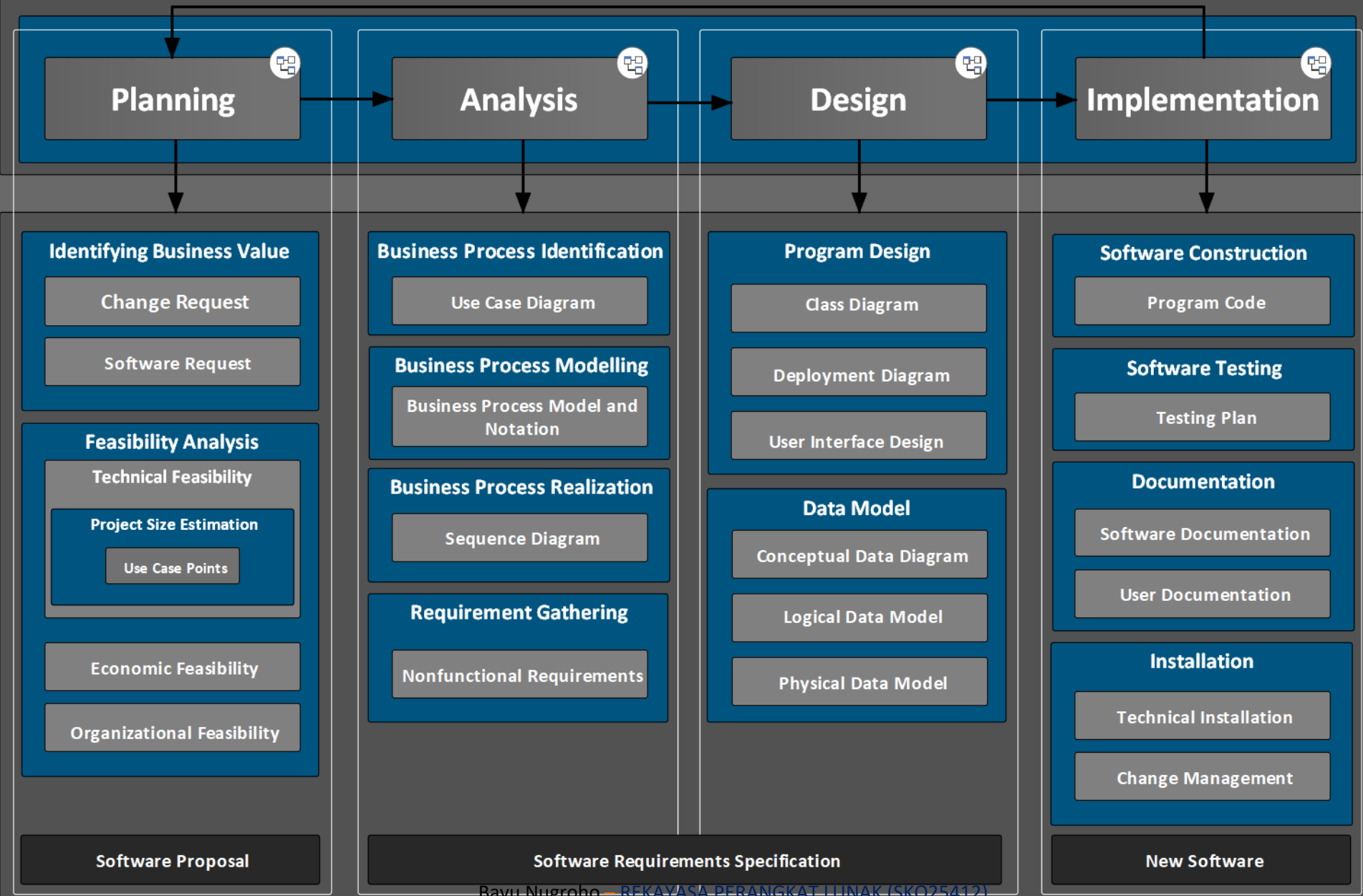
2.2 Pemodelan **User Interface Design**

2.3 Pemodelan **Data Model**

2.4 Pemodelan **Deployment Diagram**

Application Development Governance

Software Development Life Cycle

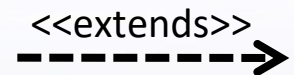
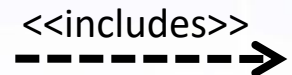
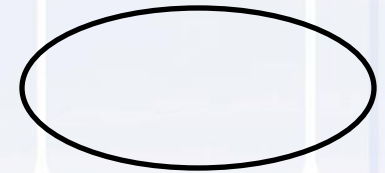
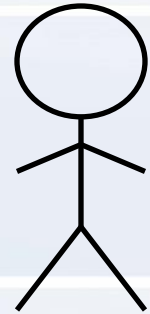


Use Case Diagram

- Summarized into a **single picture**
- All of the use cases for the part of the system being modeled
- Use case represents the discrete **activities performed by the user**
- Use Case Diagram tells **what the system will do**
- Good for **communicating with users**

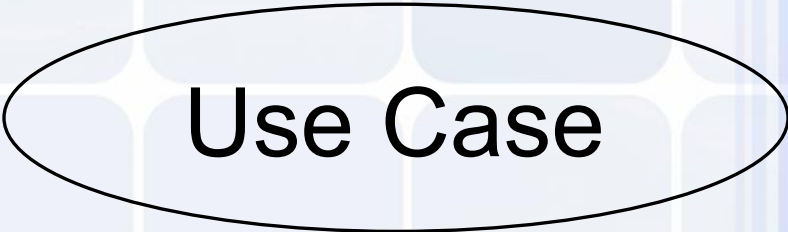
Use Case Diagram Syntax

- **Actor**
 - person or system that derives benefit from and is external to the subject
- **Use Case**
 - Represents a major piece of system functionality
- **Association Relationship**
- **Include Relationship**
- **Extend Relationship**
- **Generalization Relationship**



Use Case

- A major piece of **system functionality**
- Can **extend** other Use Cases
- Placed inside system boundary
- Labeled with descriptive **verb - noun phrase**



Use Case

System Boundary

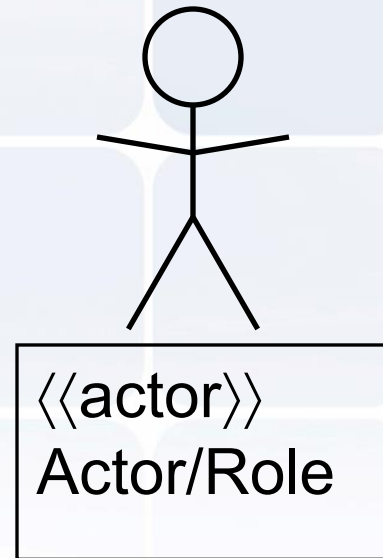
- Includes the **name of the system** inside or on top
- Represents the **scope of the system**
- **Actors are outside** the scope of the system

Boundary



Actor

- A **person** or **another system** that interacts with the current system
- A **role**, not a specific user
- **Provides input, receives output, or both**



Association Relationship

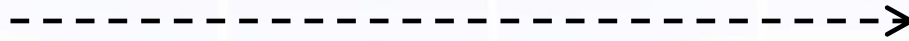
- **Links** actor and the Use Case
- Shows **two-way communication**
 - If one-way, arrows are used



Extends Relationship

- **Extends** Use Case to include **Optional** behavior
- **Arrow points** from the extension Use Case **to** the base Use Case

<<extend>>



Make Payment
Arrangement

<<extend>>

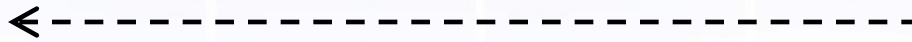


Make
Appointment

Include Relationship

- **Include** one Use Case from **within** another
- **Arrow points** from base Use Case **to the included Use Case**

«include»



Make New
Patient Appointment

«include»

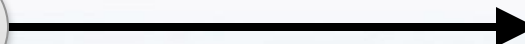
Create New
Patient

Generalization Relationship

- A specialized Use Case to a more generalized Use Case
- **Arrow points** from specialized **to** general Use Case

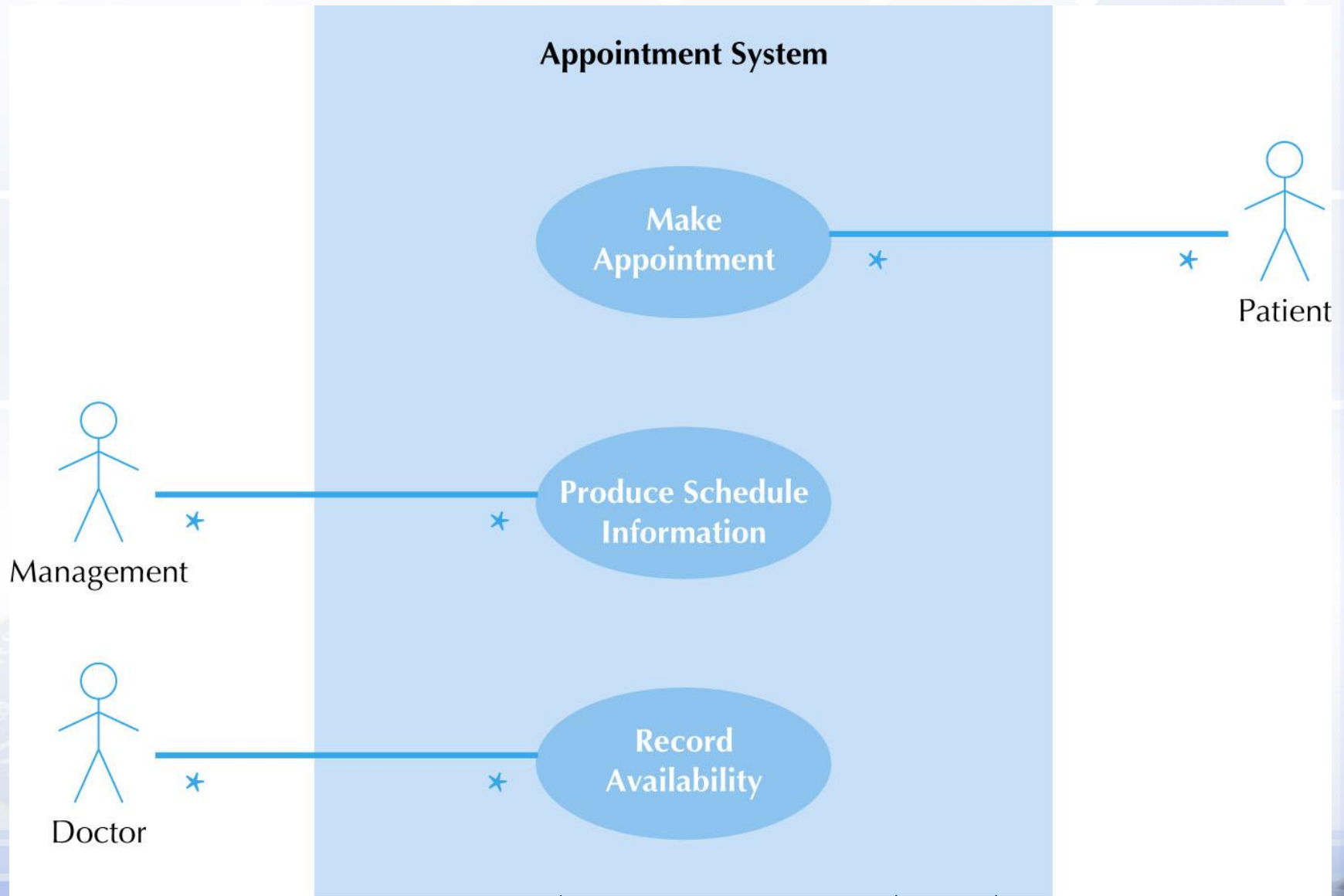


Make Old
Appointment

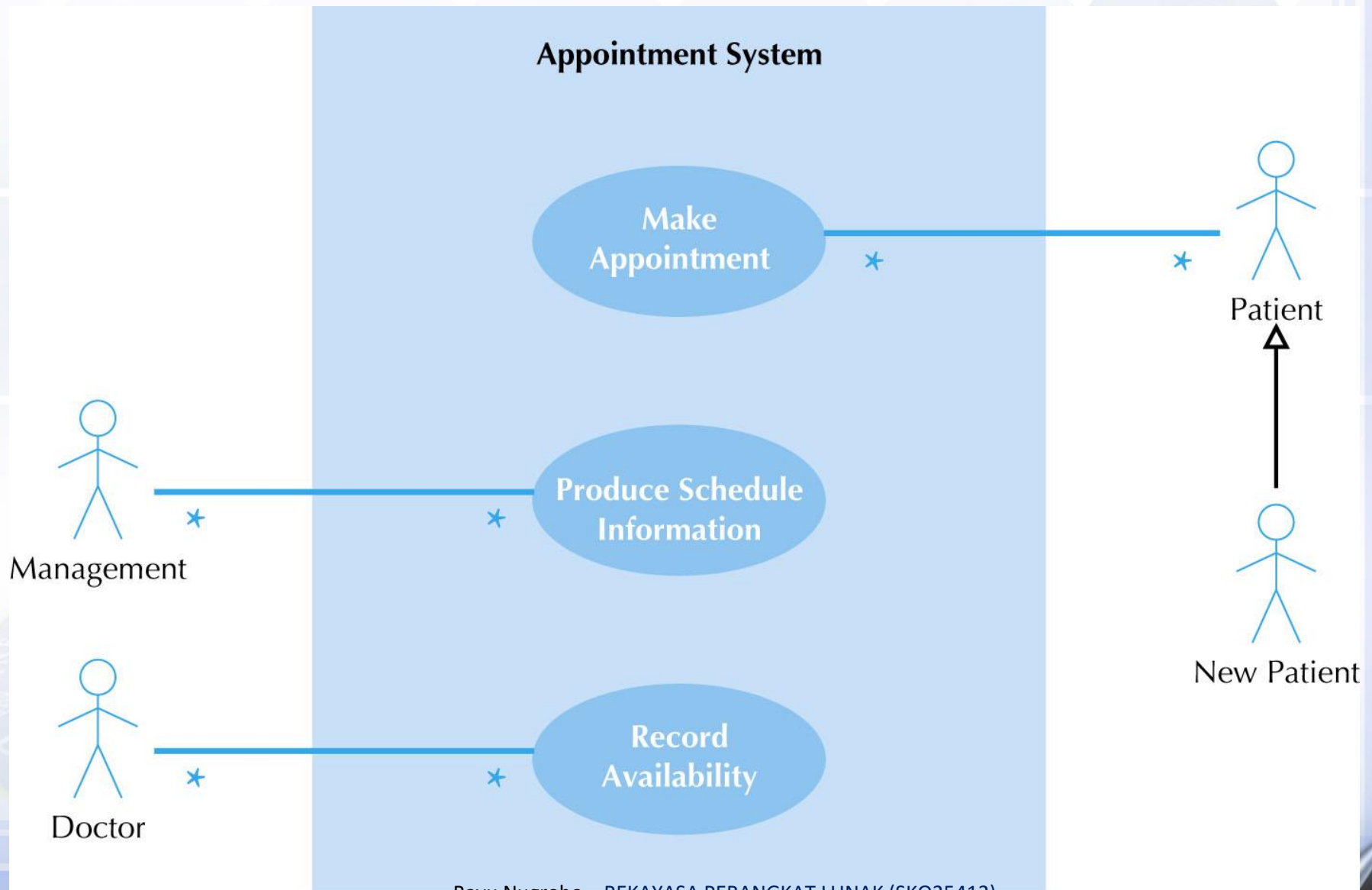


Make
Appointment

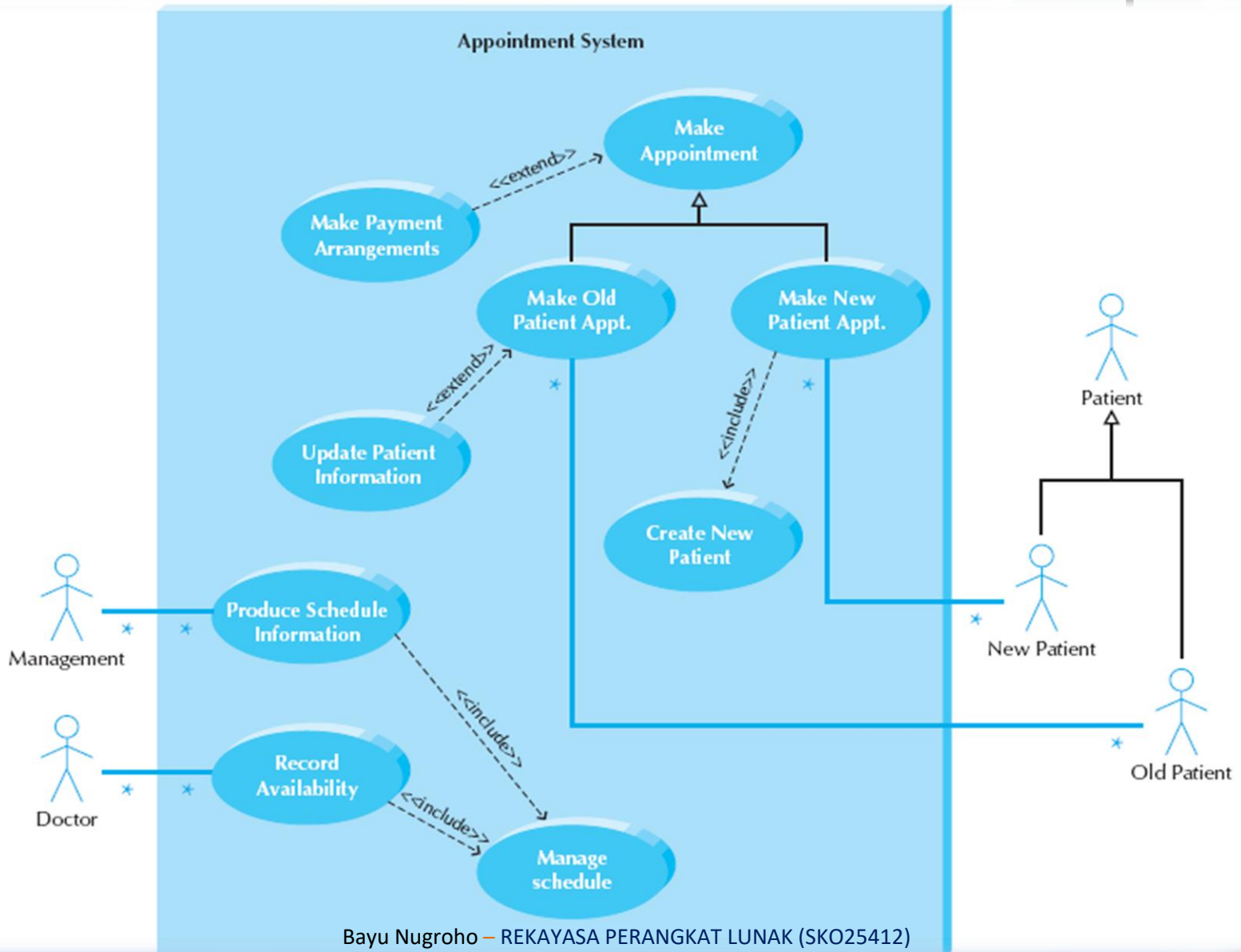
Use Case Diagram for Appointment System



Use Case Diagram with Specialized Actor



Extend and Include Relationships

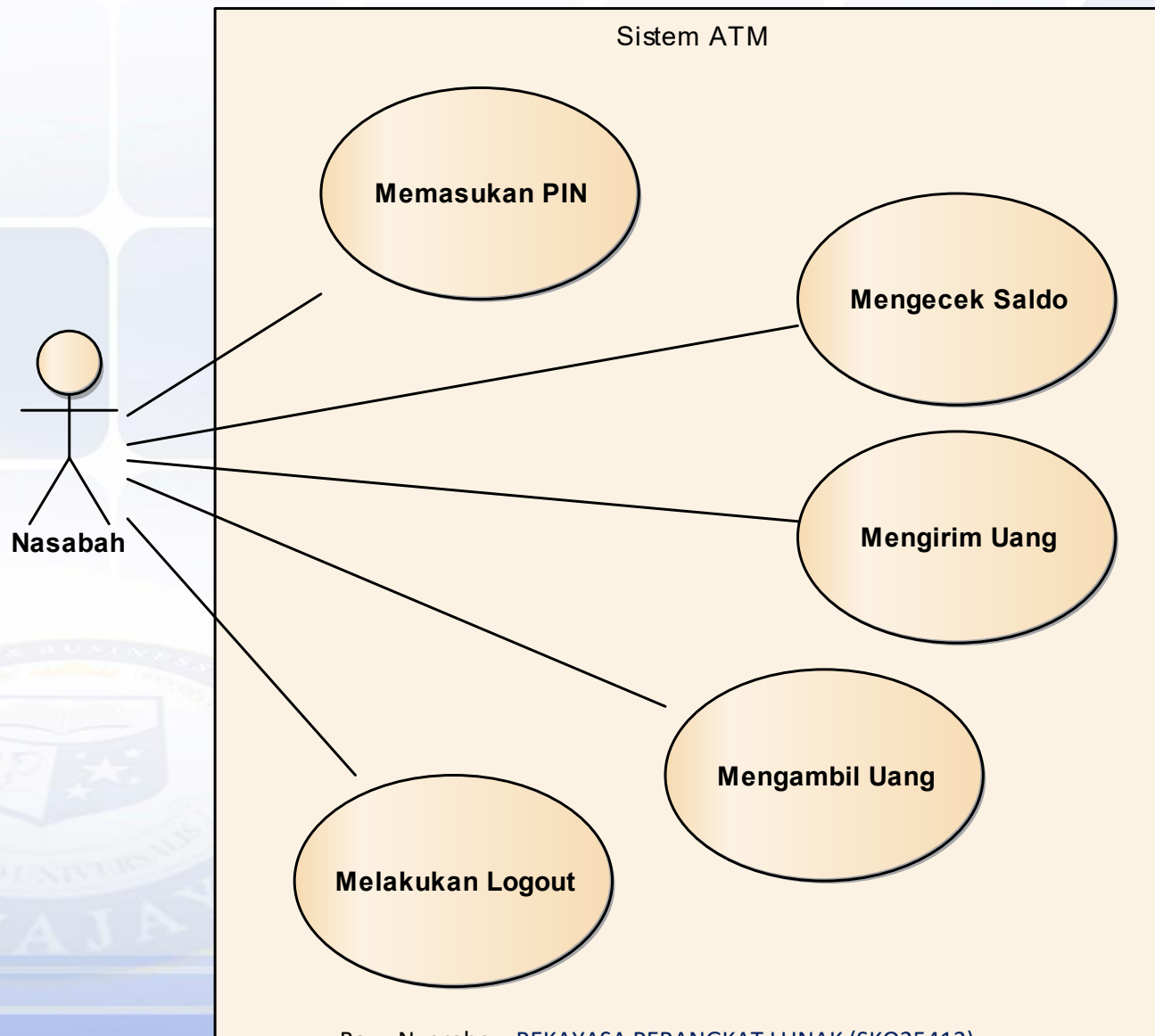


Studi Kasus: Use Case Diagram Sistem ATM



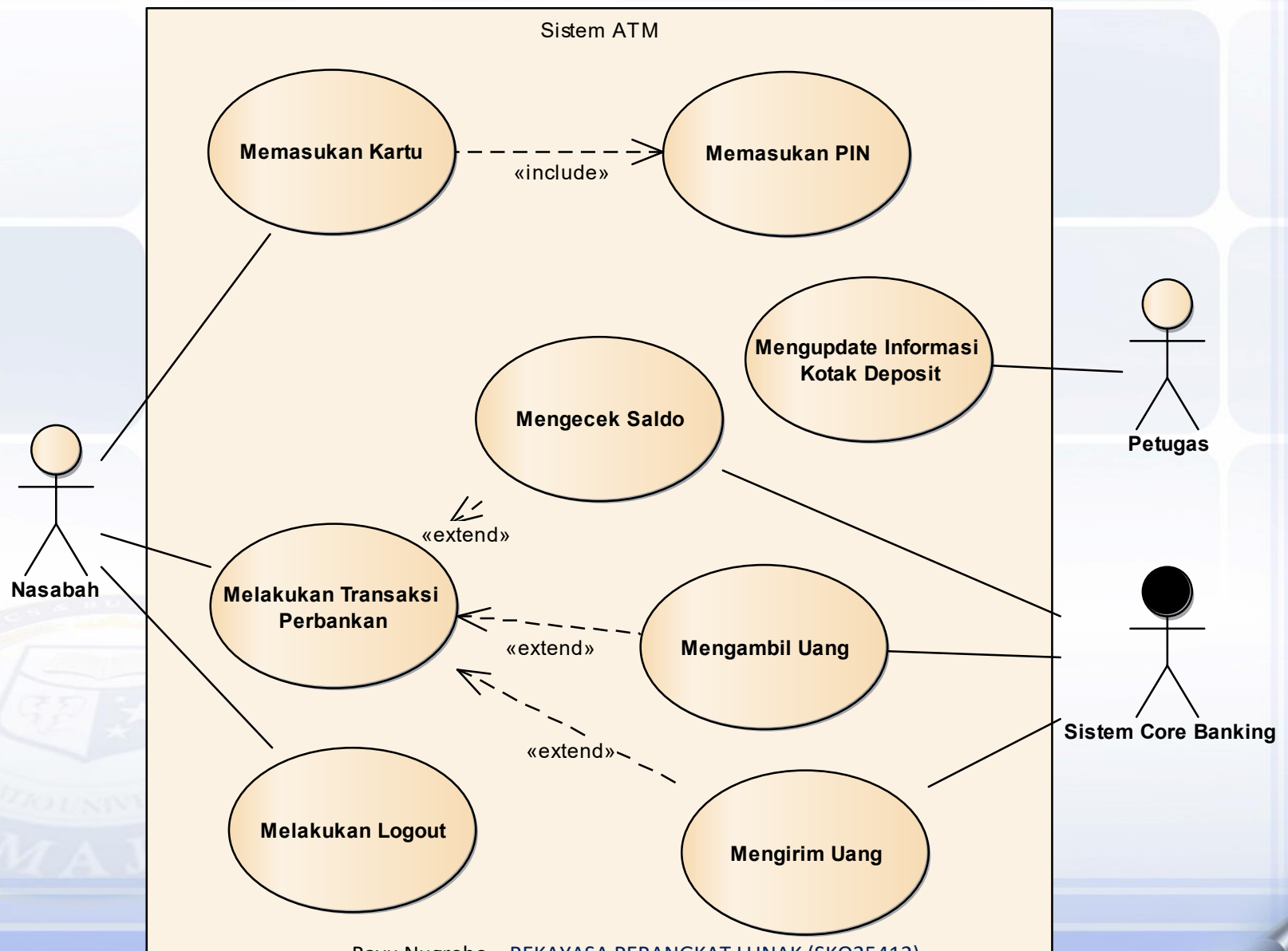
Use Case Diagram Sistem ATM

(versi Sederhana)



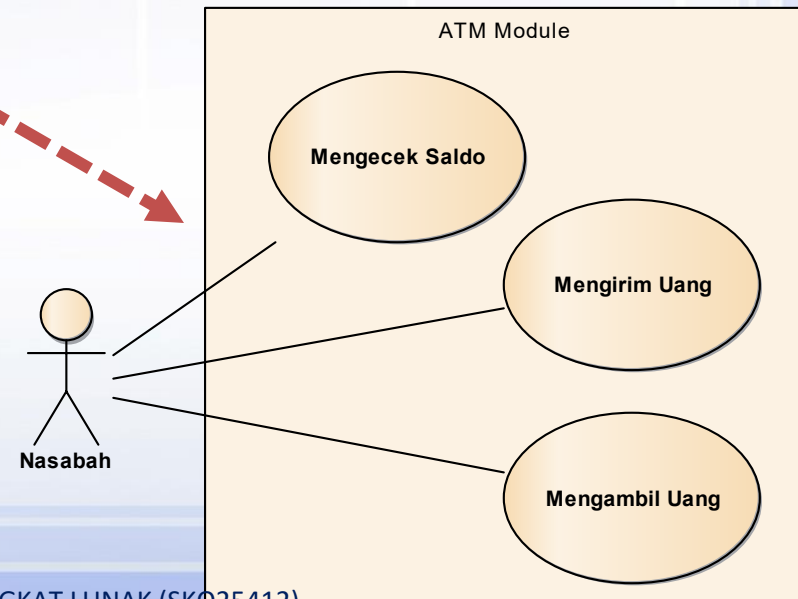
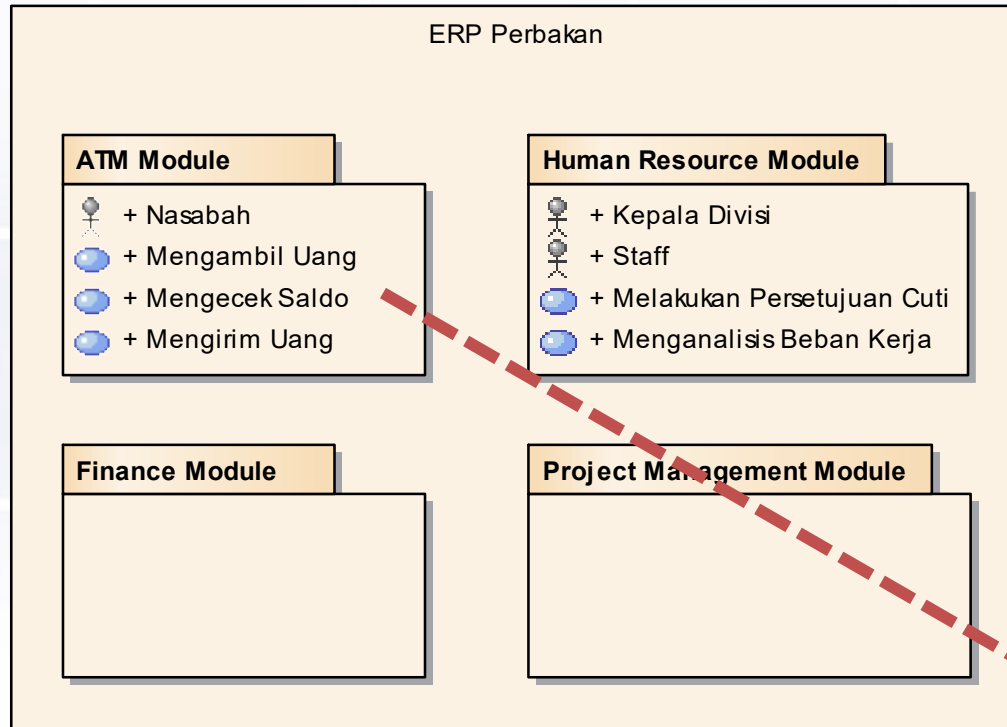
Use Case Diagram Sistem ATM

(Versi Include dan Extends)



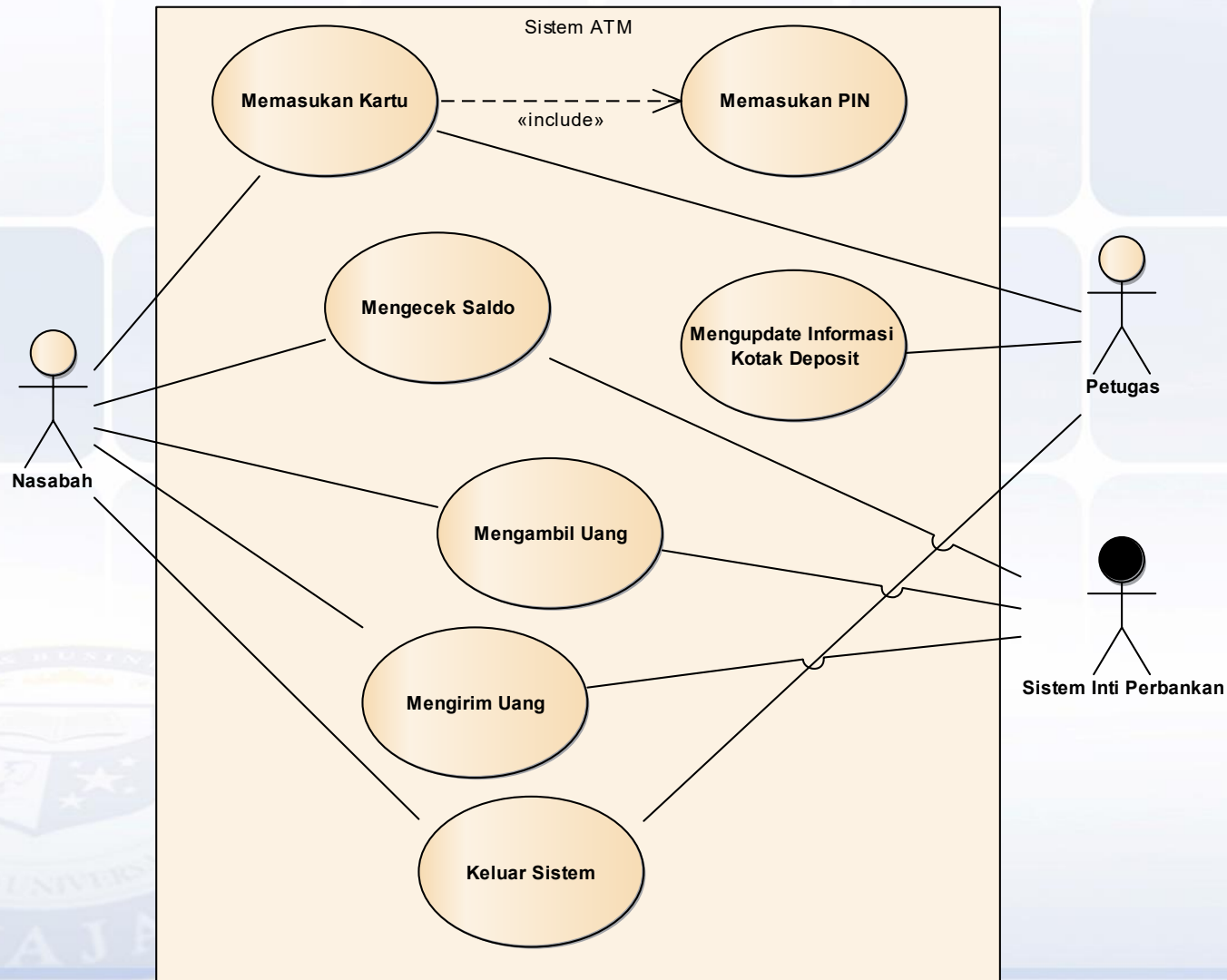
Use Case Diagram ERP Perbankan

(Sistem Lebih Kompleks)



Use Case Diagram Sistem ATM

(Versi Normal)





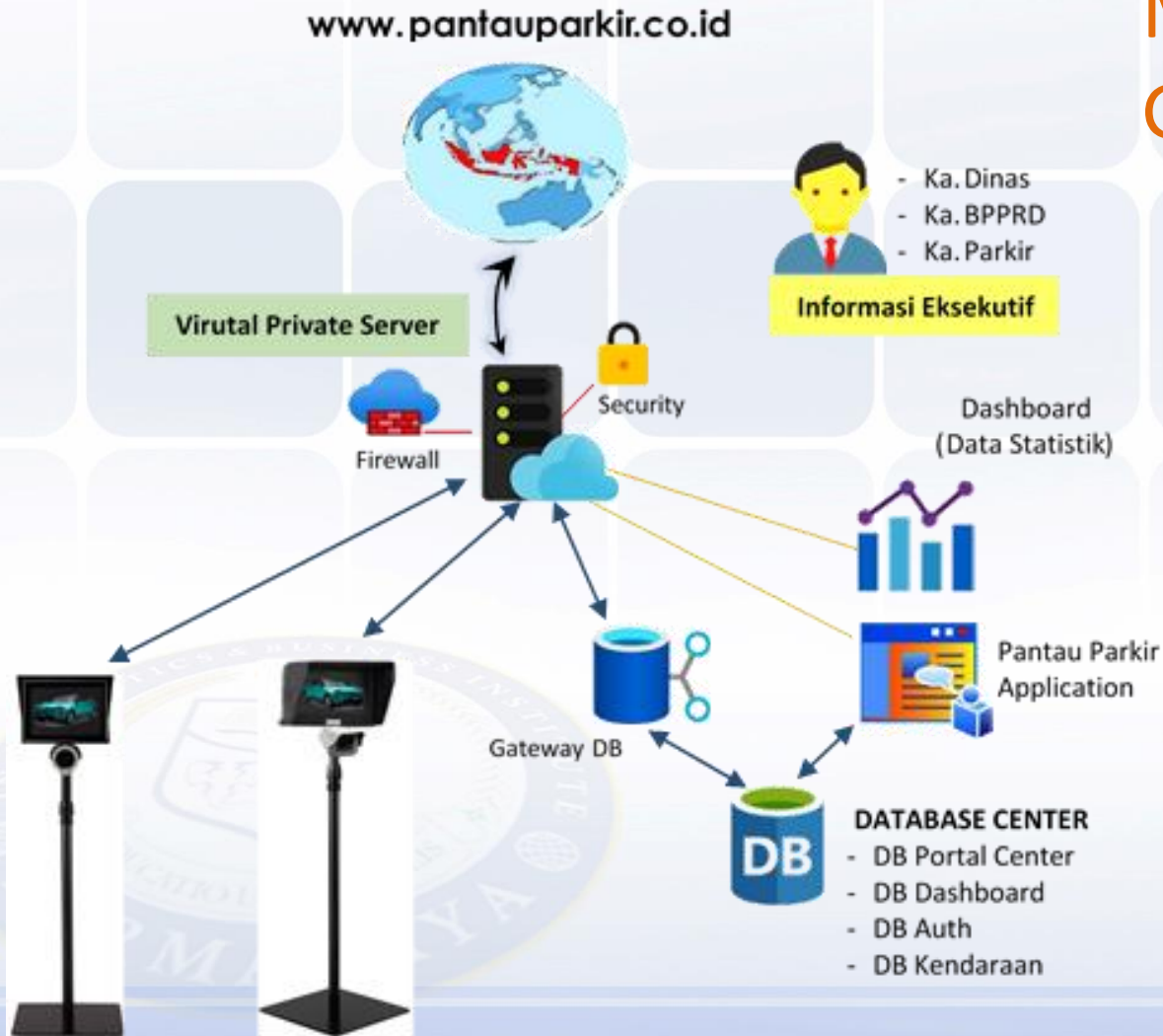
Systems Development Life Cycle (SDLC)

- 1. Planning:** Mengapa Software harus Dikembangkan?
 - System request, feasibility analysis, project size estimation
- 2. Analysis:** Siapa, Kapan Digunakan dan Alur Kerja Software?
 - Hdd/Soft, Requirement gathering dan business process modeling
- 3. Design:** Bagaimana Bekerja dan Komposisi dari Software?
 - Program design, user interface design, data design
- 4. Implementation:** Konstruksi dan Penyerahan Software
 - System construction, testing, documentation, installation

1. Representative Sistem (Gambar Sistem)

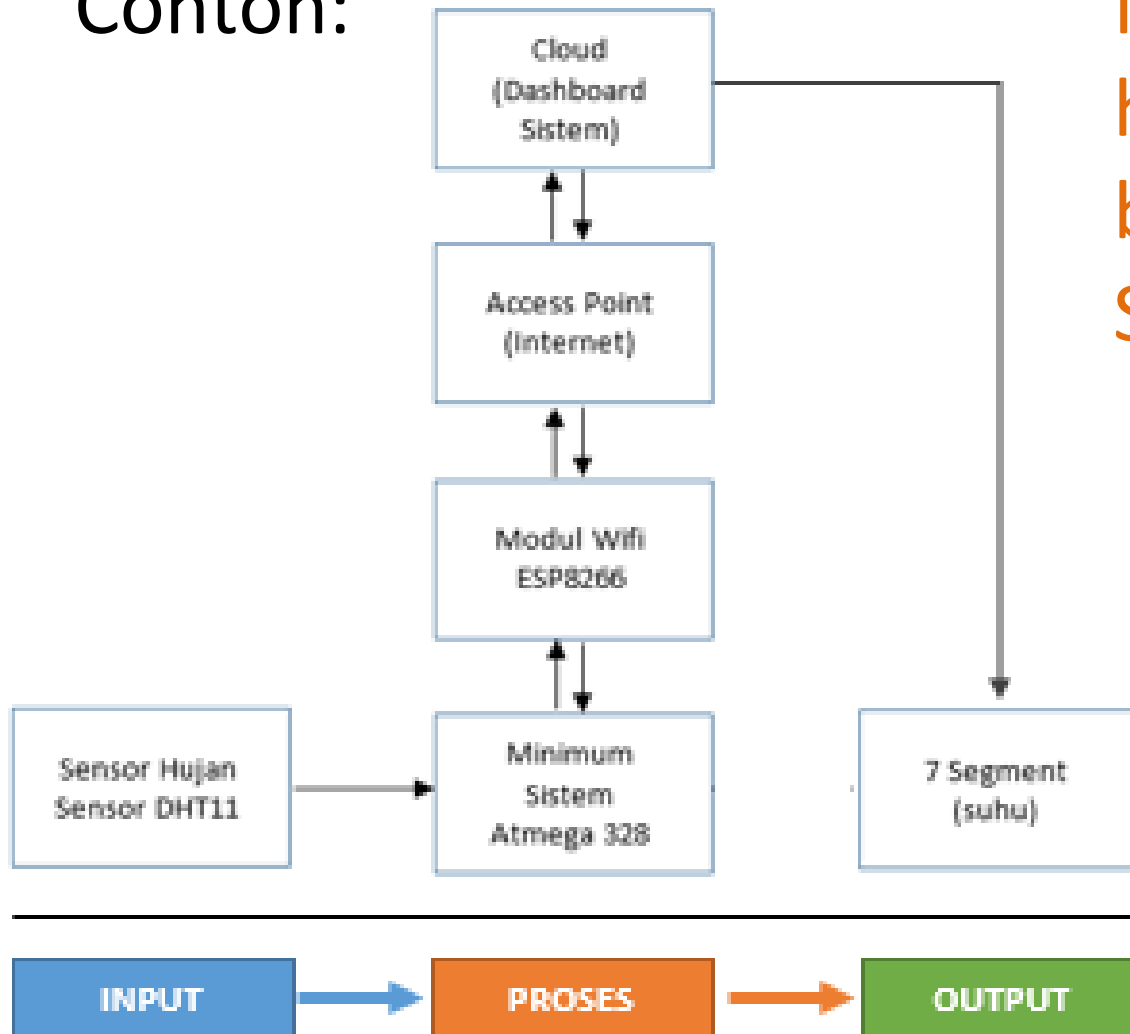
Contoh:

Merepresentasikan Gambaran sistem



2. BLOK Diagram Sistem

Contoh:



Menggambarkan hubungan antar bagian alur kerja Sistem

3. Software Environment

Mendokumentasikan
Kebutuhan Sistem

Contoh:

Pendukung Aplikasi/System



```
graph TD; A[Pendukung Aplikasi/System] --> B[HARDWARE]; A --> C[SOFTWARE];
```

HARDWARE

(Server)

Processor: Ghz

Memmmory: Ghz

Hardisk : TB

dll

SOFTWARE

Operating System:

Web Browser:

Database:

dll

4. Reqrutment Gathering:

- Functional

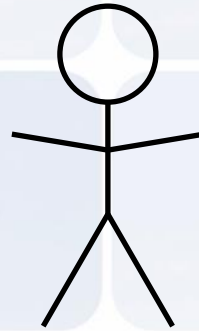
Fitur, Alur Kerja, dan aksi apa saja yang harus bisa dilakukan oleh system (**Contoh: fitur LOGIN, Kolom Pencarian, Tombol Checkout, dll**)

- Non Functional

Standar performa, keamanan data, kecepatan akses, dan kompatibilitas (**Contoh: Slide Hal.51**)

5. Identifikasi Proses Bisnis dengan :

USE CASE DIAGRAM



⟨⟨actor⟩⟩
Actor/Role



end

Pustaka :

BRAINMATICS

Romi Satria Wahono.net

System Analysis & Design

An Object-Oriented Approach with UML

(Alan Dennis, Barbara Haley Wixom, David Tegarden)

Modern Systems Analysis and Design

(Joseph S. Valacich, Joey F. George)

Systems Analysis and Design

(Kenneth E. Kendall, Julie E. Kendall)

System Analysis & Design

Ninth Edition

(Gary B. Shelly, Harry J. Rosenblatt)