Systems Analysis and Design

Romi Satria Wahono
romi@romisatriawahono.net
http://romisatriawahono.net
08118228331

Romi Satria Wahono

- SMA Taruna Nusantara Magelang (1993)
- B.Eng, M.Eng and Ph.D in Software Engineering
  Saitama University Japan (1994-2004)
  Universiti Teknikal Malaysia Melaka (2014)
- Core Competency in Enterprise Architecture, Software Engineering and Machine Learning
- LIPI Researcher (2004-2007)
- Founder and CEO:
  - PT Brainmatics Cipta Informatika (2005)
  - PT IlmuKomputerCom Braindevs Sistema (2014)
- Professional Member of IEEE, ACM and PMI
- IT and Research Award Winners from WSIS (United Nations), Kemdikbud, Ristekdikti, LIPI, etc
- SCOPUS/ISI Indexed Q1 Journal Reviewer: Information and Software Technology, Journal of Systems and Software, Software: Practice and Experience, Empirical Software Engineering, etc
- Industrial IT Certifications: TOGAF, ITIL, CCAI, CCNA, etc
- Enterprise Architecture Consultant: KPK, RistekDikti, INSW, BPPT, Kemsos Kemenkeu (Itjend, DJBC, DJPK), Telkom, FIF, PLN, PJB, Pertamina EP, etc

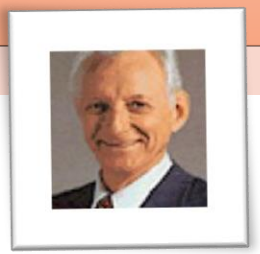
Learning Design

Educational Objectives
(Benjamin Bloom)

- Cognitive
- Affective
- Psychomotor

Criterion Referenced Instruction
(Robert Mager)

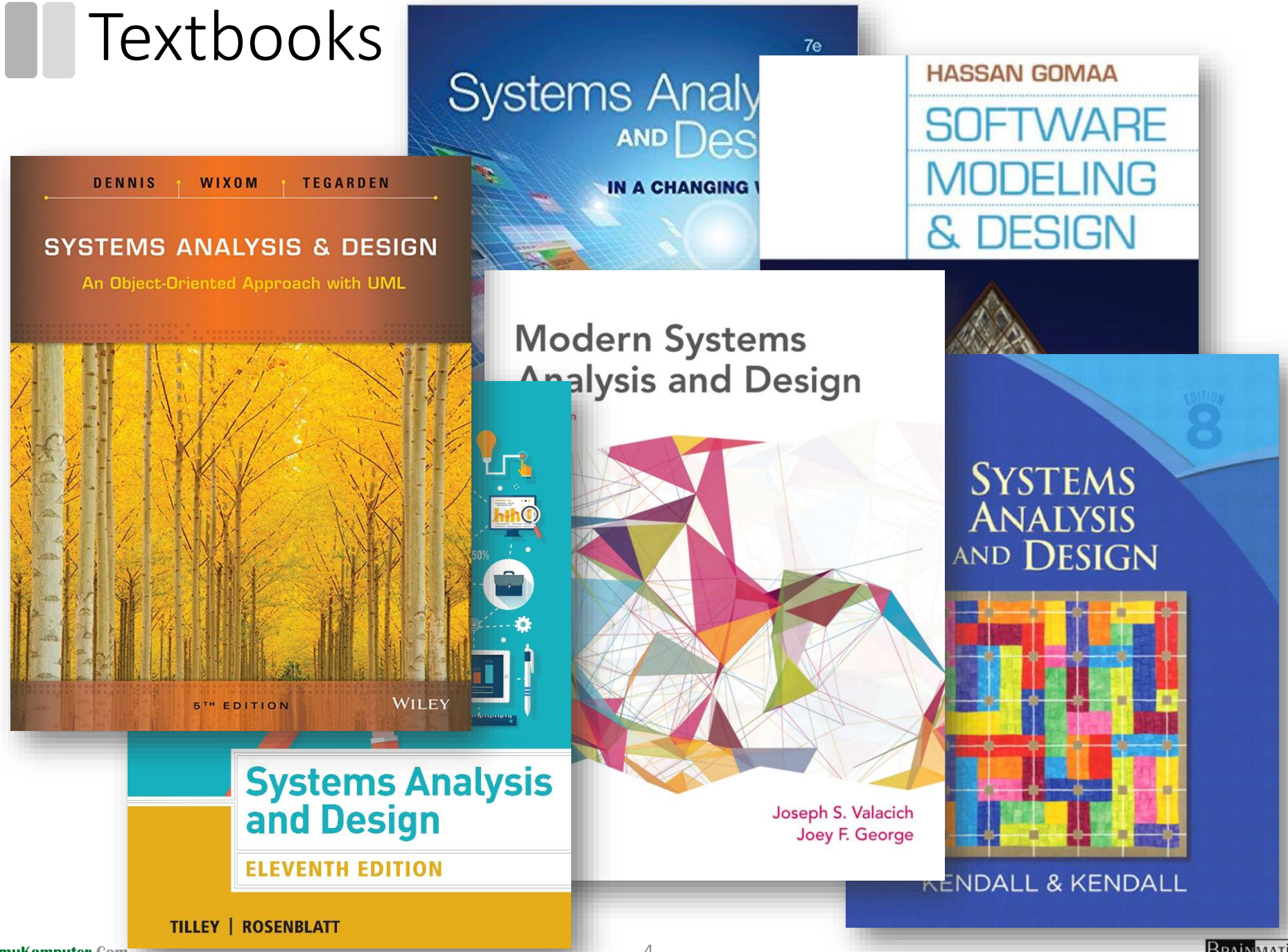
- Competencies
- Performance
- Evaluation



Minimalism
(John Carroll)

- Start Immediately
- Minimize the Reading
- Error Recognition
- Self-Contained

Textbooks


7e
Systems Analy
AND Des
IN A CHANGING
HASSAN GOMAA
SOFTWARE
MODELING
& DESIGN
DENNIS WIXOM TEGARDEN
SYSTEMS ANALYSIS & DESIGN
An Object-Oriented Approach with UML
5TH EDITION
WILEY
Modern Systems
Analysis and Design
Joseph S. Valacich
Joey F. George
EDITION 8
SYSTEMS
ANALYSIS
AND DESIGN
KENDALL & KENDALL
Systems Analysis
and Design
ELEVENTH EDITION
TILLEY | ROSENBLATT

Pre-Test

1. Sebutkan tahapan pengembangan sistem atau system development life cycle (SDLC)!
2. Sebutkan beberapa metodologi pengembangan sistem yang anda ketahui!
3. Gambarkan *requirement* di bawah dengan use case diagram!

SISTEM ELIBRARY

- Sistem elibrary memungkinkan pengguna untuk melakukan registrasi dan login
- Setelah menjadi member, pengguna dapat memodifikasi profile, serta mencari dan mendownload koleksi buku di sistem elibrary
- Admin sistem elibrary melakukan approval terhadap registrasi dan menampilkan laporan aktifitas pengguna secara individual maupun total

4. Gambarkan activity diagram, sequence diagram dan class diagram dari requirement di atas!
5. Hitung dengan metode yang anda ketahui, berapa orang dan waktu (bulan) yang dibutuhkan untuk mengembangkan sistem di atas!

Course Outline

1. Pengantar

1.1 Apa dan Mengapa Systems Analysis and Design?
1.2 Siklus Pengembangan Software
1.3 Metodologi Pengembangan Software

2. Systems Planning

2.1 Identifikasi Business Value dengan System Request
2.2 Analisis Kelayakan Pengembangan Software
2.3 Estimasi Usaha Pengembangan Software

3. Systems Analysis

3.1 Requirement Gathering
3.2 Identifikasi Proses Bisnis dengan Use Case Diagram
3.3 Pemodelan Proses Bisnis dengan AD atau BPMN
3.4 Realisasi Proses Bisnis dengan Sequence Diagram

4. Systems Design

4.1 Paradigma Berorientasi Objek
4.2 Pemodelan Class Diagram
4.3 Pemodelan User Interface Design
4.4 Pemodelan Data Model
4.5 Pemodelan Deployment Diagram

5. Systems Implementation

5.1 Konstruksi Software
5.2 Pengujian Software
5.3 Pembuatan Dokumentasi
5.4 Instalasi dan Manajemen Perubahan
5.5 Software Engineering Laws

1. Pengantar

1.1 Apa dan Mengapa Systems Analysis and Design?

1.2 Systems Development Life Cycle

1.3 Systems Development Methodology

1.1 Apa dan Mengapa Systems Analysis and Design?

Kegagalan Project Software

50% lebih project teknologi informasi gagal
(42% - Standish Group, 53% - General Accounting Office)

- Dibatalkan sebelum selesai
- Selesai tapi tidak pernah dipakai
- Tidak bermanfaat bagi pengguna
- Tidak sesuai dengan keinginan pengguna

Size Berbanding Lurus dengan Kegagalan

Organization	Year	Problems and Damage Cost
Canada Central Government	2014	Government website portal, projects with ongoing problems, $37.4 million loss
US State government	2012	Healthcare exchange website, cancelled, client and supplier both sued each other $200 million loss
Hudson Bay (Canada)	2005	Inventory system problems lead to $33.3 million loss
UK Inland Revenue	2005	$3.45 billion tax-credit overpayment caused by software errors
Avis Europe PLC (UK)	2004	Enterprise resource planning (ERP) system cancelled after $54.5 million spent
Ford Motor Co.	2004	Purchasing system abandoned after deployment costing approximately $400 million
Hewlett-Packard Co.	2004	ERP system problems contribute to $160 million loss
AT&T Wireless	2004	Customer relations management (CRM) system upgrade problems lead to $100 million loss

Keunikan dari Software

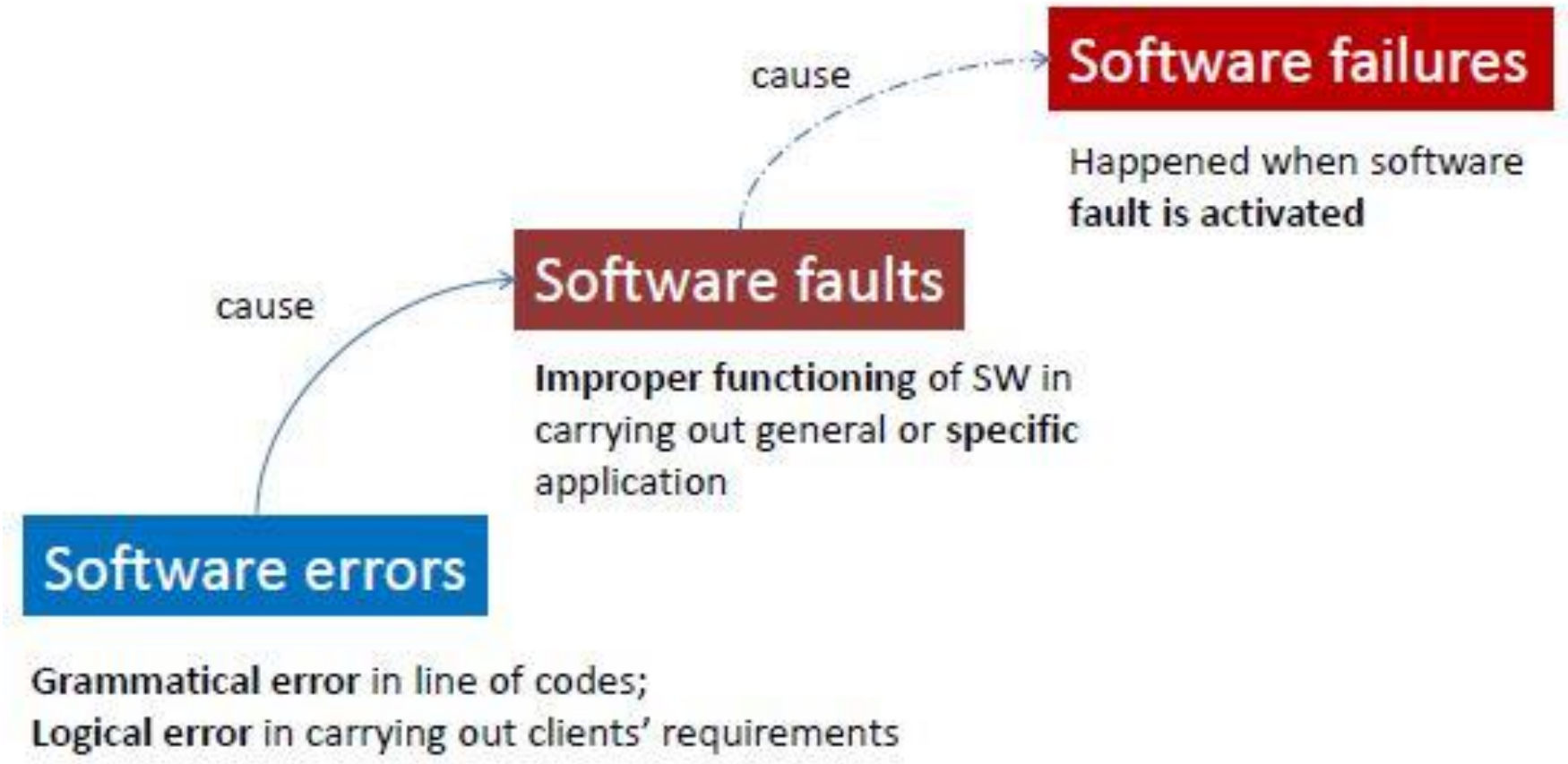
Karakteristik	Software	Hardware
Kompleksitas	Tingkat kompleksitas dari produk software tinggi, dengan kemungkinan perubahan parameter dan fungsi yang sangat beragam	Tingkat kompleksitas produk lain rendah, dengan kemungkinan perubahan parameter dan fungsi tidak beragam
Sumber Daya Manusia	Kuantitas SDM Tidak berhubungan dengan Kualitas dan Kecepatan Kerja	Kuantitas SDM berhubungan dengan Kualitas dan Kecepatan Kerja
Visibilitas Produk	Produk tidak terlihat dengan kasat mata, termasuk bila ada cacat (*defect*) dari produk	Produk terlihat dengan kasat mata, termasuk bila ada cacat (*defect*) dari produk

Brooks' Law (1975)

Adding manpower to a late project makes it later

(Endres, 2003) *[L36]*

Software Errors vs Faults vs Failures



Software Errors != Failures

- Suatu perusahaan PT ABC memproduksi software yang akan ditanam ke dalam suatu device
- Salah satu fungsi yang terdapat pada software adalah akan mematikan device secara otomatis apabila suhu ruangan lebih besar daripada 30^{o} celcius
- Programmer salah menuliskan logika menjadi:

```
...
if (suhu > 3) shutdownDevice();
...
```

- Error ini tidak pernah menyebabkan failure pada software, dan perusahaan PT ABC sampai saat ini terkenal sebagai perusahaan yang memproduksi software tanpa bug
- Jelaskan mengapa bisa terjadi demikian!

Warranty Lawsuits

- Mortenson vs. Timeberline Software (TS) (≈1993)
  - Mortenson menggunakan software yang diproduksi TS untuk membuka tender pembangunan rumah sakit
  - Software memiliki bug sehingga memenangkan perusahaan yang mengajukan proposal paling mahal (kerugian 2 miliar USD)
  - TS tahu tentang bug itu, tapi tidak mengirimkan update ke Mortenson
  - Pengadilan di Amerika Serikat memenangkan perusahaan TS

- Uniform Computer Information Transaction Act (UCITA) allows software manufacturers to:
  - disclaim all liability for defects
  - prevent the transfer of software from person to person

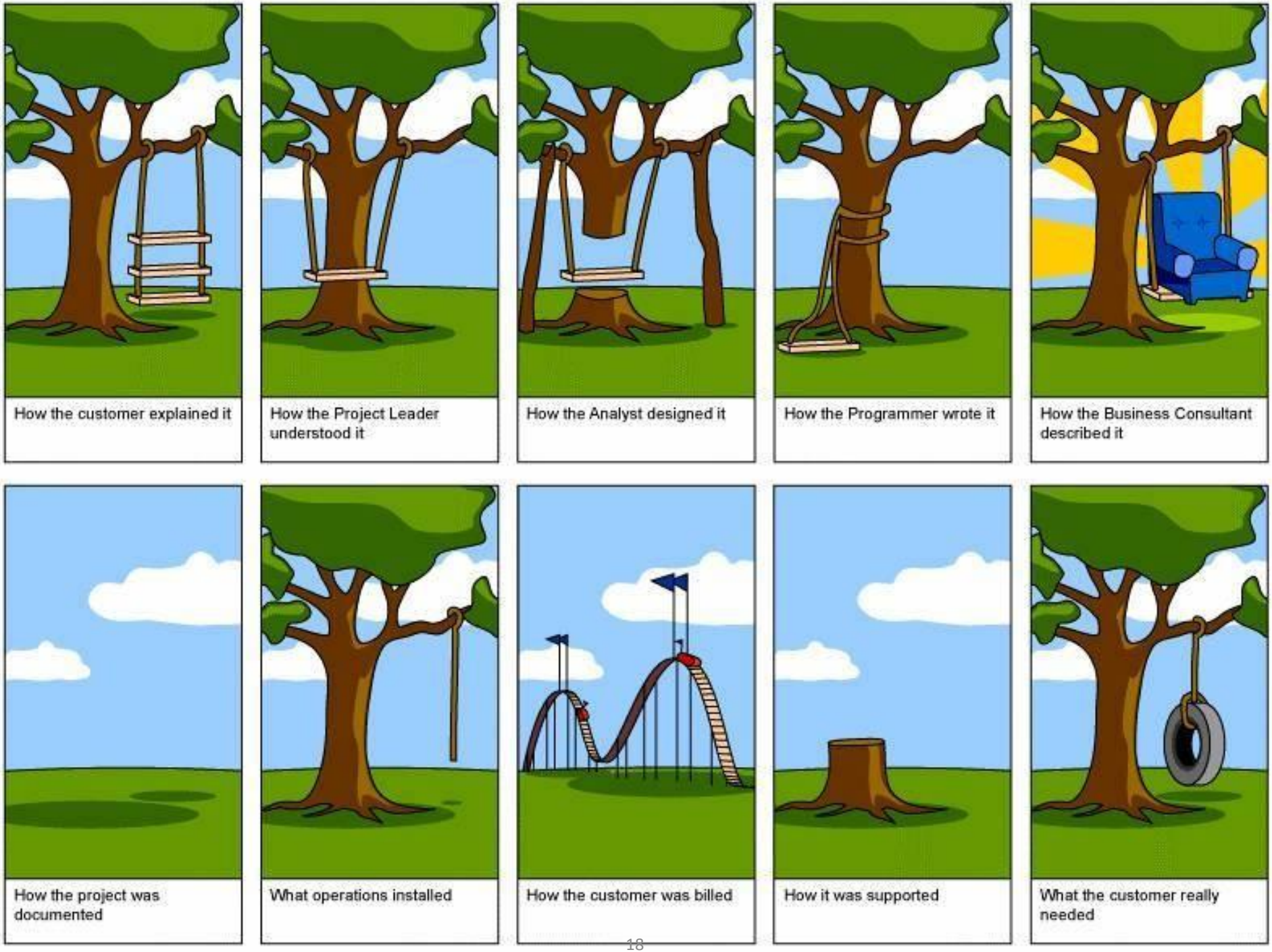
Disclaimer of Warranties

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, MICROSOFT AND ITS SUPPLIERS PROVIDE TO YOU THE SOFTWARE COMPONENT, AND ANY (IF ANY) SUPPORT SERVICES RELATED TO THE SOFTWARE COMPONENT ("SUPPORT SERVICES") **AS IS AND WITH ALL FAULTS**; AND MICROSOFT AND ITS SUPPLIERS HEREBY DISCLAIM WITH RESPECT TO THE SOFTWARE COMPONENT AND SUPPORT SERVICES ALL WARRANTIES AND CONDITIONS, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY (IF ANY) WARRANTIES OR CONDITIONS OF OR RELATED TO: TITLE, NON-INFRINGEMENT, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, LACK OF VIRUSES, ACCURACY OR COMPLETENESS OF RESPONSES, RESULTS, LACK OF NEGLIGENCE OR LACK OF WORKMANLIKE EFFORT, QUIET ENJOYMENT, QUIET POSSESSION, AND CORRESPONDENCE TO DESCRIPTION. **THE ENTIRE RISK ARISING OUT OF USE OR PERFORMANCE OF THE SOFTWARE COMPONENT AND ANY SUPPORT SERVICES REMAINS WITH YOU**.

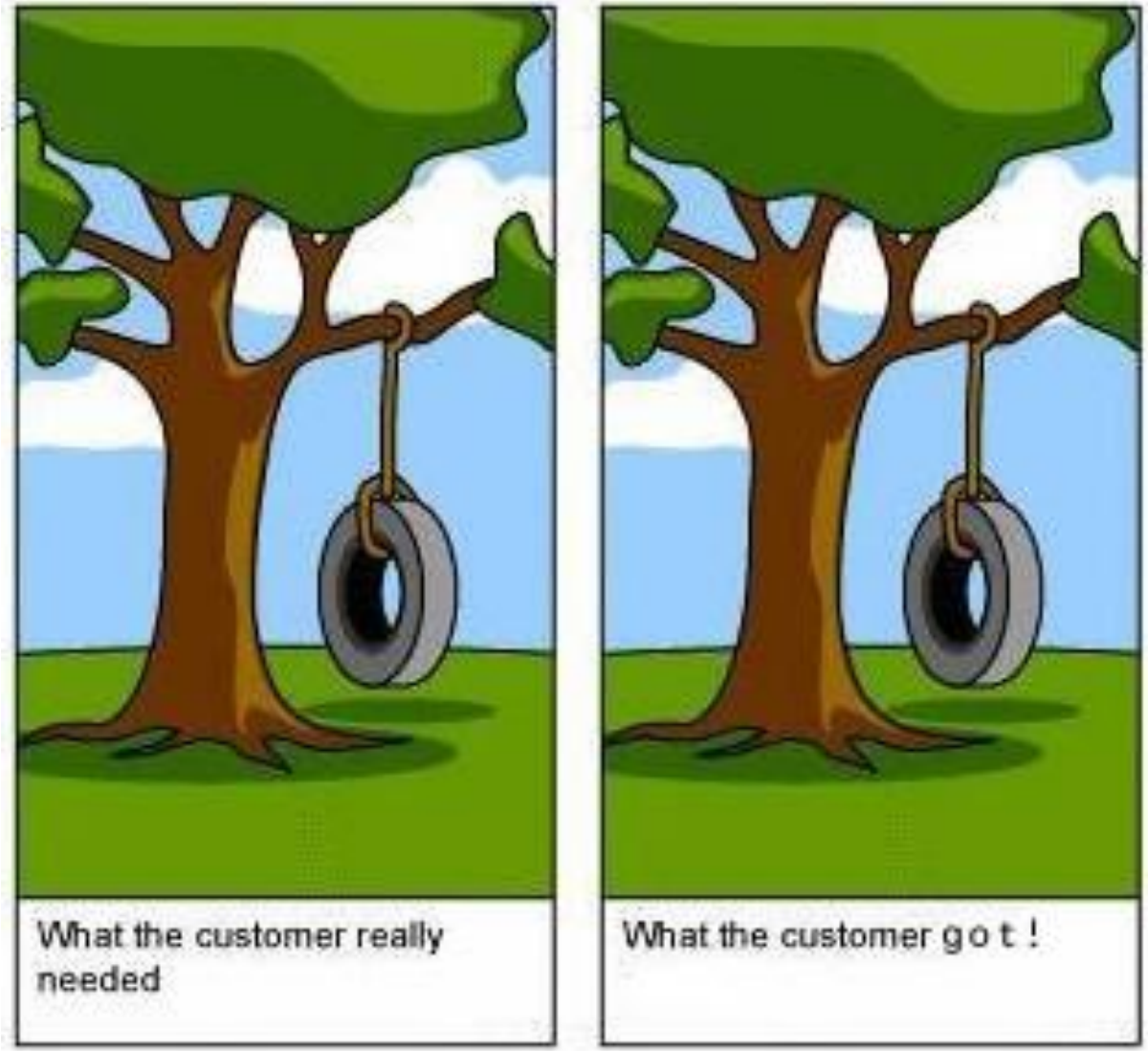
Software Engineering Problem

Building software will always be hard, inherently no silver bullet

(Brooks, 1987)


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Tantangan Pengembangan Software



Glass' Law (1998)

Requirement deficiencies are the prime source of project failures

(Endres, 2003) *[L1]*

Software Berkualitas?

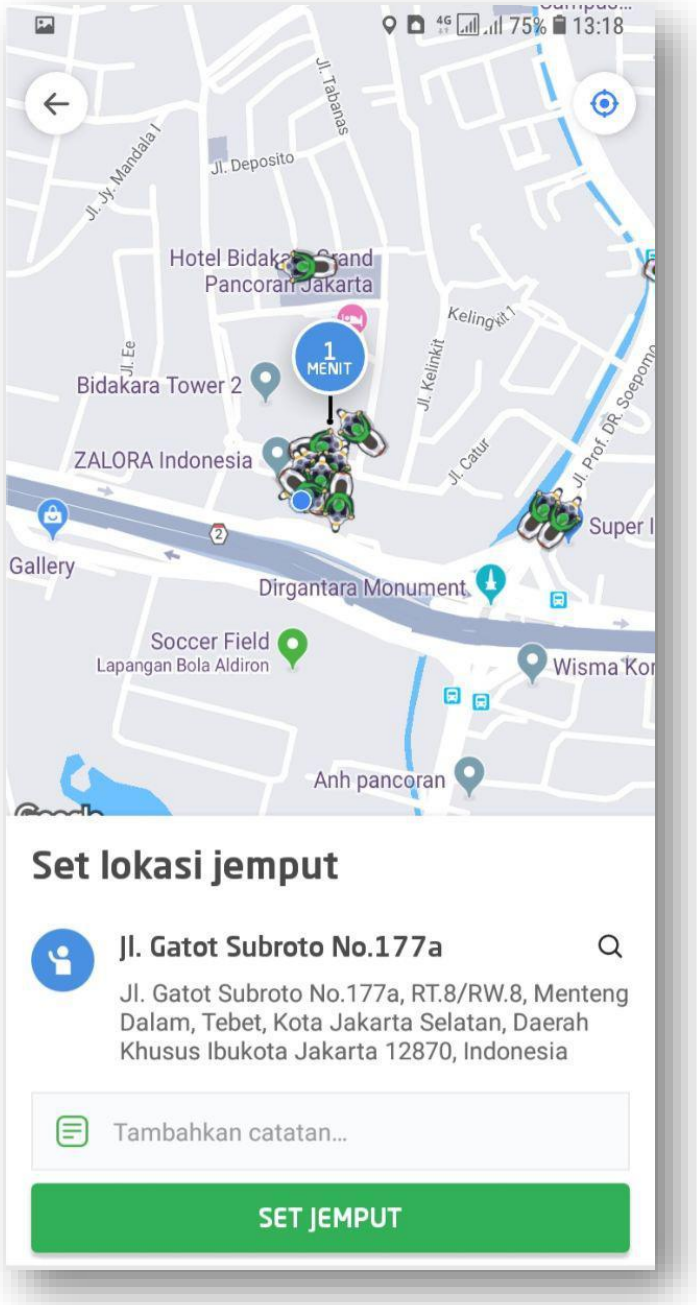
Software quality is *(IEEE, 1991)*:

1. The degree to which a system, component, or

Sesuai Kebutuhan

2. The degree to which a system, component, or process meets customer

Ada Keuntungan



Komputer Datang untuk Efisiensi!

- Alan Turing dengan komputer yang dibuat berhasil mendekripsi pesan enigma dengan cepat
  - Dilakukan manusia puluhan atau ratusan tahun, dengan komputer hanya 3-4 jam
  - Menyelesaikan perang yang harusnya bisa terjadi puluhan tahun, menjadi hanya 3 tahun, dan menghemat miliaran USD

- Komputer adalah alat untuk efisiensi, untuk mengurangi *cost*, mempercepat waktu, meningkatkan *income*
  - Investasi pengadaan komputer dan software harus bisa meningkatkan Return on Investment (ROI), mempercepat Break Event Point (BEP) pada organisasi

Software Sistem Kita Berkualitas?

1. KTP Manual

3 Hari

2. 

Komputer

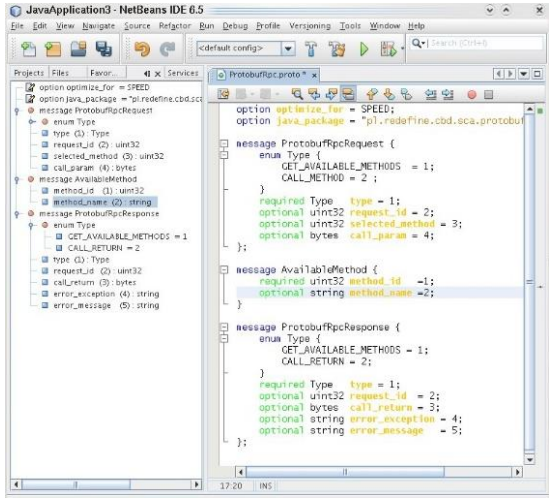
3 Minggu!

3. KTP-EL

1 Tahun!

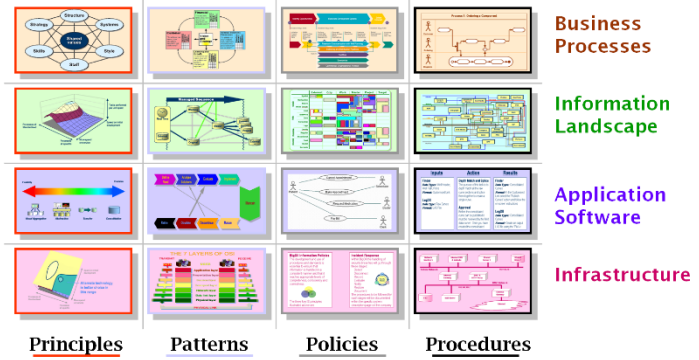
Software Development Evolution

1. How to Write a Code (Coder or Programmer)
   - Menguasai Bahasa Pemrograman

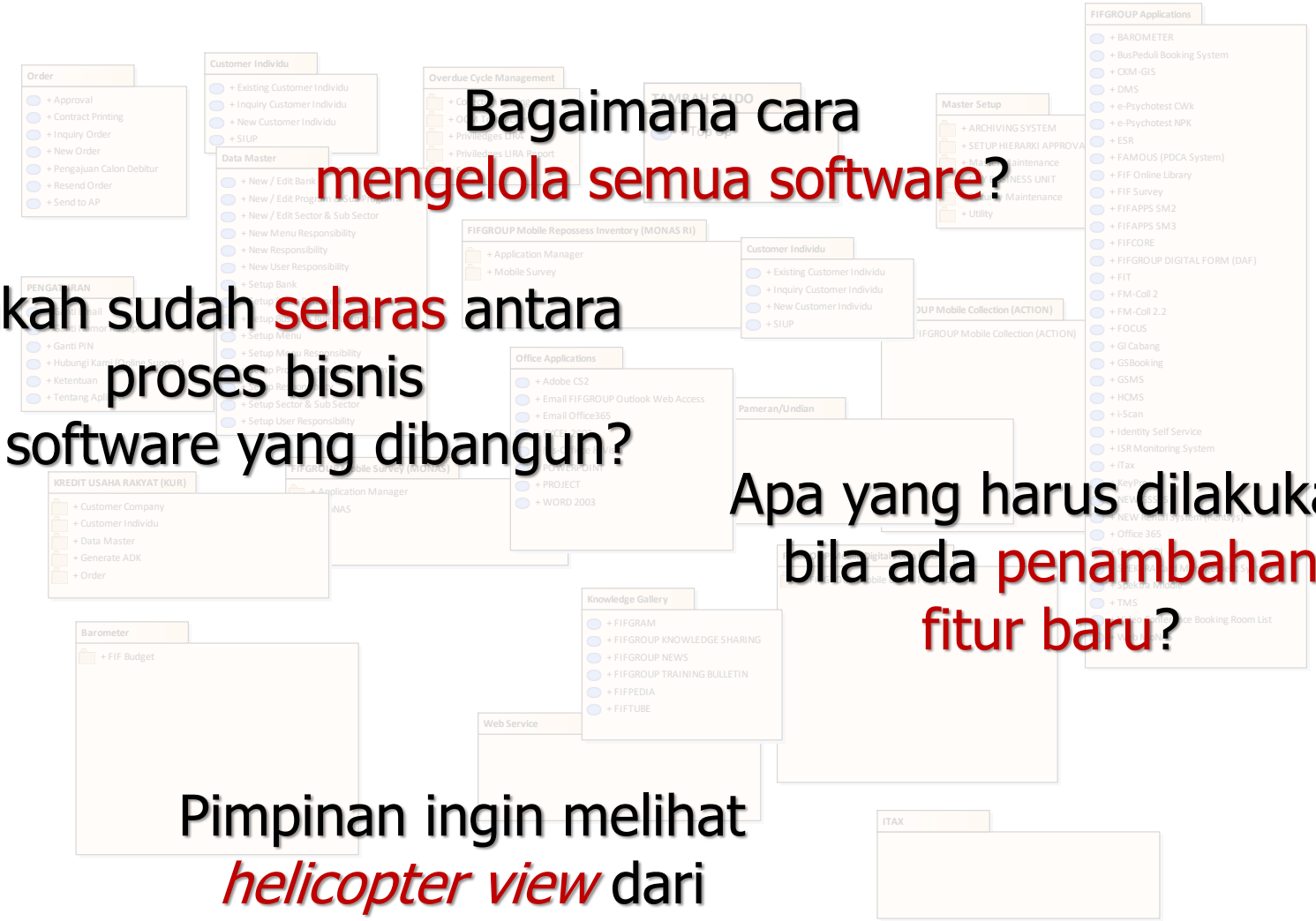


2. How to Develop a Software (Software Engineer)
   - Menguasai Metodologi Pengembangan Software

3. How to Manage Software (Enterprise Architect)
   - Menguasai Manajemen Software



Software Berkembang, Semakin Banyak dan Tumbuh Besar!

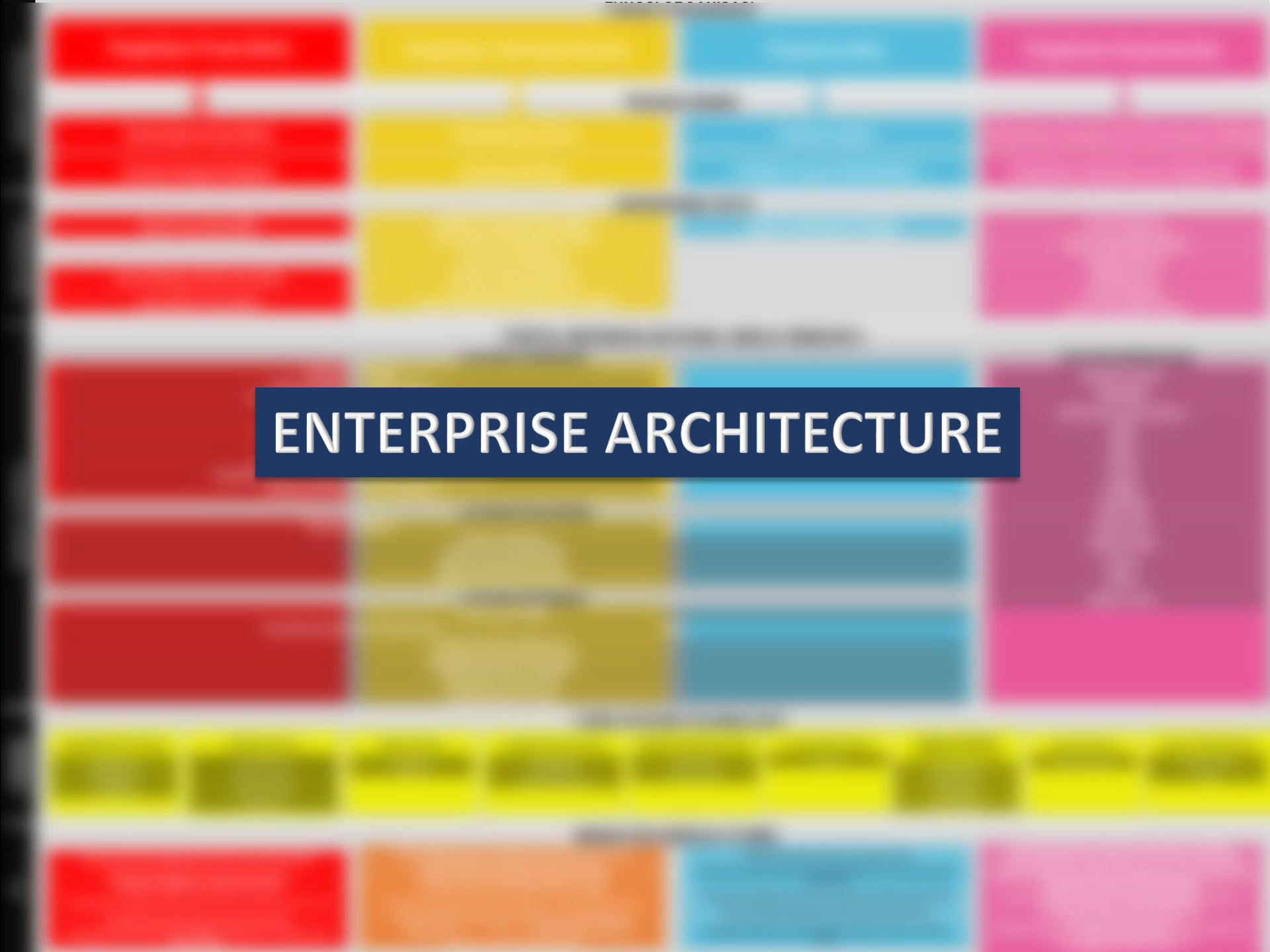


Bagaimana cara
mengelola semua software?

Apakah sudah selaras antara
proses bisnis
dan software yang dibangun?

Apa yang harus dilakukan
bila ada penambahan
fitur baru?

Pimpinan ingin melihat
helicopter view dari
semua software yang ada?


ENTERPRISE ARCHITECTURE

Morris–Ferguson Law (1993)

Architecture wins over technology

(Endres, 2003) *[L48]*

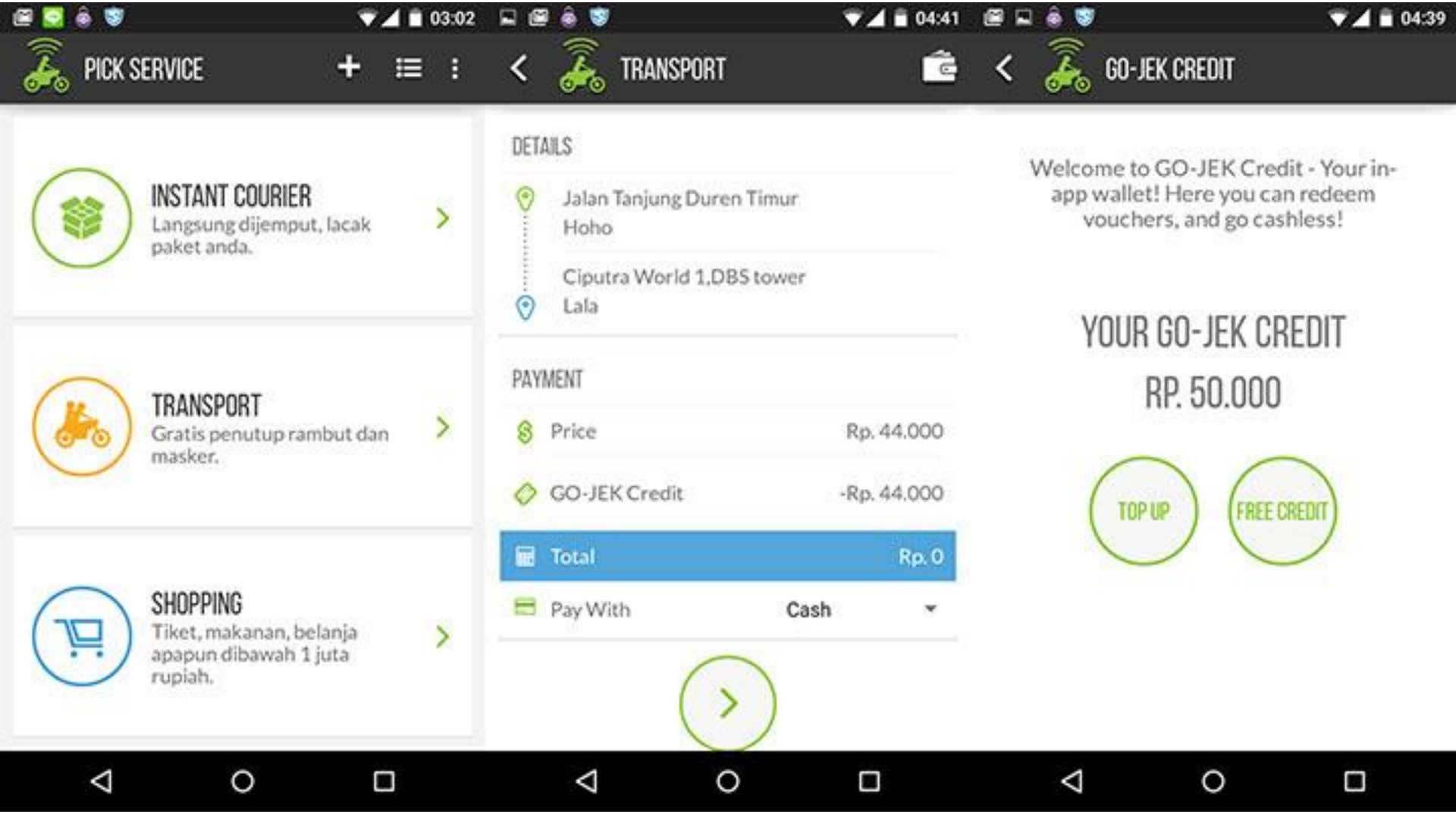
Kemampuan Coding?

- Penting!

- Tapi lebih penting lagi kemampuan membaca kebutuhan riil masyarakat dan kemampuan analisis kelayakan dari software yang kita kembangkan

- Mendidik programmer lebih mudah daripada system analyst, software engineer atau enterprise architect

Berapa Harga Kode Gojek?



Berapa Harga (Ide) Software Gojek?

Bos Gojek dan Bukalapak Masuk Daftar Orang Terkaya di Indonesia

Reporter: Imam Hamdi
Editor: Kodrat Setiawan

Kamis, 26 Juli 2018 06:21 WIB

0 KOMENTAR | 71 | 0 | 72

Pendiri dan CEO Go-Jek (Gojek) Indonesia, Nadiem Makarim. TEMPO/Ratih Purnama

TEMPO.CO, Jakarta - Empat pendiri start up atau perusahaan rintisan masu
terkaya di Indonesia versi Majalah Globe Asia yang dirilis pada Juni 2018. M
Unardy, 30 tahun, pendiri Traveloka; Wiliam Tanu Jaya, 36 tahun, (Tokopedia
tahun, (Bukalapak) dan Nadiem Makarim, 33 tahun, (Gojek).

Baca juga: Bos Djarum Masih Orang Terkaya di Indonesia

Keempat nama pendiri start up tersebut merupakan wajah baru yang masu
negara ini. Globe Asia mencatat kekayaan Ferry yang menduduki urutan 14
juta. Wiliam berada di urutan 148 (US$ 130 juta), Achmad di urutan 149 (US
di urutan 150 (US$ 100 juta).

Google Tanam Modal, Valuasi Go-Jek Capai Rp 53 Triliun?

FATIMAH KARTINI BOHANG
Kompas.com - 18/01/2018, 16:37 WIB

Ojek Online yang manggkal di bawah kolong flyover dekat Stasiun Tebet (Stanly)

KOMPAS.com — Raksasa internet Google dikabarkan menggelontorkan dana segar ke layanan *ride-sharing* Go-Jek pada awal 2018 ini. Kesepakatan keduanya diteken paling cepat pekan depan.

Tak diumbar jumlah pastinya, tetapi investasi itu menaikkan valuasi Go-Jek menjadi 4 miliar dollar AS atau setara Rp 53,3 triliun. Setidaknya begitu menurut sumber dalam yang familiar dengan penanaman modal ini.

Jika hal itu benar, Go-Jek yang hanya fokus di Indonesia semakin dekat mengejar Grab yang bisnisnya lebih luas seantero Asia Tenggara. Valuasi terakhir Grab disebutkan berada di angka 6 miliar dollar AS atau senilai Rp 80

Norman's Law (1993)

It takes 5000 hours to turn a novice into an expert

(Endres, 2003)

[L43]

1.2 Siklus Pengembangan Software

Siklus Pengembangan Software:

Alur, Peran, dan Tahapan (*Deliverable*) *(Tilley, 2012) (Dennis, 2016) (Valacich, 2017)*

1. **User/Product Owner** membawa permintaan kebutuhan (perubahan) software (System Request) ke System Analyst
2. **System Analyst** membuat analisis kelayakan (Feasibility Analysis) dari System Request tersebut

Planning
(System Proposal)

3. Setelah dinyatakan layak, System Analyst melakukan analysis dan design, dan hasilnya adalah System Specification
   - **Business Analyst** membantu System Analyst memahami proses bisnis dari software yang akan dibangun

Analysis and Design
(System Specification)

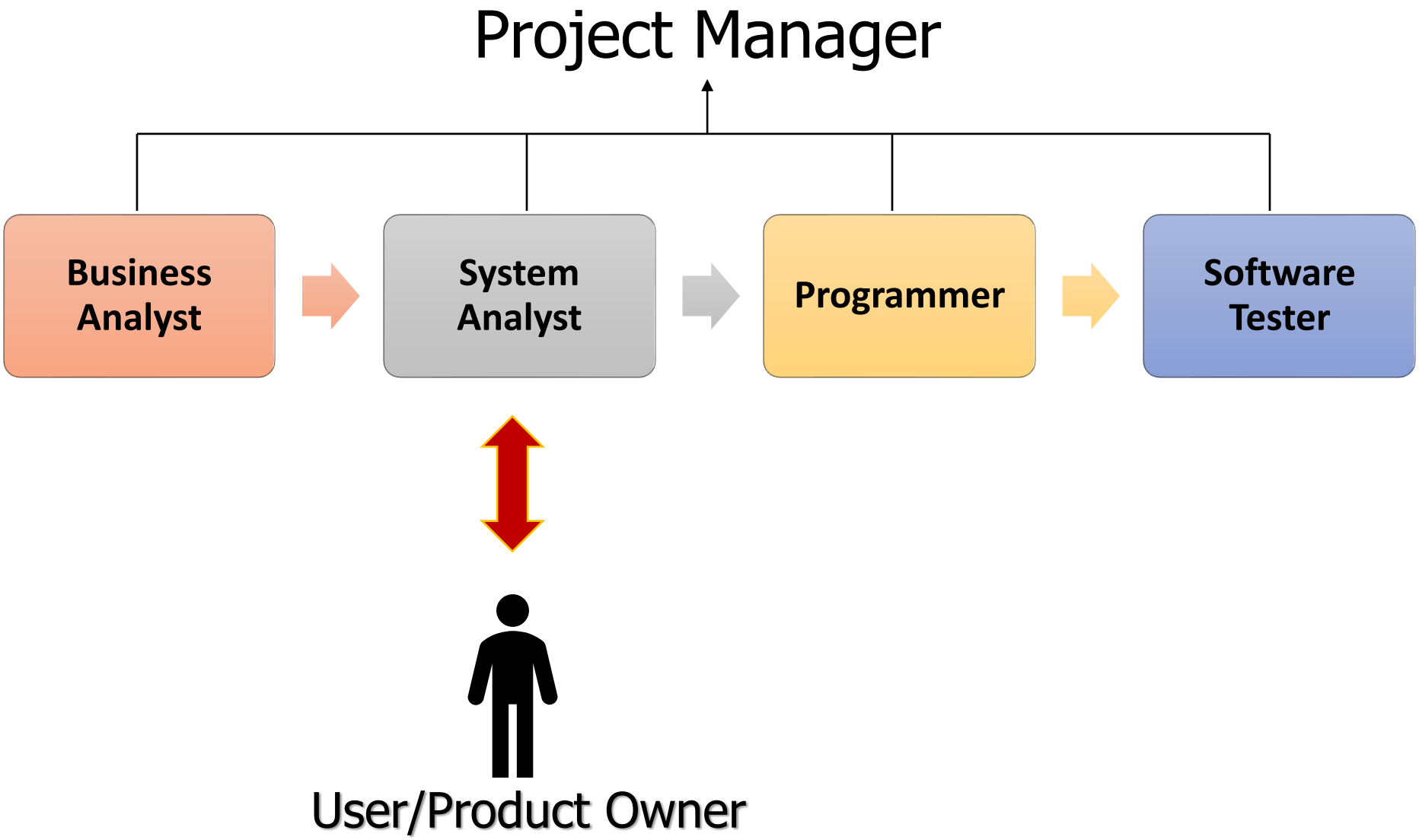
4. System Specification diserahkan oleh System Anayst ke **Programmer** untuk dilakukan Konstruksi (*Coding*)
5. Hasil Konstruksi berupa Kode Program diserahkan ke **Software Tester** untuk dilakukan Pengujian (*Unit, Integration, System, User Acceptance Testing*)
6. Instalasi (*delivery*) software dan manajemen perubahan
   - Software = Kode Program + Dokumentasi (Pengembangan dan Penggunaan)

Implementation
(Software)

7. Siklus kembali ke 1 apabila ada permintaan perubahan (Permintaan Perubahan Software)

Maintenance
(Updated Software)

Peran dalam Pengembangan Software



User/Product Owner

- Pihak yang membutuhkan software (*requirement*), yang menganalisis bahwa ada business value (benefit) apabila suatu proses bisnis diautomasi menggunakan software
  - Kebutuhan dan business value ini diwujudkan dalam bentuk System Request

- Diperlukan skill, pengetahuan dan pemahaman tentang domain bidang dan proses bisnis
  - Bukan di bidang komputernya

Business Analyst

- Fokus pekerjaan membantu user/product owner di masalah domain dan bidang (business) dari software
  - Nilai bisnis (business value) dari software (sistem)
  - Perbaikan bisnis proses yang diperlukan untuk pengembangan software

- Diperlukan skill, pengetahuan dan pemahaman tentang domain bidang dan proses bisnis
  - Bukan di bidang komputernya

System Analyst

- Orang yang menganalisis kebutuhan bisnis, mengidentifikasi peluang untuk perbaikan, mendesain software (sistem) untuk mengimplementasikan idenya

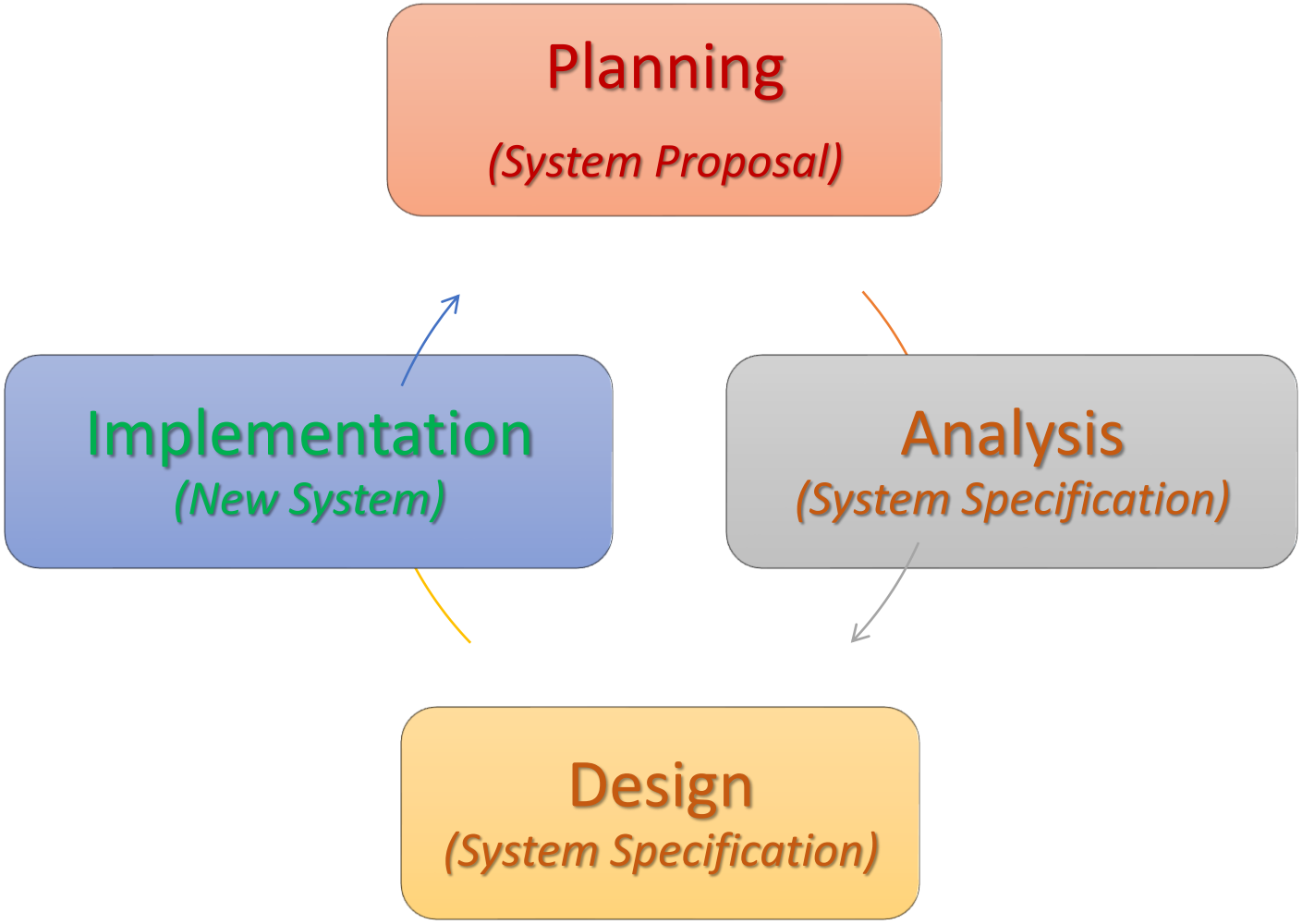
- Fokus pekerjaan ke software dan sistem informasinya, bukan domain atau bidangnya
  - Bagaimana software bisa meningkatkan ROI dan mempercepat BEP dari layanan (organisasi)
  - Mendesain software dan sistem baru
  - Menjamin standard kualitas dari software dan sistem yang dibangun

- Diperlukan skill, pengetahuan dan pemahaman tentang design analysis, programming, business (*lesser degree*)

Project Manager

- Tugas utama di perencanaan, schedule, budget dan monitoring pekerjaan
- Harus bisa memberi jaminan bahwa kebutuhan dan business value yang diharapkan user/product owner bisa terpenuhi
- Mampu mengelola member team development
  - Yang tidak semua memiliki kemampuan teknis dan komunikasi yang baik
- Apabila sumber daya manusia terbatas, atau project skala kecil, bisa dirangkap oleh Systems Analyst

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

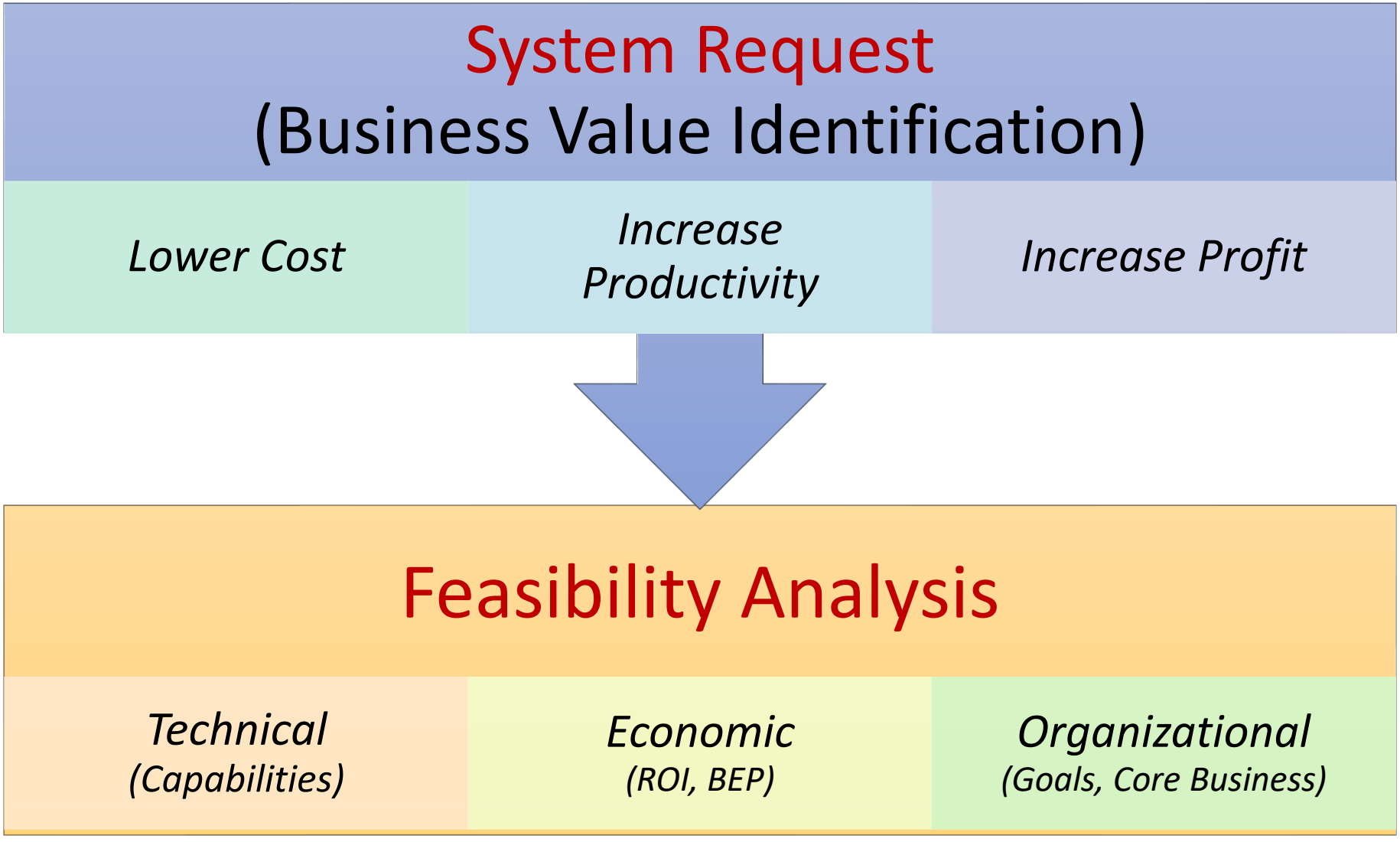
Planning

1. Identifikasi Business Value (System Request)
    1. Lower Costs
    2. Increase Productivity
    3. Increase Profits

2. Analisis Kelayakan
    1. Technical Feasibility
    2. Economic Feasibility
    3. Organizational Feasibility

(System Proposal)

Planning


System Request
(Business Value Identification)
Lower Cost
Increase Productivity
Increase Profit
Feasibility Analysis
Technical
(Capabilities)
Economic
(ROI, BEP)
Organizational
(Goals, Core Business)

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
2. Meningkatkan efisiensi penanganan masalah pelanggan melalui internet

Business Requirements:

Sistem yang mendukung penjualan musik secara online. Fitur-fitur yang harus ada:

1. Fitur Pencarian Produk
2. Fitur Pencarian Toko yang Menyediakan Stok Produk
3. Fitur Pemesanan Produk Melalui Toko yang Menyediakan
4. Fitur Pembayaran dengan Berbagai Pilihan Pembayaran

Business Value:

Intangible Value:

- Meningkatkan kenyamanan dan kepuasan pelanggan
- Meningkatkan brand recognition tentang perusahaan di dunia Internet

Tangible Value:

1. Meningkatkan penjualan dari pelanggan baru lewat Internet:
   - Rp 400 juta peningkatan penjualan dari pelanggan baru dan Rp 600 juta dari pelanggan lama
2. Mengurangi biaya operasional untuk menangani komplain dari pelanggan
   - Rp 100 juta pengurangan tahunan biaya telepon untuk menangani pelanggan

Studi Kelayakan Sistem Penjualan Musik Online

Margaret Mooney dan Alec Adams membuat studi kelayakan untuk pengembangan Sistem Penjualan Musik Online

Kelayakan Teknis

Sistem penjualan musik online layak secara teknis, meskipun memiliki beberapa risiko.

Risiko Berhubungan dengan **Kefamilieran dengan Aplikasi**: Resiko Tinggi

- Divisi Marketing tidak memiliki pengalaman menggunakan sistem penjualan online
- Divisi IT memiliki pemahaman yang baik tentang sistem penjualan offline, akan tetapi tidak berpengalaman mengembangkan sistem penjualan musik online

Risiko Berhubungan dengan **Kefamilieran dengan Teknologi**: Resiko Sedang

- Divisi IT tidak menguasai masalah infrastruktur dan ISP, tetapi akan menyewa konsultan
- Divisi IT cukup familier dengan framework dan IDE yang akan digunakan
- Divisi Marketing tidak memiliki pengalaman menggunakan teknologi Web

Risiko berhubungan dengan **Ukuran Project**: Risiko Rendah

- Perusahaan memiliki total 30 orang pengembang
- Project dikerjakan oleh 5 orang pengembang dengan estimasi waktu 6 bulan

Kompatibilitas dengan sistem dan infrastruktur yang ada: Risiko Rendah

- Sistem pemesanan yang ada sekarang menggunakan *open standard*, jadi sangat kompatibel dengan sistem penjualan berbasis web yang akan dibangun

Kelayakan Ekonomi

Cost benefit analysis telah dilakukan. Sistem Penjualan musik online memiliki peluang yang baik untuk bisa meningkatkan pendapatan perusahaan.

- Return on Investment (ROI) setelah 3 tahun: **31%**
- Break-even point (BEP): **2.25 tahun**
- Total keuntungan setelah 3 tahun: **Rp. 503.559.986,-**

Kelayakan Organisasi

- Secara organisasi, resikonya rendah. Tujuan dari pengembangan sistem penjualan musik online adalah meningkatkan penjualan perusahaan. Dan ini selaras dengan KPI marketing yang ke arah peningkatan kuantitas penjualan
- Project champion dari pengembangan sistem penjualan musik online ini adalah Margaret Mooney, Vice President of Marketing

	2019	2020	2021
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:	**0**	**978,996,084**	**2,070,501,152**
Honor Tim (Analysis, Design and Implementation)	360,000,000	0	0
Honor Konsultan	90,000,000	0	0
Total Development Costs:	**450,000,000**	**0**	**0**
Honor Pengelola Web	60,000,000	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**	**669,811,321**
Cumulative NPV:	**-735,849,057**	**-166,251,335**	**503,559,986**
Return on Investment (ROI) di Tahun 3: 30.70%	**-100.00%**	**-0.145166304**	**0.306932213**
Break-even Point (BEP): 2.25 tahun			**2.248206218**

Analysis

1. Pengumpulan dan analisis kebutuhan (Requirements):
   - Siapa yang menggunakan software?
   - Apa yang dilakukan oleh software?
   - Kapan software digunakan?
2. Investigasi Software yang Ada (Baseline)
3. Identifikasi Peluang untuk Perbaikan
4. Kembangan konsep untuk software baru

(**System Specification**)

Design

1. **Program Design** (UML Diagrams)
   - Software seperti apa yang ingin dibuat
   - Komposisi dan arsitektur dari software

2. **User Interface Design**
   - Bagaimana pengguna berinteraksi dengan software
   - Pahami form/laporan yang digunakan oleh perusahaan

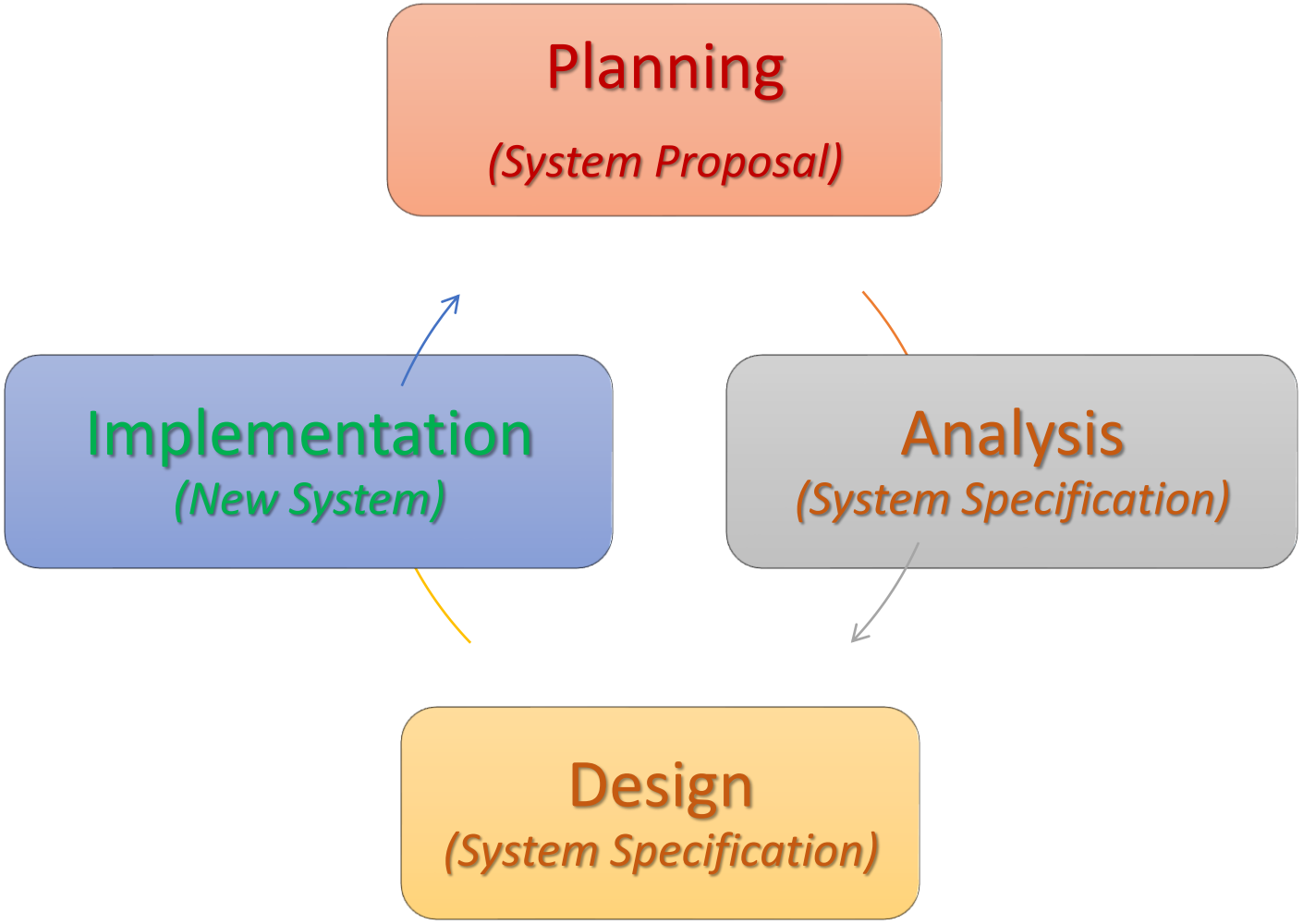
3. **Data Design**
   - Data apa yang akan disimpan
   - Format data yang disimpan
   - Dimana data akan disimpan

(**System Specification**)

Implementation

1. Konstruksi Software
   - Pembuatan kode program
2. Pengujian Software
   1. Unit Testing
   2. Integration Testing
   3. System Testing
   4. User Acceptance Test
3. Dokumentasi
   1. User Documentation
   2. System Documentation
4. Installation
   - Software lama dimatikan
   - Software baru diaktifkan (instalasi)

Siklus Pengembangan Software



(Tilley, 2012) *(Dennis, 2016)* *(Valacich, 2017)*

1.3 Metodologi Pengembangan Software (*Model Process*)

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)*

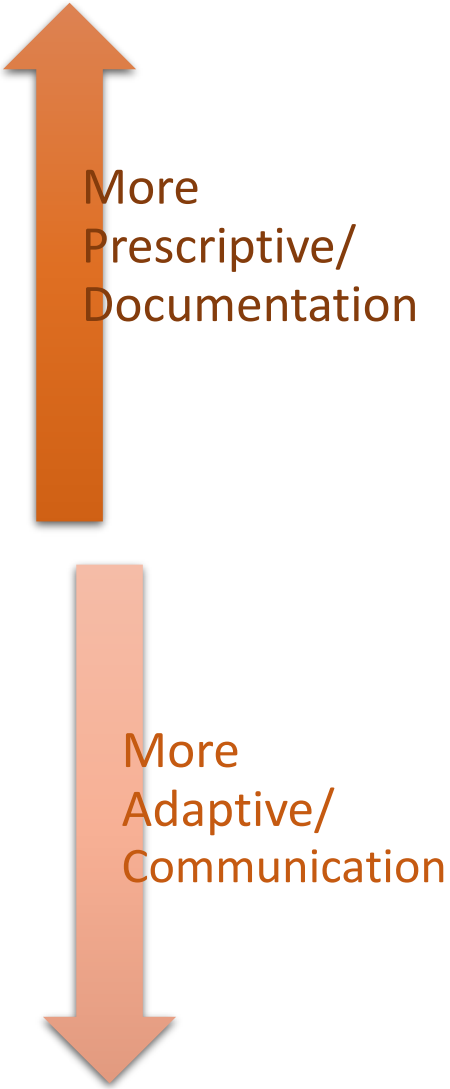
- A distinct set of activities, actions, tasks, milestones, and work products required to engineer high quality software *(Pressman, 2015)*

Major Software Development Methodologies

1. **Structured Design**
   (Prescriptive) (1967-)
   - Waterfall method
   - Parallel development

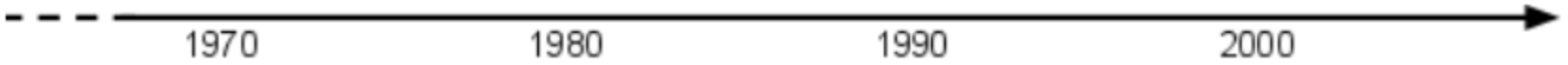
2. **Rapid Application Development**
   (Iterative) (1985-)
   - Phased Development
   - Prototyping

3. **Agile Development**
   (Adaptive) (1995-)
   - Extreme Programming (XP)
   - Scrum



Compiled from (Dennis, Wixom and Tegarden, 2016)

Software Development Methodology
Timeline



Waterfall

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

Spiral, RAD, RUP

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

Scrum, XP

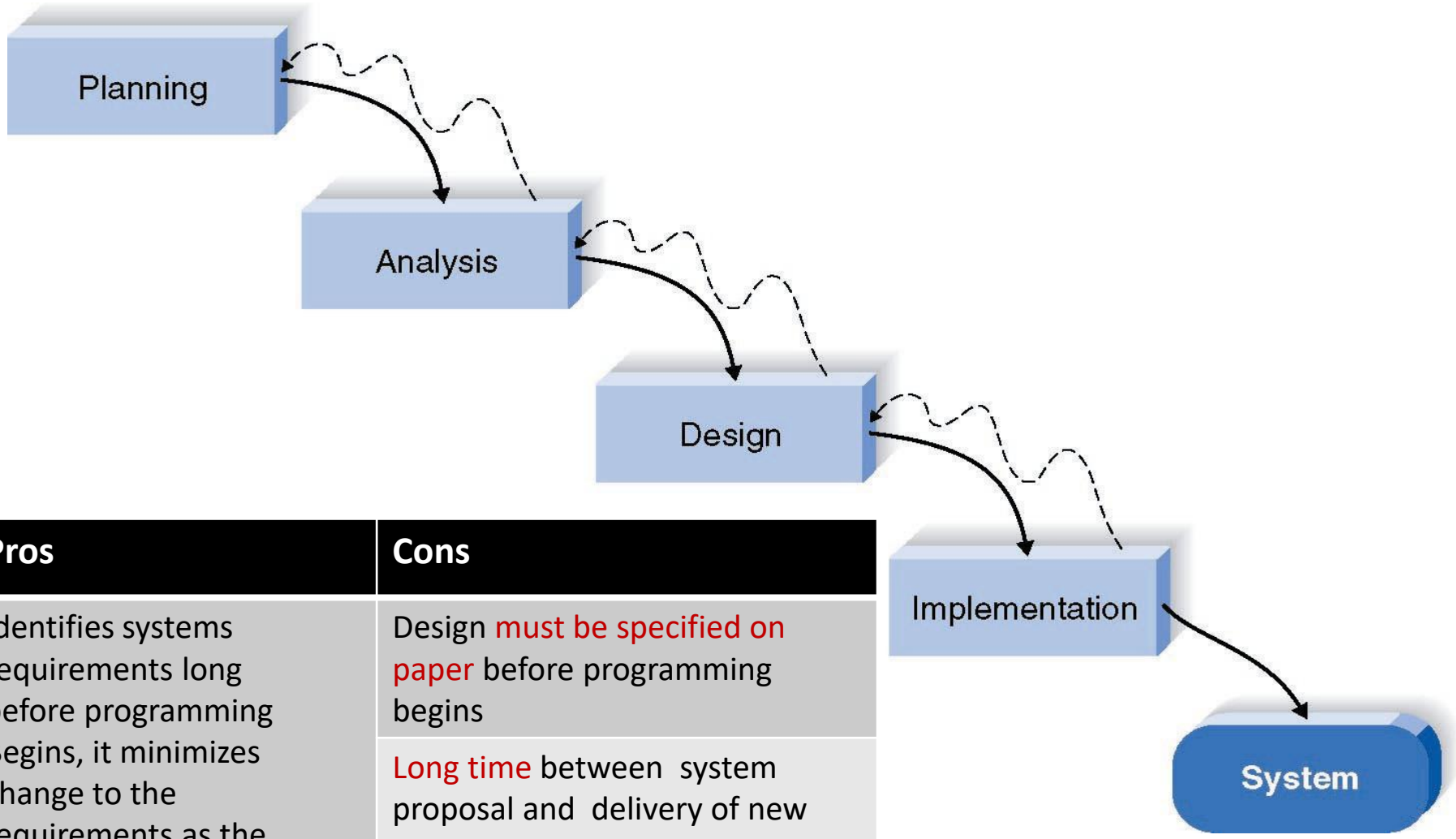
Adaptive: iterative, self-organizing teams, value driven, transparent

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	Cons
Identifies systems requirements long before programming Begins, it minimizes change to the requirements as the project proceed (mature)	Design must be specified on paper before programming begins
	Long time between system proposal and delivery of new system
	Rework is very hard

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

Business Requirement

Sistem yang menduku

1. Fitur Pencarian Pro
2. Fitur Pencarian Tok
3. Fitur Pemesanan Pr
4. Fi

Busi

Intar

Tang

Studi Kelayakan Sistem Penjualan Musik Online

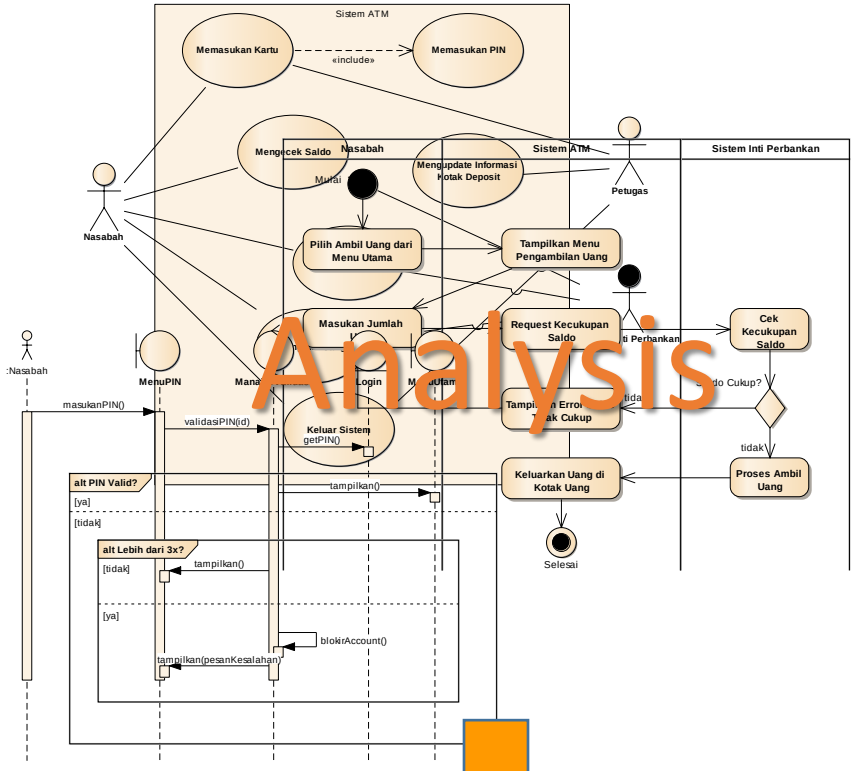
Margaret Mooney dan Alec Adams membuat studi kelayakan untuk pengembangan Sistem Penjualan Musik Online

Kelayakan Teknis

Sistem penjualan musik online layak secara teknis, meskipun memiliki beberapa risiko.

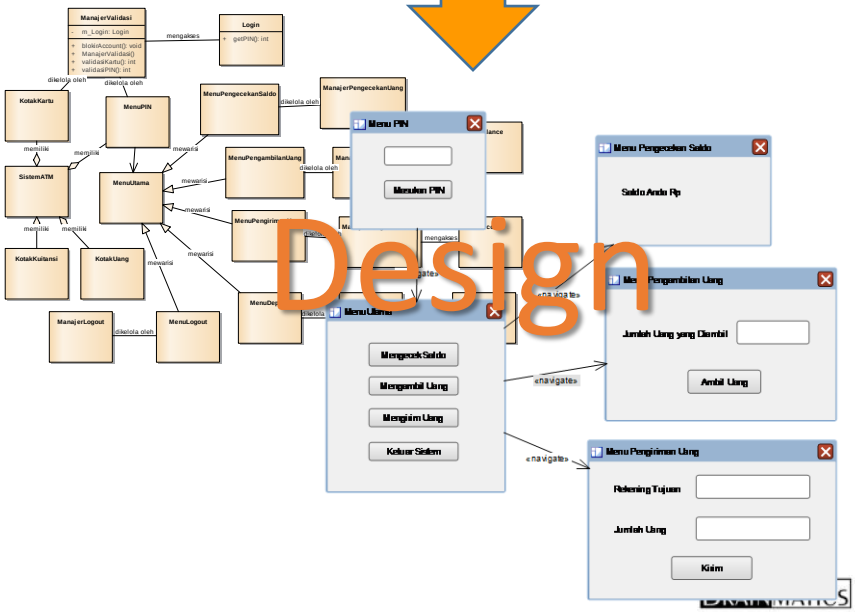
	2016	2017	2018
Peningkatan penjualan dari pelanggan baru	0	400,000,000	500,000,000
Peningkatan penjualan dari pelanggan lama	0	600,000,000	700,000,000
Pengurangan biaya operasional dan telepon	0	100,000,000	100,000,000
Total Benefits:	**0**	**1,100,000,000**	**1,300,000,000**
PV of Benefits:	System Requ	8,99 084	091 [illegible]
PV of All Benefits:	ness Value)	9 8,99 084	070,501,1 2
Honor Tim (Planning, Analysis, Design and Implementation)	60,00 ,000	0	
Honor Konsultan Infrastruktur Internet	00,000,000	0	
Total Development Costs:	**450,000,000**	**0**	**0**
Honor Pengelola Web	60,000,000	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**	**569,507,722**	**596,129,691**
Cumulative NPV:	**-735,849,057**	**-166,251,435**	**429,878,356**
Return on Investment (ROI) di Tahun 3: 26.2%	**429,878,356/1,640,622,796**		**0.262021445**
Break-even Point (BEP): 2.28 tahun	**2 + (596,129,691-429,878,356)/ 596,129,691**		**2.278884507**

Planning



Analysis

Implementation

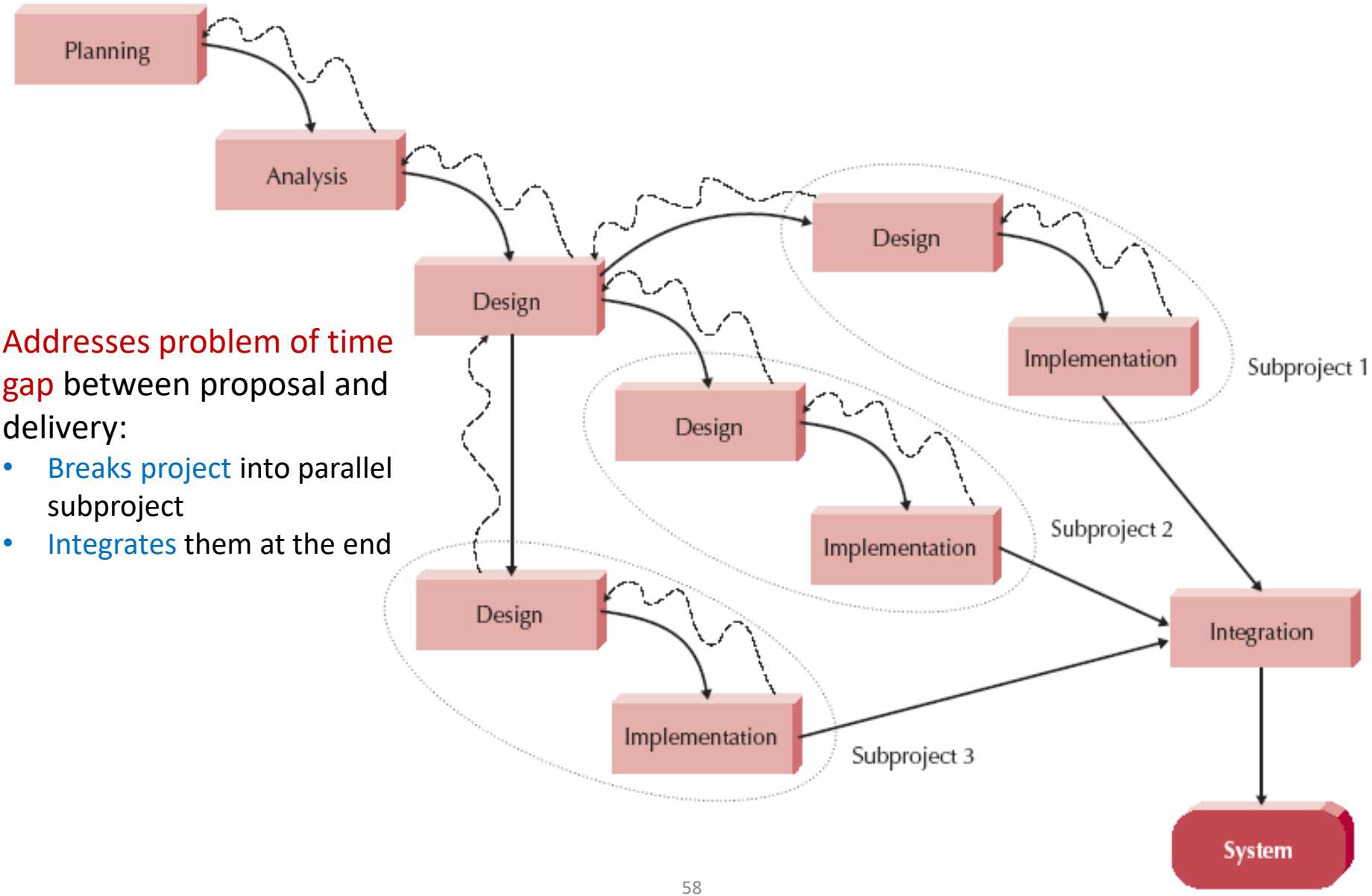


Design

Parallel Development

Addresses problem of time gap between proposal and delivery:

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

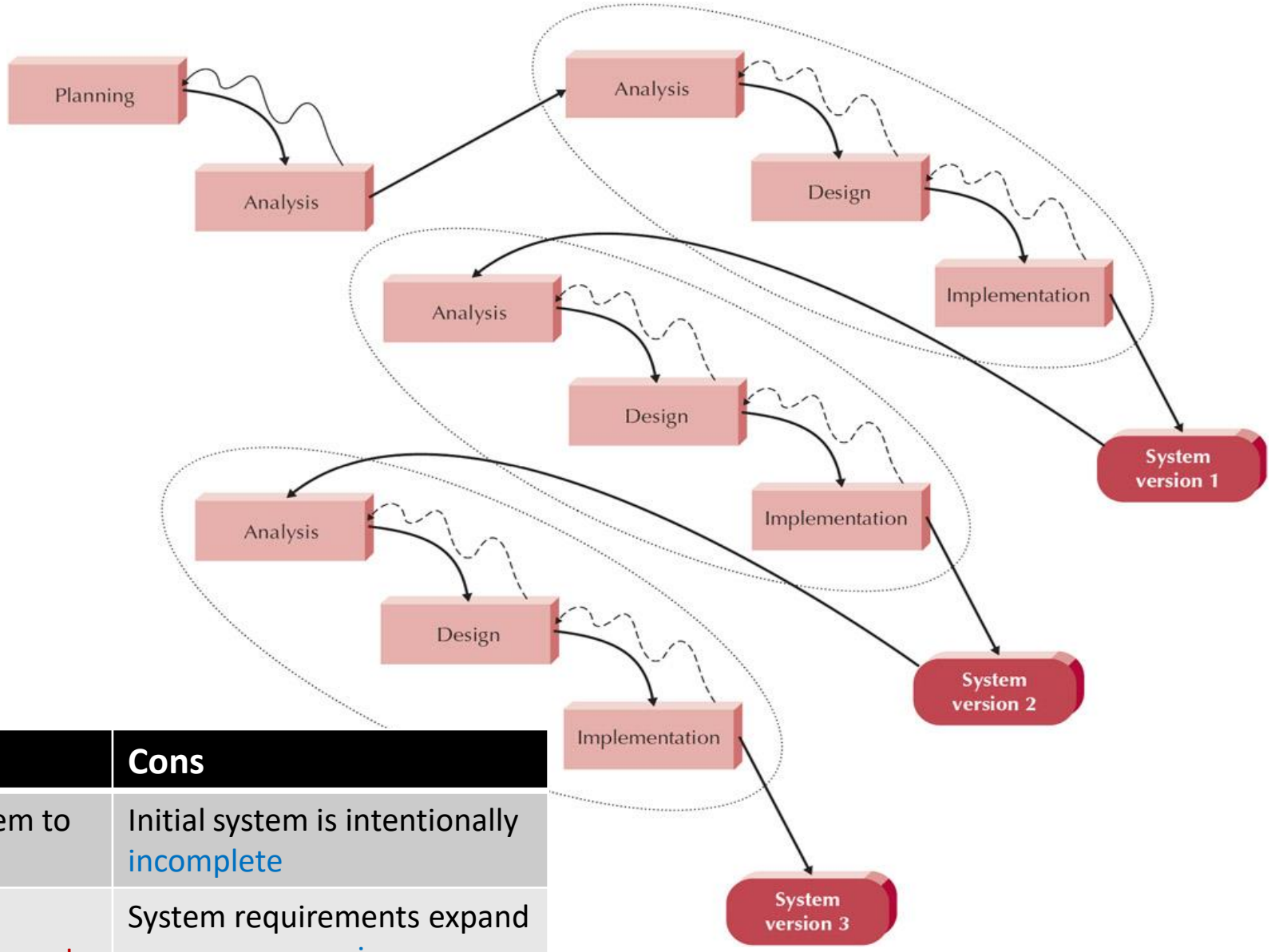


Rapid Application Development (RAD)

- 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

Margaret Mooney dan Alec Adams membuat studi kelayakan untuk pengembangan Sistem Penjualan Musik Online

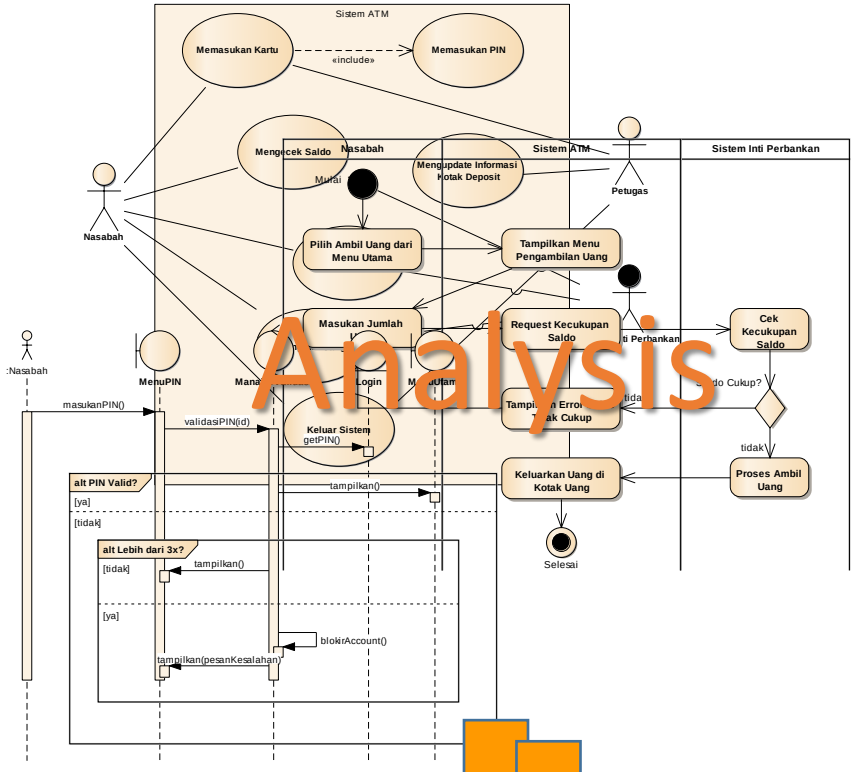
Kelayakan Teknis

Sistem penjualan musik online layak secara teknis, meskipun memiliki beberapa risiko.

	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:			
PV of All Benefits:			
Honor Tim (Planning, Analysis, Design and Implementation)			
Honor Konsultan Infrastruktur Internet			
Total Development Costs:	**450,000,000**	**0**	**0**
Honor Pengelola Web	60,000,000	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**	**569,507,722**	**596,129,691**
Cumulative NPV:	**-735,849,057**	**-166,251,335**	**429,878,356**
Return on Investment (ROI) di Tahun 3: 26.2%	**429,878,356/1,640,622,796**		**0.262021445**
Break-even Point (BEP): 2.28 tahun	**2 + (596,129,691-429,878,356)/ 596,129,691**		**2.278884507**

Planning

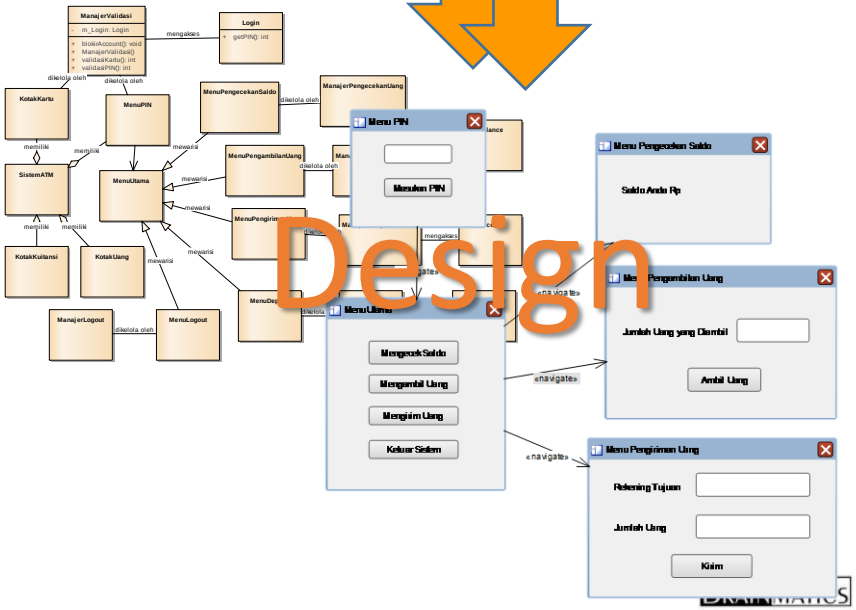
Analysis



Versi 1

Versi 2

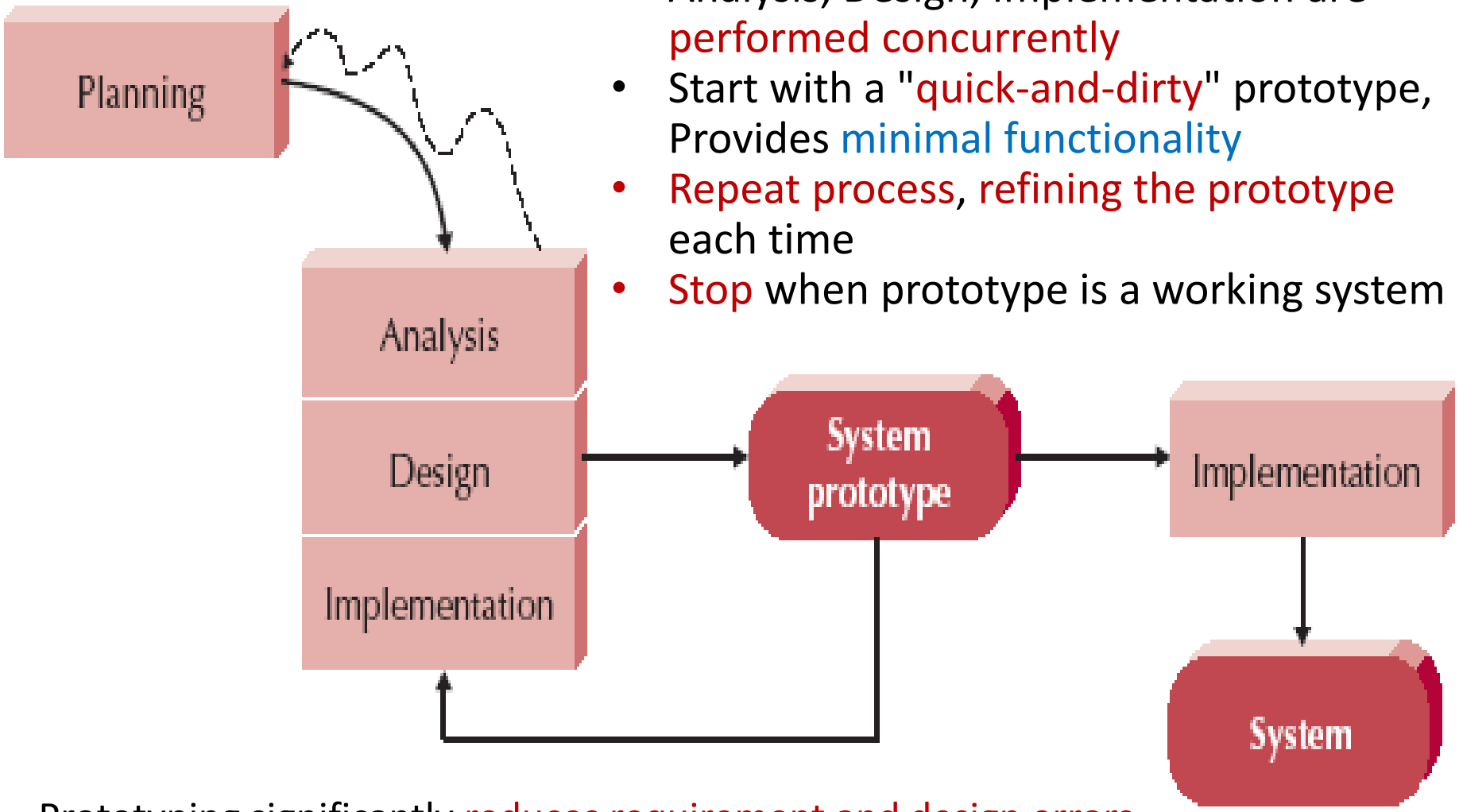
Implementation



Design

RAD: Prototyping

- Analysis, Design, Implementation are performed concurrently
- Start with a "quick-and-dirty" prototype, Provides minimal functionality
- Repeat process, refining the prototype each time
- Stop when prototype is a working system



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

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
	2. M

Business Requirement
Sistem yang menduku
1. Fitur Pencarian Pro
2. Fitur Pencarian Tok
3. Fitur Pemesanan Pr
4. Fi

Busi
Intar

Tang

Studi Kelayakan Sistem Penjualan Musik Online

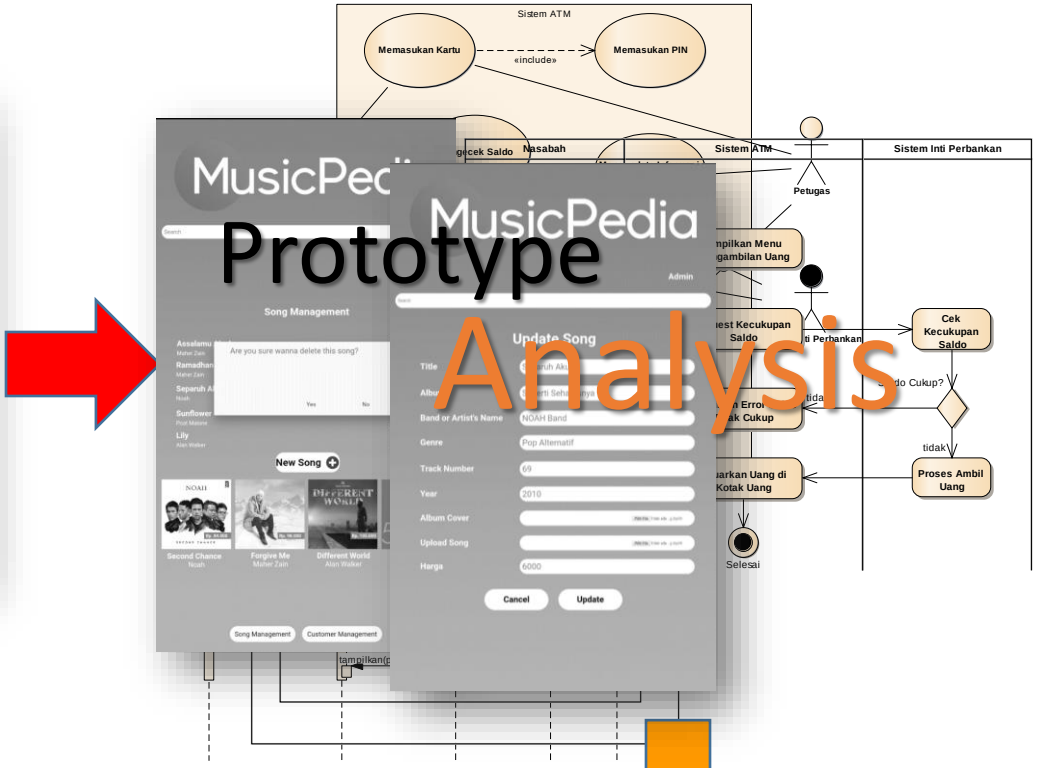
Margaret Mooney dan Alec Adams membuat studi kelayakan untuk pengembangan Sistem Penjualan Musik Online

Kelayakan Teknis

Sistem penjualan musik online layak secara teknis, meskipun memiliki beberapa risiko.

	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:	**Syste**	**8,99 084**	**091 [illegible]**
PV of All Benefits:	**(Mess alue)**	**8,99 084**	**070,501,1 2**
Honor Tim (Planning, A ysis, Design d Implementation)	60,00 ,000	0	
Honor Konsultan Infra truktur Internet	00,000,000	0	
Total Development Costs:	**450,000,000**	**0**	**0**
Honor Pengelola Web	60,000,000	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**	**569,507,722**	**596,129,691**
Cumulative NPV:	**-735,849,057**	**-166,251,335**	**429,878,356**
Return on Investment (ROI) di Tahun 3: 26.2%	**429,878,356/1,640,622,796**		**0.262021445**
Break-even Point (BEP): 2.28 tahun	**2 + (596,129,691-429,878,356)/ 596,129,691**		**2.278884507**

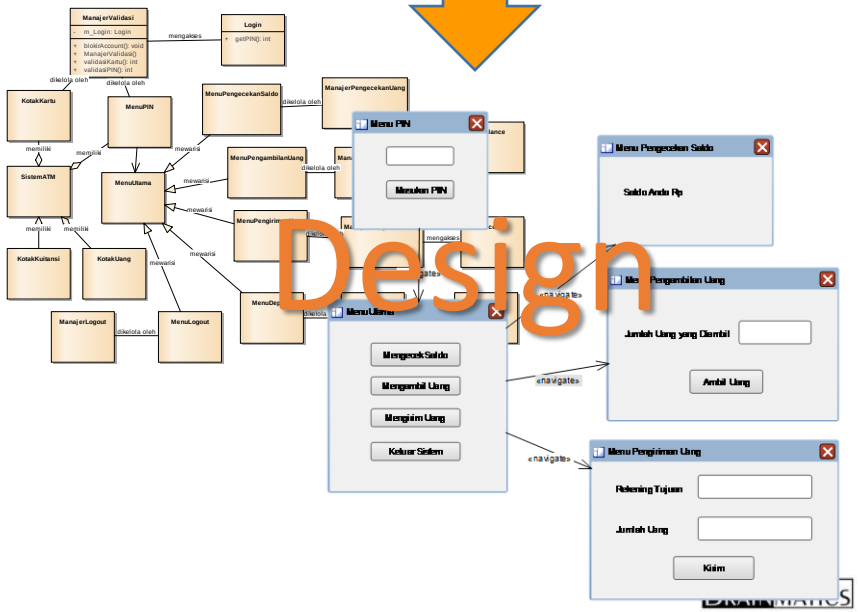
Planning



Prototype

Analysis

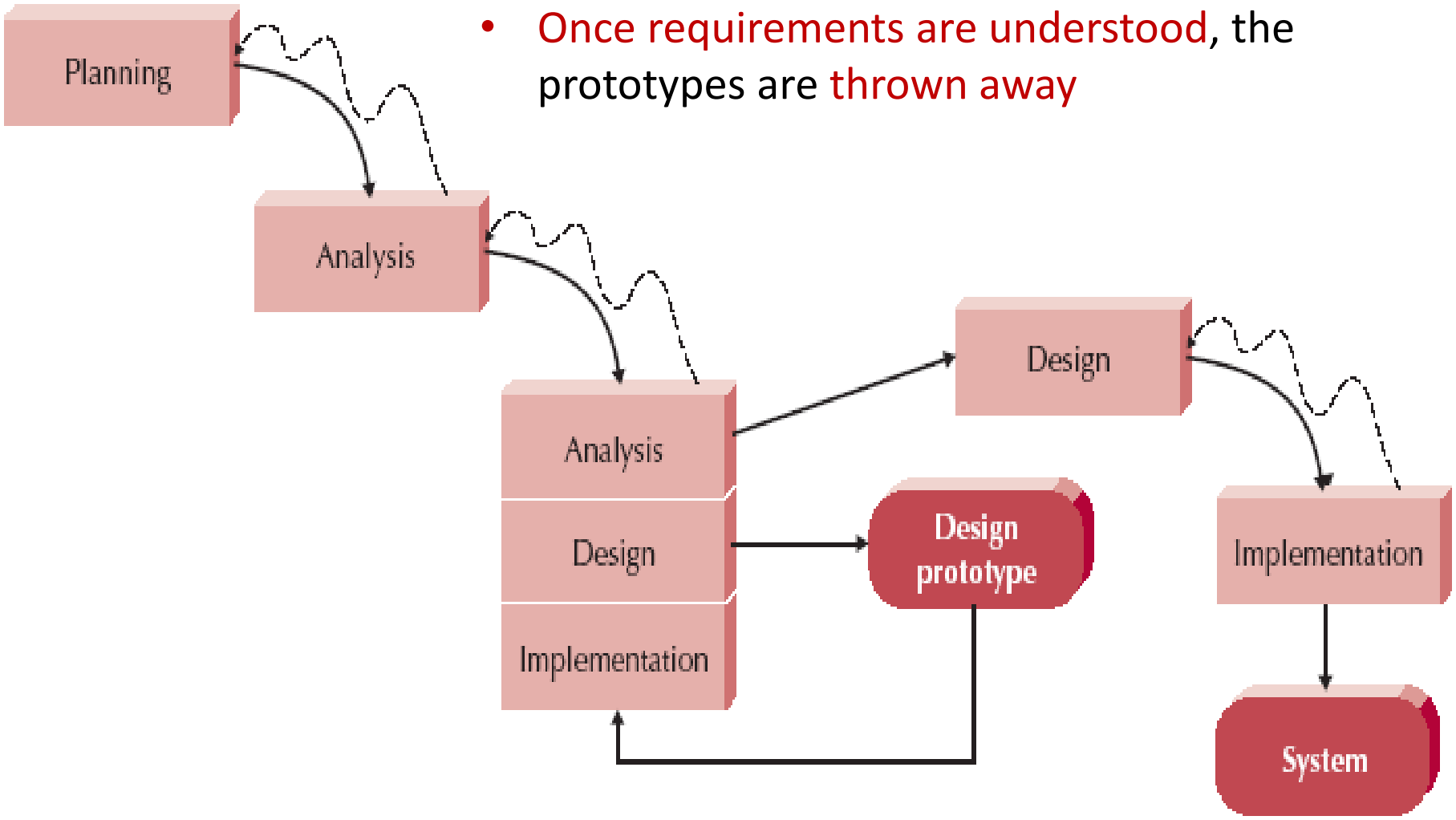
Implementation



Design

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



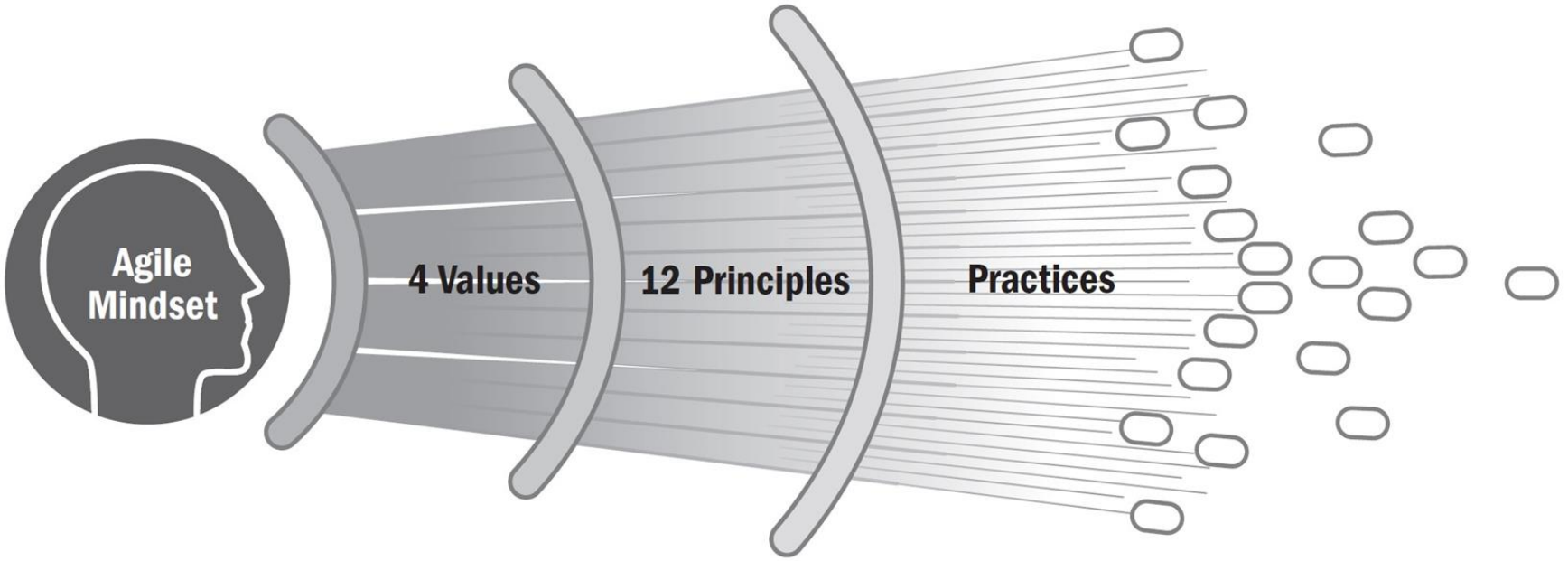
Agile Development

- Menggunakan beberapa aturan yang mudah dipahami dan diikuti

- Mempercepat proses SDLC
  - Mengurangi pemodelan dan dokumentasi
  - Mengembangakan 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

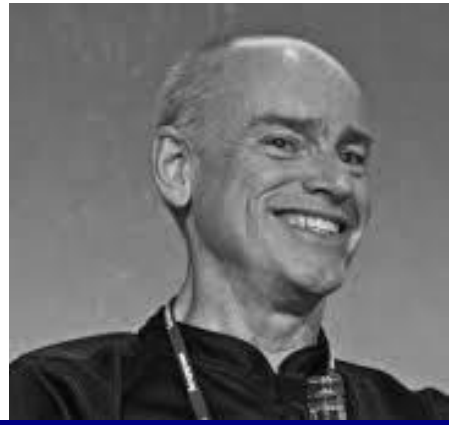
Responding to change over

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

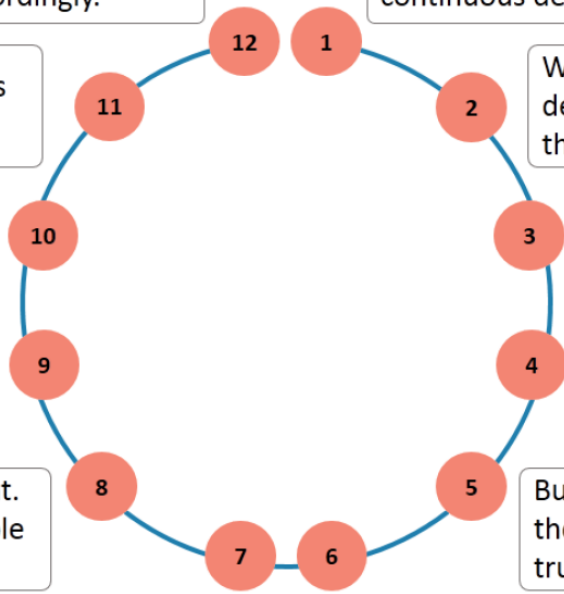
Jeff Sutherland



Ken Schwaber

12 Agile Principles (Manifesto)

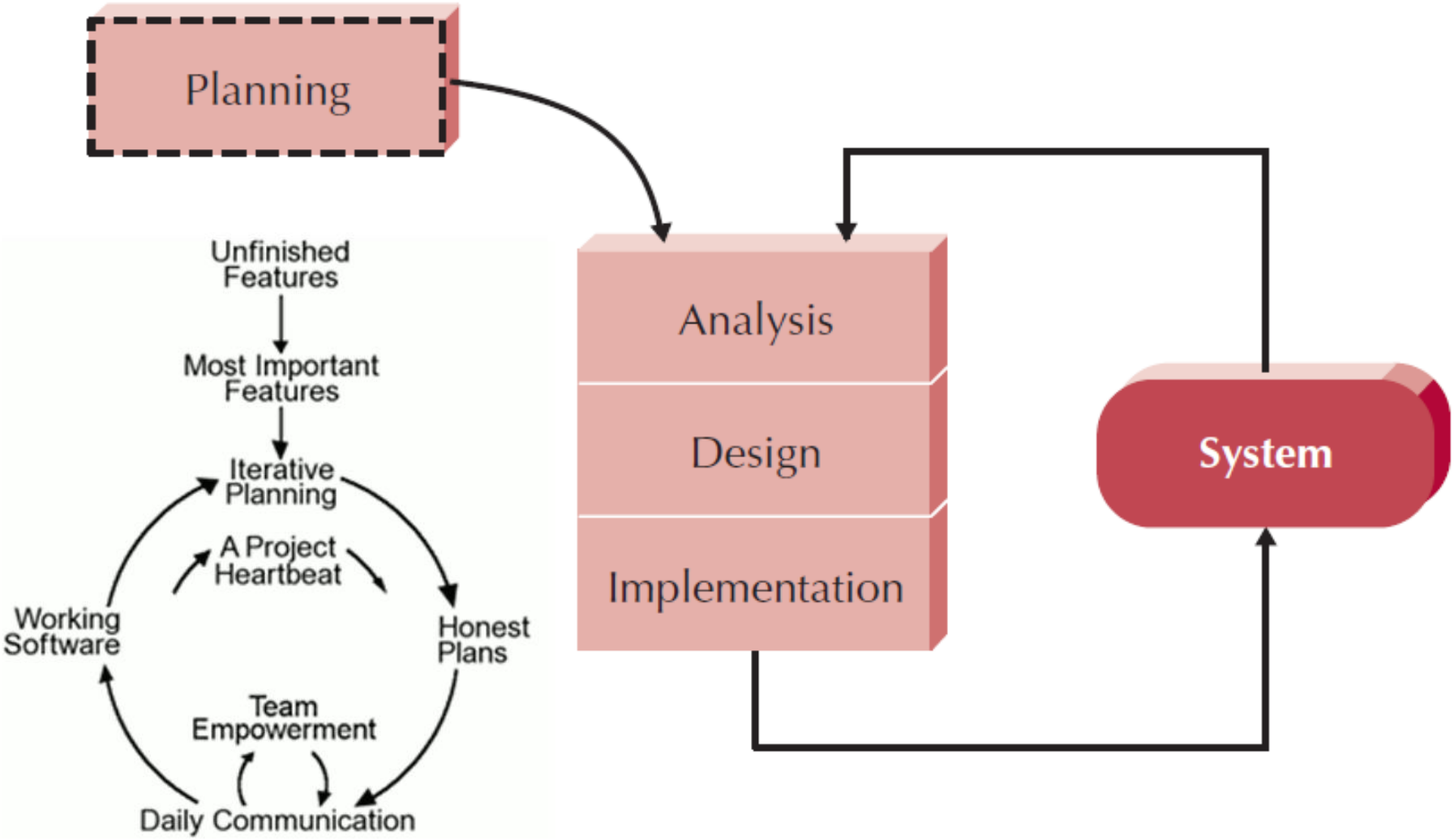
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from every couple of weeks to couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity, the art of maximizing the amount of work not done, is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.



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)

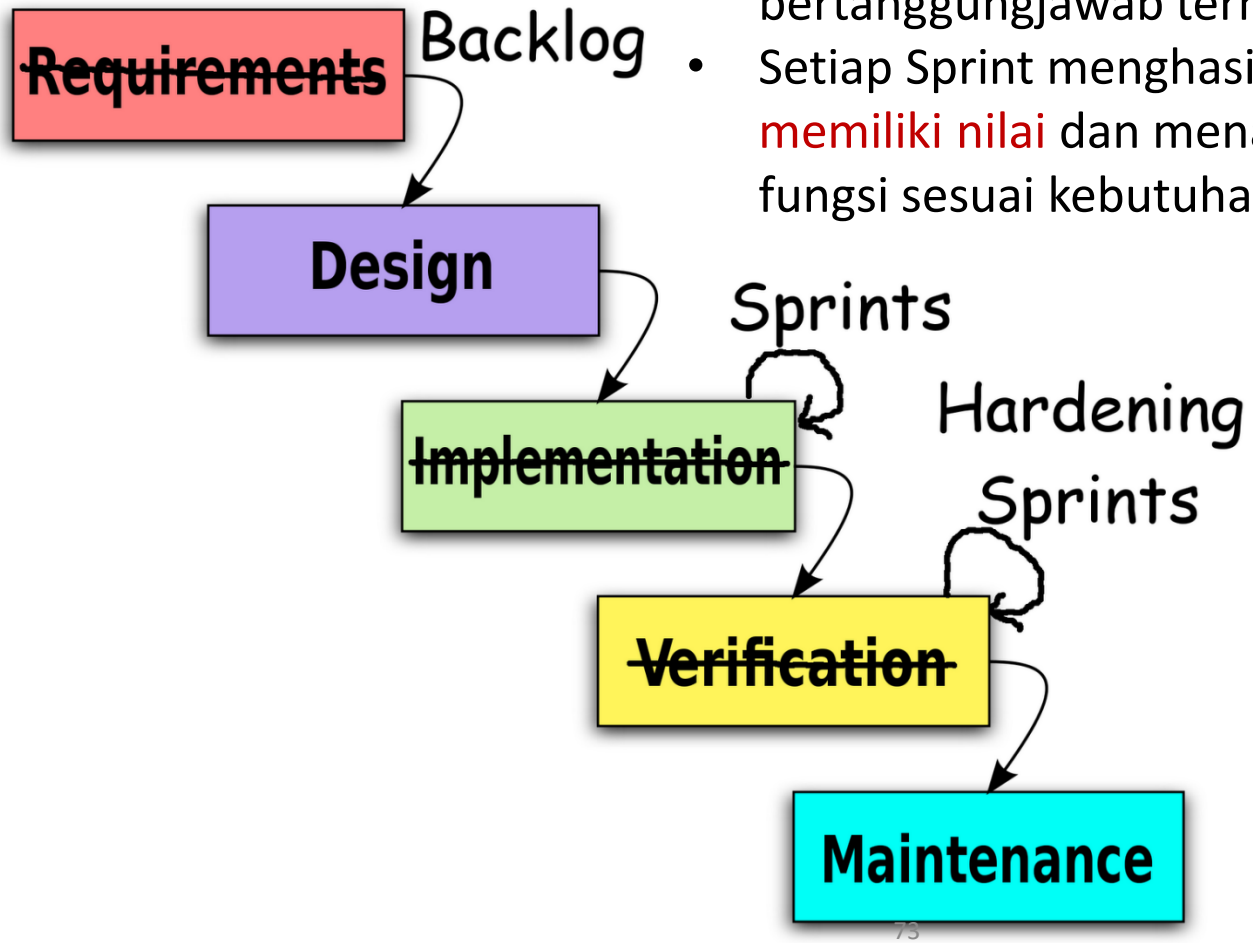


XP Values

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

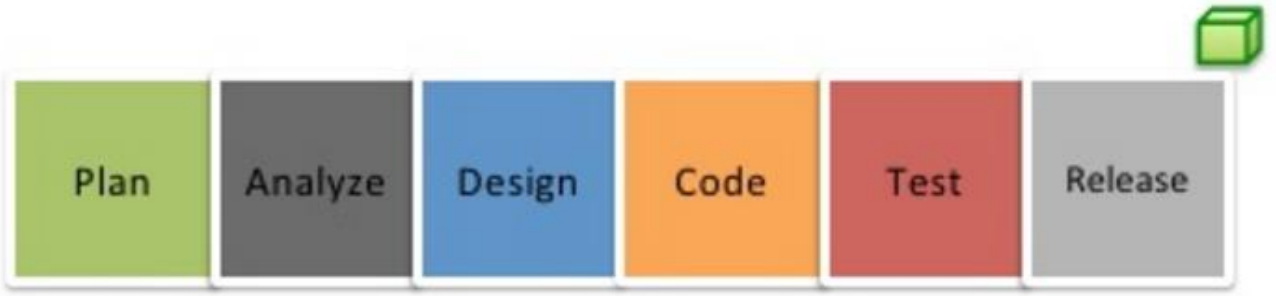
- Flatter World in Software Engineering
- Introduced Scrum for software development in 1993

Waterfall vs Scrum

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

Waterfall (Defined)

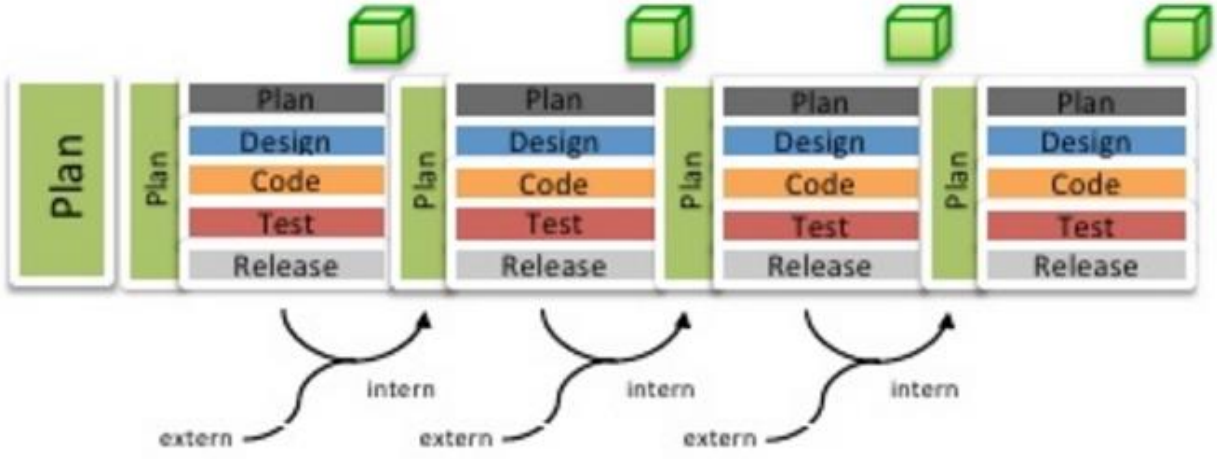
Plan for the entire project up-front



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

Scrum (Empirical)

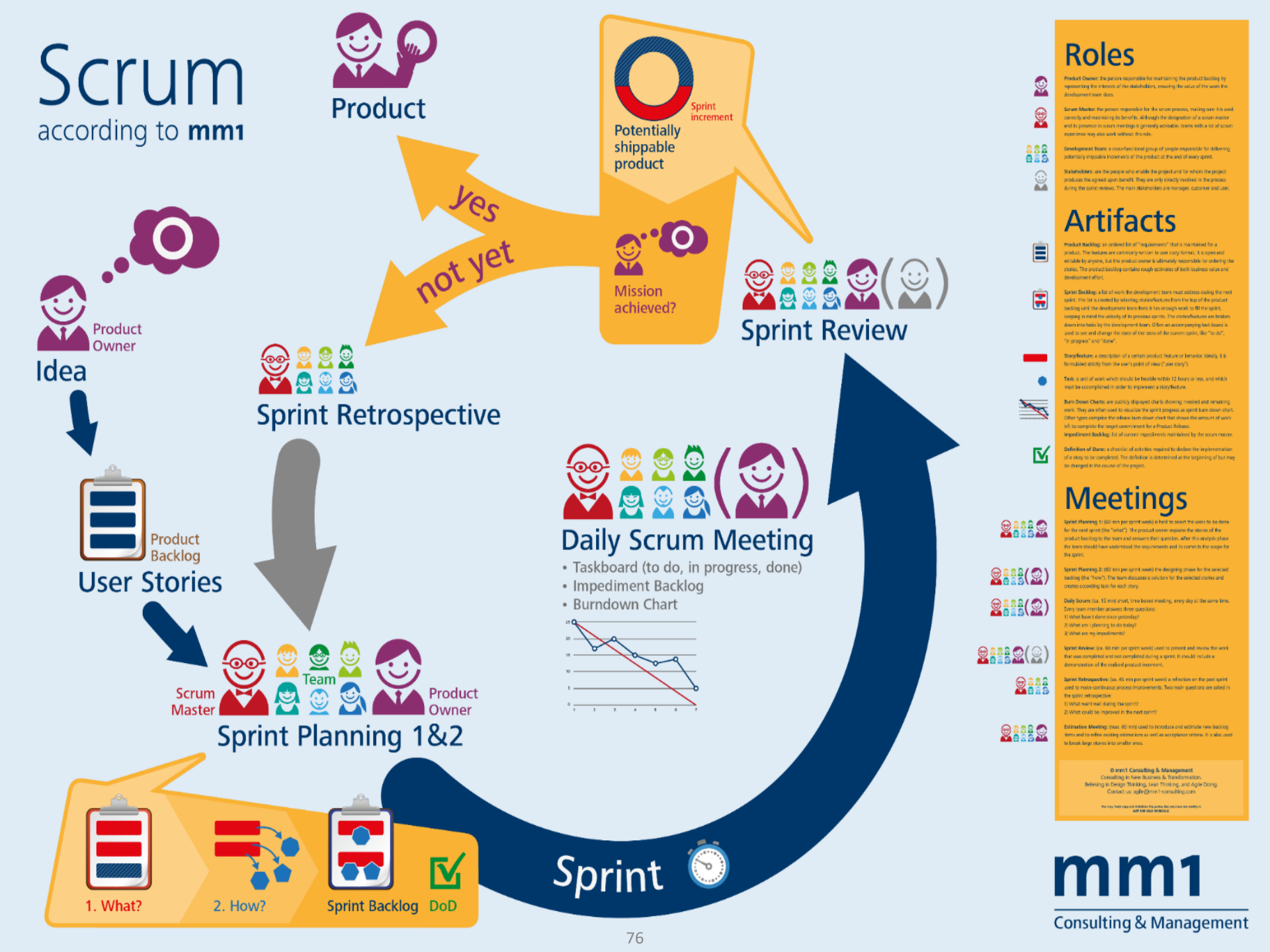
Plan a little for the entire project and then a little for each Sprint



(source: ADM)

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

Scrum
according to **mm1**



Roles

Product Owner: the person responsible for maintaining the product backlog by representing the interests of the stakeholders, ensuring the value of the work the development team does.

Scrum Master: the person responsible for the scrum process, making sure it is used correctly and maximizing its benefits. Although the designation of a scrum master and its presence in scrum meetings is generally advisable, teams with a lot of scrum experience may also work without this role.

Development Team: a cross-functional group of people responsible for delivering potentially shippable increments of the product at the end of every sprint.

Stakeholders: are the people who enable the project and for whom the project produces the agreed-upon benefit. They are only directly involved in the process during the sprint reviews. The main stakeholders are managers, customer and user.

Artifacts

Product Backlog: an ordered list of "requirements" that is maintained for a product. The features are commonly written in user story format. It is open and editable by anyone, but the product owner is ultimately responsible for ordering the stories. The product backlog contains rough estimates of both business value and development effort.

Sprint Backlog: a list of work the development team must address during the next sprint. The list is created by selecting stories/features from the top of the product backlog until the development team feels it has enough work to fill the sprint, keeping in mind the velocity of its previous sprints. The stories/features are broken down into tasks by the development team. Often an accompanying task board is used to see and change the state of the tasks of the current sprint, like "to do", "in progress" and "done".

Story/feature: a description of a certain product feature or behavior. Ideally, it is formulated strictly from the user's point of view ("user story").

Task: a unit of work which should be feasible within 12 hours or less, and which must be accomplished in order to implement a story/feature.

Burn Down Charts: are publicly displayed charts showing invested and remaining work. They are often used to visualize the sprint progress as sprint burn down chart. Other types comprise the release burn down chart that shows the amount of work left to complete the target commitment for a Product Release.

Impediment Backlog: list of current impediments maintained by the scrum master.

Definition of Done: a checklist of activities required to declare the implementation of a story to be completed. The definition is determined at the beginning of but may be changed in the course of the project.

Meetings

Sprint Planning 1: (60 min per sprint week) it held to select the work to be done for the next sprint (the "what"). the product owner explains the stories of the product backlog to the team and answers their question. After this analysis phase the team should have understood the requirements and its comes in the scope for the sprint.

Sprint Planning 2: (60 min per sprint week) the designing phase for the selected backlog (the "how"). The team discusses a solution for the selected stories and creates according task for each story.

Daily Scrum: (ca. 15 min) short, time boxed meeting, every day at the same time. Every team member answers three questions:
1) What have I done since yesterday?
2) What am I planning to do today?
3) What are my impediments?

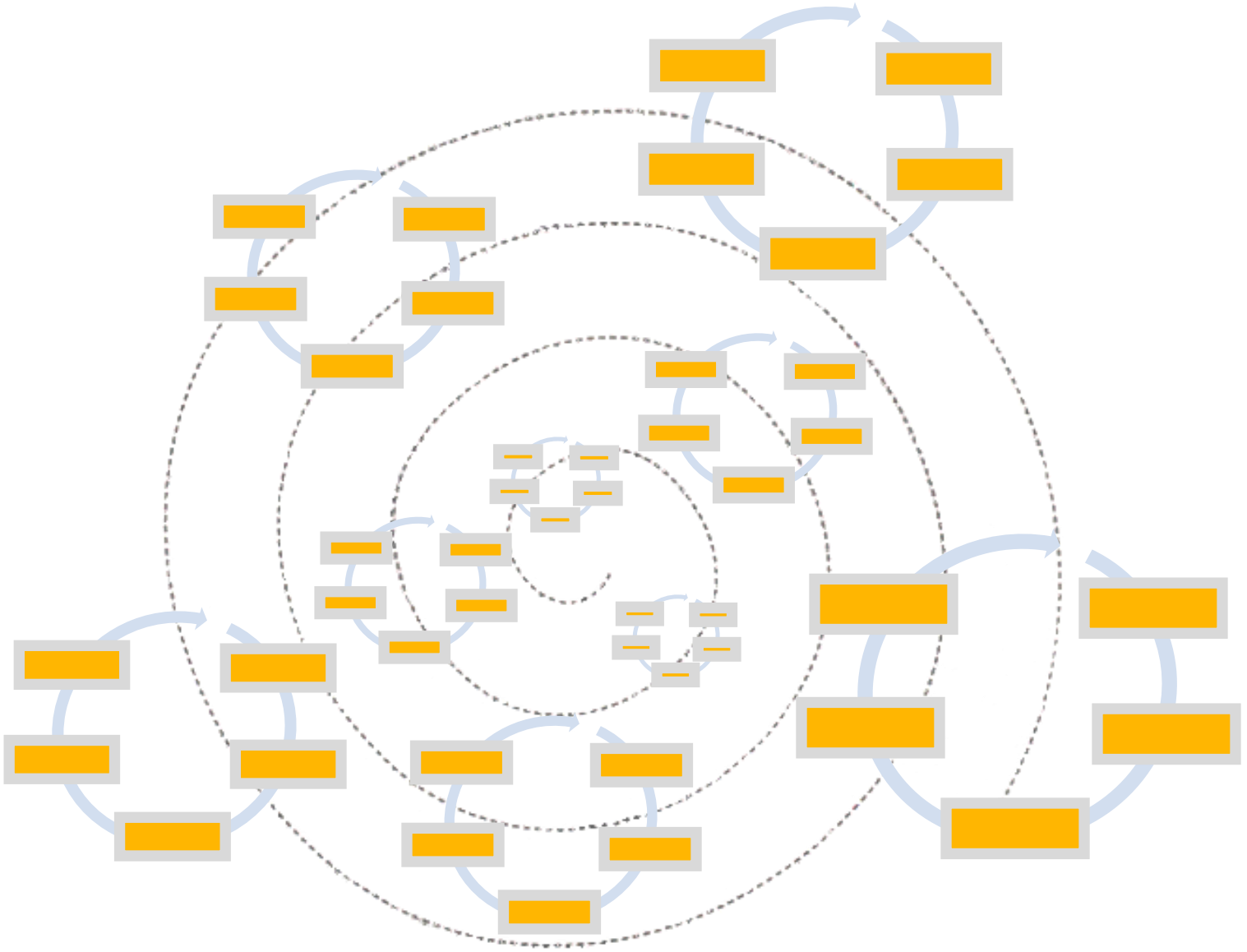
Sprint Review: (ca. 60 min per sprint week) used to present and review the work that was completed and not completed during a sprint. It should include a demonstration of the realized product increment.

Sprint Retrospective: (ca. 45 min per sprint week) a reflection on the past sprint used to make continuous process improvements. Two main questions are asked in the sprint retrospective:
1) What went well during the sprint?
2) What could be improved in the next sprint?

Estimation Meeting: (max. 60 min) used to introduce and estimate new backlog items and to refine existing estimations as well as acceptance criteria. It is also used to break large stories into smaller ones.

mm1
Consulting & Management

Scrum Helicopter View




PRIORITY
TASKS
IN-PROGRESS
DONE

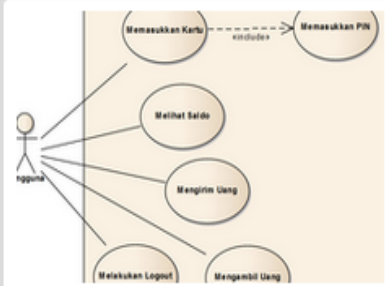
Boards

Trello

Sistem ATM

Intelligent Systems ★ Private

Product Backlog



Use Case Diagram: Sistem ATM

1

Melakukan Login

Melihat Saldo

Mengirim Uang

Mengambil Uang

Melakukan Logout

Add a card...

Sprint Backlog

MenuLogin UI Design

Card Validation

PIN Validation

MenuUtama UI Design

MenuMelihatSaldo UI Design

Balance Check Processing

MenuMengirimUang UI Design

Money Sending Processing

MenuMengambilUang UI Design

Withdrawal Processing

MenuLogout UI Design

Add a card...

To Do

Add a card...

Doing

Add a card...

Done

Add a card...

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

Margaret Mooney dan Alec Adams membuat studi kelayakan untuk pengembangan Sistem Penjualan Musik Online

Kelayakan Teknis

Sistem penjualan musik online layak secara teknis, meskipun memiliki beberapa risiko.

	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:			
PV of All Benefits:			
Honor Tim (Planning, Analysis, Design and Implementation)			
Honor Konsultan Infrastruktur Internet			
Total Development Costs:	**450,000,000**	**0**	**0**
Honor Pengelola Web	60,000,000	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**	**569,507,722**	**596,129,691**
Cumulative NPV:	**-735,849,057**	**-166,251,435**	**429,878,356**
Return on Investment (ROI) di Tahun 3: 26.2%	**429,878,356/1,640,622,796**		**0.262021445**
Break-even Point (BEP): 2.28 tahun	**2 + (596,129,691-429,878,356)/ 596,129,691**		**2.278884507**

Planning

Sprint 1

Analysis

Sprint 2

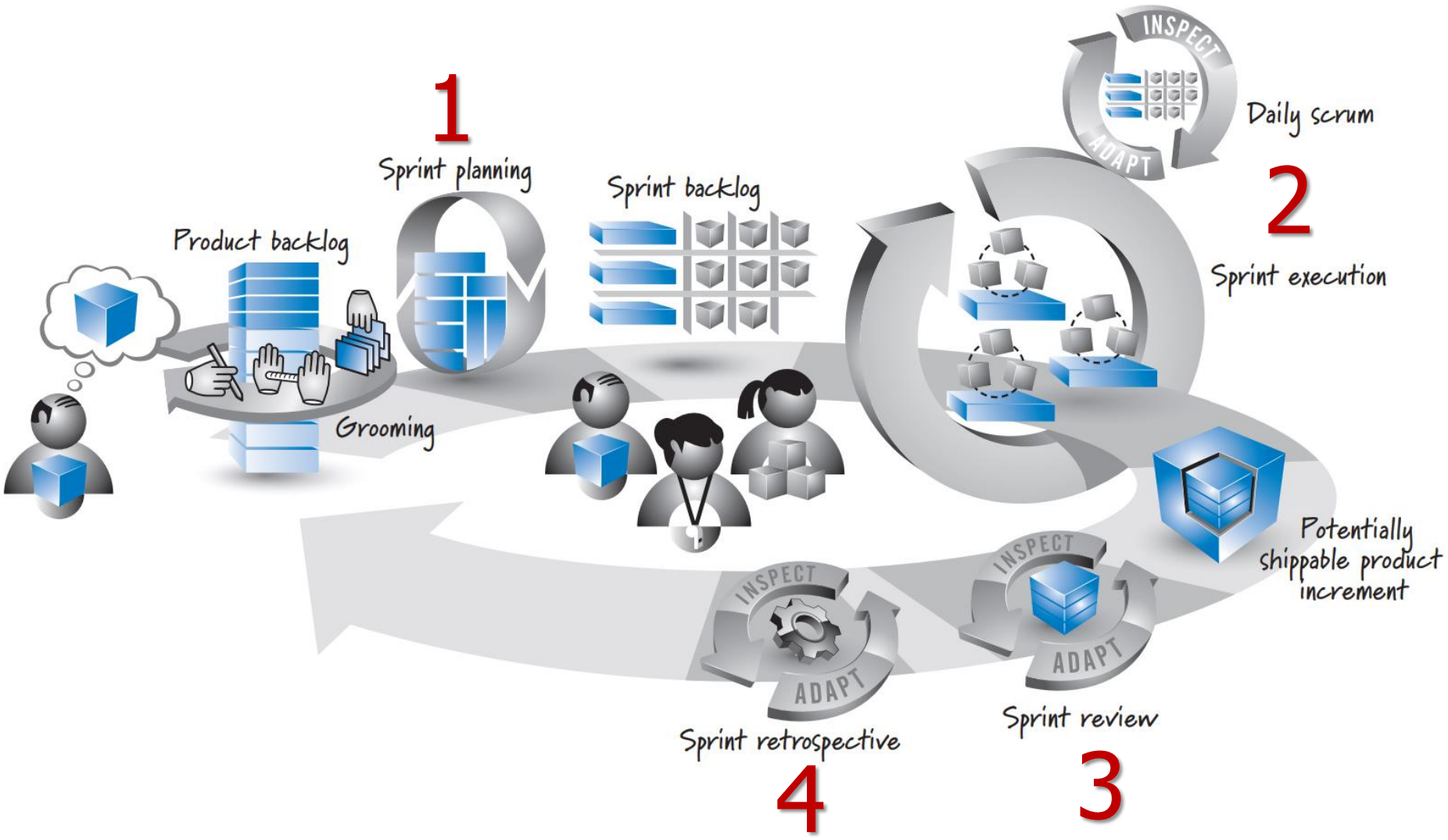
Hasil Sprint 1

Hasil Sprint 2

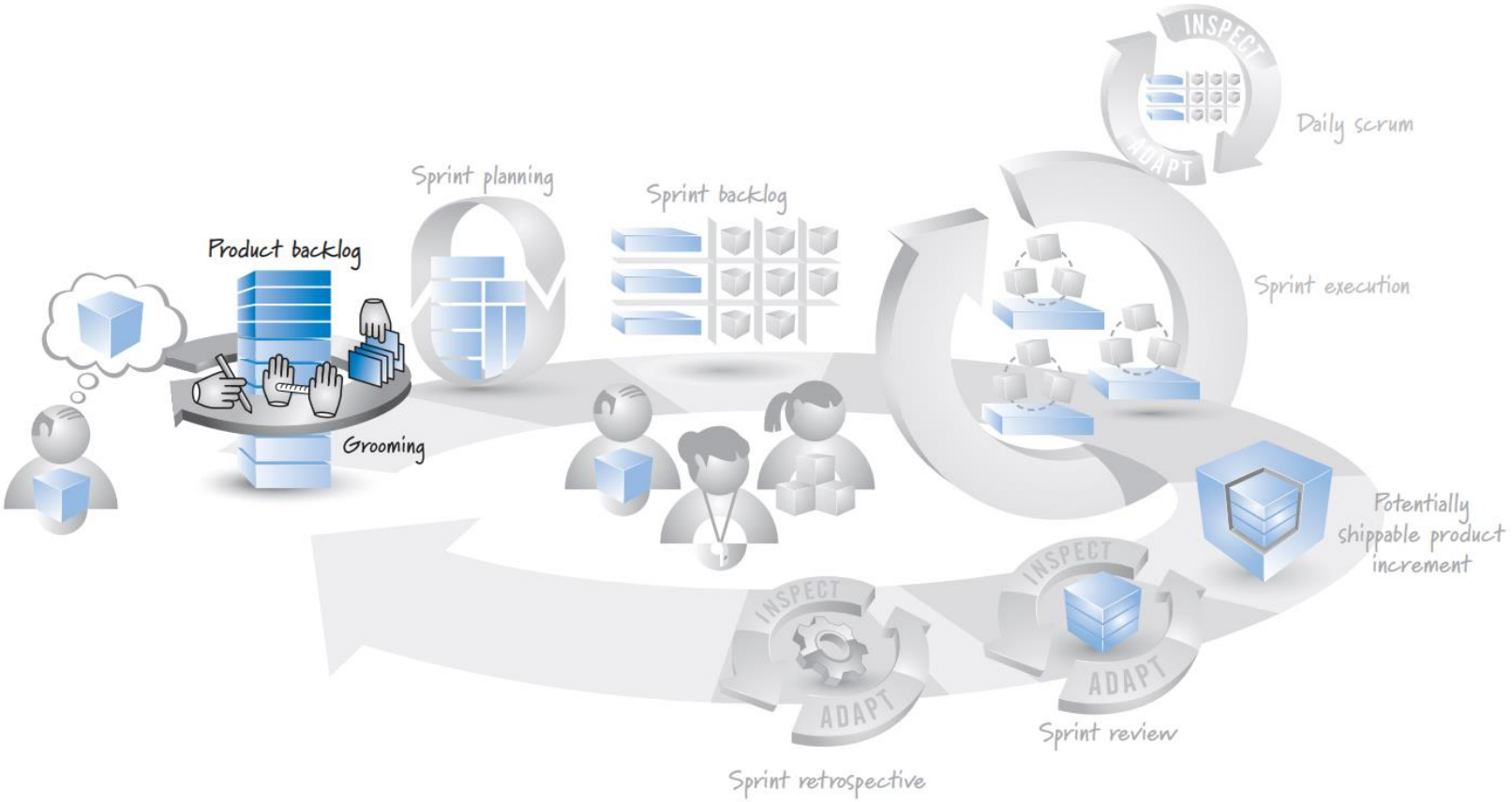
Implementation

Design

Scrum Events

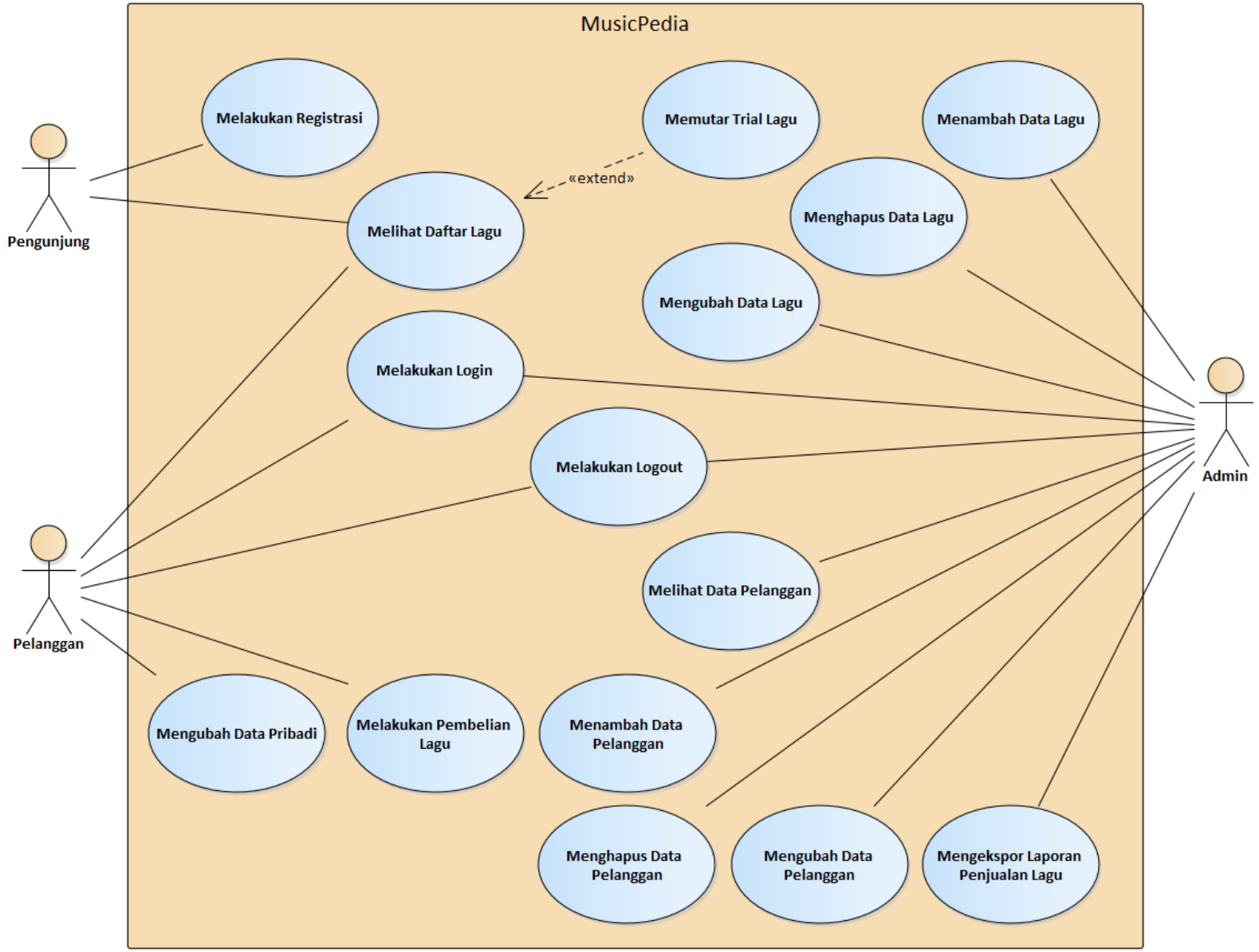


Product Backlog Grooming

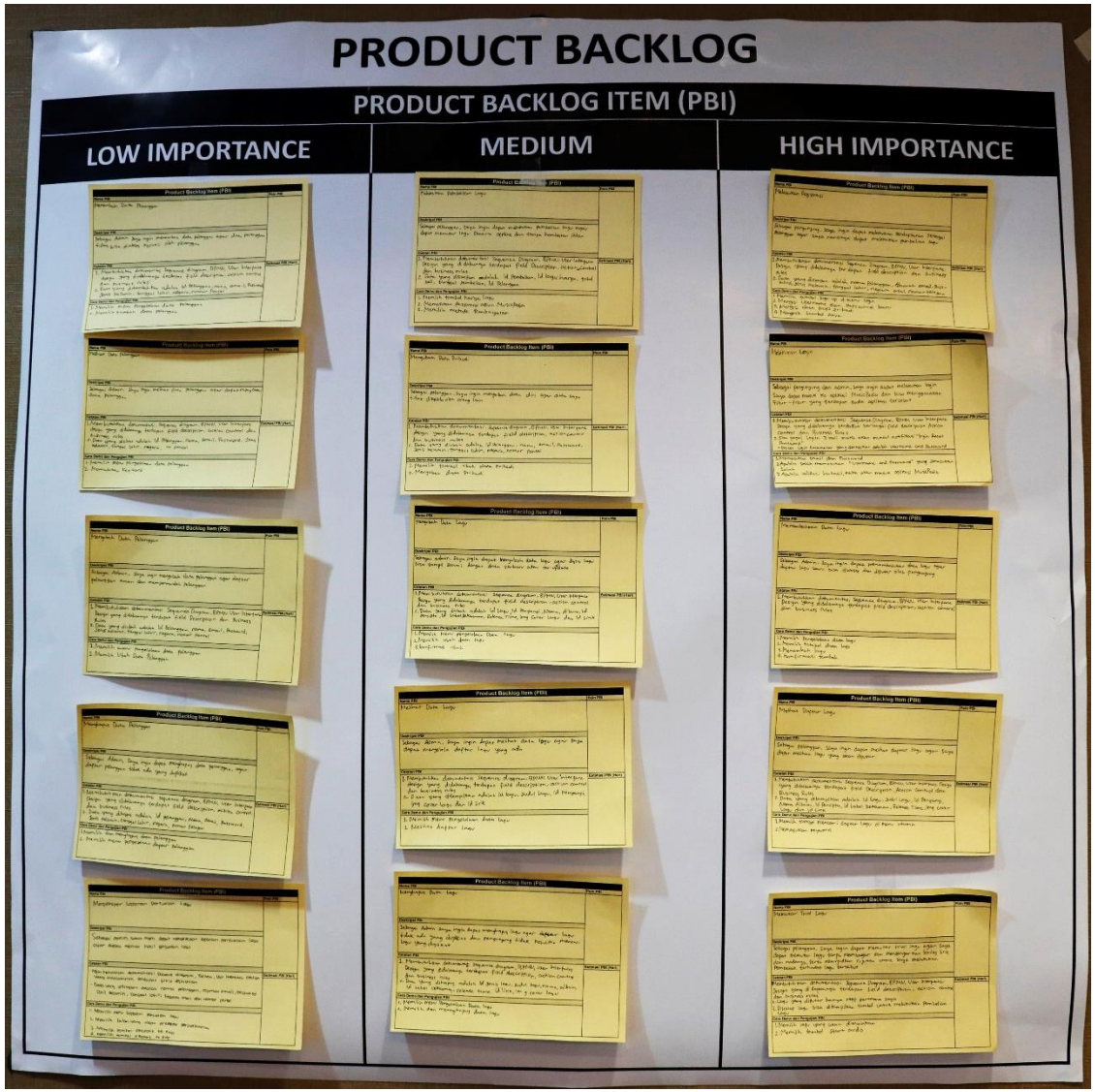


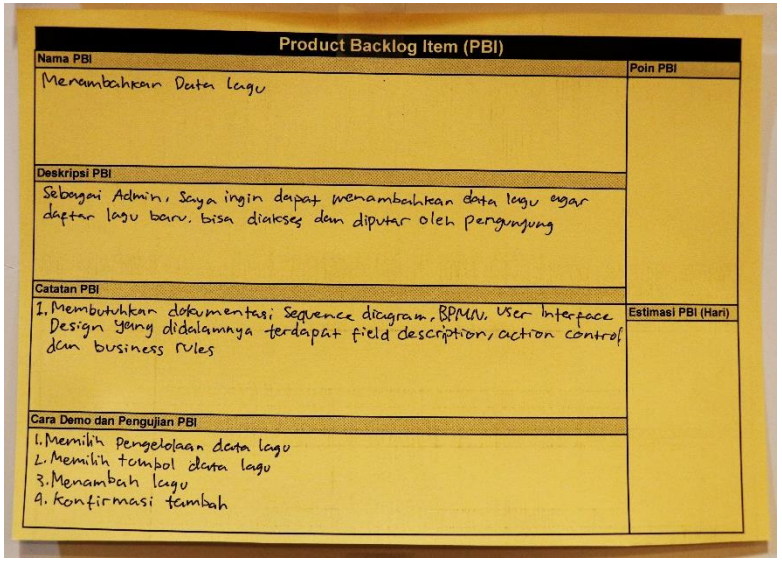
- Product Backlog adalah daftar urutan seluruh fitur, fungsi, kebutuhan, peningkatan dan perbaikan yang diperlukan sebuah produk
- Product Backlog Grooming adalah kegiatan yang mengacu pada 3 kegiatan utama PBI yaitu membuat dan memperbaiki, mengestimasi serta mengatur prioritas

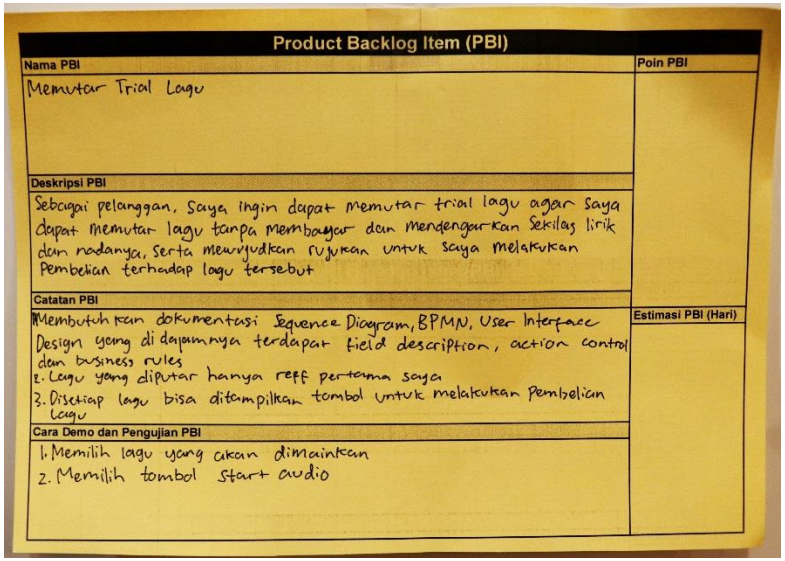
Use Case Diagram Aplikasi MusicPedia



Hasil Product Backlog Grooming (Offline)







Hasil Product Backlog Grooming (Sparx EA)

LOW IMPORTANT	MEDIUM	HIGH IMPORTANT
Menambah Data Pelanggan *Deskripsi PBI:* *See more*	**Melakukan Pembelian Lagu** *Deskripsi PBI:* *See more*	**Melakukan Registrasi** *Deksripsi PBI:* *See more*
Melihat Data Pelanggan *Deskripsi PBI* *See more*	**Mengubah Data Pribadi** *Deskripsi PBI:* *See more*	**Melakukan Login** *Deskripsi PBI:* *See more*
Mengubah Data Pelanggan *Deskripsi PBI* *See more*	**Mengubah Data Lagu** *Deskripsi PBI:* *See more*	**Menambah Data Lagu** *Deskripsi PBI:* *See more*
Menghapus Data Pelanggan *Deskripsi PBI* *See more*	**Menghapus Data Lagu** *Deskripsi PBI:* *See more*	**Melihat Daftar Lagu** *Deskripsi PBI:* *See more*
Mengekspor Laporan Penjualan Lagu *Deskripsi PBI:* *See more*	**Melakukan Logout** *Deskripsi PBI:* *See more*	**Memutar Trial Lagu** *Deskripsi PBI:* *See more*

Melakukan Registrasi

Deksripsi 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

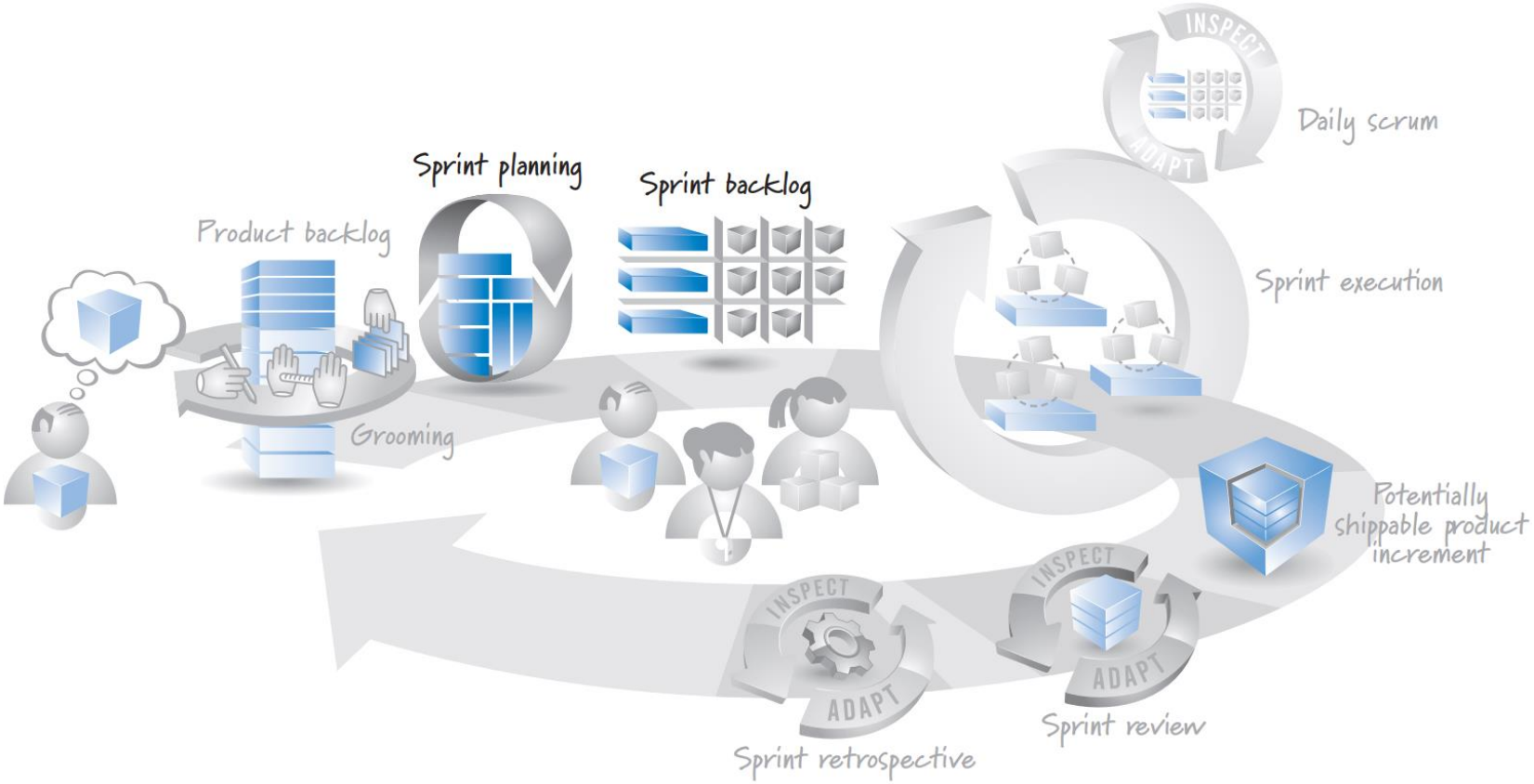
Menambah Data Lagu

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

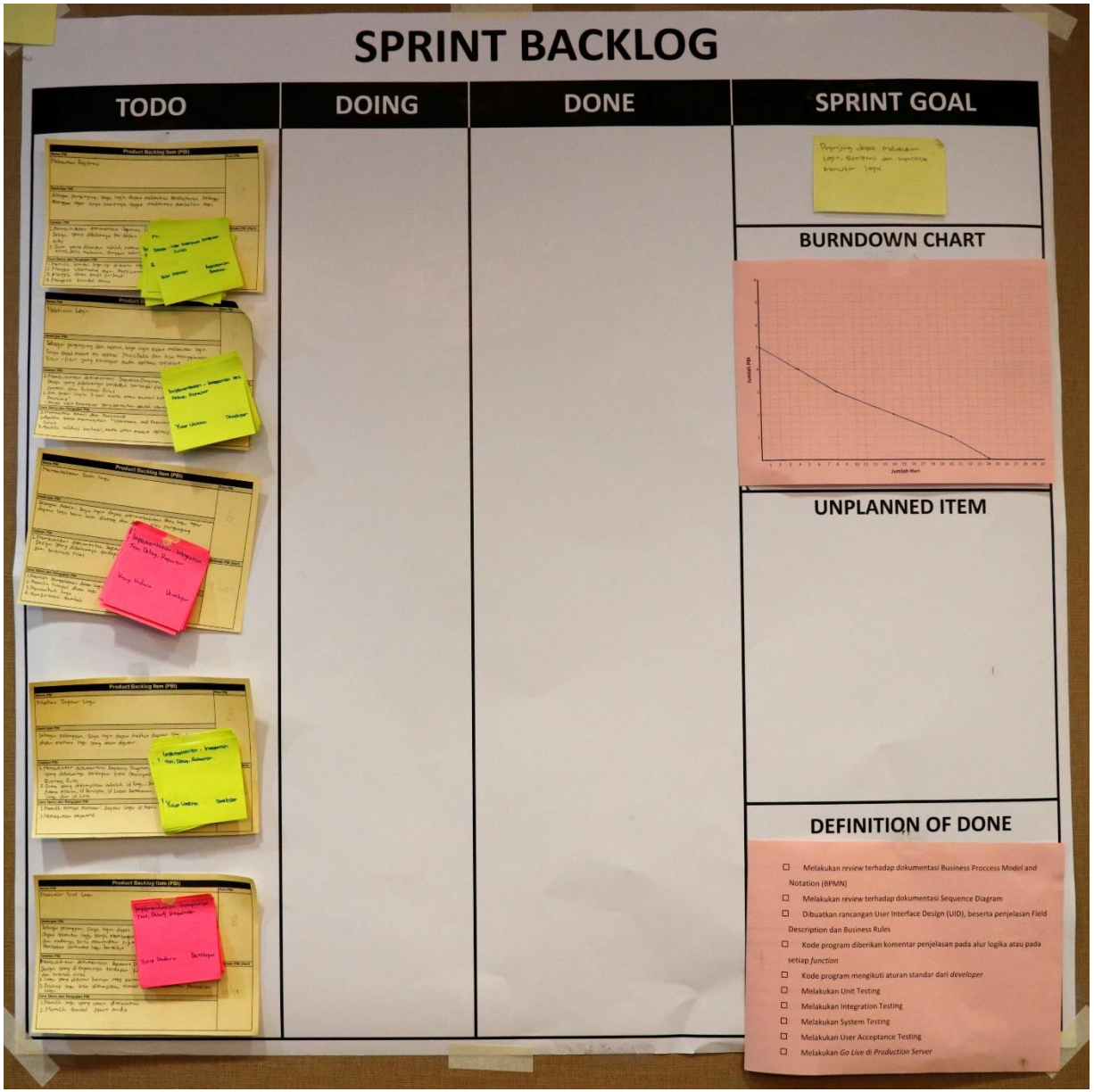
Cara Demo dan Pengujian PBI:
1. Memilih Menu Pengelolaan Data Lagu
2. Memilih Tambah Data Lagu
3. Menambah Lagu
4. Konfirmasi Tambah

1 Sprint Planning

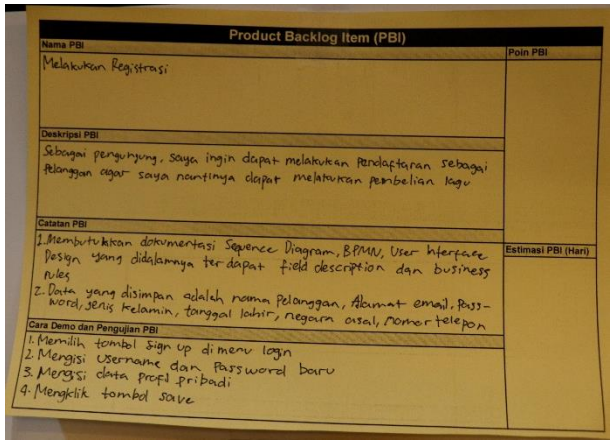


- Sprint Planning adalah proses perencanaan pekerjaan yang akan dikerjakan dalam satu Sprint
- Perencanaan ini dilakukan secara kolaboratif oleh seluruh anggota Scrum Team

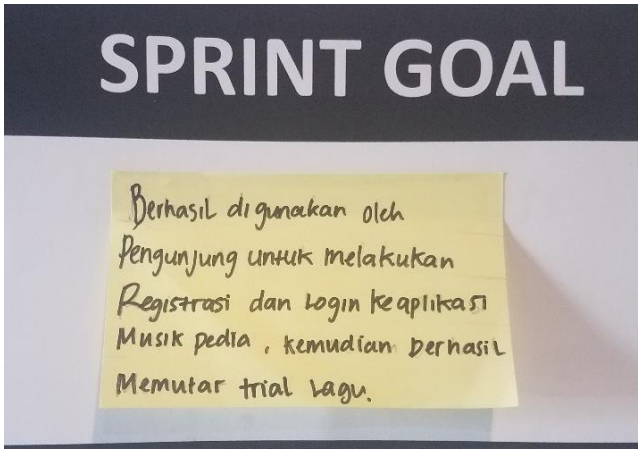
Hasil Sprint Planning (Sprint 1)



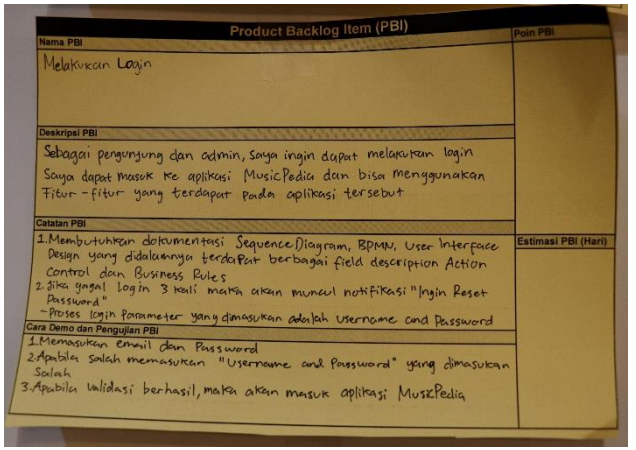
Hasil Sprint Planning (Sprint 1)



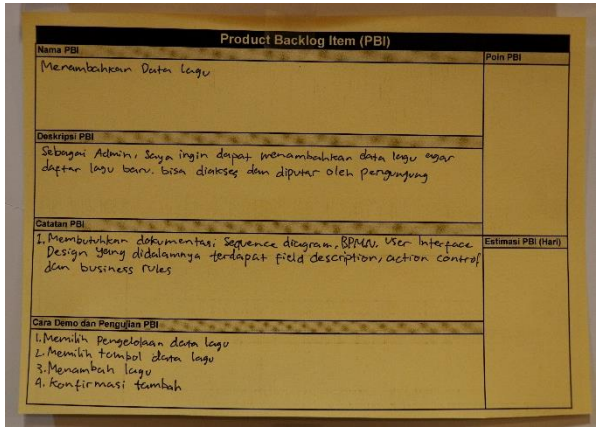
PBI Prioritas Pertama



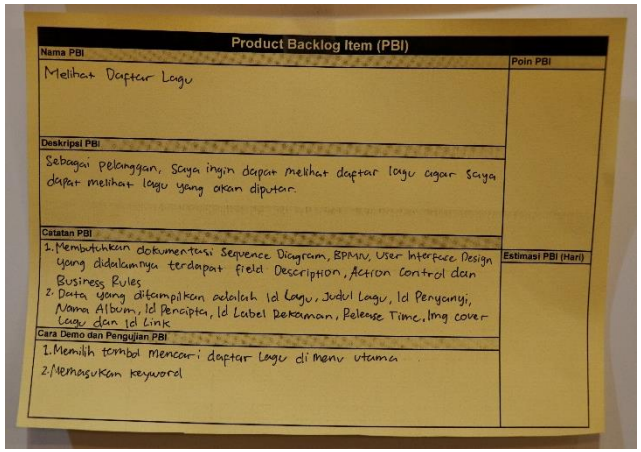
Sprint Goal



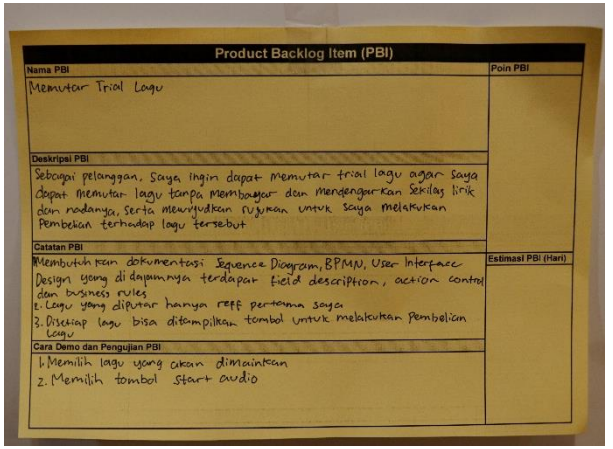
PBI Prioritas Kedua



PBI Prioritas Ketiga

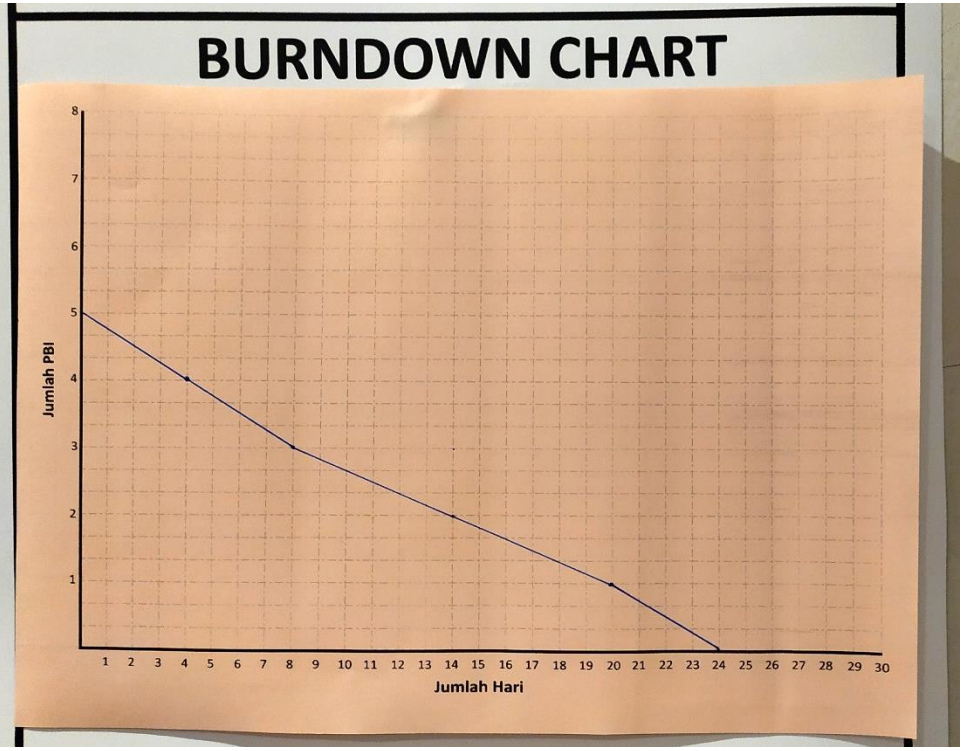


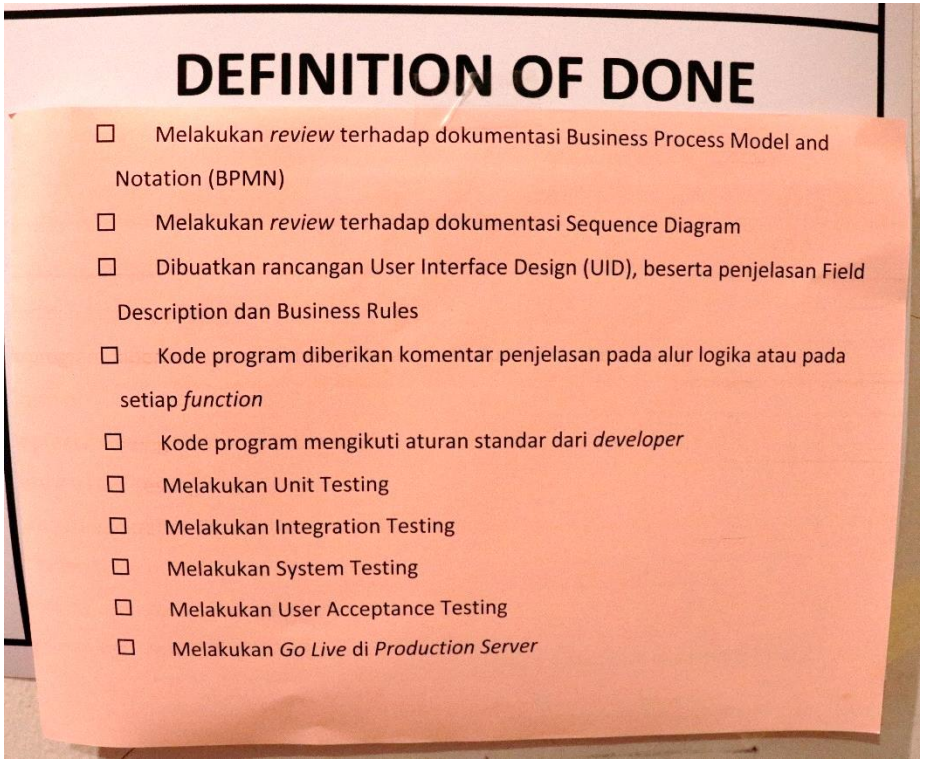
PBI Prioritas Keempat



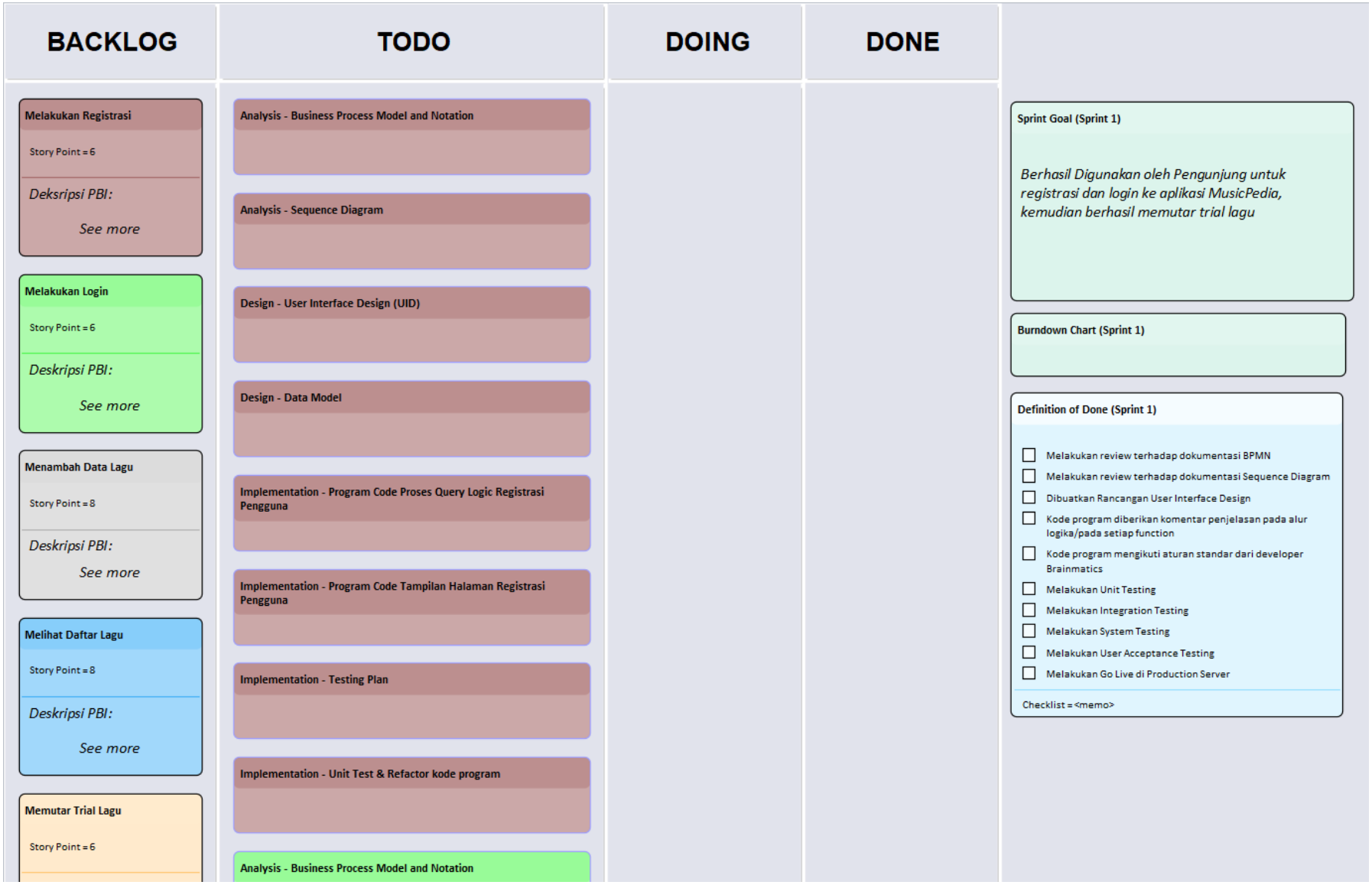
PBI Prioritas Kelima

Hasil Sprint Planning (Sprint 1)

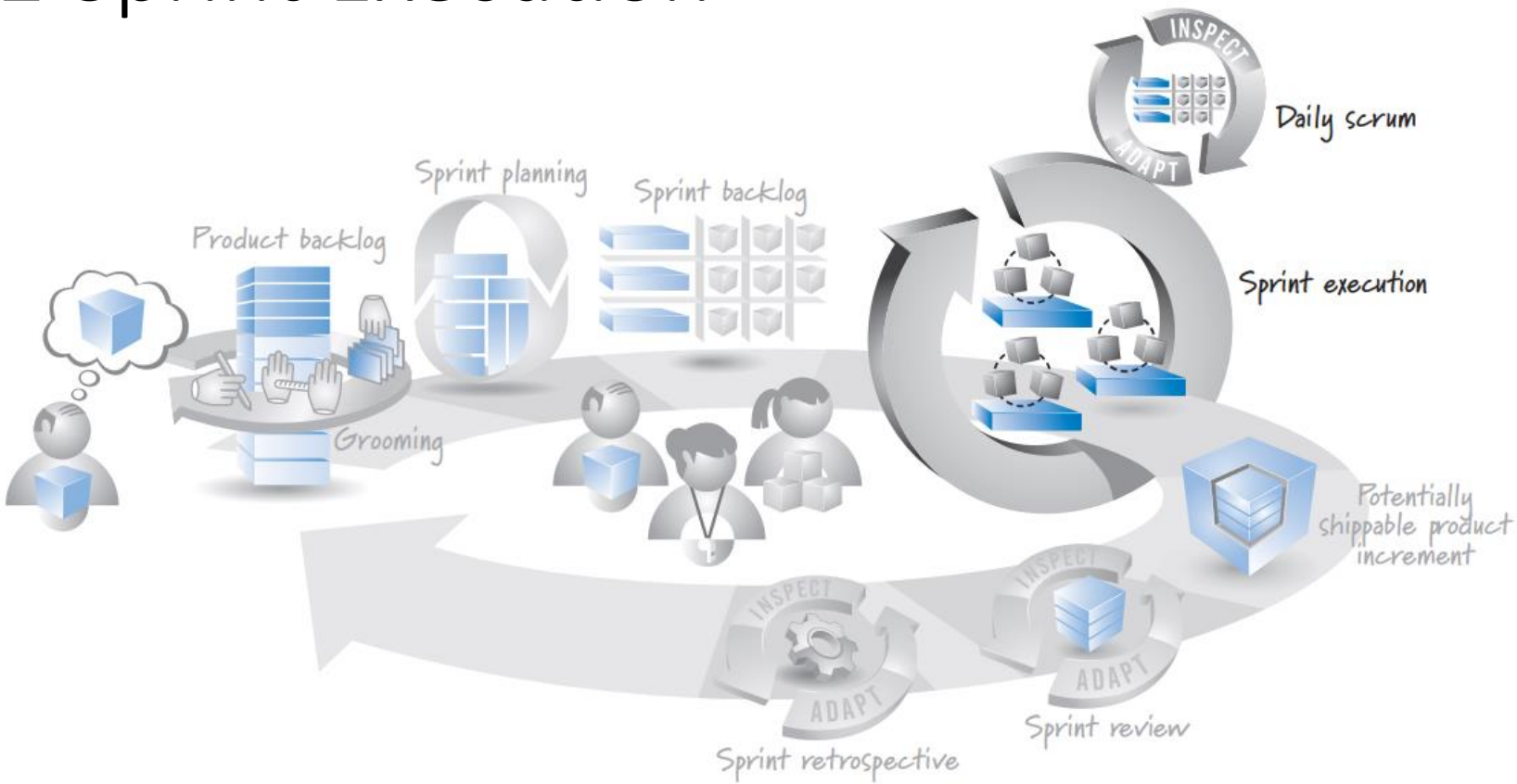




Hasil Sprint Planning (Sprint 1)

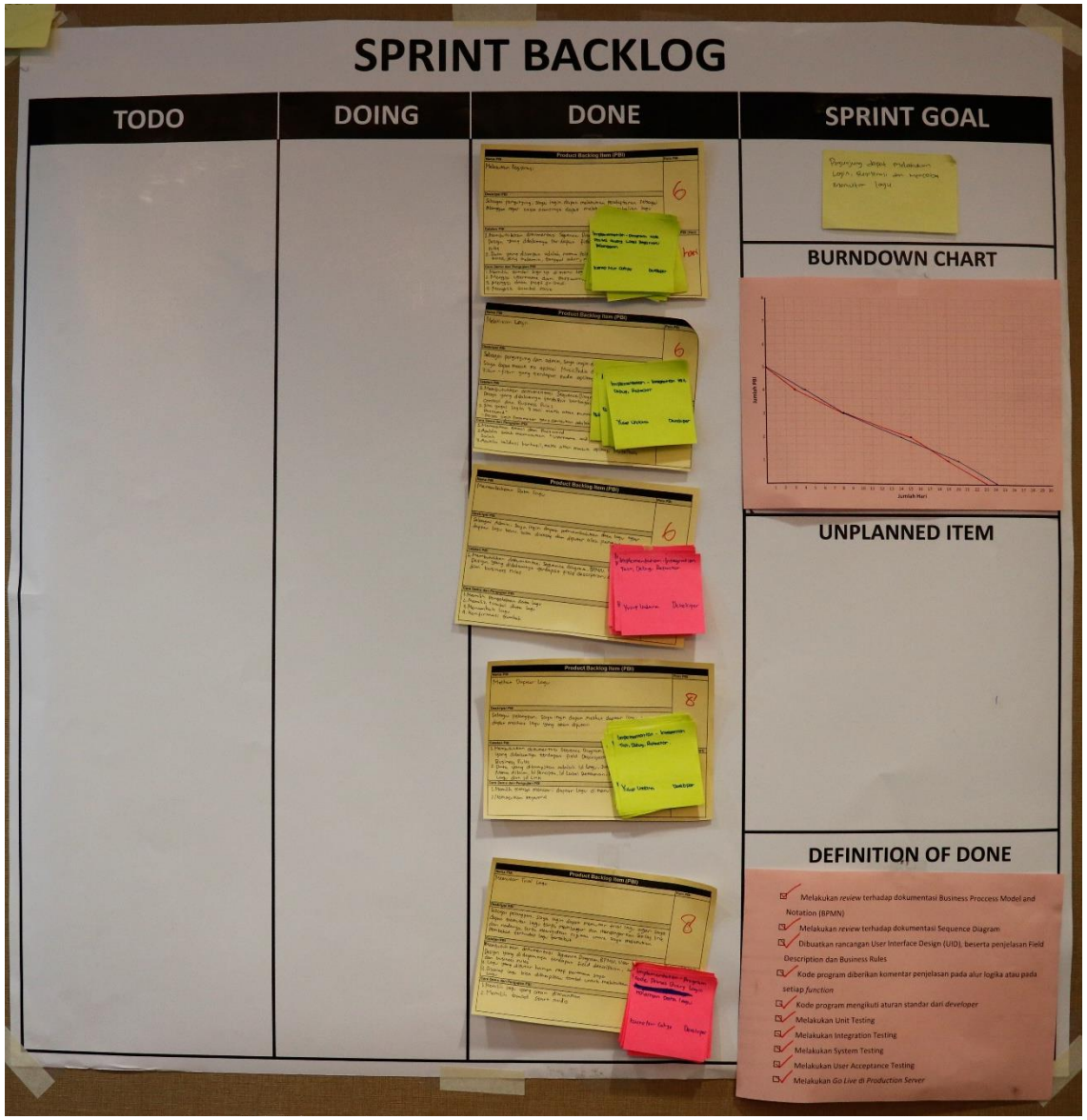


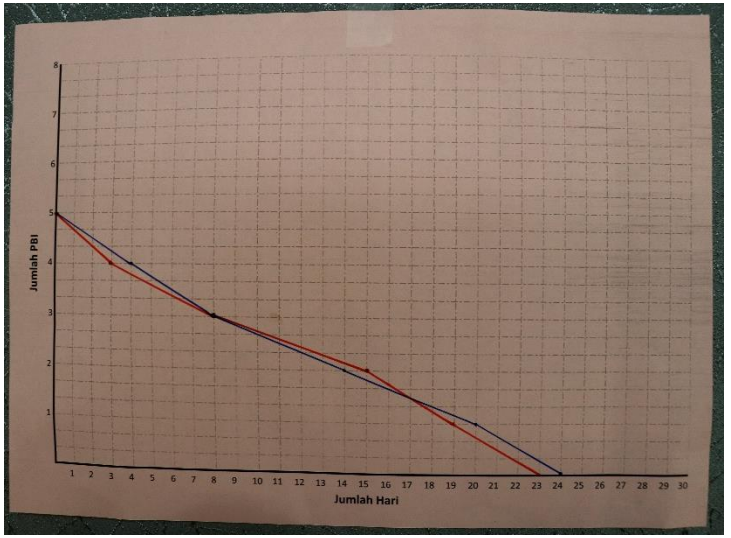
2 Sprint Execution

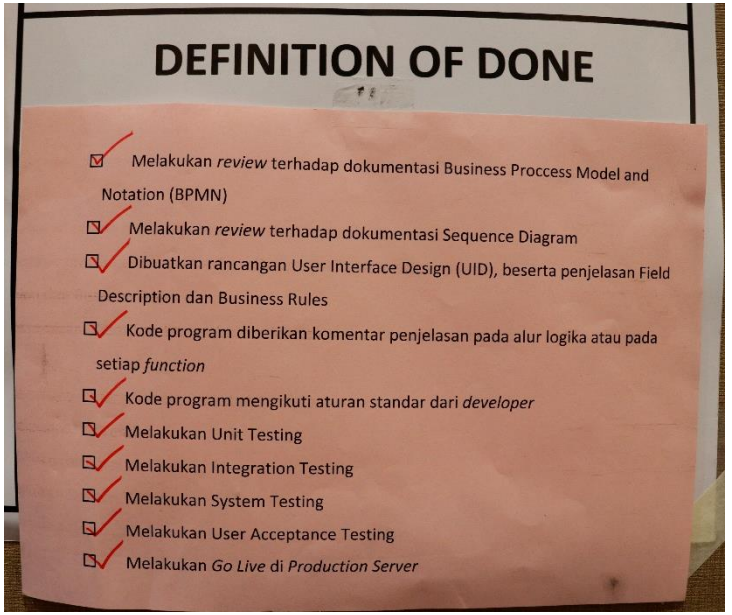


- Sprint Execution adalah pekerjaan yang dilakukan Scrum Team untuk mencapai Sprint Goal
- Prinsip Scrum Execution adalah *Task Planning, Daily Scrums, Task Performance, Communicating,* dan *Flow Management*
- Sprint Execution dapat diasumsikan sebagai *mini project* yang bertujuan untuk menghasilkan potentially shippable product increment

Hasil Sprint Execution (Sprint 1)

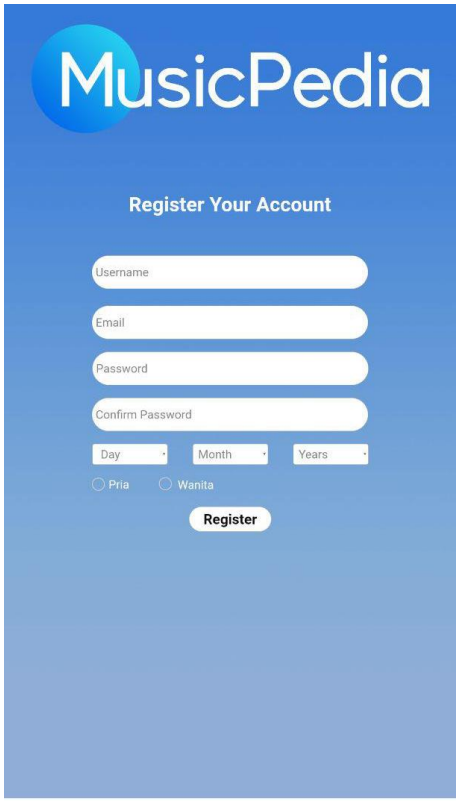




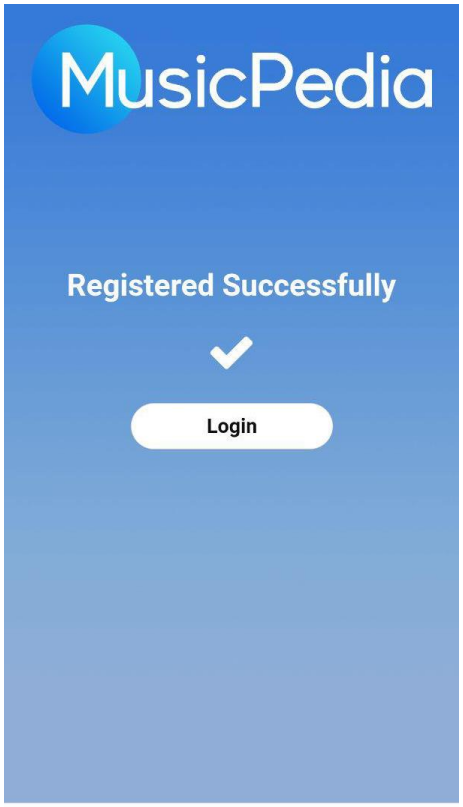


Hasil Sprint Execution (Sprint 1)

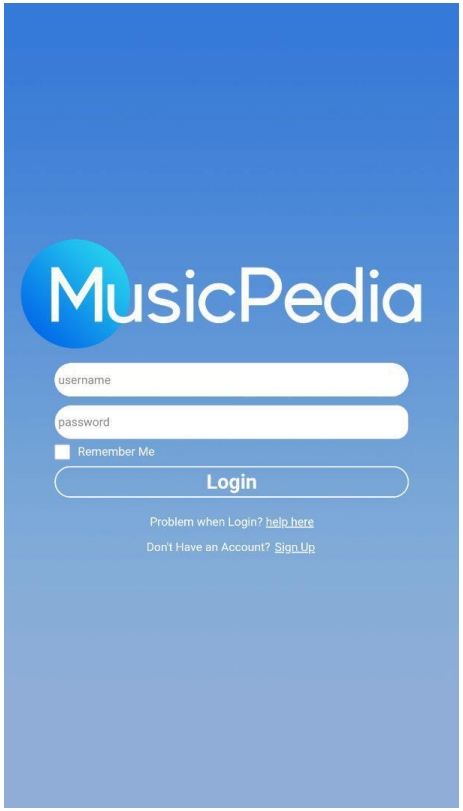
Splash Screen



Halaman Registrasi



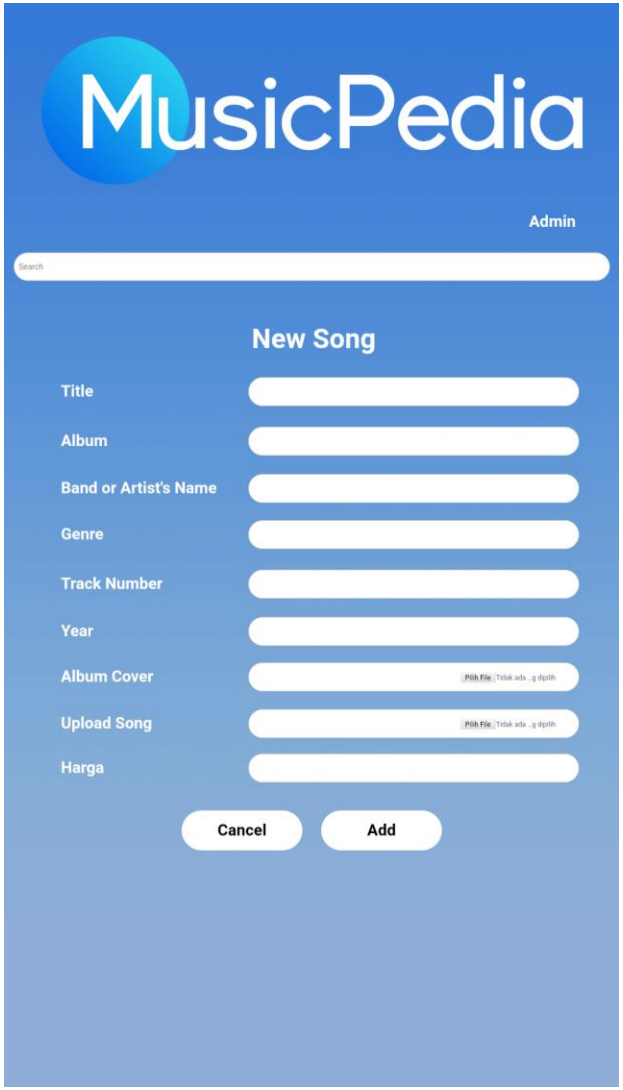
Notifikasi Berhasil Registrasi



Halaman Login

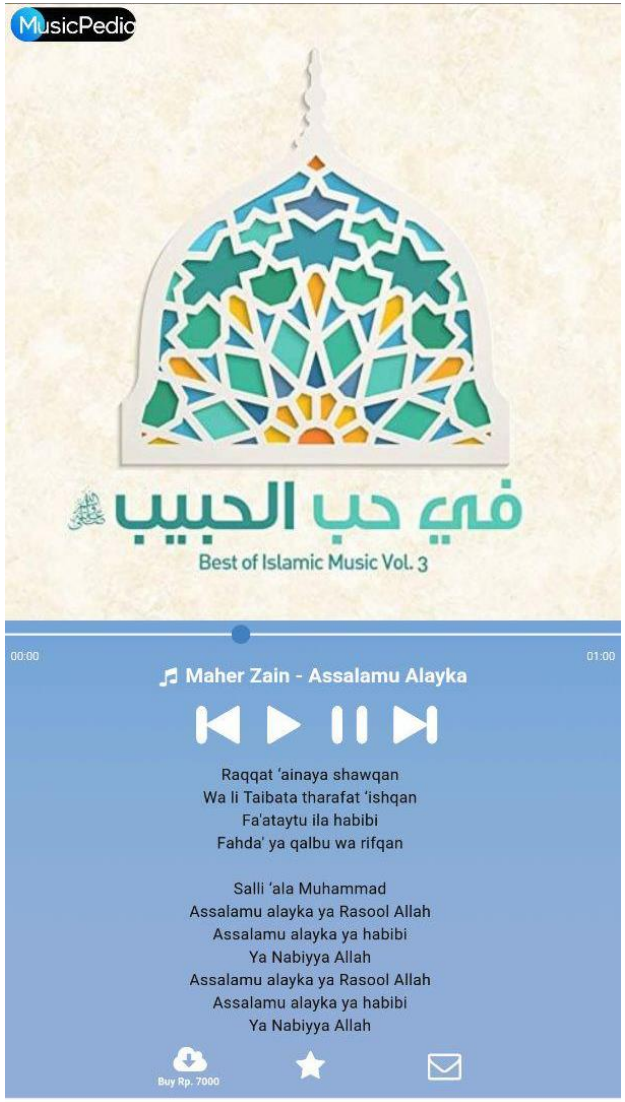
Sprint 1 Goal: Pengunjung dapat melakukan Registrasi, login, dan memutar trial lagu

Hasil Sprint Execution (Sprint 1)



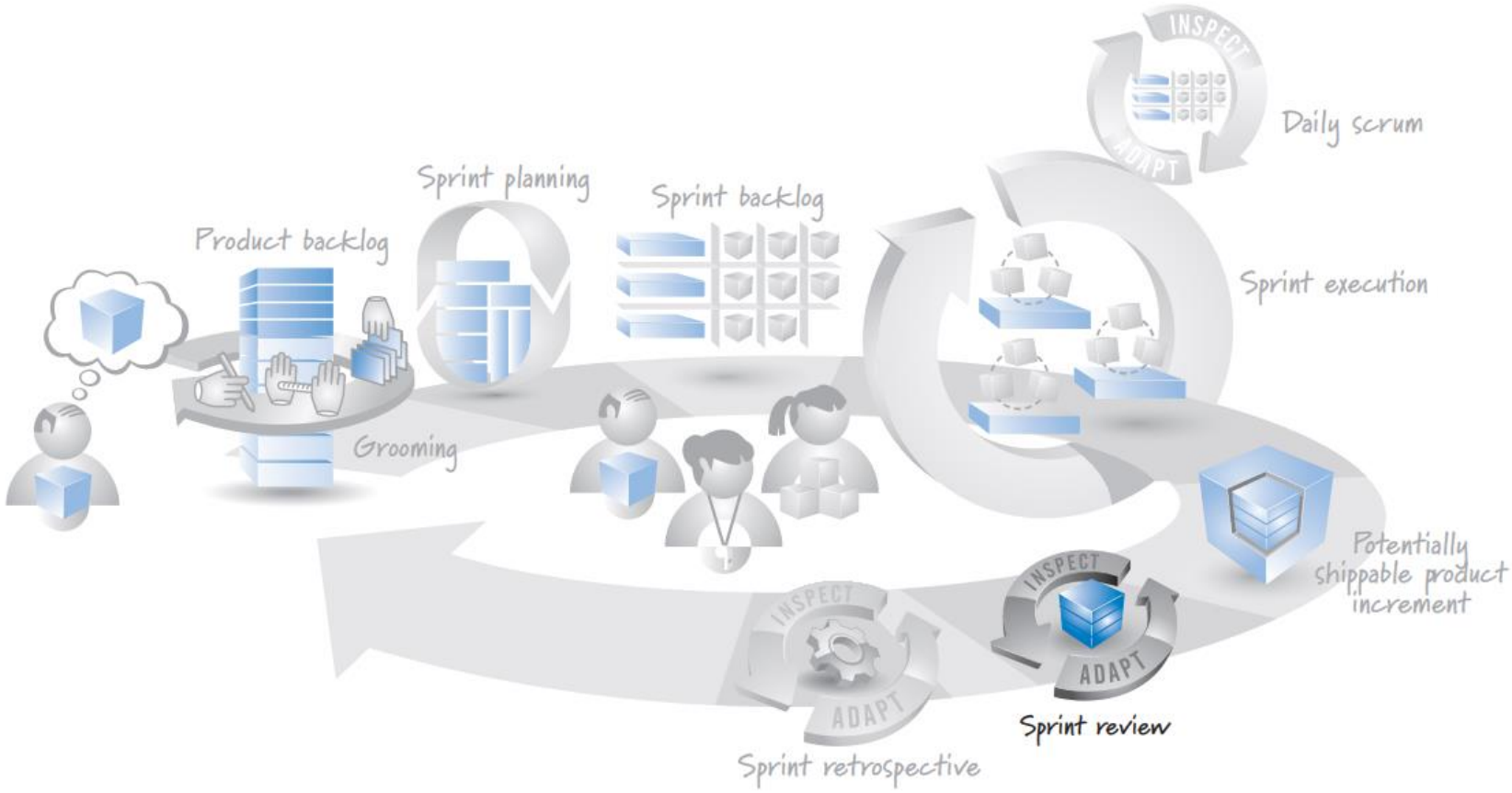
Halaman Penambahan Data Lagu

Halaman Melihat Data Lagu



Memutar Trial Lagu

3 Sprint Review



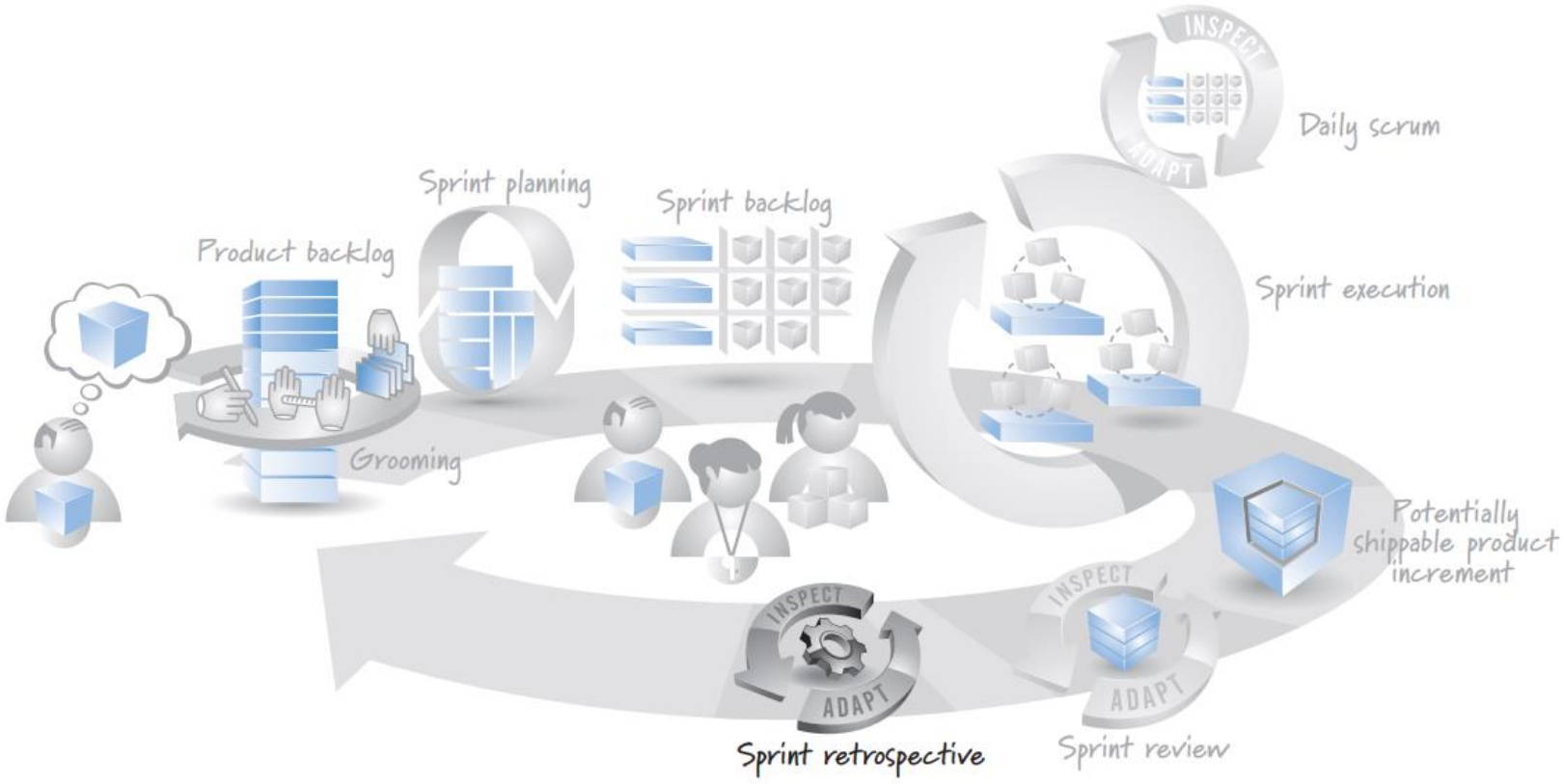
- Sprint Review adalah Sprint yang dilakukan untuk memeriksa dan menyesuaikan progres pekerjaan
- Sprint Review terjadi di dekat akhir sprint setelah Sprint Execution dan sebelum (atau kadang setelah) Sprint Retrospective

Hasil Sprint Review

- Hasil Sprint Review dari Sprint pertama dari pengembangan aplikasi MusicPedia

Rank	PBI	Accepted?	Notes
1	Melakukan Registrasi	OK	-
2	Melakukan Login	OK	-
3	Menambah Data Lagu	X	Terdapat 3 error yang muncul saat proses input data lagu
4	Melihat Daftar Lagu	X	Kode program tidak rapi
5	Memutar Trial Lagu	X	*Test Case* tidak dituliskan

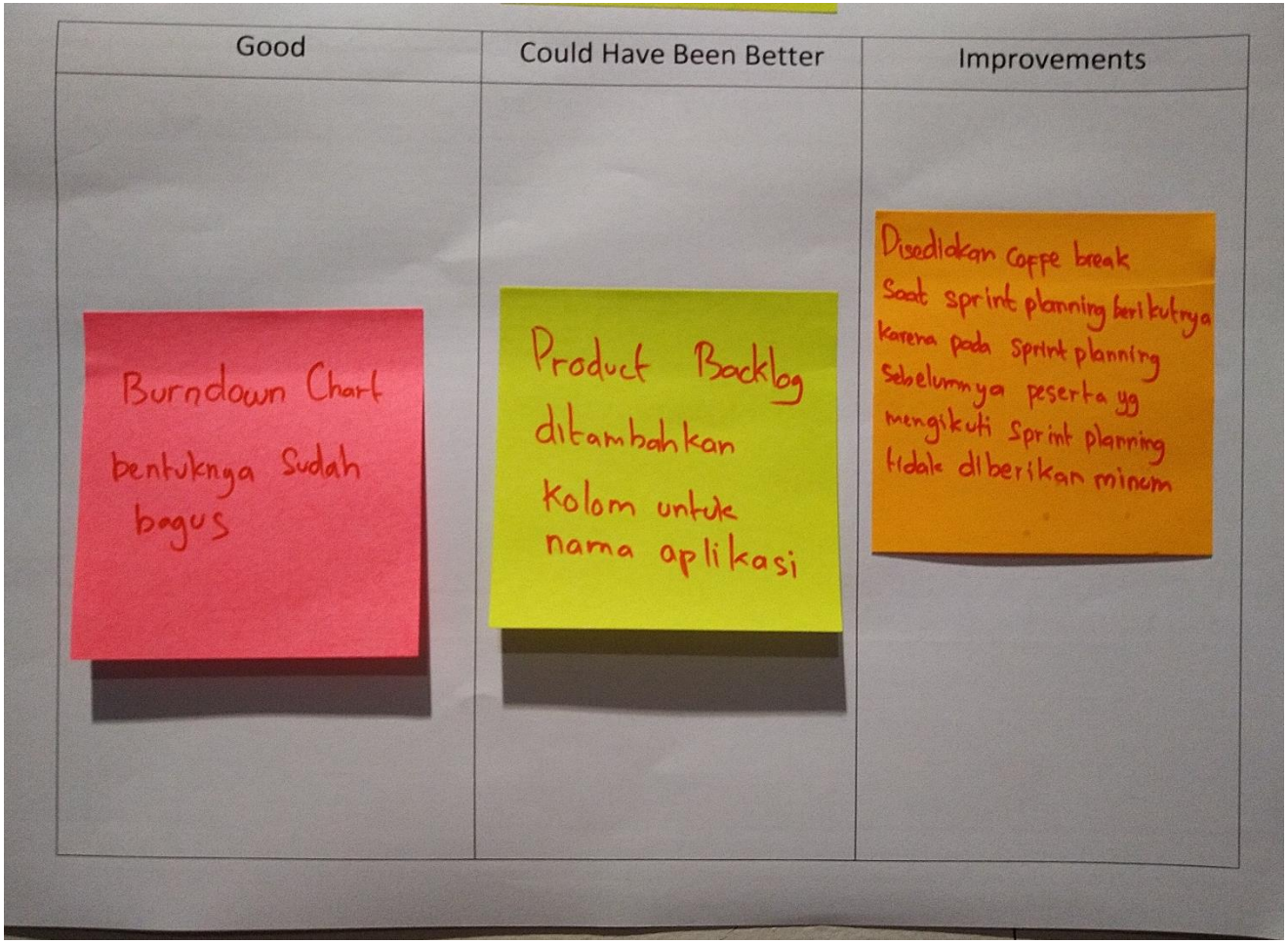
4 Sprint Retrospective



- Sprint Retrospective adalah sebuah kesempatan bagi Scrum Team untuk menginspeksi dirinya sendiri dan membuat perencanaan mengenai peningkatan yang akan dilakukan di Sprint berikutnya
- Sprint Retrospective Dilakukan setelah Sprint Review dan sebelum Sprint Planning berikutnya

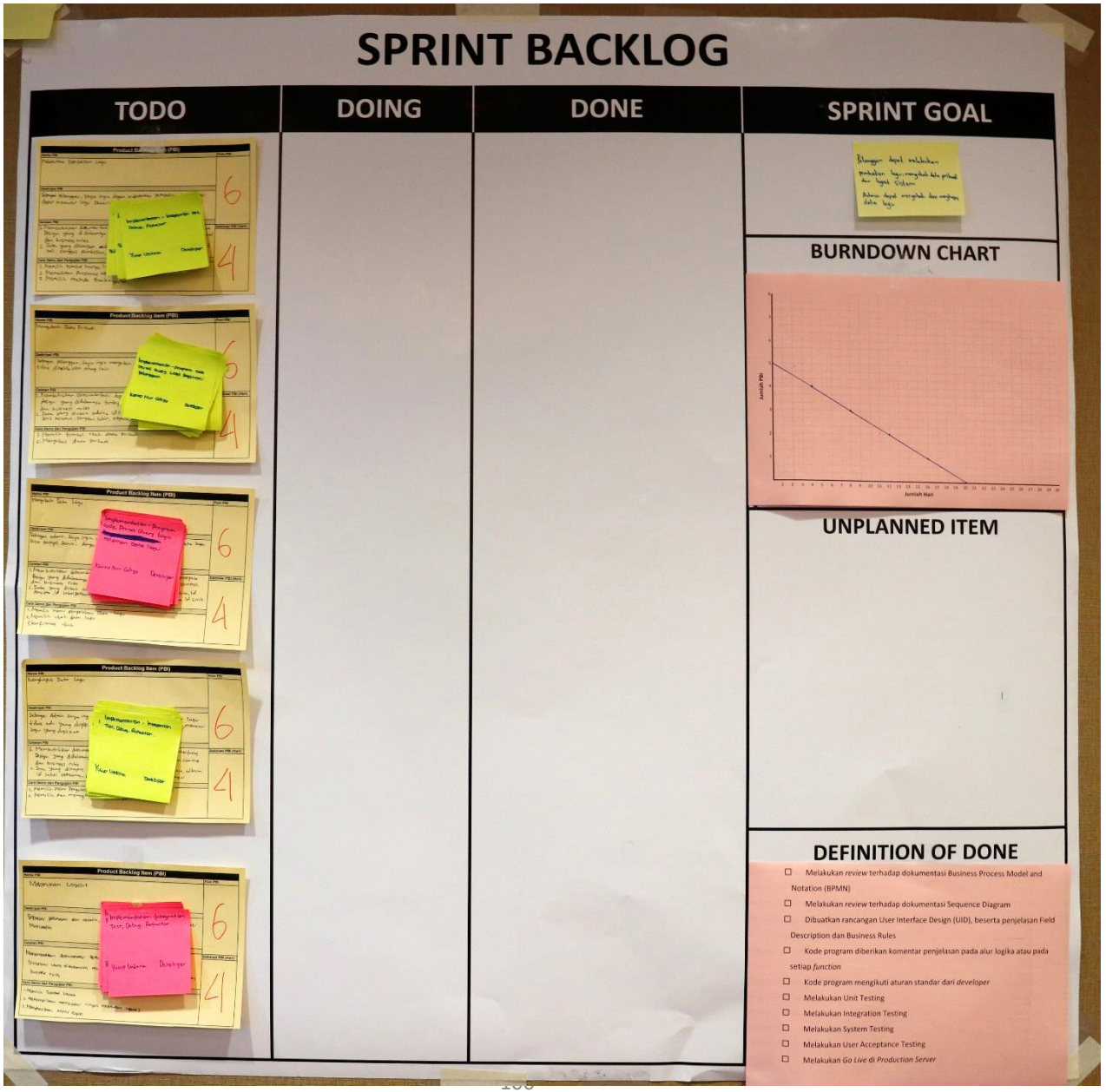
Hasil Sprint Retrospective

Hasil Sprint Retrospective dari Sprint pertama pengembangan aplikasi MusicPedia

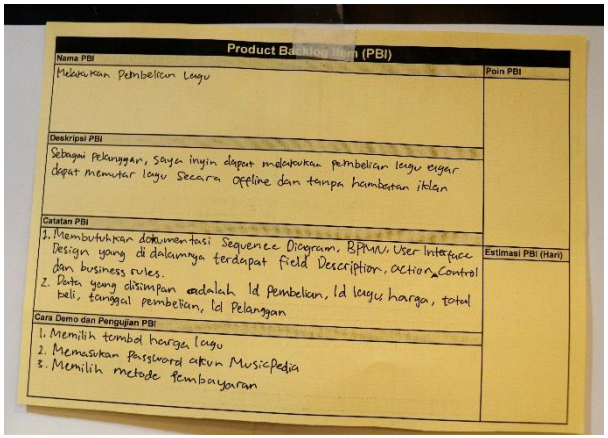


Sprint Execution (Sprint 2 – Sprint *n*)

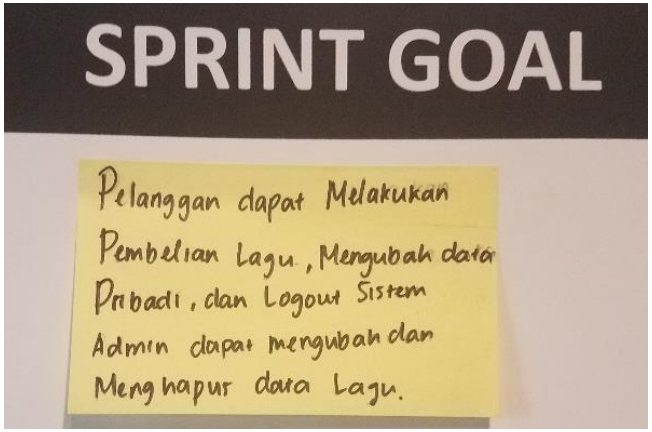
Hasil Sprint Planning (Sprint 2)



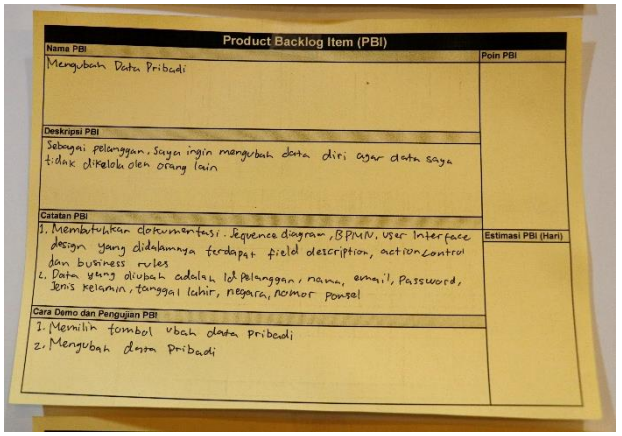
Hasil Sprint Planning (Sprint 2)



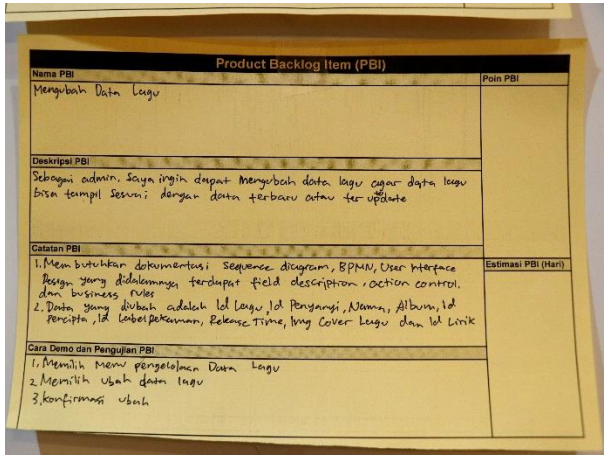
PBI Prioritas Pertama



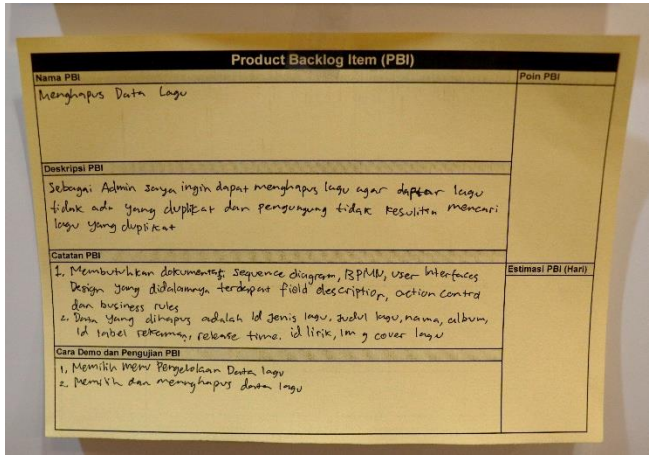
Sprint Goal



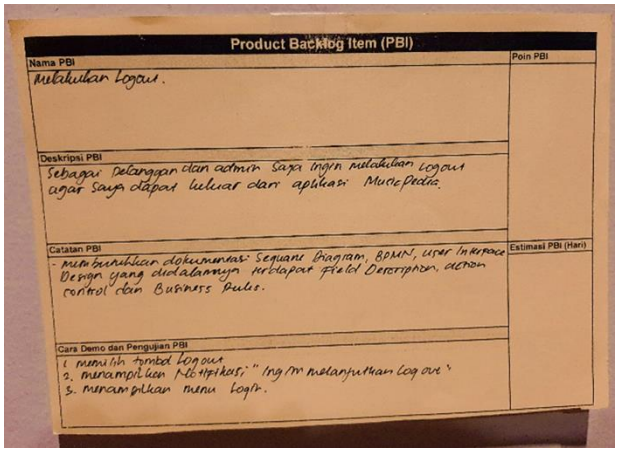
PBI Prioritas Kedua



PBI Prioritas Ketiga

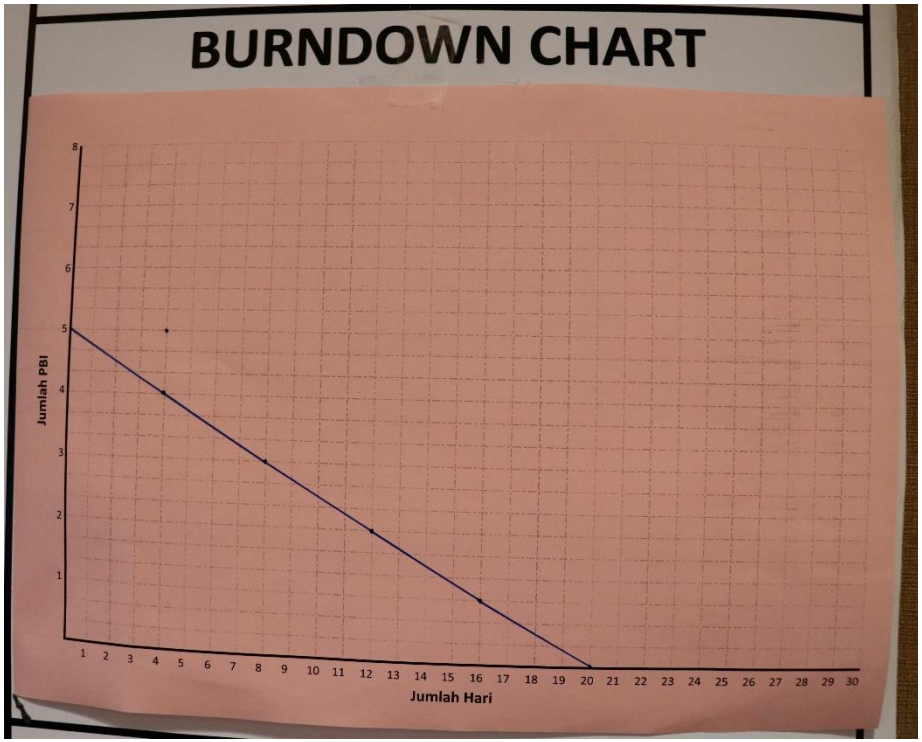


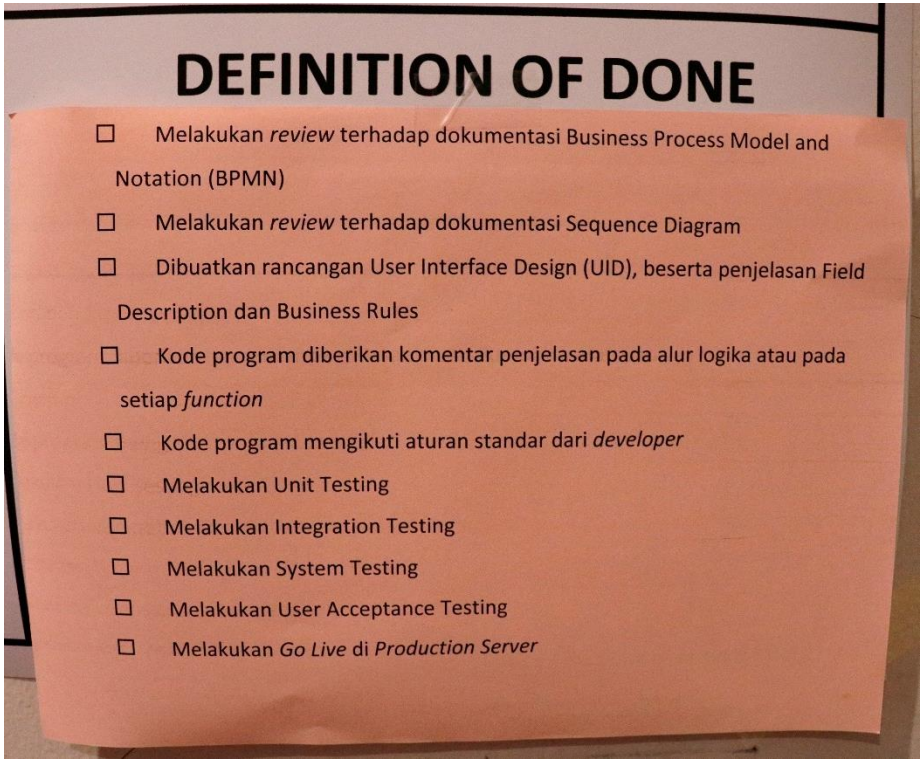
PBI Prioritas Keempat



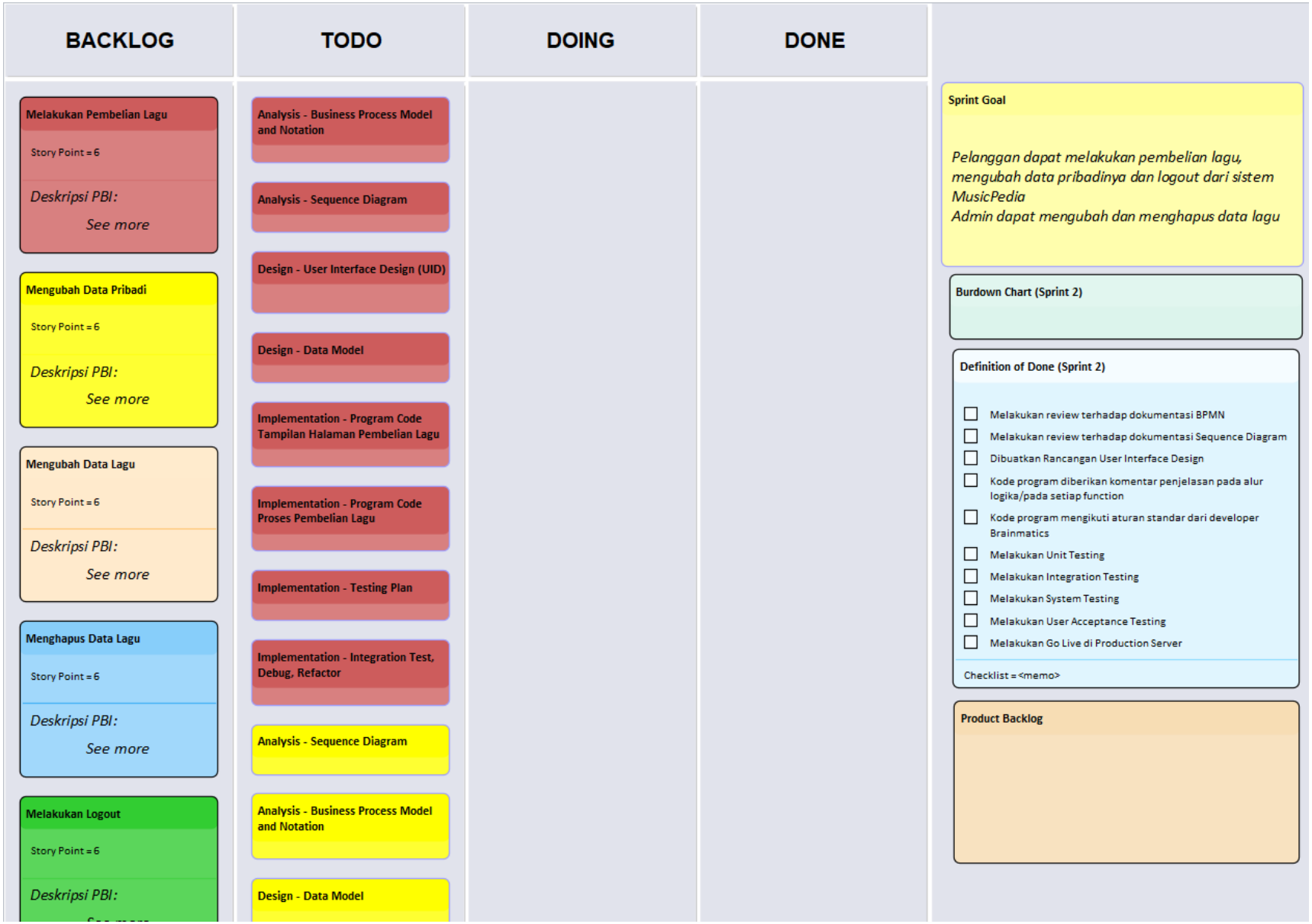
PBI Prioritas Kelima

Hasil Sprint Planning (Sprint 2)

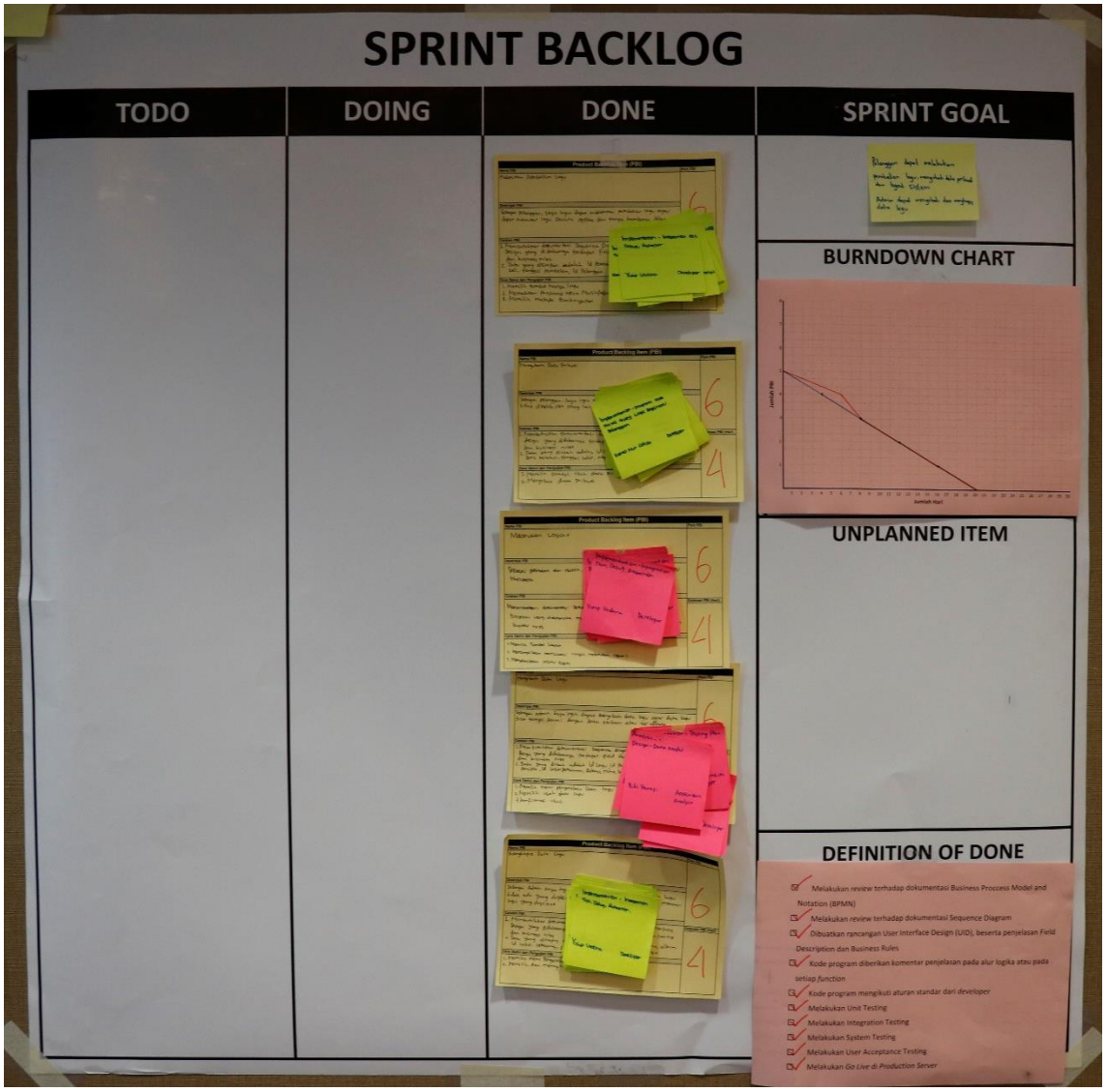


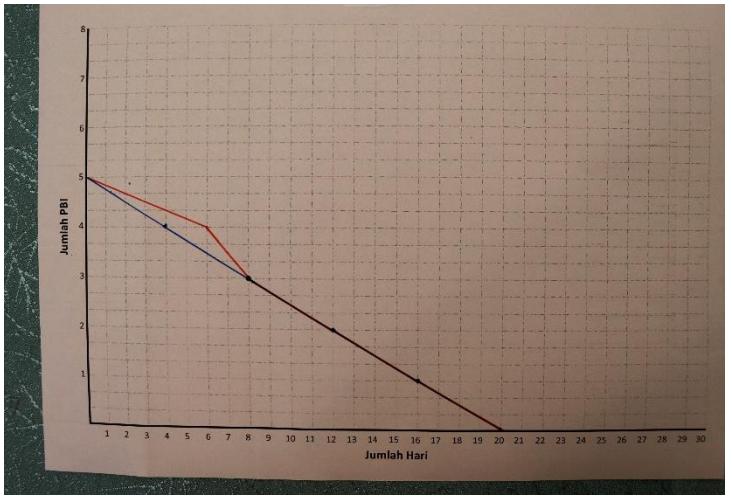


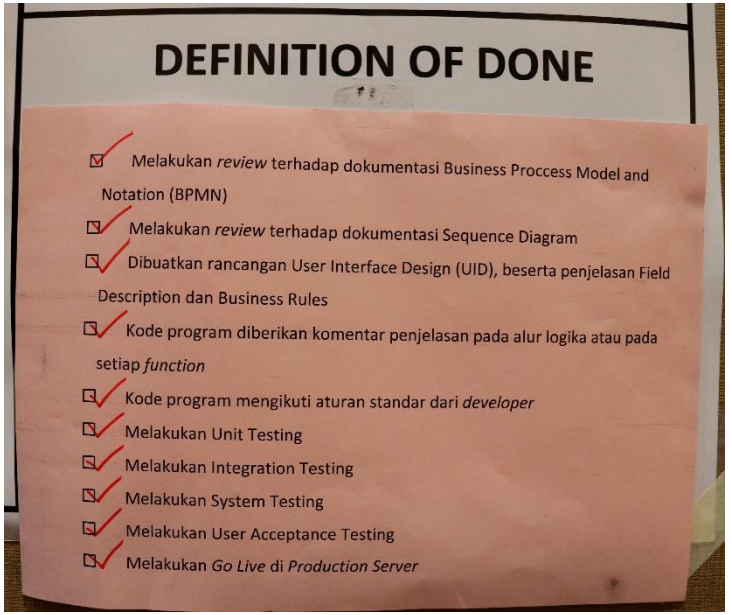
Hasil Sprint Planning (Sprint 2)



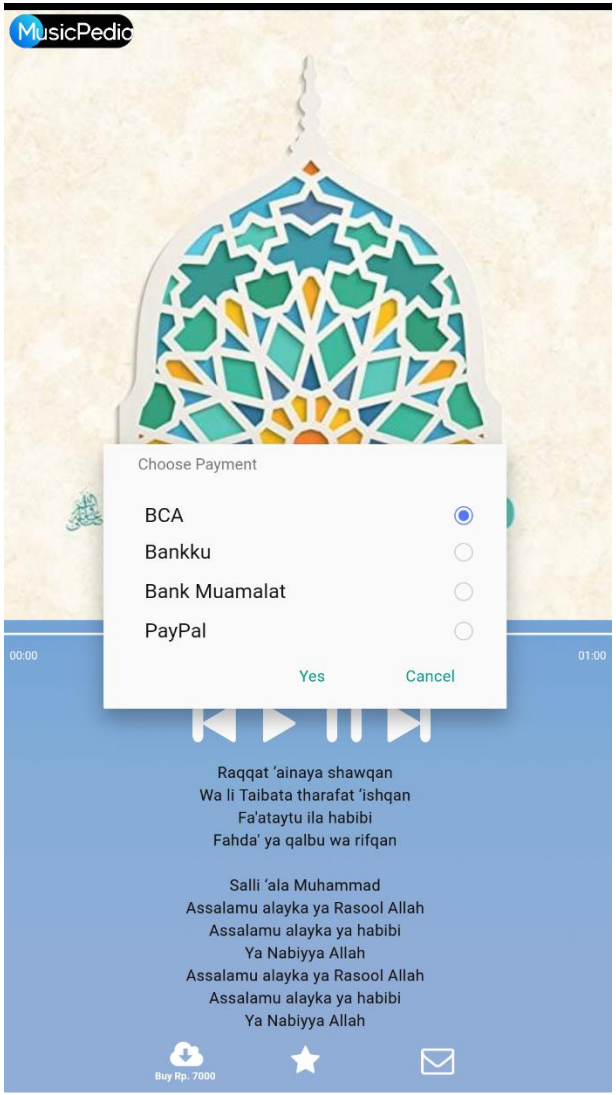
Sprint Execution (Sprint 2)



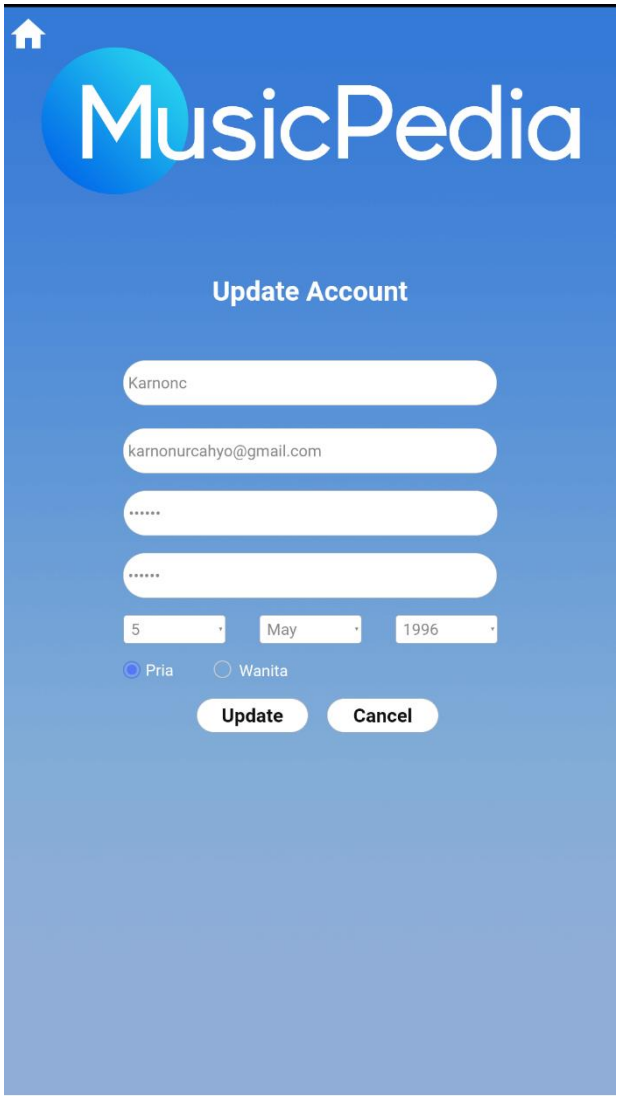




Hasil Sprint Execution (Sprint 2)



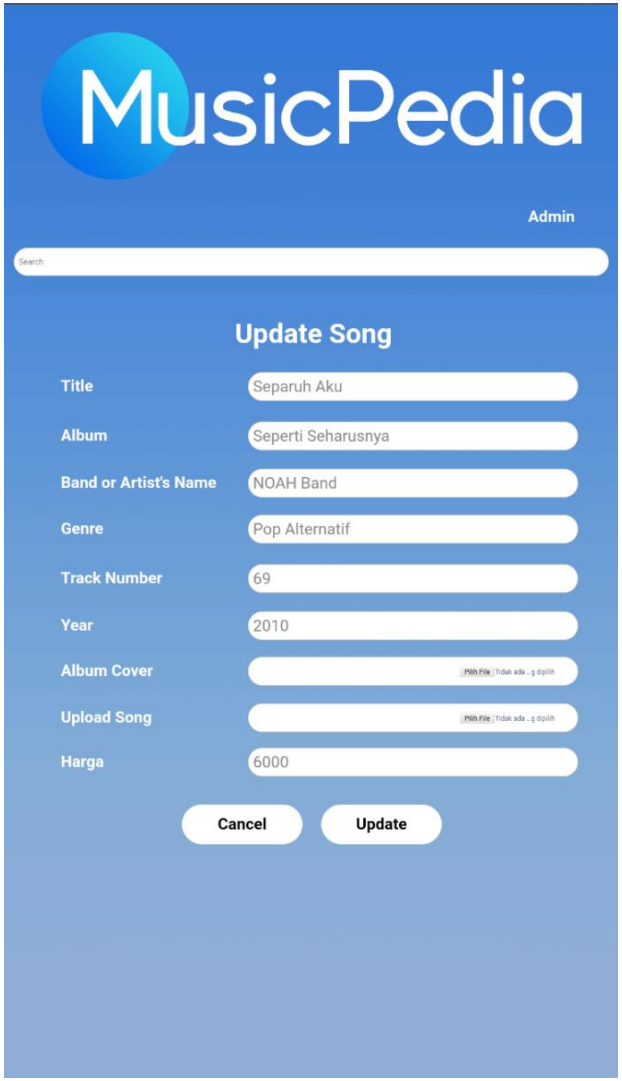
Melakukan Pembelian Lagu



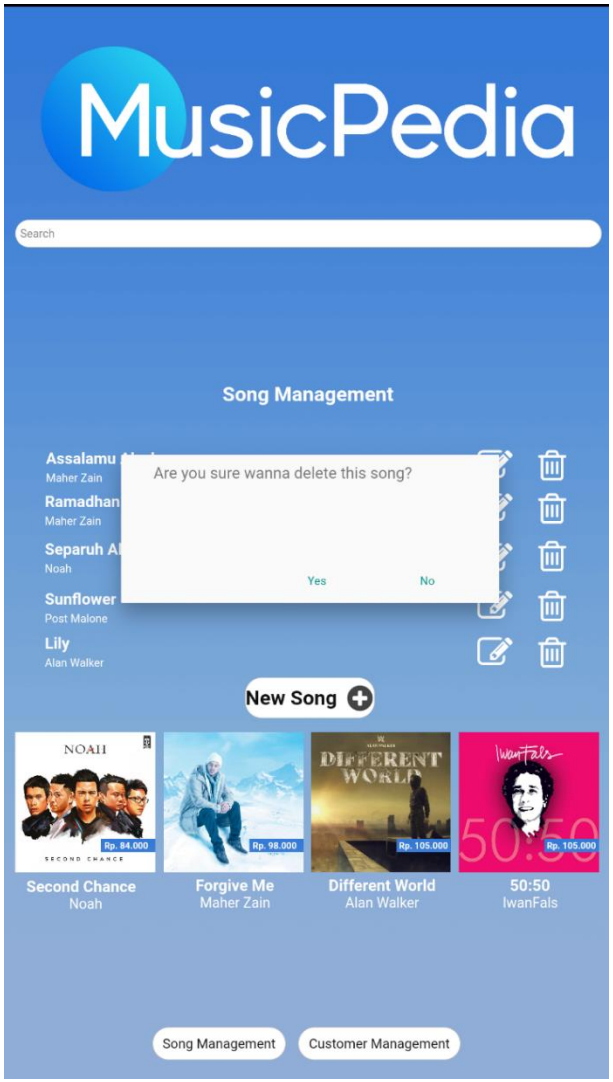
Mengubah Data Pribadi

Sprint 2 Goal: Pelanggan dapat melakukan pembelian lagu, mengubah data pribadi, dan admin dapat mengubah dan menghapus data lagu

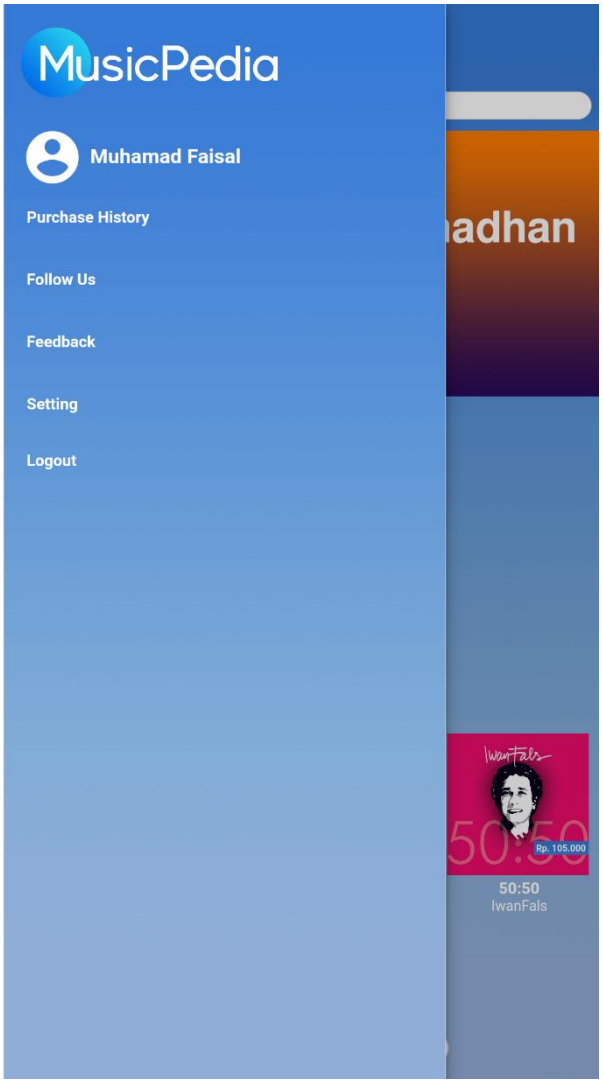
Hasil Sprint Execution (Sprint 2)



Mengubah Data Lagu

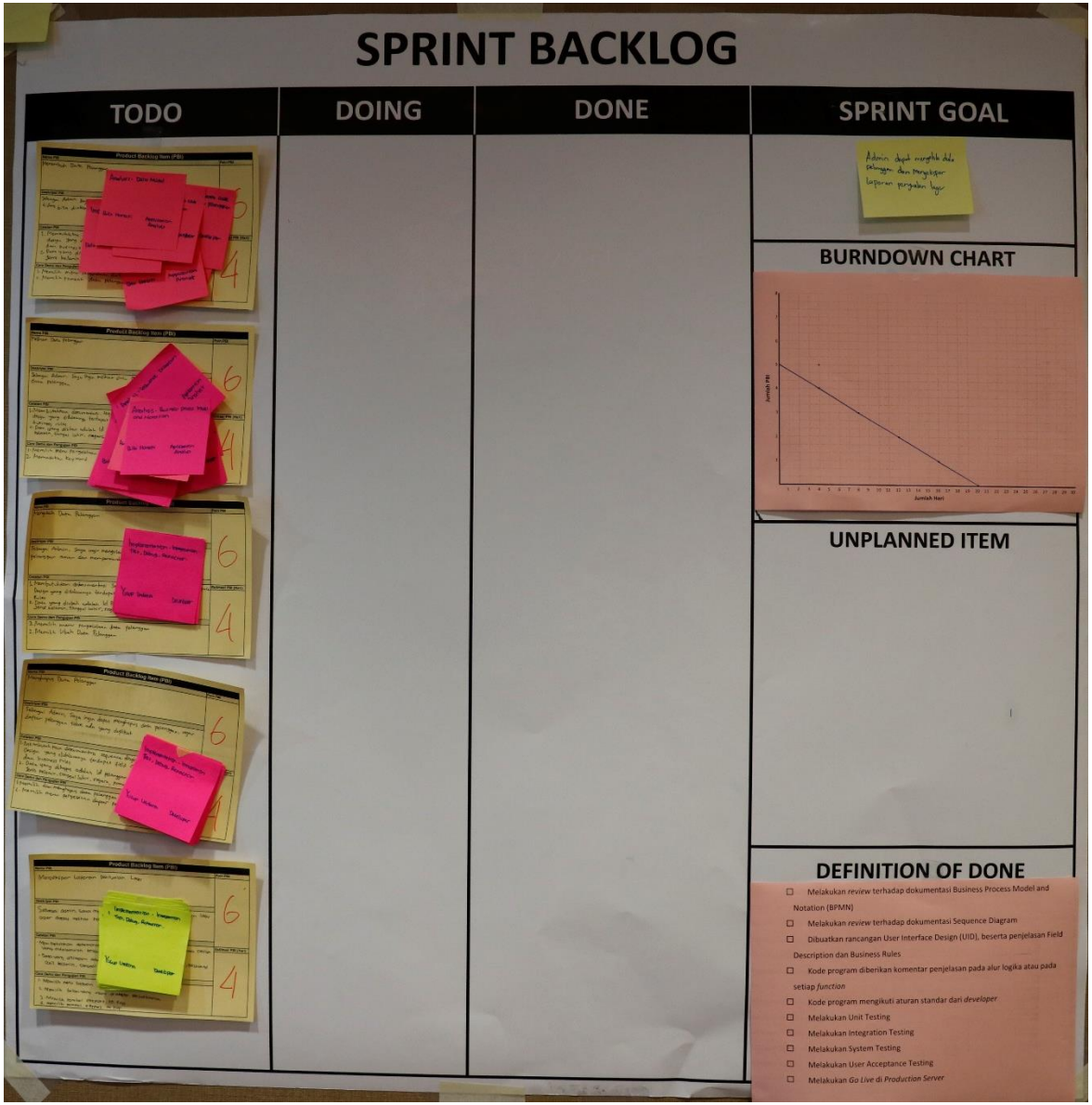


Menghapus Data Lagu

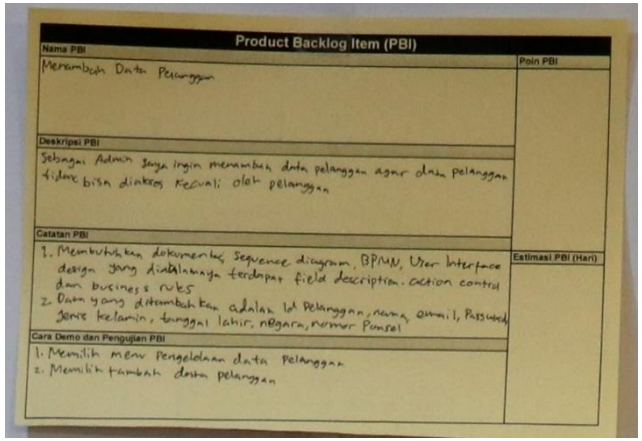


Melakukan Logout

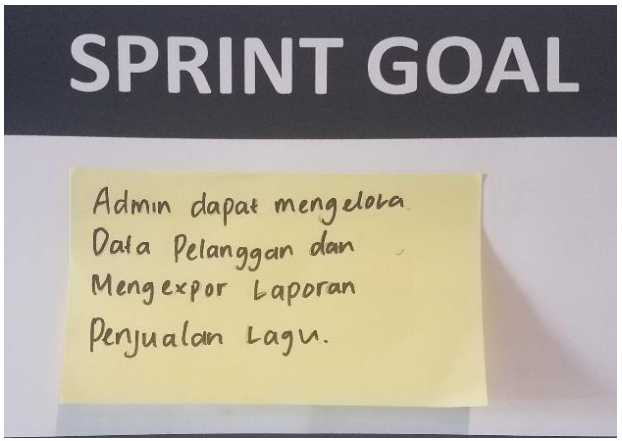
Hasil Sprint Planning (Sprint 3)



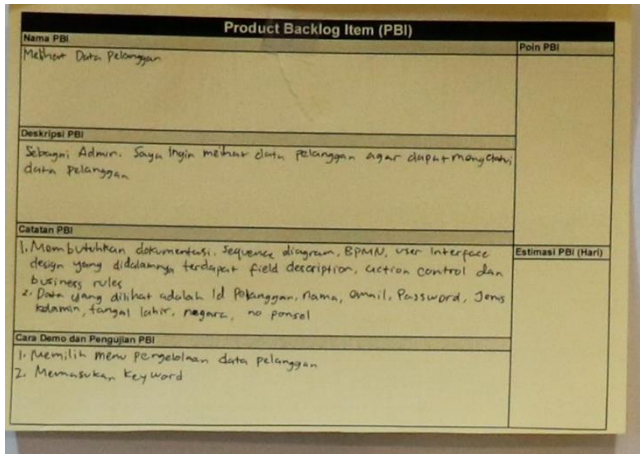
Hasil Sprint Planning (Sprint 3)



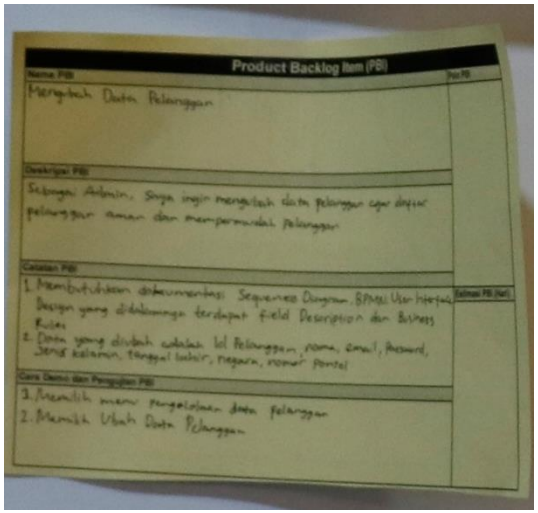
PBI Prioritas Pertama



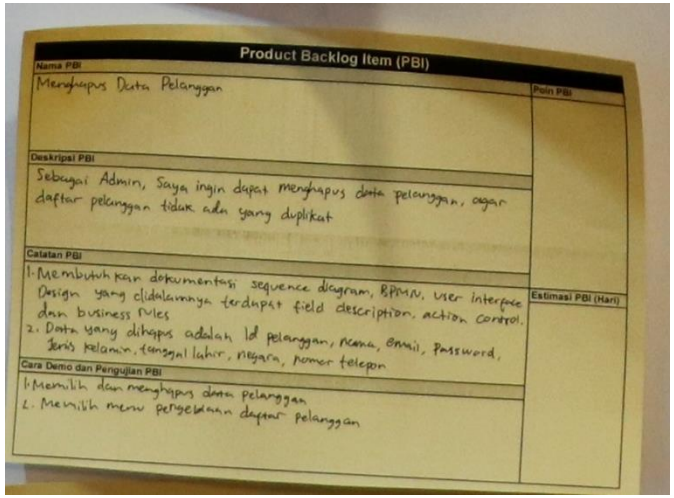
Sprint Goal



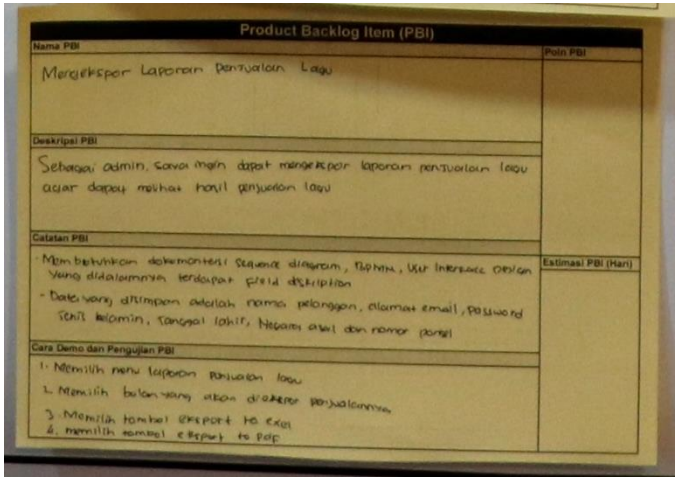
PBI Prioritas Kedua



PBI Prioritas Ketiga

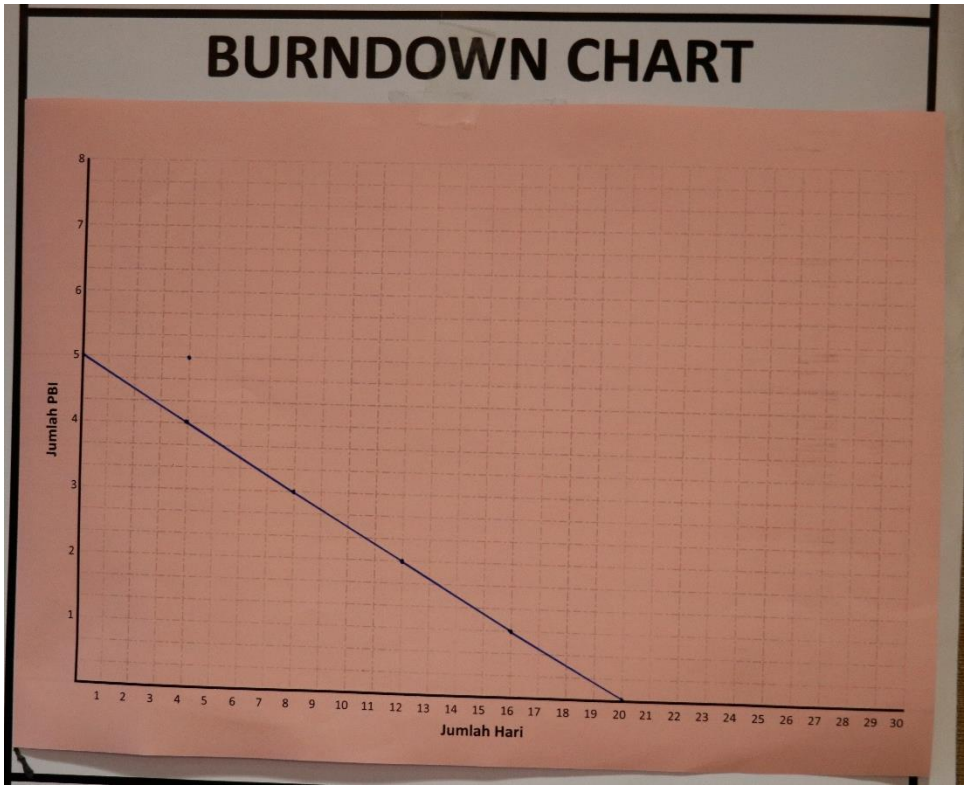


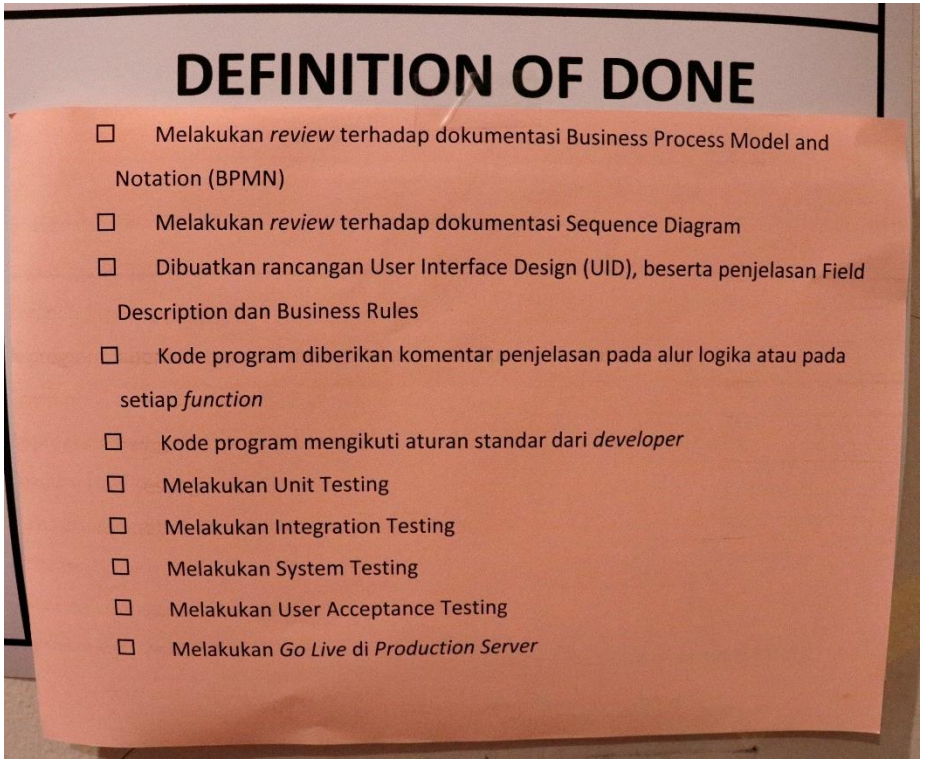
PBI Prioritas Keempat



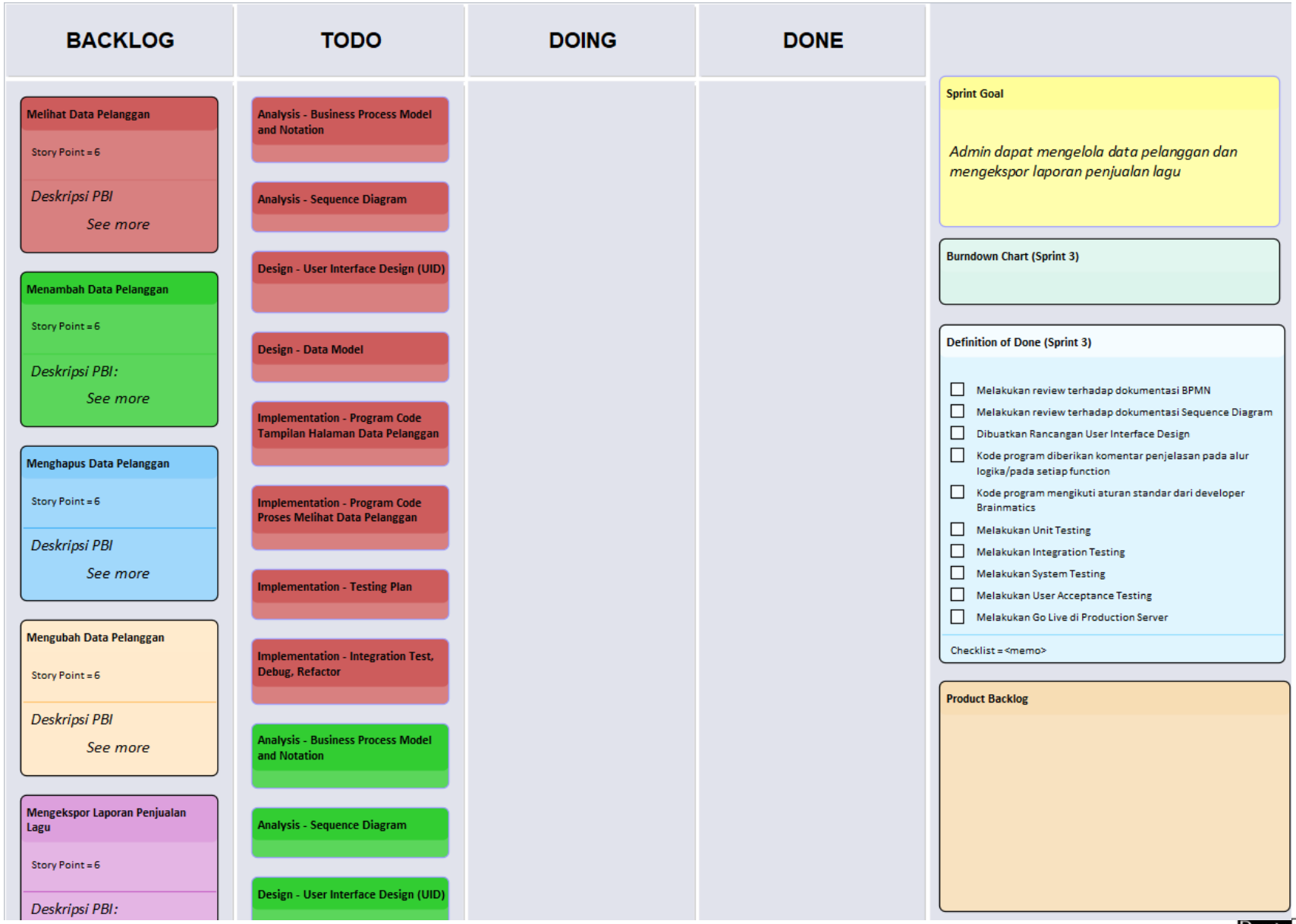
PBI Prioritas Kelima

Hasil Sprint Planning (Sprint 3)

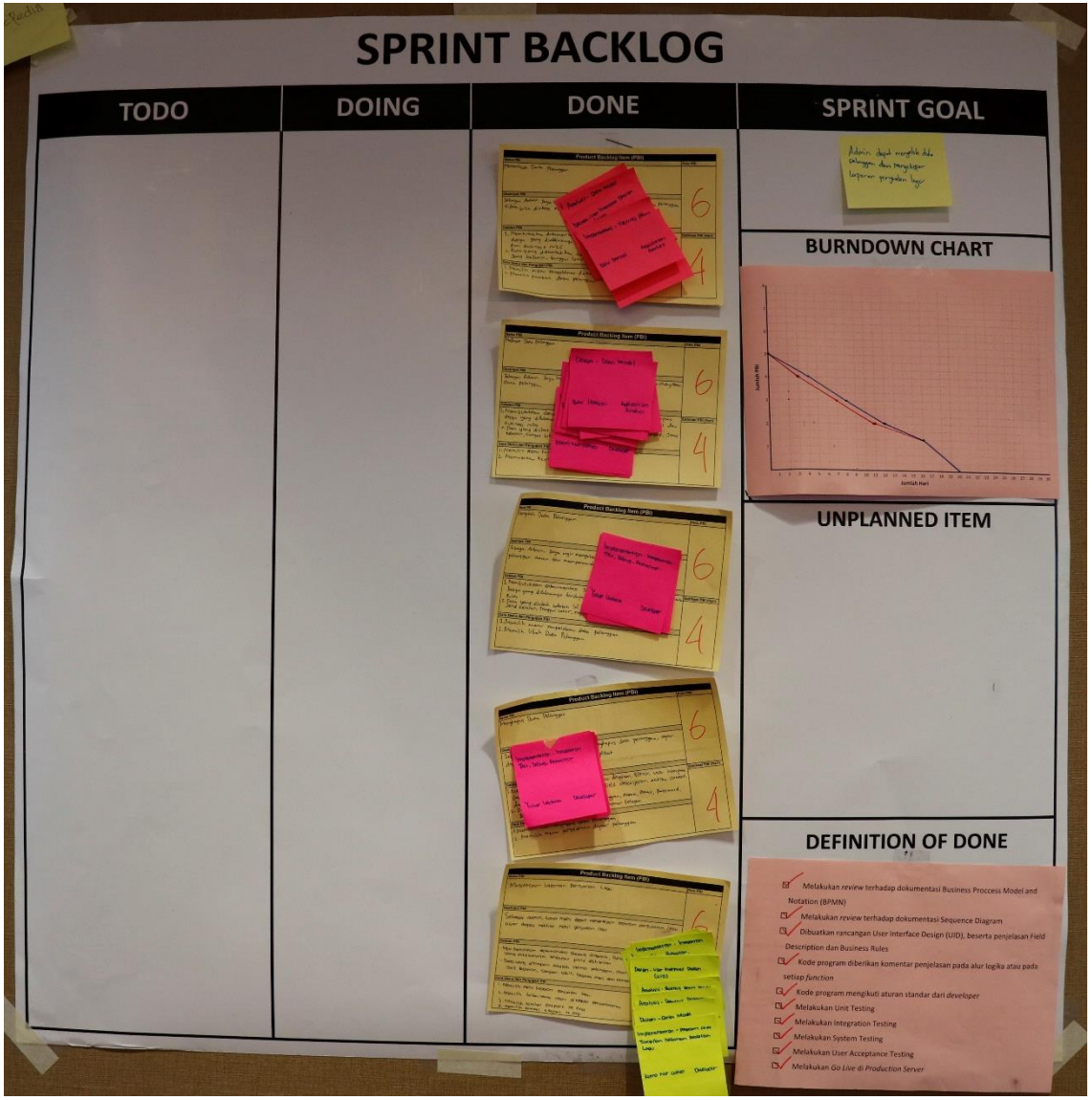


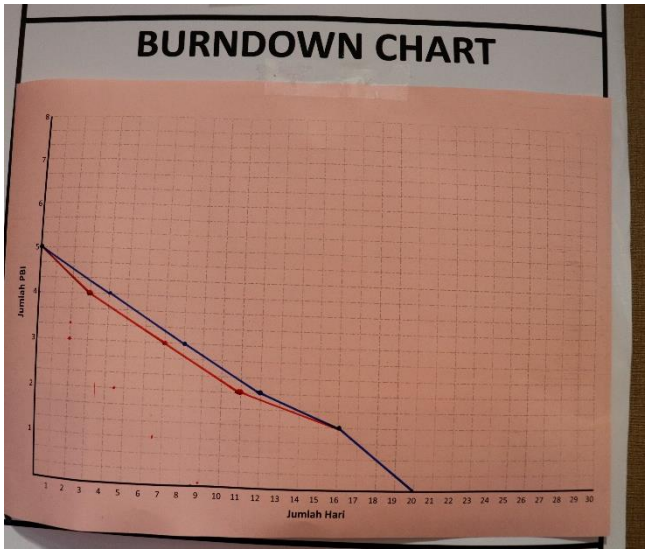


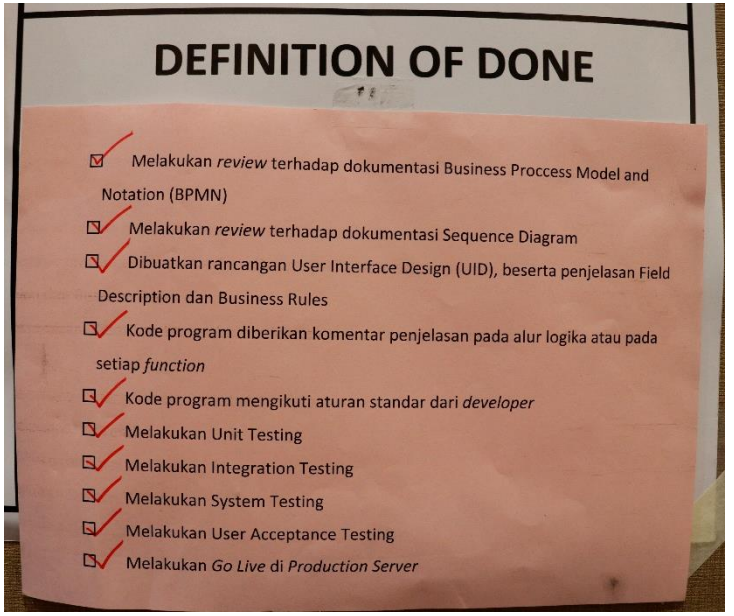
Hasil Sprint Planning (Sprint 3)



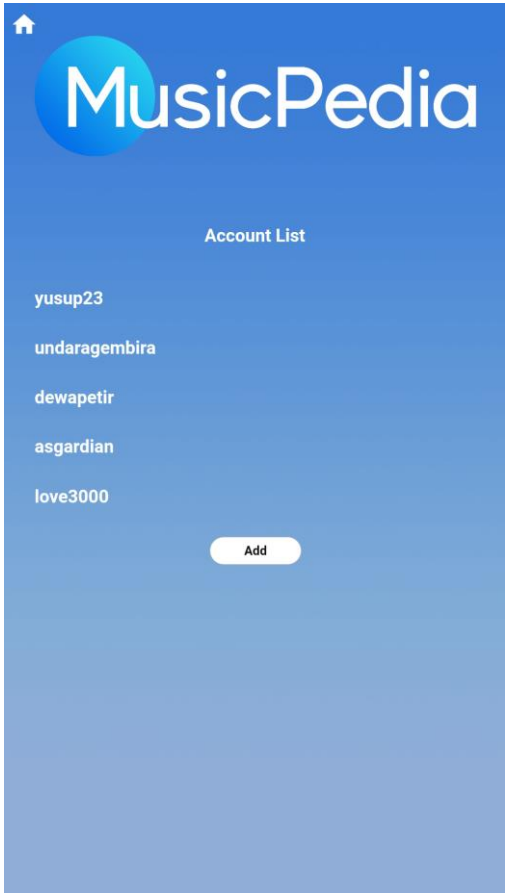
Sprint Execution (Sprint 3)



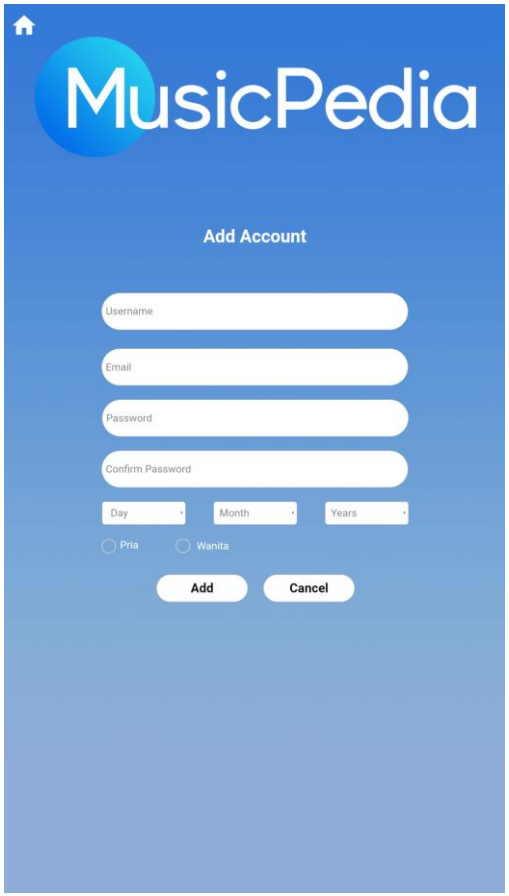




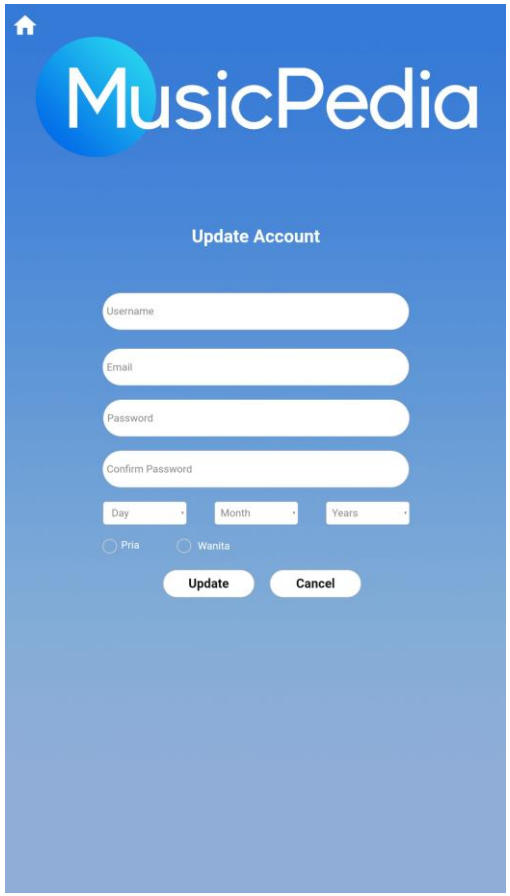
Hasil Sprint Execution (Sprint 3)



Melihat Data Pelanggan



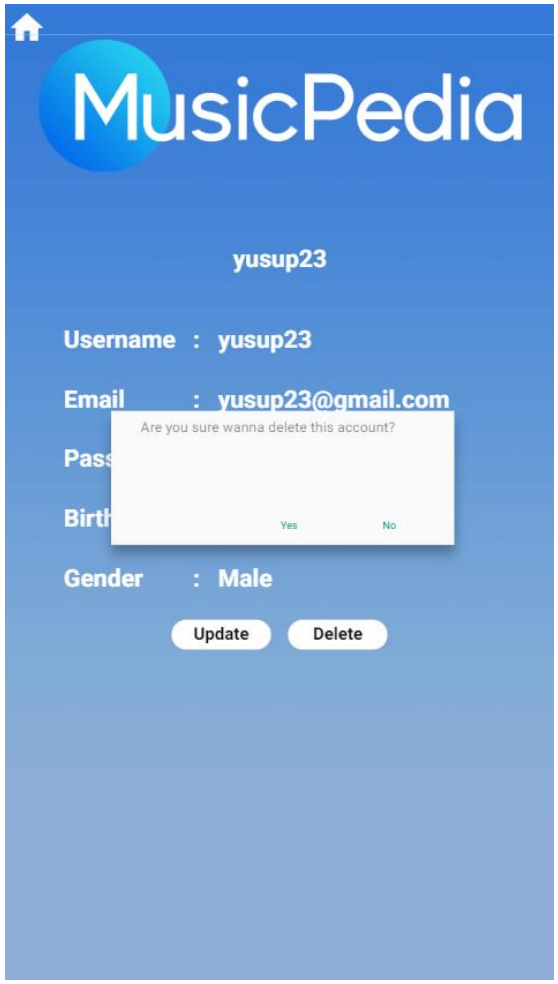
Menambah Data Pelanggan



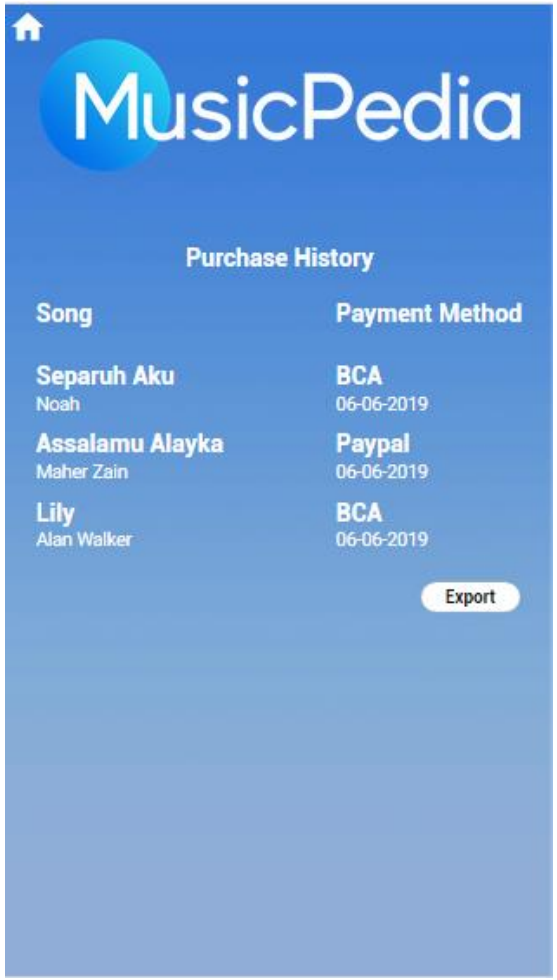
Mengubah Data Pelanggan

Sprint 3 Goal: Admin dapat mengelola data pelanggan dan mengekspor laporan penjualan lagu

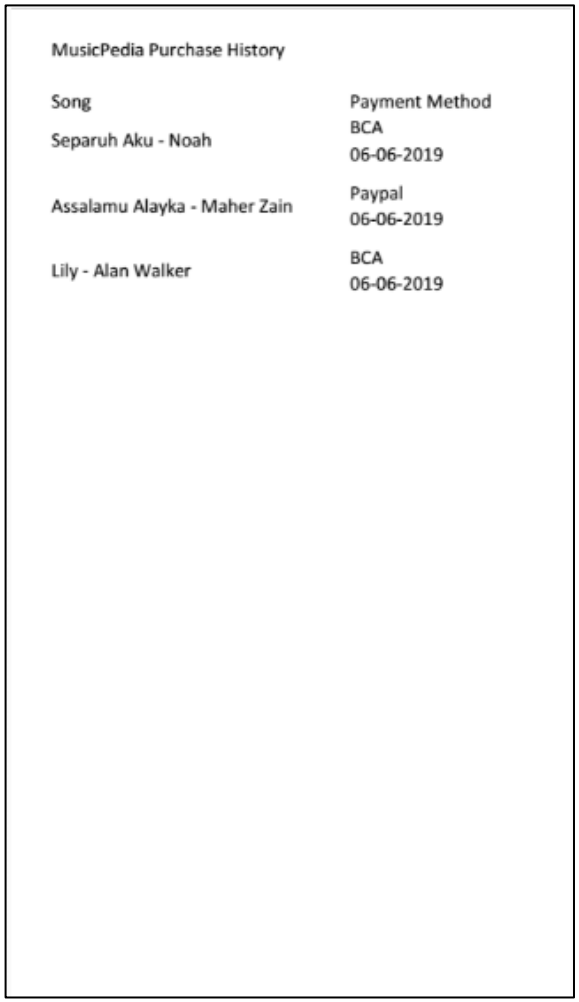
Hasil Sprint Execution (Sprint 3)



Menghapus Data Pelanggan

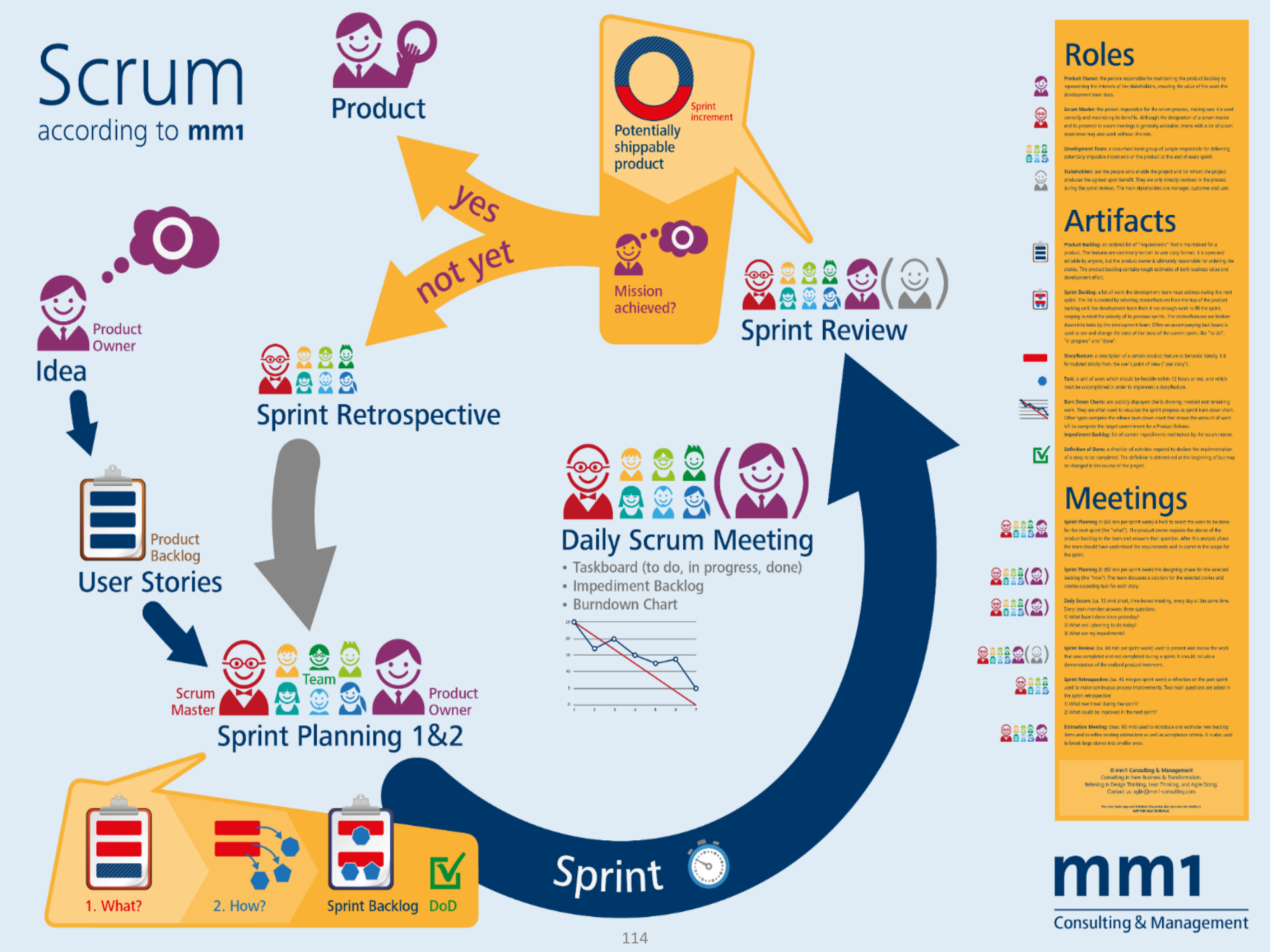


Mengekspor Laporan Penjualan Lagu



Hasil Ekspor Laporan Penjualan Lagu

Scrum
according to **mm1**



Roles

Product Owner: the person responsible for maintaining the product backlog by representing the interests of the stakeholders, ensuring the value of the work the development team does.

Scrum Master: the person responsible for the scrum process, making sure it is used correctly and maximizing its benefits. Although the designation of a scrum master and its presence in scrum meetings is generally advisable, teams with a lot of scrum experience may also work without this role.

Development Team: a cross-functional group of people responsible for delivering potentially shippable increments of the product at the end of every sprint.

Stakeholders: are the people who enable the project and for whom the project produces the agreed-upon benefit. They are only directly involved in the process during the sprint reviews. The main stakeholders are managers, customer and user.

Artifacts

Product Backlog: an ordered list of "requirements" that is maintained for a product. The features are commonly written in user story format. It is open and editable by anyone, but the product owner is ultimately responsible for ordering the stories. The product backlog contains rough estimates of both business value and development effort.

Sprint Backlog: a list of work the development team must address during the next sprint. The list is created by selecting stories/features from the top of the product backlog until the development team feels it has enough work to fill the sprint, keeping in mind the velocity of its previous sprints. The stories/features are broken down into tasks by the development team. Often an accompanying task board is used to see and change the state of the tasks of the current sprint, like "to do", "in progress" and "done".

Story/feature: a description of a certain product feature or behavior. Ideally, it is formulated strictly from the user's point of view ("user story").

Task: a unit of work which should be feasible within 12 hours or less, and which must be accomplished in order to implement a story/feature.

Burn Down Charts: are publicly displayed charts showing invested and remaining work. They are often used to visualize the sprint progress as sprint burn down chart. Other types comprise the release burn down chart that shows the amount of work left to complete the target commitment for a Product Release.

Impediment Backlog: list of current impediments maintained by the scrum master.

Definition of Done: a checklist of activities required to declare the implementation of a story to be completed. The definition is determined at the beginning of but may be changed in the course of the project.

Meetings

Sprint Planning 1: (60 min per sprint week) it held to select the work to be done for the next sprint (the "what"). the product owner explains the stories of the product backlog to the team and answers their question. After this analysis phase the team should have understood the requirements and its comes in the scope for the sprint.

Sprint Planning 2: (60 min per sprint week) the designing phase for the selected backlog (the "how"). The team discusses a solution for the selected stories and creates according task for each story.

Daily Scrum: (ca. 15 min) short, time boxed meeting, every day at the same time. Every team member answers three questions:
1) What have I done since yesterday?
2) What am I planning to do today?
3) What are my impediments?

Sprint Review: (ca. 60 min per sprint week) used to present and review the work that was completed and not completed during a sprint. It should include a demonstration of the realized product increment.

Sprint Retrospective: (ca. 45 min per sprint week) a reflection on the past sprint used to make continuous process improvements. Two main questions are asked in the sprint retrospective:
1) What went well during the sprint?
2) What could be improved in the next sprint?

Estimation Meeting: (max. 60 min) used to introduce and estimate new backlog items and to refine existing estimations as well as acceptance criteria. It is also used to break large stories into smaller ones.

mm1
Consulting & Management

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

Boards

Trello

Thesis Project

Intelligent Systems ☆ Private

1. LITERATURE REVIEWS

[README] PERATURAN BIMBINGAN TESIS
3 2

Dedi Sutopo
4 1 14/19

Jimmi Adrian
2 1

Singgih Ardianto
11

Tri Santoso
3 2

Mia Rosmiati
3 1

Esti Mulyani
1 2 0/1

Bambang Ismanto
1 1 0/3

2. RESEARCH PROBLEMS (RP) AND QUESTIONS (RQ)

[README] PENENTUAN MASALAH PENELITIAN
11 4

Jaya Chandra
3 2 6/70

Sukmawati Anggraeni Putri
7 1 0/18

Al Riza Khadafy
2 3 0/1

Anjar Nugroho
2 1 0/2

Hilda Rachmi
2 1

Norma Yunita
2 2 0/5

Rakhmat Purnomo
18 4

3. PROPOSED MODEL DEVELOPMENT

[README] STRUKTUR TESIS DAN KESALAHAN PENULISAN TESIS
3

[README] PANDUAN UMUM EYD 2009
1

Erna S.R
2 4 0/1

Ibnu Fajar
11 3 2/12

Muhammad Firman Suwarya
13 4 6/25

Arif Setiawan
4 1 7/18

Add a card...

4. EXPERIMENTS AND RESULT ANALYSIS

[README] JENIS-JENIS EKSPERIMEN
1

Heri Sutrisno
5 4 0/4

Slamet Sucipto
29 3 27/29

Misno
11 1 48/54 Feb 15

Marsiska Ariesta Putri
1 1 Apr 18, 2014

Safuan
18 2 15/27

Dian Pratama Putra
35 3 24/45

Add a card...

5. THESIS FINISHING AND DEFENSE

[README] PERSIAPAN PRESENTASI SIDANG TESIS
1

Romi Satria Wahono [CONTOH CARD]
12 3 0/16

Ade Hikmah F. Udjir
14 0/5

Jeniva Nilna
2 5

Wiwik Rachmanto

Add a card...

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

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 dikembangakn 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 Analyst adalah orang yang menganalisis kebutuhan bisnis, mengidentifikasi peluang untuk perbaikan, mendesain software (sistem) untuk mengimplementasikan idenya
- 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

2. Systems Planning

2.1 Identifikasi *Business Value* dengan *System Request*

2.2 Analisis Kelayakan Pengembangan Software

2.3 Estimasi Usaha Pengembangan Software

Siklus Pengembangan Software:

Alur, Peran, dan Tahapan (*Deliverable*) *(Tilley, 2012) (Dennis, 2016) (Valacich, 2017)*

1. **User/Product Owner** membawa permintaan kebutuhan (perubahan) software (System Request) ke System Analyst
2. **System Analyst** membuat analisis kelayakan (Feasibility Analysis) dari System Request tersebut

Planning
(System Proposal)

3. Setelah dinyatakan layak, System Analyst melakukan analysis dan design, dan hasilnya adalah System Specification
   - **Business Analyst** membantu System Analyst memahami proses bisnis dari software yang akan dibangun

Analysis and Design
(System Specification)

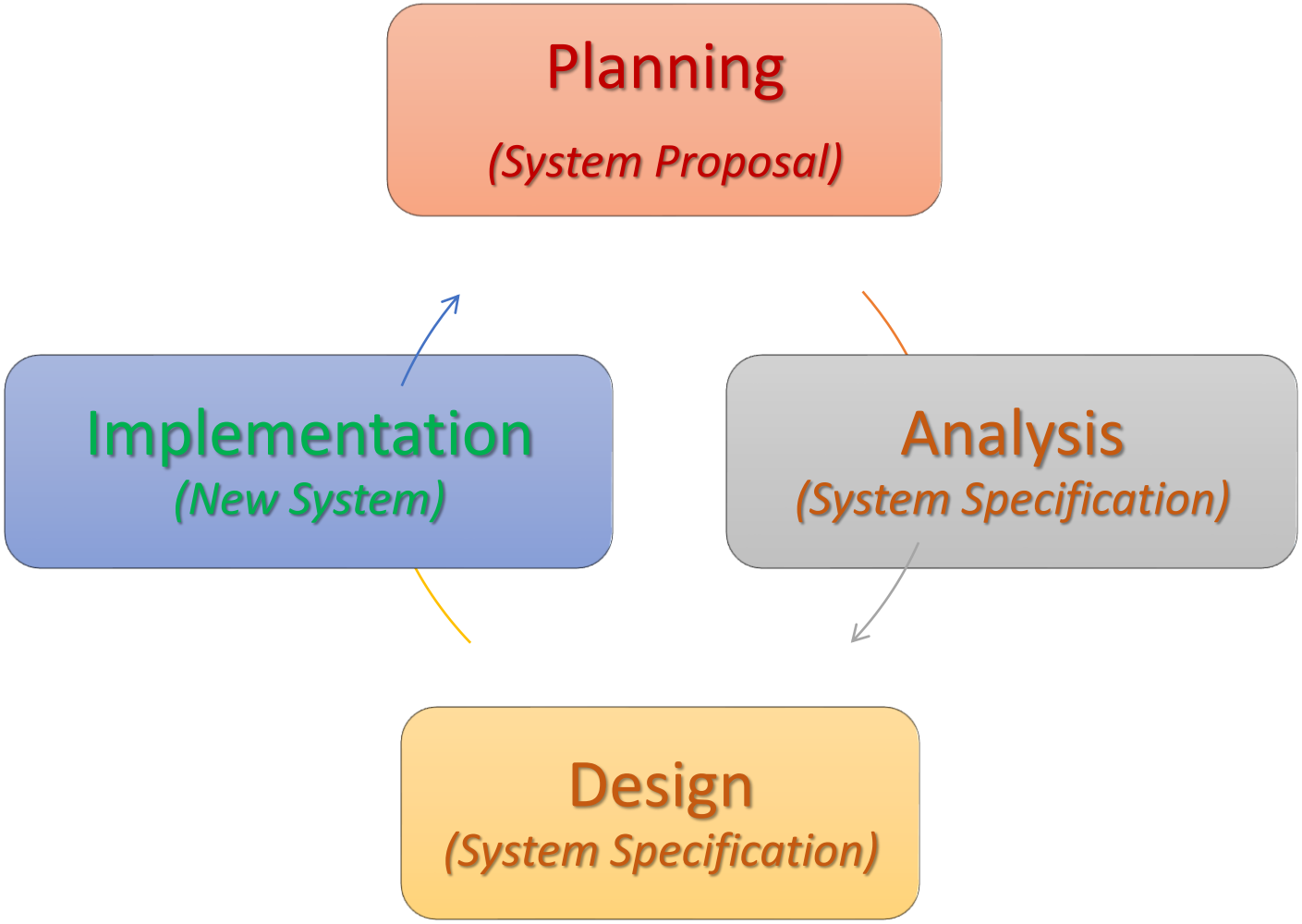
4. System Specification diserahkan oleh System Anayst ke **Programmer** untuk dilakukan Konstruksi (*Coding*)
5. Hasil Konstruksi berupa Kode Program diserahkan ke **Software Tester** untuk dilakukan Pengujian (*Unit, Integration, System, User Acceptance Testing*)
6. Instalasi (*delivery*) software dan manajemen perubahan
   - Software = Kode Program + Dokumentasi (Pengembangan dan Penggunaan)

Implementation
(Software)

7. Siklus kembali ke 1 apabila ada permintaan perubahan (Permintaan Perubahan Software)

Maintenance
(Updated Software)

Siklus Pengembangan Software



(Tilley, 2012) *(Dennis, 2016)* *(Valacich, 2017)*

2.1 Identifikasi *Business Value* dengan *System Request*

Kapan Project Software Diinisiasi?

- Ketika ada seseorang yang melihat peluang menciptakan business value dengan menggunakan software dan teknologi informasi
  - Seseorang itu kemudian membuat System Request

- System Request kemudian dianalisis kelayakannya (*Feasibility Analysis*) untuk menentukan apakah akan diteruskan projectnya atau tidak
  - Di dalam analisis kelayakan, dilakukan juga penghitungan usaha pengembangan software (butuh berapa bulan dan berapa orang)
    - sehingga analisis kelayakan ekonomi bisa dibangun dengan akurat

Planning

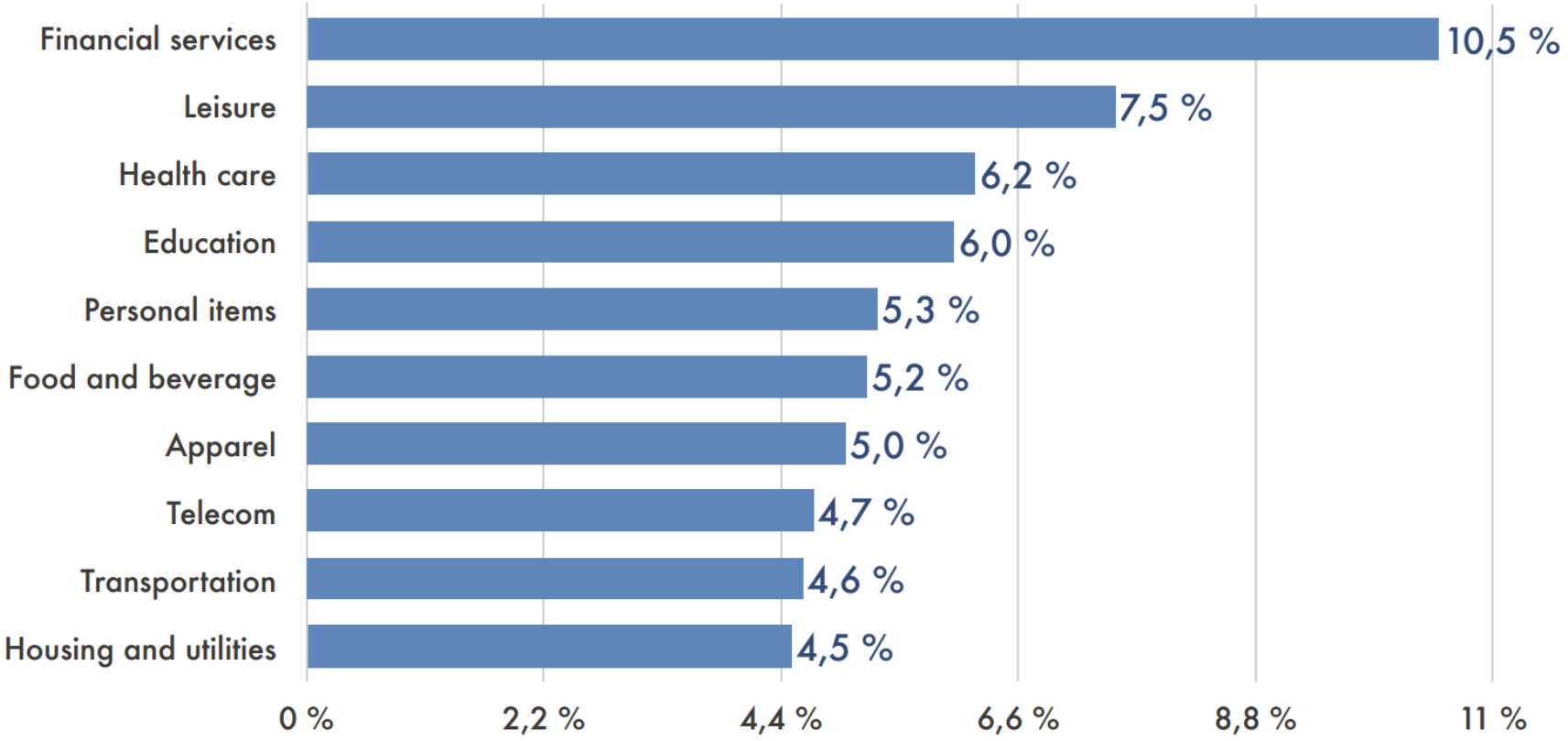
System Request
(Business Value Identification)

Lower Cost	*Increase Productivity*	*Increase Profit*

Feasibility Analysis

Technical *(Capabilities)*	*Economic* *(ROI, BEP)*	*Organizational* *(Goals, Core Business)*

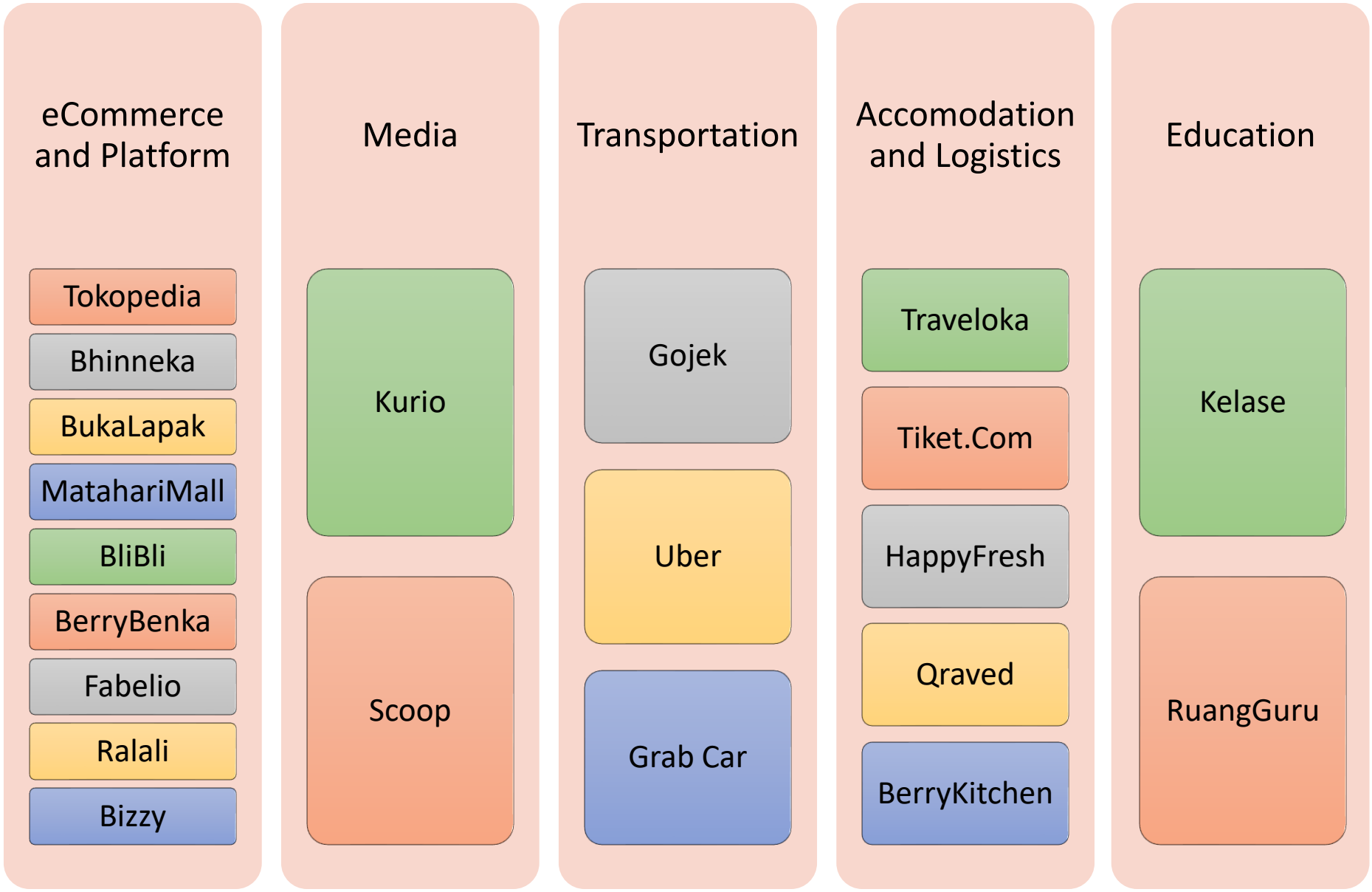
Consumer Spending in Indonesia (McKinsey, 2013)



RAPID URBANIZATION, RISING INCOME LEVELS, FAVORABLE DEMOGRAPHIC PATTERNS AND CHANGING LIFESTYLE TRENDS ARE JUST SOME OF THE FACTORS THAT ARE BOOSTING CONSUMER SPENDING IN INDONESIA

Source: McKinsey 2013

Indonesia Software Innovation Maps



System Request

Elemen	Deskripsi	Contoh
Business Need	• The business-related reason for initiating the software development project • Reason prompting the project, and why the project should be funded?	• Meningkatkan penjualan • Mengurangi biaya operasional • Meningkatkan produktifitas pegawai • Meningkatkan kualitas layanan • Mengurangi kebocoran/kecurangan • Mengurangi cacat produksi • Meningkatkan efisiensi kerja
Business Value	• The benefits that the software will create for the organization • Tangible value (a quantiable value) and intangible value (intuitive believe)	• Peningkatan penjualan 3% • Pengurangan biaya operasional 10% • Peningkatan produktifitas pegawai 10% (dihitung rasio pekerjaan dan gaji) • Pengurangan cacat produksi 20% • Peningkatan efisiensi kerja 20%
Business Requirements	• The business capabilities that software will provide • Can be replaced by Use Case Diagram	• Fitur registrasi, login, dan logout • Fitur pengelolaan data pengguna • Fitur pengiriman notifikasi otomatis • Fitur cetak laporan bulanan dan tahunan

Software Berkualitas?

Software quality is *(IEEE, 1991)*:

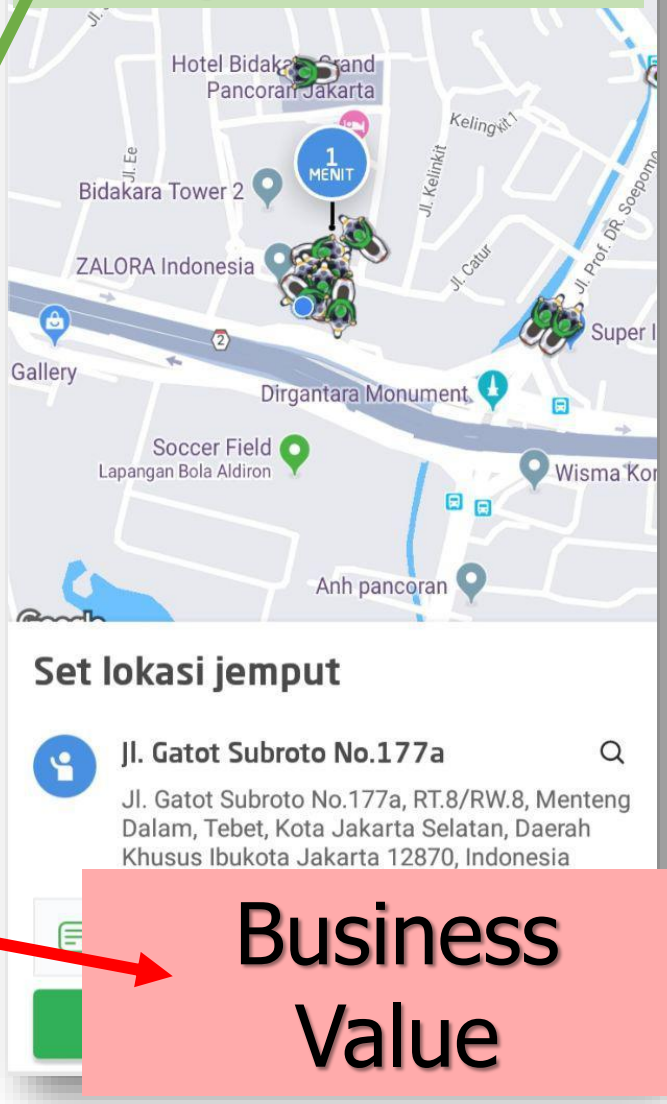
1. The degree to which a system, component, or

Sesuai Kebutuhan

2. The degree to which a system, component, or process meets customer

Ada Keuntungan

Business Requirements



Business Value

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 2. Meningkatkan efisiensi penanganan masalah pelanggan melalui internet

Business Requirements:

Sistem yang mendukung penjualan musik secara online. Fitur-fitur yang harus ada:

1. Fitur Pencarian Produk
2. Fitur Pencarian Toko yang Menyediakan Stok Produk
3. Fitur Pemesanan Produk Melalui Toko yang Menyediakan
4. Fitur Pembayaran dengan Berbagai Pilihan Pembayaran

Business Value:

Intangible Value:

- Meningkatkan kenyamanan dan kepuasan pelanggan
- Meningkatkan brand recognition tentang perusahaan di dunia Internet

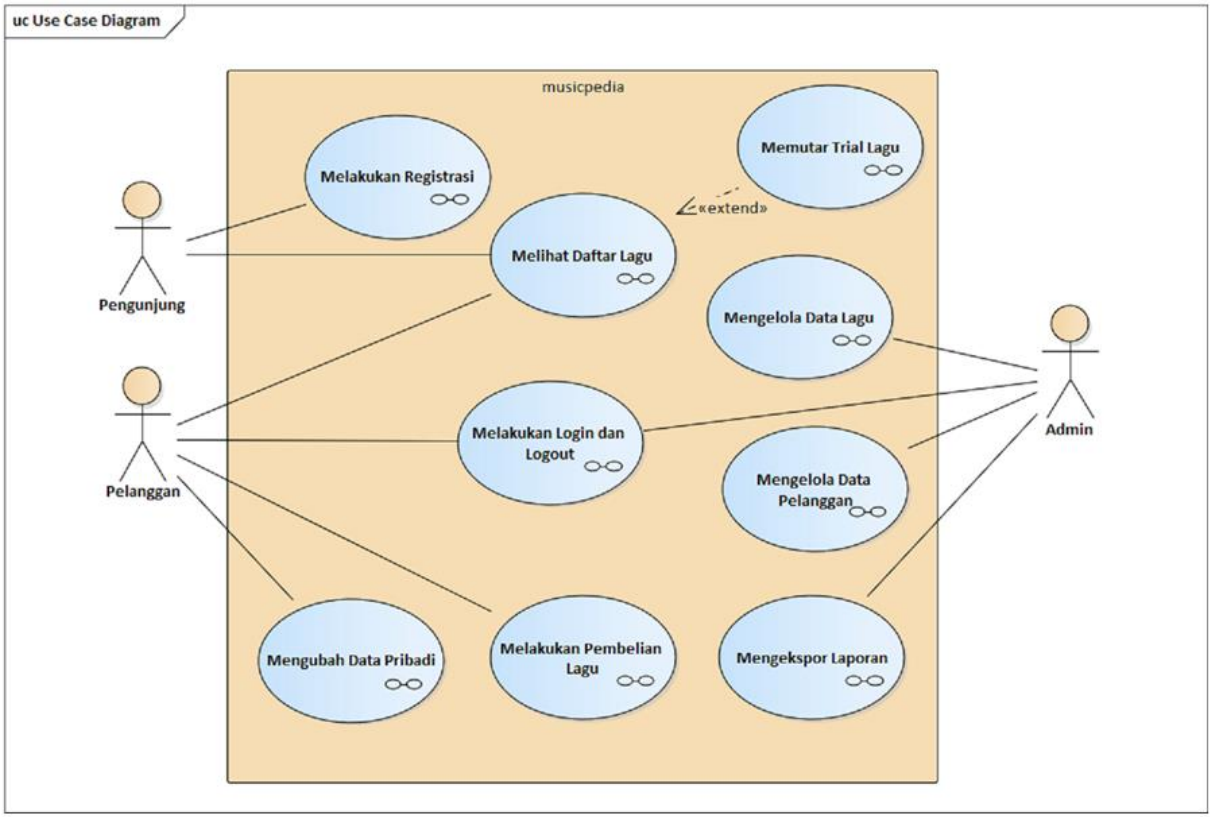
Tangible Value:

1. Meningkatkan penjualan dari pelanggan baru lewat Internet:
   - Rp 400 juta peningkatan penjualan dari pelanggan baru dan Rp 600 juta dari pelanggan lama
2. Mengurangi biaya operasional untuk menangani komplain dari pelanggan
   - Rp 100 juta pengurangan tahunan biaya telepon untuk menangani pelanggan

Software Request

musicpedia				
Date	26 Oktober 2018			
Description	Musicpedia adalah aplikasi layanan download musik & audio dimana saja dan kapan saja, menawarkan akses lengkap ke jutaan lagu dari semua artis papan atas di industri musik barat maupun musik lokal, mendengarkan musik baru dan top musik dunia dalam bentuk audio mp3 secara offline			
Project Sponsor	Wahyu Utomo, VP Business Development, PT Musika Indonesia			
Business Need	1. Tidak Setuju	2. Ragu-Ragu	3. Setuju	4. Sangat Setuju
Aplikasi yang dikembangkan mampu meningkatkan pendapatan perusahaan?	☐	☐	☐	☑
Aplikasi yang dikembangkan mampu mengurangi biaya operasional perusahaan?	☐	☐	☑	☐
Aplikasi yang dikembangkan mampu meningkatkan produktifitas kerja pegawai?	☐	☑	☐	☐
Aplikasi yang dikembangkan mampu meningkatkan nilai tambah perusahaan yang bersifat intangible?	☐	☐	☑	☐
Business Value	Intangible Value: a. Meningkatkan brand recognition perusahaan di dunia internet b. Meningkatkan produktivitas kerja pegawai dan mengurangi kuantitas pegawai Tangible Value: a. Mengurangi biaya operasional perusahaan: - Sewa ruangan: Rp120.000.000,- - Biaya komunikasi: Rp6.000.000,- b. Meningkatkan penjualan musik: Rp400.000.000,-			

Business Requirements



Change Request

Rental System				
Date	25 Oktober 2018			
Module Name	Inquiry Rental Invoice			
Type of Change Request	☐ Enchancement	☑ Defect		
Description of Request	Fitur upload pada module Inquiry Rental Invoice tidak berjalan, muncul error "Upload error, file could not be uploaded for some reason. Please go back and fix this"			
Priority	☐ Low	☐ Medium	☐ High	☑ Mandatory
Reason of Change	Fitur ini adalah fitur utama agar rental invoice bisa dikirimkan ke dealer cabang dan dealer-dealer perwakilan cabang, prioritas perbaikannya mandatory karena setiap sore rental invoice harus dikirimkanke dealer cabang			
Other Module Impacted	• Inquiry Rental Request • Inquiry Rental Order • Retire Rented Items			
Attachments of References	☑ Yes	☐ No		
	Link: https://fifo365-my.sharepoint.com/personal/out900/image5.jpg			

Exercise: Membuat System Request

1. Lihat contoh System Request untuk Sistem Penjualan Musik Online
2. Pikirkan suatu sistem* yang saat ini dibutuhkan oleh perusahaan atau organisasi anda
3. Buat System Request dari sistem tersebut

* System request yang anda buat akan menjadi studi kasus dari pembelajaran kita selama course berlangsung

Contoh Peningkatan Produktifitas Kerja di Pekerjaan Analisis Data dan Informasi

- Gaji Pegawai: Rp. 10.000.000
- Pekerjaan analisis data dan informasi hanya 25% dari semua pekerjaan yang menjadi kewajiban pegawai tersebut (nilai total pekerjaan bulanan Rp. 2.500.000)
- Kuantitas pekerjaan perbulan:
  - 5 pola hasil analisis data dan informasi (nilai 1 pekerjaan Rp. 500.000)
- Kuantitas pekerjaan perbulan dengan adanya aplikasi baru
  - 10 pola hasil analisis data dan informasi (nilai 1 pekerjaan Rp. 250.000)

- Benefit berupa peningkatan produktifitas di pekerjaan analisis data dan informasi dengan adanya aplikasi:
  - Perbulan Rp. 250.000 x 5 pekerjaan = Rp. 1.250.000 (pertahun Rp. 15.000.000)

Contoh Peningkatan Produktifitas Kerja di Pekerjaan Analisis Beban Kerja

- Gaji Pegawai: Rp. 10.000.000
- Pekerjaan analisis beban kerja satu-satunya job description (uraian jabatan) dari pegawai tersebut
  (nilai total pekerjaan bulanan Rp. 10.000.000)
- Kuantitas pekerjaan perbulan:
  - 10 hasil analisis beban kerja
    (nilai 1 pekerjaan Rp. 1000.000)
- Kuantitas pekerjaan perbulan dengan adanya aplikasi baru
  - 20 hasil analisis beban kerja
    (nilai 1 pekerjaan Rp. 500.000)

- Benefit berupa peningkatan produktifitas di pekerjaan analisis beban kerja dengan adanya aplikasi:
  - Perbulan Rp. 500.000 x 10 pekerjaan = Rp. 5000.000
    (pertahun Rp. 60.000.000)

Contoh Pengurangan Kecurangan Pembayaran Pajak dengan Aplikasi

- Kecurangan pembayaran pajak: Rp. 100.000.000.000 pertahun
- Dengan adanya aplikasi, akan mengurangi kecurangan penerimaan pajak 20%
- Benefit penanganan kecurangan pembayaran pajak di Rp. 20.000.000.000

2.2 Analisis Kelayakan Pengembangan Software

Siklus Pengembangan Software:

Alur, Peran, dan Tahapan (*Deliverable*) *(Tilley, 2012) (Dennis, 2016) (Valacich, 2017)*

1. **User/Product Owner** membawa permintaan kebutuhan (perubahan) software (System Request) ke System Analyst
2. **System Analyst** membuat analisis kelayakan (Feasibility Analysis) dari System Request tersebut

Planning
(System Proposal)

3. Setelah dinyatakan layak, System Analyst melakukan analysis dan design, dan hasilnya adalah System Specification
   - **Business Analyst** membantu System Analyst memahami proses bisnis dari software yang akan dibangun

Analysis and Design
(System Specification)

4. System Specification diserahkan oleh System Anayst ke **Programmer** untuk dilakukan Konstruksi (*Coding*)
5. Hasil Konstruksi berupa Kode Program diserahkan ke **Software Tester** untuk dilakukan Pengujian (*Unit, Integration, System, User Acceptance Testing*)
6. Instalasi (*delivery*) software dan manajemen perubahan
   - Software = Kode Program + Dokumentasi (Pengembangan dan Penggunaan)

Implementation
(Software)

7. Siklus kembali ke 1 apabila ada permintaan perubahan (Permintaan Perubahan Software)

Maintenance
(Updated Software)

Template Analisis Kelayakan

Technical Feasibility: Apakah Kita Mampu Membangun dan Menggunakan?

1. Familiarity with Application: Pemahaman pengembang dan pengguna terhadap aplikasi
2. Familiarity with Technology: Pemahaman pengembang dan pengguna terhadap teknologi yang mendukung aplikasi
3. Project Size: Jumlah pengembang dan waktu yang dibutuhkan
4. Compatibility: Kompatibilitas dengan sistem yang ada di organisasi

Economic Feasibility: Apakah Menguntungkan untuk Dibangun?

1. Break-even Point (BEP): Waktu balik modal
2. Return on Investment (ROI): Persentase pengembalian investasi

Organizational Feasibility: Kalau Kita Bangun, Apakah Akan Digunakan?

1. Anggota tim
2. Apakah software selaras dengan visi dan misi organisasi?
3. Apakah software sesuai dengan tugas dan fungsi unit kerja organisasi?
4. Apakah software mengautomasi proses bisnis unit kerja organisasi?

Technical Feasibility

1. Familiarity with application	• Pengguna familier terhadap pengoperasian software sejenis ini • Pengembang familier terhadap pengembangan software sejenis ini
2. Familiarity with technology	• Pengguna familier dengan teknologi pendukung aplikasi • Pengembang familier dengan platform, bahasa pemrograman dan tool IDE untuk pengembangan
3. Project size	• Jumlah pengembang yang dibutuhkan (Man Month) • Waktu yang dibutuhkan (Month)
4. Compatibility	• Kompatibilitas dengan sistem yang ada di organisasi

Economic Feasiblity

Calculation	Definition	Formula
Present Value (PV)	The amount of an investment today compared to that same amount in the future, taking into account inflation and time.	$\frac{\text{Amount}}{(1 + \text{interest rate})^n}$ n = number of years in future
Net Present Value (NPV)	The present value of benefit less the present value of costs.	PV Benefits – PV Costs
Return on Investment (ROI)	The amount of revenues or cost savings results from a given investment.	$\frac{\text{Total benefits} - \text{Total costs}}{\text{Total costs}}$
Break-Even Point	The point in time at which the costs of the project equal the value it has delivered.	$\frac{\text{Yearly NPV*} - \text{Cumulative NPV}}{\text{Yearly NPV*}}$ *Use the Yearly NPV amount from the first year in which the project has a positive cash flow. Add the above amount to the year in which the project has a positive cash flow.

	2019	2020	2021
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:	**0**	**978,996,084**	**2,070,501,152**
Honor Tim (Planning, Analysis, Design and Implementation)	360,000,000	0	0
Honor Konsultan Infrastruktur Internet	90,000,000	0	0
Total Development Costs:	**450,000,000**	**0**	**0**
Honor Pengelola Web	60,000,000	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**	**-166,251,335**	**429,878,356**
Return on Investment (ROI) di Tahun 3: 26.2%	**429,878,356/1,640,622,796**		**0.262021445**
Break-even Point (BEP): 2.28 tahun	**2 + (596,129,691-429,878,356)/ 596,129,691**		**2.278884507**

System Request (Business Value)

Present Value (PV)

- The amount of an investment today compared to the same amount *n* years in the future
- Taking into account inflation and time

$$PV = \frac{\text{Amount}}{(1 + \text{Interest Rate})^n}$$

$$\frac{537{,}201}{(1+0.03)^5} = 463{,}395$$

	2003	2004	2005	2006	2007	Total
Increased sales	500,000	530,000	561,800	595,508	631,238	
Reduction in customer complaint calls	70,000	70,000	70,000	70,000	70,000	
Reduced inventory costs	68,000	68,000	68,000	68,000	68,000	
Total Benefits:	638,000	668,000	699,800	733,508	769,238	
PV of Benefits:	619,417	629,654	640,416	651,712	663,552	3,204,752
PV of All Benefits:	619,417	1,249,072	1,889,488	2,541,200	3,204,752	
2 Servers @ $125,000	250,000	0	0	0	0	
Printer	100,000	0	0	0	0	
Software licenses	34,825	0	0	0	0	
Server software	10,945	0	0	0	0	
Development labor	1,236,525	0	0	0	0	
Total Development Costs:	1,632,295	0	0	0	0	
Hardware	54,000	81,261	81,261	81,261	81,261	
Software	20,000	20,000	20,000	20,000	20,000	
Operational labor	111,788	116,260	120,910	125,746	130,776	
Total Operational Costs:	185,788	217,521	222,171	227,007	232,037	
Total Costs:	1,818,083	217,521	222,171	227,007	232,037	
PV of Costs:	1,765,129	205,034	203,318	201,693	200,157	2,575,331
PV of All Costs:	1,765,129	1,970,163	2,173,481	2,375,174	2,575,331	
Total Project Benefits—Costs:	(1,180,083)	450,479	477,629	506,501	537,201	
Yearly NPV:	(1,145,712)	424,620	437,098	450,019	463,395	629,421
Cumulative NPV:	(1,145,712)	(721,091)	(283,993)	166,026	629,421	

Return on Investment (ROI)

The amount of revenue or cost savings results from a given investment

$$PV = \frac{\text{Total Benefits - Total Costs}}{\text{Total Costs}}$$

$$\frac{3{,}204{,}752 - 2{,}575{,}331}{2{,}575{,}331} = 0.2444$$

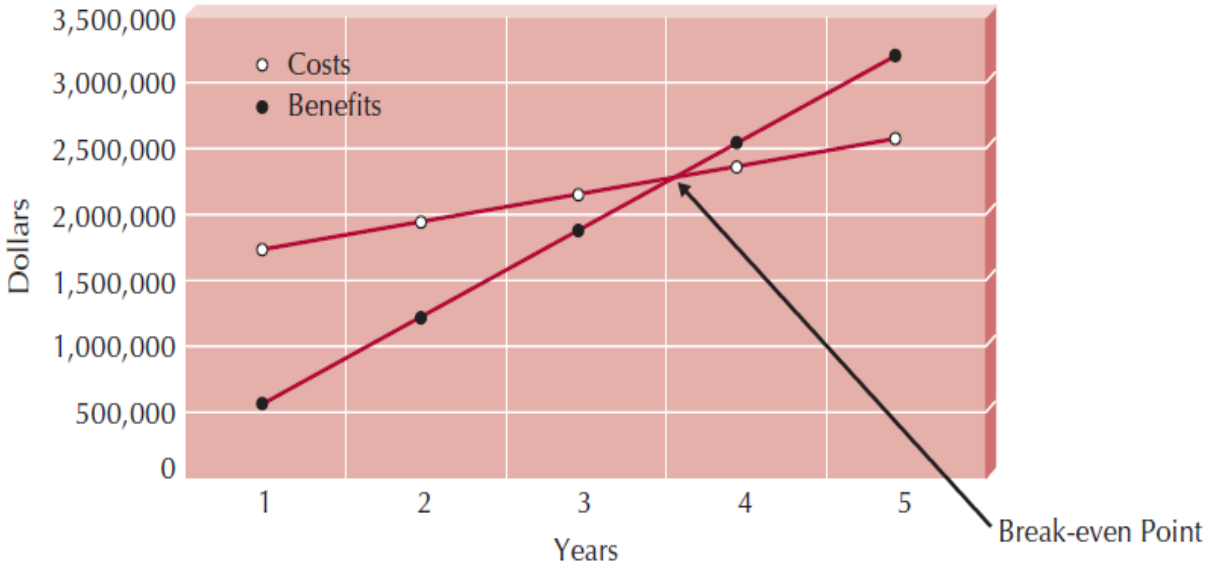
	2003	2004	2005	2006	2007	Total
Increased sales	500,000	530,000	561,800	595,508	631,238	
Reduction in customer complaint calls	70,000	70,000	70,000	70,000	70,000	
Reduced inventory costs	68,000	68,000	68,000	68,000	68,000	
Total Benefits:	638,000	668,000	699,800	733,508	769,238	
PV of Benefits:	619,417	629,654	640,416	651,712	663,552	3,204,752
PV of All Benefits:	619,417	1,249,072	1,889,488	2,541,200	3,204,752	
2 Servers @ $125,000	250,000	0	0	0	0	
Printer	100,000	0	0	0	0	
Software licenses	34,825	0	0	0	0	
Server software	10,945	0	0	0	0	
Development labor	1,236,525	0	0	0	0	
Total Development Costs:	1,632,295	0	0	0	0	
Hardware	54,000	81,261	81,261	81,261	81,261	
Software	20,000	20,000	20,000	20,000	20,000	
Operational labor	111,788	116,260	120,910	125,746	130,776	
Total Operational Costs:	185,788	217,521	222,171	227,007	232,037	
Total Costs:	1,818,083	217,521	222,171	227,007	232,037	
PV of Costs:	1,765,129	205,034	203,318	201,693	200,157	2,575,331
PV of All Costs:	1,765,129	1,970,163	2,173,481	2,375,174	2,575,331	
Total Project Benefits—Costs:	(1,180,083)	450,479	477,629	506,501	537,201	
Yearly NPV:	(1,145,712)	424,620	437,098	450,019	463,395	629,421
Cumulative NPV:	(1,145,712)	(721,091)	(283,993)	166,026	629,421	
Return on Investment:	24.44%	(629,421/2,575,331)				
Break-even Point:	3.63 years	(break-even occurs in year 4; [450,019 – 166,026] / 450,019 = 0.63)				
Intangible Benefits:	This service is currently provided by competitors Improved customer satisfaction					

Break Even Point (BEP)

- The point in time when the costs of the project equal the value it has delivered

$$BEP = \frac{\text{Yearly NPV}^* - \text{Cumulative NPV}}{\text{Yearly}^* \text{ NPV}}$$

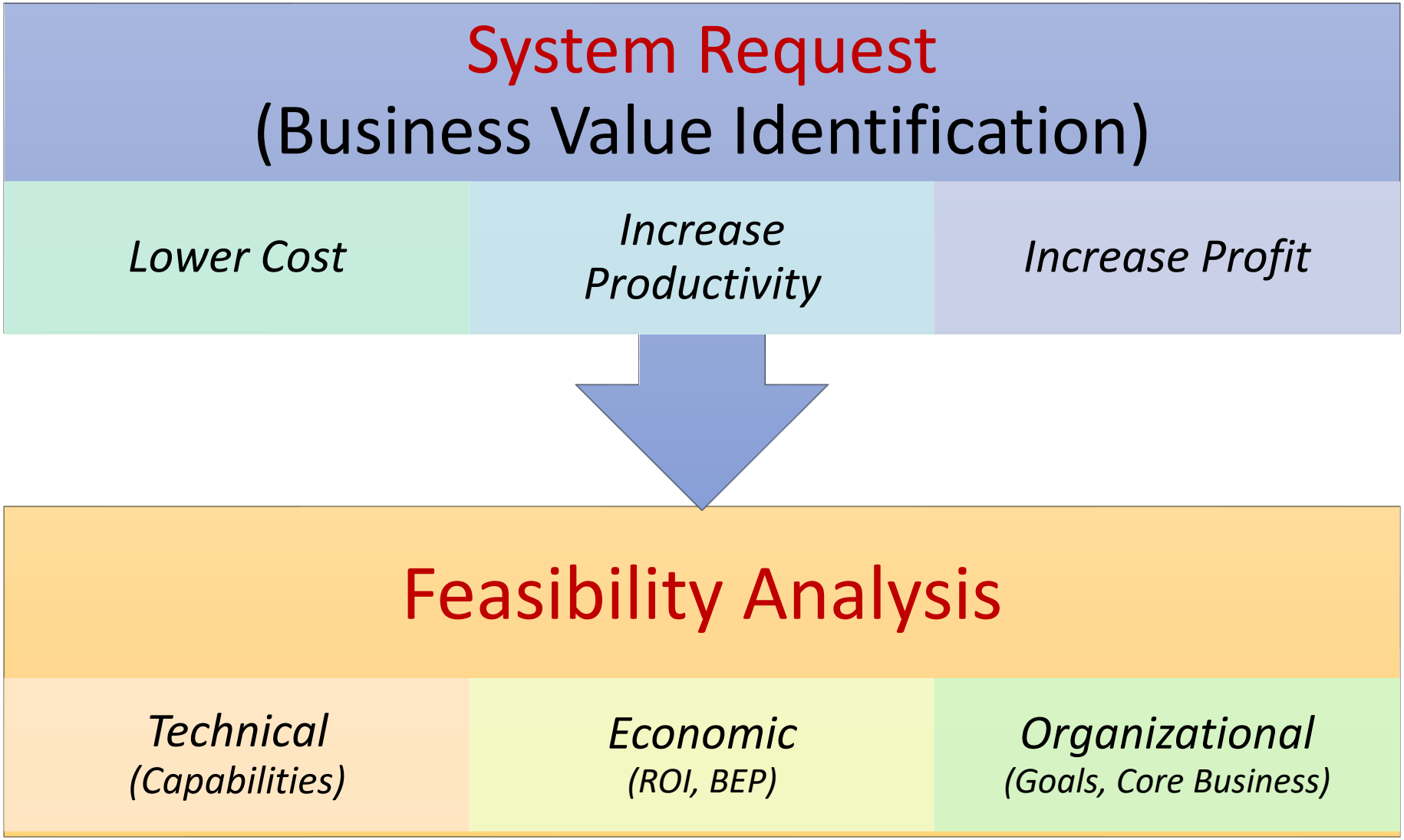
- Use the yearly NPV amount from the first year in which project has positive cash flow



Organizational Feasibility

- Anggota Tim
  - User/Product Owner:
  - Project Manager:
  - Developer (System Analyst/Business Analyst/Programmer/Tester):
- Apakah software selaras dengan visi dan misi organisasi?
- Apakah software sesuai dengan tugas dan fungsi unit kerja organisasi?
- Apakah software mengautomasi proses bisnis unit kerja organisasi?

Planning


System Request
(Business Value Identification)
Lower Cost
Increase Productivity
Increase Profit
Feasibility Analysis
Technical
(Capabilities)
Economic
(ROI, BEP)
Organizational
(Goals, Core Business)

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
2. Memberikan layanan pendukung dengan menggunakan internet

Business Requirements:

Sistem yang mendukung penjualan musik secara online. Fitur-fitur yang harus ada:

1. Fitur Pencarian Produk
2. Fitur Pencarian Toko yang Menyediakan Stok Produk
3. Fitur Pemesanan Produk Melalui Toko yang Menyediakan
4. Fitur Pembayaran dengan Berbagai Pilihan Pembayaran

Business Value:

Intangible Value:

- Meningkatkan kenyamanan dan kepuasan pelanggan
- Meningkatkan brand recognition tentang perusahaan di dunia Internet

Tangible Value:

1. Meningkatkan penjualan dari pelanggan baru lewat Internet:
   - Rp 400 juta peningkatan penjualan dari pelanggan baru dan Rp 600 juta dari pelanggan lama
2. Mengurangi biaya operasional untuk menangani komplain dari pelanggan
   - Rp 100 juta pengurangan tahunan biaya telepon untuk menangani pelanggan

Studi Kelayakan Sistem Penjualan Musik Online

Margaret Mooney dan Alec Adams membuat studi kelayakan untuk pengembangan Sistem Penjualan Musik Online

Kelayakan Teknis

Sistem penjualan musik online layak secara teknis, meskipun memiliki beberapa risiko.

Risiko Berhubungan dengan **Kefamilieran dengan Aplikasi**: Resiko Tinggi

- Divisi Marketing tidak memiliki pengalaman menggunakan sistem penjualan online
- Divisi IT memiliki pemahaman yang baik tentang sistem penjualan offline, akan tetapi tidak berpengalaman mengembangkan sistem penjualan musik online

Risiko Berhubungan dengan **Kefamilieran dengan Teknologi**: Resiko Sedang

- Divisi IT tidak menguasai masalah infrastruktur dan ISP, tetapi akan menyewa konsultan
- Divisi IT cukup familier dengan framework dan IDE yang akan digunakan
- Divisi Marketing tidak memiliki pengalaman menggunakan teknologi Web

Risiko berhubungan dengan **Ukuran Project**: Risiko Rendah

- Perusahaan memiliki total 30 orang pengembang
- Project dikerjakan oleh 5 orang pengembang dengan estimasi waktu 6 bulan

Kompatibilitas dengan sistem dan infrastruktur yang ada: Risiko Rendah

- Sistem pemesanan yang ada sekarang menggunakan *open standard*, jadi sangat kompatibel dengan sistem penjualan berbasis web yang akan dibangun

Kelayakan Ekonomi

Cost benefit analysis telah dilakukan. Sistem Penjualan musik online memiliki peluang yang baik untuk bisa meningkatkan pendapatan perusahaan.

- Return on Investment (ROI) setelah 3 tahun: **26,2%**
- Break-even point (BEP): **2,28 tahun**
- Total keuntungan setelah 3 tahun: **Rp. 429.878.356,-**

Kelayakan Organisasi

- Anggota Tim
  - User/Product Owner: Margaret Mooney, VP Marketing
  - Project Manager: Joko
  - Developer: Wahyu, Mansyur, Haris, Azmi
- Apakah software selaras dengan visi dan misi organisasi? Ya
- Apakah software sesuai dengan tugas dan fungsi unit kerja organisasi? Ya. Tujuan dari pengembangan sistem penjualan musik online adalah meningkatkan penjualan perusahaan. Dan ini selaras dengan KPI marketing yang ke arah peningkatan kuantitas penjualan
- Apakah software mengautomasi proses bisnis unit kerja organisasi? Ya
- Secara organisasi, resikonya rendah.

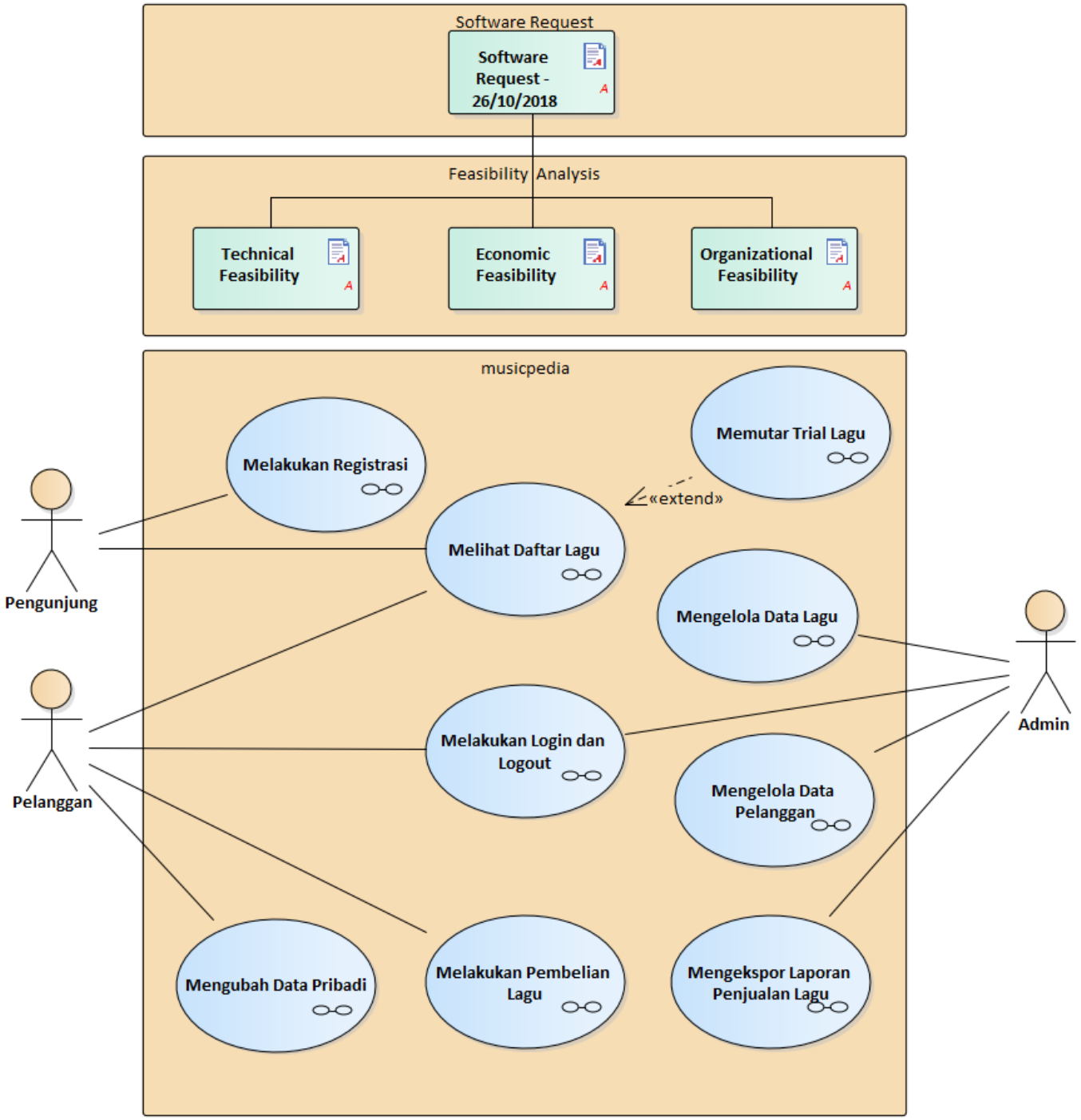
	2016	2017	2018
Peningkatan penjualan dari pelanggan baru	0	400,000,000	500,000,000
Peningkatan penjualan dari pelanggan lama	0	600,000,000	700,000,000
Pengurangan biaya operasional dan telepon	0	100,000,000	100,000,000
Total Benefits:	**0**	**1,100,000,000**	**1,300,000,000**
PV of Benefits:	**0**	**978,996,084**	**1,091,505,068**
PV of All Benefits:	**0**	**978,996,084**	**2,070,501,152**
Honor Tim (Planning, Analysis, Design and Implementation)	360,000,000	0	0
Honor Konsultan Infrastruktur Internet	90,000,000	0	0
Total Development Costs:	**450,000,000**	**0**	**0**
Honor Pengelola Web	60,000,000	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**	**-166,251,335**	**429,878,356**
Return on Investment (ROI) di Tahun 3: 26.2%	**429,878,356/1,640,622,796**		**0.262021445**
Break-even Point (BEP): 2.28 tahun	**2 + (596,129,691-429,878,356)/ 596,129,691**		**2.278884507**

System Request (Business Value)

Exercise: Membuat Feasibility Analysis

1. Lihat contoh Feasibility Analysis untuk Sistem Penjualan Musik Online
2. Perhatikan kembali System Request yang sebelumnya sudah kita buat
3. Buat Feasibility Analysis dari system yang akan kita buat tersebut

4. Kirim file EAPX ke romi@brainmatics.com untuk dilakukan review bersama


Software Request
Software Request - 26/10/2018
Feasibility Analysis
Technical Feasibility
Economic Feasibility
Organizational Feasibility
musicpedia
Melakukan Registrasi
Memutar Trial Lagu
«extend»
Melihat Daftar Lagu
Mengelola Data Lagu
Melakukan Login dan Logout
Mengelola Data Pelanggan
Mengubah Data Pribadi
Melakukan Pembelian Lagu
Mengekspor Laporan Penjualan Lagu
Pengunjung
Pelanggan
Admin

Technical Feasibility

musicpedia				
Date: 26 Oktober 2018				
Penjelasan isian	1. Sangat Kurang	2. Kurang	3. Baik	4. Sangat Baik

Kefamiliaran dengan Aplikasi	1	2	3	4
Pengguna familiar terhadap pengoperasian aplikasi ini.	☐	☐	☑	☐
Pengembang familiar terhadap pengembangan aplikasi ini.	☐	☐	☐	☑
Kefamiliaran dengan Teknologi	**1**	**2**	**3**	**4**
Pengguna familiar dengan teknologi pendukung aplikasi.	☐	☑	☐	☐
Pengembang familiar mengembangkan aplikasi dengan platform, bahasa pemrograman dan tool IDE yang dipilih.	☐	☐	☐	☑
Ukuran Proyek				
Jumlah pengembang yang dibutuhkan.	7 Man/Month			
Waktu yang dibutuhkan dalam mengembangkan aplikasi ini.	6 Month			
Kompatibilitas	**1**	**2**	**3**	**4**
Kebutuhan pengguna terhadap kompatibilitas aplikasi untuk terintegrasi dengan aplikasi lain.	☐	☐	☑	☐
Kompatibilitas aplikasi terhadap teknologi yang ada pada organisasi.	☐	☐	☐	☑

Secara analisi kelayakan teknis, apakah aplikasi layak dikembangkan sesuai kriteria di atas?	☑ Layak	☐ Tidak Layak

Technical Feasibility

Use Case Points

Tahap 1 - Menghitung Person Hours (PH)				
Use Case Points (UCP)		Person Hours Multiplier (PHM)	Person Hours (PH)	
51		20	1020	
51		28	1428	
Tahap 2 - Menghitung Person Month (PM)				
PHM	Person Hours (PH)	Lama Bekerja Perhari	Jumlah Bekerja Sebulan	Person Months (PM)
20	1020	8	22	5.80
	1020	10	26	3.92
28	1428	8	22	8.11
	1428	10	26	5.49
Tahap 3 - Menghitung Time (Month)				
PHM	Formula Penghitung Waktu		Jumlah Bekerja Sebulan	Waktu dalam Bulan (M)
20	3 * PM^(1/3)		22	5.39
			26	4.73
28			22	6.03
			26	5.29

Economic Feasibility

Cost-Benefit Analysis

Tahun	2019	2020	2021	2022
Peningkatan Pendapatan Penjualan Lagu		400,000,000	400,000,000	400,000,000
Pengurangan Biaya Sewa Ruangan		120,000,000	120,000,000	120,000,000
Pengurangan Biaya Komunikasi		6,000,000	6,000,000	6,000,000
Total Benefits	**0**	**526,000,000**	**526,000,000**	**526,000,000**
PV of Benefits	**0**	**468,138,127**	**441,639,743**	**416,641,267**
PV of All Benefits	**0**	**468,138,127**	**909,777,870**	**884,779,394**
Honor Tim (Analysis, Design and Implementation)	250,000,000	120,000,000	120,000,000	120,000,000
Total Development Costs	**250,000,000**	**120,000,000**	**120,000,000**	**120,000,000**
Honor Pengelola Web	72,000,000	72,000,000	72,000,000	72,000,000
Biaya Lisensi Software	10,000,000	10,000,000	10,000,000	10,000,000
Hardware upgrades	50,000,000	50,000,000	50,000,000	50,000,000
Biaya Komunikasi	1,000,000	1,000,000	1,000,000	1,000,000
Biaya Marketing	50,000,000	50,000,000	50,000,000	50,000,000
Total Operational Costs	**183,000,000**	**183,000,000**	**183,000,000**	**183,000,000**
Total Costs	**433,000,000**	**303,000,000**	**303,000,000**	**303,000,000**
PV of Costs	**408,490,566**	**269,668,921**	**153,650,329**	**144,953,140**
PV of all Costs	**408,490,566**	**678,159,487**	**831,809,816**	**976,762,957**
Total Project Costs Less Benefits	**-433,000,000**	**223,000,000**	**223,000,000**	**223,000,000**
Yearly NPV	**-408,490,566**	**198,469,206**	**187,235,100**	**176,636,887**
Cumulative NPV	**-408,490,566**	**-210,021,360**	**-22,786,260**	**153,850,627**
Return on Investment (ROI)	**-100.00%**	**-0.309693168**	**-0.027393593**	**0.15751071**
Break-even Point (BEP)				**3.129000574**

Organizational Feasibility

musicpedia			
Date	**26 Oktober 2018**		
Anggota Tim			
User/Product Owner	Wahyu Utomo		
Project Manager	Haris Dermawan		
System Analyst	Risa Dhani Horasman Purba		
Business Analyst	Mulyana		
Programmer	Achmad Fatkarrofiqi		
Tester	Januar Sapareza		
Apakah aplikasi ini mendukung visi dan misi organisasi?			
Ya			
Apakah aplikasi ini sesuai dengan tugas, fungsi dan KPI unit kerja anda?			
Ya			
Apakah aplikasi ini selaras dengan proses bisnis unit kerja anda?			
Ya			
Secara analisis kelayakan organisasi, apakah aplikasi layak dikembangkan sesuai kriteria di atas?		☑ Layak	☐ Tidak Layak

Exercise: Membuat System Request dan Feasibility Analysis dengan Sparx EA

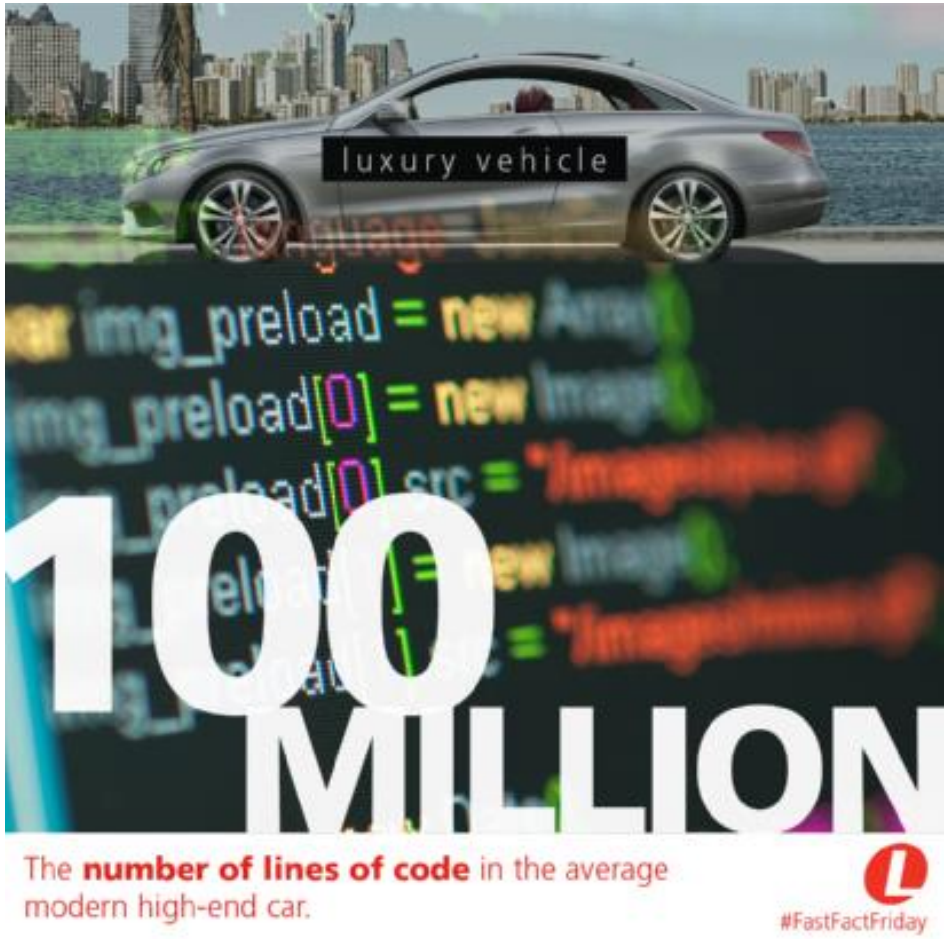
1. Lihat contoh System Request dan Feasibility Analysis untuk Sistem Penjualan Musik Online
2. Perhatikan kembali System Request dan Feasibility Analysis yang sebelumnya sudah kita buat
3. Revisi System Request dan Feasibility Analysis menggunakan template di Sparx EA, dan tambahkan Use Case Diagram pada System Request
4. Kirim file EAPX ke romi@brainmatics.com untuk dilakukan review bersama

2.3 Estimasi Usaha Pengembangan Software

“Size” of Software Systems

Year	Operating System	SLOC (Million)
1993	Windows NT 3.1	4-5[1]
1994	Windows NT 3.5	7–8[1]
1996	Windows NT 4.0	11–12[1]
2000	Windows 2000	more than 29[1]
2001	Windows XP	45[2][3]
2003	Windows Server 2003	50[1]

Source: Wikipedia



"Size" of Software Systems

Organizations	Function Points	Lines of Code
Internal Revenue Service	150,000	7,500,000
Banks	125,000	6,250,000
Insurance companies	125,000	6,250,000
Credit card companies	125,000	6,250,000
Credit bureaus	120,000	6,000,000
Census Bureau	100,000	5,000,000
State tax boards	90,000	4,500,000
Airlines	75,000	3,750,000
Police organizations	75,000	3,750,000
Hospitals	75,000	3,750,000
Web-based stores	75,000	3,750,000
Municipal tax boards	50,000	2,500,000
Motor vehicle department	50,000	2,500,000
Physicians offices	30,000	1,500,000
Dental offices	30,000	1,500,000
Schools/universities	25,000	1,250,000
Clubs and associations	20,000	1,000,000
Retail stores	20,000	1,000,000

Caper Jones, The Economic of Software Quality (2012)

Software Size (Effort) Estimation Methods

1. Simply Method

(Industry Std Percentages)

1. Use the time spent for planning
2. Along with industry standard percentages
3. Estimate the overall time for the project

2. Function Points

(Allen Albrecht, 1979)

1. Estimate System Size (Function Points)
2. Estimate Effort Required (Person-Month)
3. Estimate Time Required (Month)

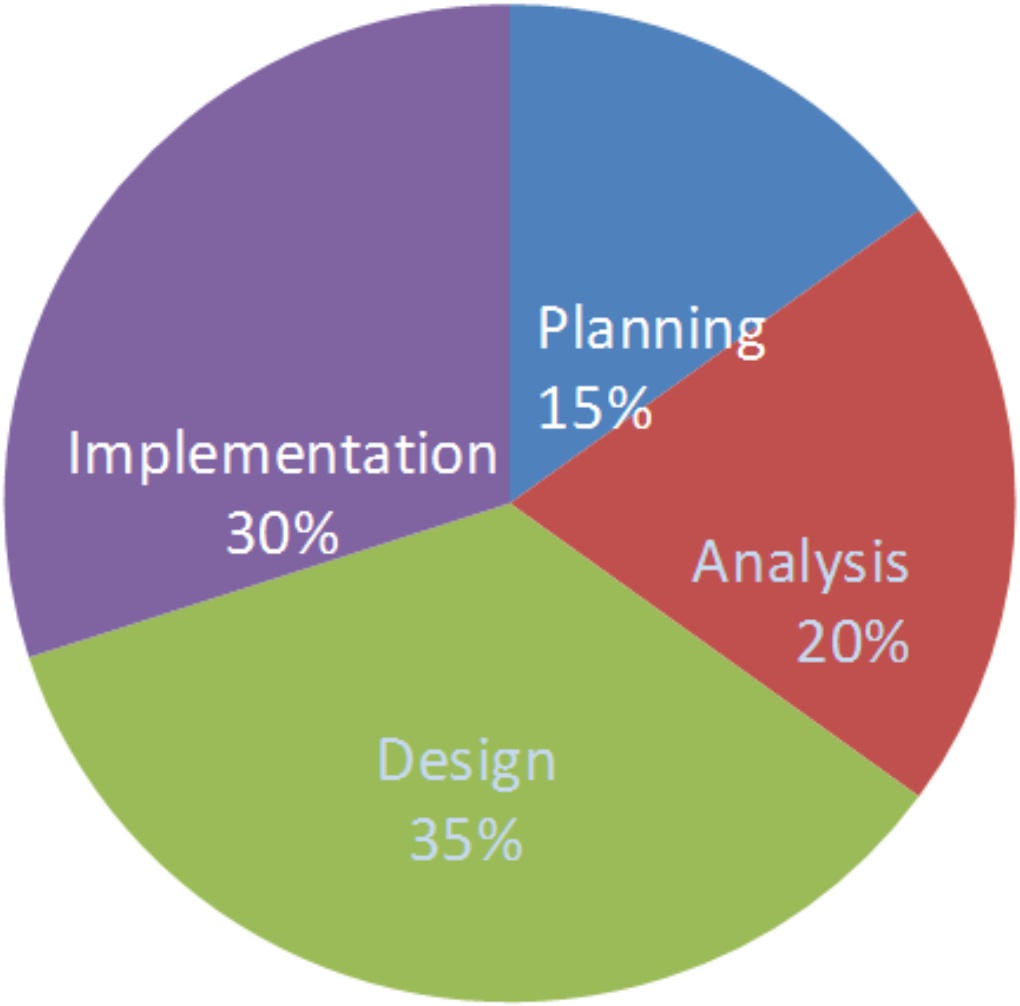
3. Use Case Points

(Gustav Karner, 1993)

1. Estimate System Size (Use Case Points)
2. Estimate Effort Required (Person-Month)
3. Estimate Time Required (Month)

2.3.1 Simply Method

Distribusi Effort Pengembangan Software



(Dennis, 2016) *(Jones, 2012)*

Estimate the Overall Time

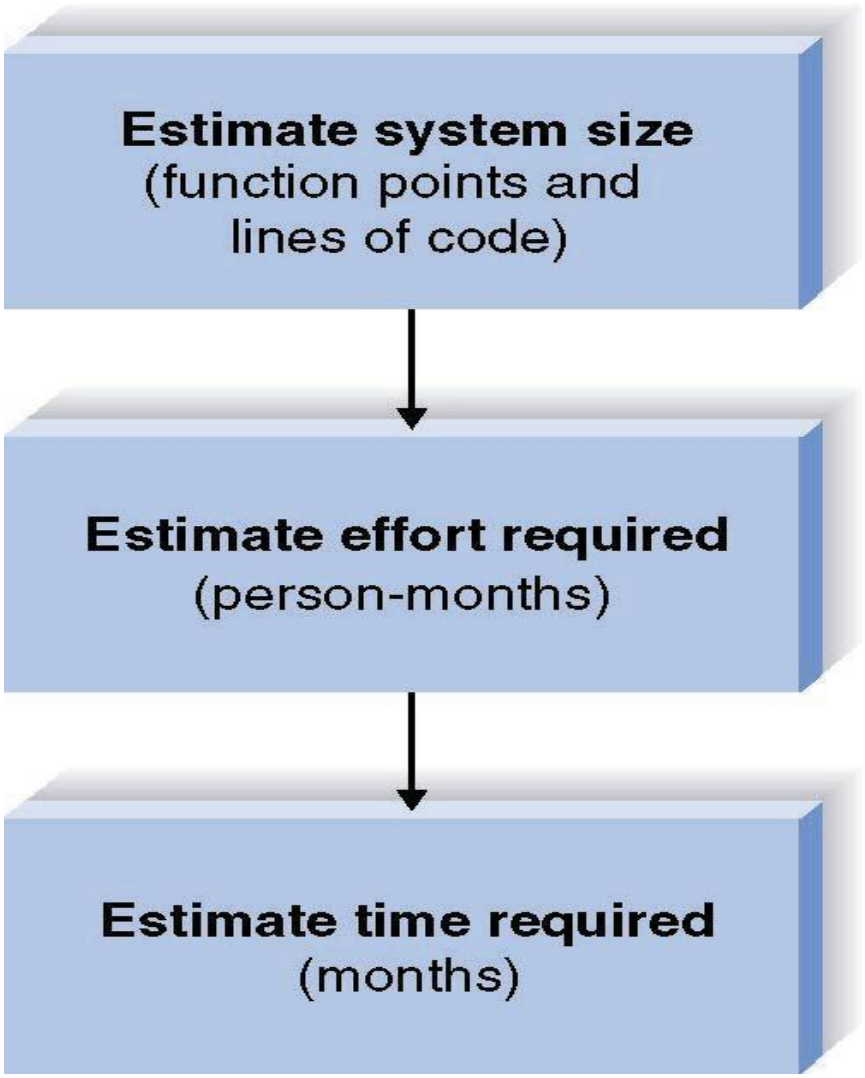
	Planning (Actual)	Analysis	Design	Implementation
Effort Distribution	15%	20%	35%	30%
Effort in Time (Month)	4 month	5.33 month	9.33 month	8 month
Effort in Person (Man)	2 person	2.6 person	4.6 person	4 person
Formula per Phase	Actual Time and Person	0.2 * (Planning/0.15)	0.35 * (Planning/0.15)	0.3 * (Planning/0.15)
Overall Time and Person	Planning/0.15			

$$\text{Planning time} = 0.15 \times \text{Overall time}$$

$$\text{Overall time} = \frac{\text{Planning time}}{0.15} \qquad \text{Analysis time} = 0.2 \times \frac{\text{Planning time}}{0.15}$$

2.3.2 Function Points

Function Point



(Allen Albrecht, 1979)

A. Function Points Estimation:
Step One (TUFP)

Complexity				
Description	Low	Medium	High	Total
Inputs	__x 3	__x 4	__x 6	____
Outputs	__x 4	__x 5	__x 7	____
Queries	__x 3	__x 4	__x 6	____
Files	__x 7	__x 10	__x 15	____
Program Interfaces	__x 5	__x 7	__x 10	____
TOTAL UNADJUSTED FUNCTION POINTS				____

Example: CD Selection System

System Components:

Description		Complexity			
	Total Number	Low	Medium	High	Total
Inputs	6	3 × 3	2 × 4	1 × 6	23
Outputs	19	4 × 4	10 × 5	5 × 7	101
Queries	10	7 × 3	0 × 4	3 × 6	39
Files	15	0 × 7	15 × 10	0 × 15	150
Program Interfaces	3	1 × 5	0 × 7	2 × 10	25
Total Unadjusted Function Points (TUFP):					338

Function Points Estimation:
Step Two (Processing Complexity)

	Scale of 0 to 3
Data Communications	_____
Heavy Use Configuration	_____
Transaction Rate	_____
End-User efficiency	_____
Complex Processing	_____
Installation Ease	_____
Multiple sites	_____
Performance	_____
Distributed functions	_____
On-line data entry	_____
On-line update	_____
Reusability	_____
Operational Ease	_____
Extensibility	_____
Processing Complexity (PC)	_____

Example: Sistem Penjualan Musik Online

Data communications	3
Heavy use configuration	0
Transaction rate	0
End-user efficiency	0
Complex processing	0
Installation ease	0
Multiple sites	0
Performance	0
Distributed functions	2
Online data entry	2
Online update	0
Reusability	0
Operational ease	0
Extensibility	0
Total Processing Complexity (PC):	7

Function Point Estimation:
Step Three (TAFP)

Processing Complexity (PC) = 7
(From Step Two)

Adjusted Processing
Complexity (PCA) = 0.65 + (0.01 * 7) = 0.72

Total Adjusted
Function Points (TAFP): 338 * 0.72 = 243
(From Step One)

Adjusted Processing Complexity

Choose standard Adjusted Project Complexity (PCA) from the range:

1. 0.65	Simple systems
2. 1.0	"Normal" systems
3. 1.35	Complex systems

Konversi Function Points ke Lines of Code

Language	LOC/TAFP
C	130
COBOL	110
JAVA	55
C++	50
Turbo Pascal	50
Visual Basic	30
PowerBuilder	15
HTML	15
Packages (e.g., Access, Excel)	10-40

Source: Capers Jones, Software Productivity Research

Language	QSM SLOC/FP Data			
	Avg	Median	Low	High
ABAP (SAP) *	28	18	16	60
ASP*	51	54	15	69
Assembler *	119	98	25	320
Brio +	14	14	13	16
C *	97	99	39	333
C++ *	50	53	25	80
C# *	54	59	29	70
COBOL *	61	55	23	297
Cognos Impromptu Scripts +	47	42	30	100
Cross System Products (CSP) +	20	18	10	38
Cool:Gen/IEF *	32	24	10	82
Datastage	71	65	31	157
Excel *	209	191	131	315
Focus *	43	45	45	45
FoxPro	36	35	34	38
HTML *	34	40	14	48
J2EE *	46	49	15	67
Java *	53	53	14	134
JavaScript *	47	53	31	63
JCL *	62	48	25	221
LINC II	29	30	22	38
Lotus Notes *	23	21	19	40
Natural *	40	34	34	53
.NET *	57	60	53	60

Lines of Codes (LOC)

Line of Codes (LOC) = TAFP * LOC/TAFP

Example:

If TAFP = 243 Then we build the software using Java

*LOC = 243 * 55 = 13365 line of codes*

Contoh Jenis Aplikasi dan FP

Organizations	Function Points	Lines of Code
Internal Revenue Service	150,000	7,500,000
Banks	125,000	6,250,000
Insurance companies	125,000	6,250,000
Credit card companies	125,000	6,250,000
Credit bureaus	120,000	6,000,000
Census Bureau	100,000	5,000,000
State tax boards	90,000	4,500,000
Airlines	75,000	3,750,000
Police organizations	75,000	3,750,000
Hospitals	75,000	3,750,000
Web-based stores	75,000	3,750,000
Municipal tax boards	50,000	2,500,000
Motor vehicle department	50,000	2,500,000
Physicians offices	30,000	1,500,000
Dental offices	30,000	1,500,000
Schools/universities	25,000	1,250,000
Clubs and associations	20,000	1,000,000
Retail stores	20,000	1,000,000

Caper Jones, The Economic of Software Quality (2012)

B. Estimating Effort

Effort = 1.4 * thousands-of- lines-of-code
(in Person- Months)

Example:

If LOC = 13365 Then...
*Effort = 1.4 * 13365/1000 = 18.711 Person Months*

C. Estimating Time

Time (in Months) = $3.0 * \text{person-months}^{1/3}$

Example:

If LOC = 13365 Then...

*Effort = 1.4 * 13.365 = 18.711 person-months*

Time = $3.0 * 18.711^{1/3}$ = 7.9 month

Boehm's Third Law (1981):

Development effort is a non-linear function of product size

Hitung Size dari Sistem dengan Function Point (TUFP – TAFP – LOC – PM – M)

- Sebuah perusahaan membutuhkan sistem *job seeker* untuk pencari kerja dan perusahaan pembuka lowongan pekerjaan
- Sistem memungkinkan pencari kerja untuk **menginput** data curriculum vitae. Di sisi lain, perusahan pembuka lowongan kerja bisa menginput data perusahaan dan lowongan pekerjaan yang disediakan
- Pencari kerja dapat melakukan **pencarian (query)** tentang lowongan pekerjaan apa saja yang tersedia, pembuka lowongan kerja dapat mencari siapa saja yang sudah mendaftar di suatu lowongan pekerjaan
- Sistem mampu **memproduksi laporan** statistik lengkap tentang pencari kerja, perusahaan, jenis lowongan pekerjaan dan tren lowongan kerja yang sedang populer
- Laporan statistik disajikan dalam dua format file, bentuk infografik (image), juga tersedia dalam bentuk file pdf yang bisa didownload (file)
- Sistem akan dikembangkan dengan menggunakan bahasa Java

TUFP

Complexity				
Description	Low	Medium	High	Total
Inputs	__x 3	__x 4	__x 6	____
Outputs	__x 4	__x 5	__x 7	____
Queries	__x 3	__x 4	__x 6	____
Files	__x 7	__x 10	__x 15	____
Program Interfaces	__x 5	__x 7	__x 10	____
		TOTAL UNADJUSTED FUNCTION POINTS		____

TUFP

	Fungsi	Bobot (Low)	Total
Input	3	3	9
Output	4	4	16
Queries	2	3	6
File	2	7	14
Program Interface	5	5	25
TUFP			**70**

Processing Complexity

Data Communications	1
Heavy Use Configuration	0
Transaction Rate	0
End-User efficiency	0
Complex Processing	0
Installation Ease	0
Multiple sites	0
Performance	0
Distributed functions	0
On-line data entry	2
On-line update	2
Reusability	0
Operational Ease	1
Extensibility	0

TOTAL: 6

TAFP

1. Processing Complexity (PC) = 6

2. Adjusted Processing Complexity (PCA)

   0.65 + (0.01 * 6) = 0.71

3. Total Adjusted Function Points (**TAFP**)

   70 * 0.71 = **49.7**

LOC → Effort (ManMonth)→ Time (Month)

1. LOC = 55*49.7 = 2733.5

2. Effort = 1.4*2.7335 = 3.83 MM

3. Time = 3.0 * 3.83 $^{(1/3)}$ = 4.6 M

Example: Schedule

	Fase	M	PM	Cost
1	Planning	1	3	15
2	Analysis	2	3	30
3	Design	2	4	50
4	Implementation	4	5	80
				175jt

Exercise: Software Size Estimation dengan Function Points

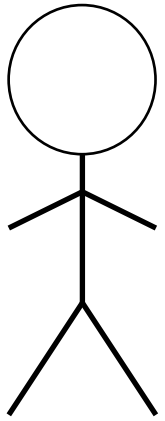
1. Lihat contoh Feasibility Analysis untuk Sistem Penjualan Musik Online
2. Perhatikan kembali System Request yang sebelumnya sudah kita buat
3. Buat Feasibility Analysis dari system yang akan kita buat tersebut
4. Lakukan Software Size Estimation dengan menggunakan Function Point Approach untuk melengkapi Technical dan Economic Feasibility

2.3.3 Use Case Points

Use Case Points

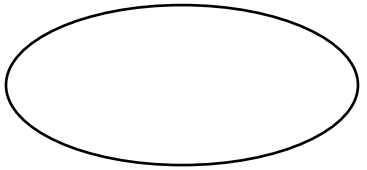
Unadjusted Actor Weighting (UAW)

Actor Type	Description	Weighting Factor
Simple (Easy)	External System with well-defined API	1
Average (Medium)	External System using a protocol-based interface, e.g., HTTP, TCT/IP, SQL	2
Complex (Difficult)	Human	3



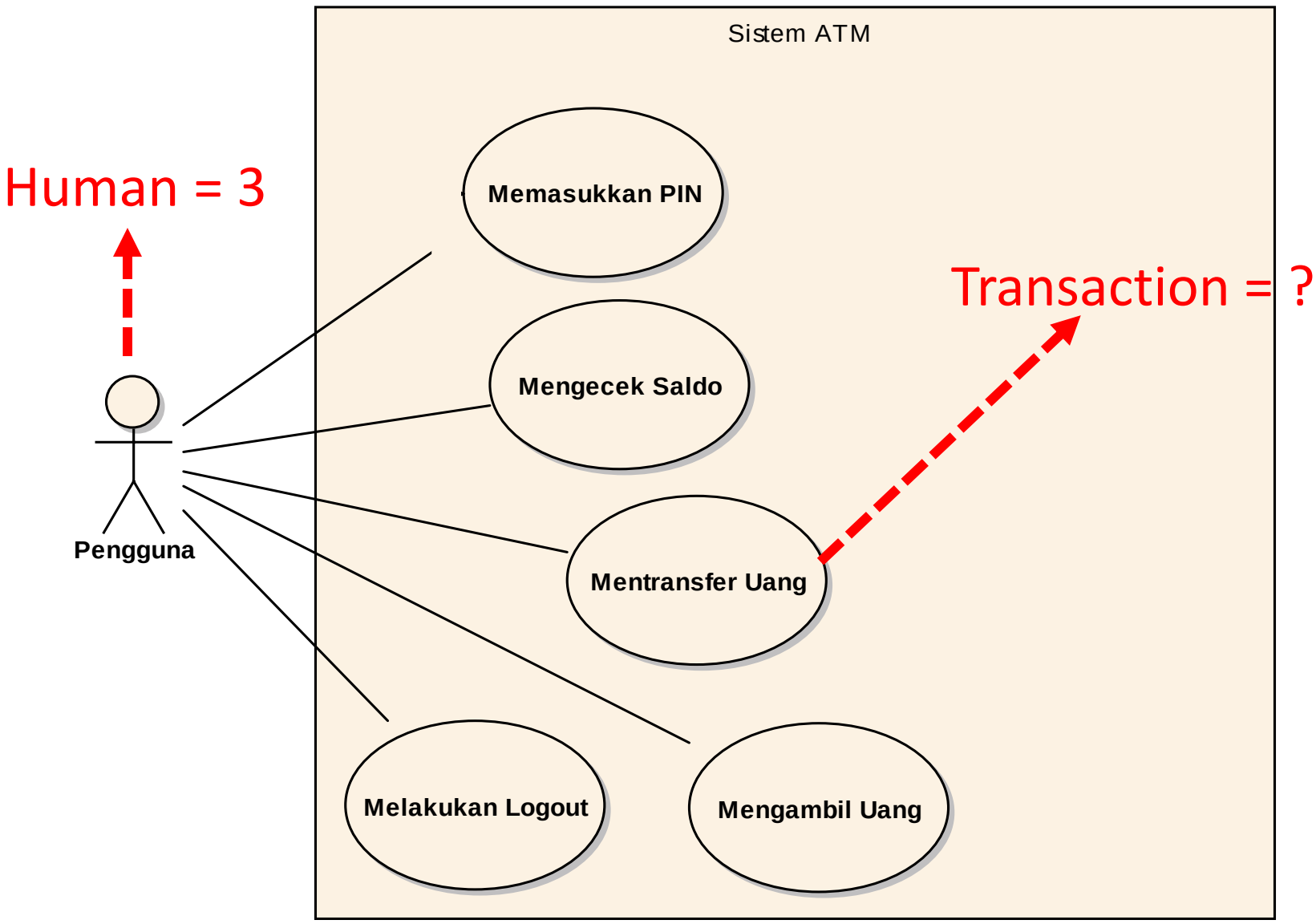
Unadjusted Use Case Weighting (UUCW)

Use-Case Type	Description	Weighting Factor
Simple (Easy)	1-3 transactions	5
Average (Medium)	4-7 transactions	10
Complex (Difficult)	More than 7 transactions	15

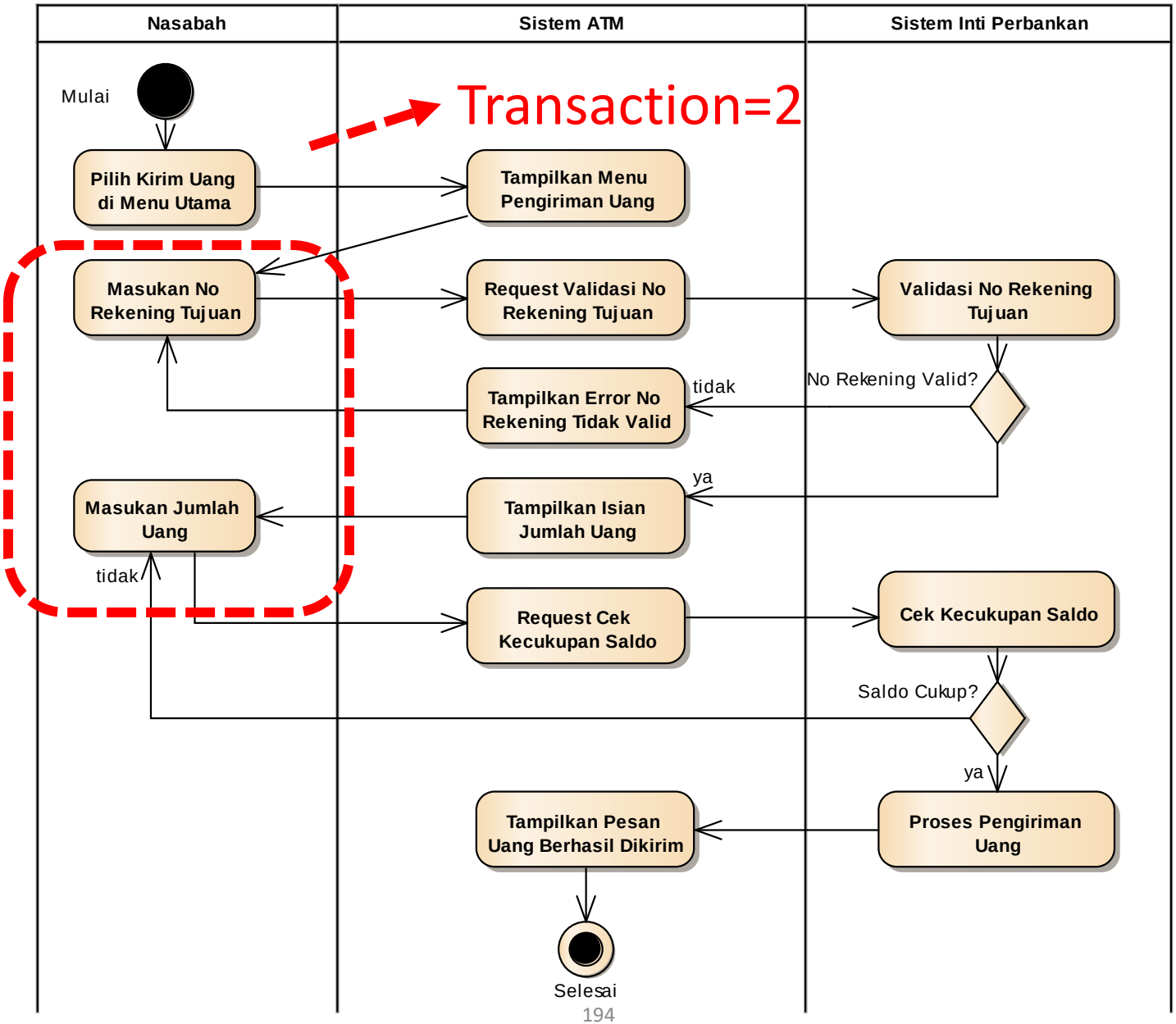


Unadjusted Use Case Points (UUCP) = UAW + UUCW

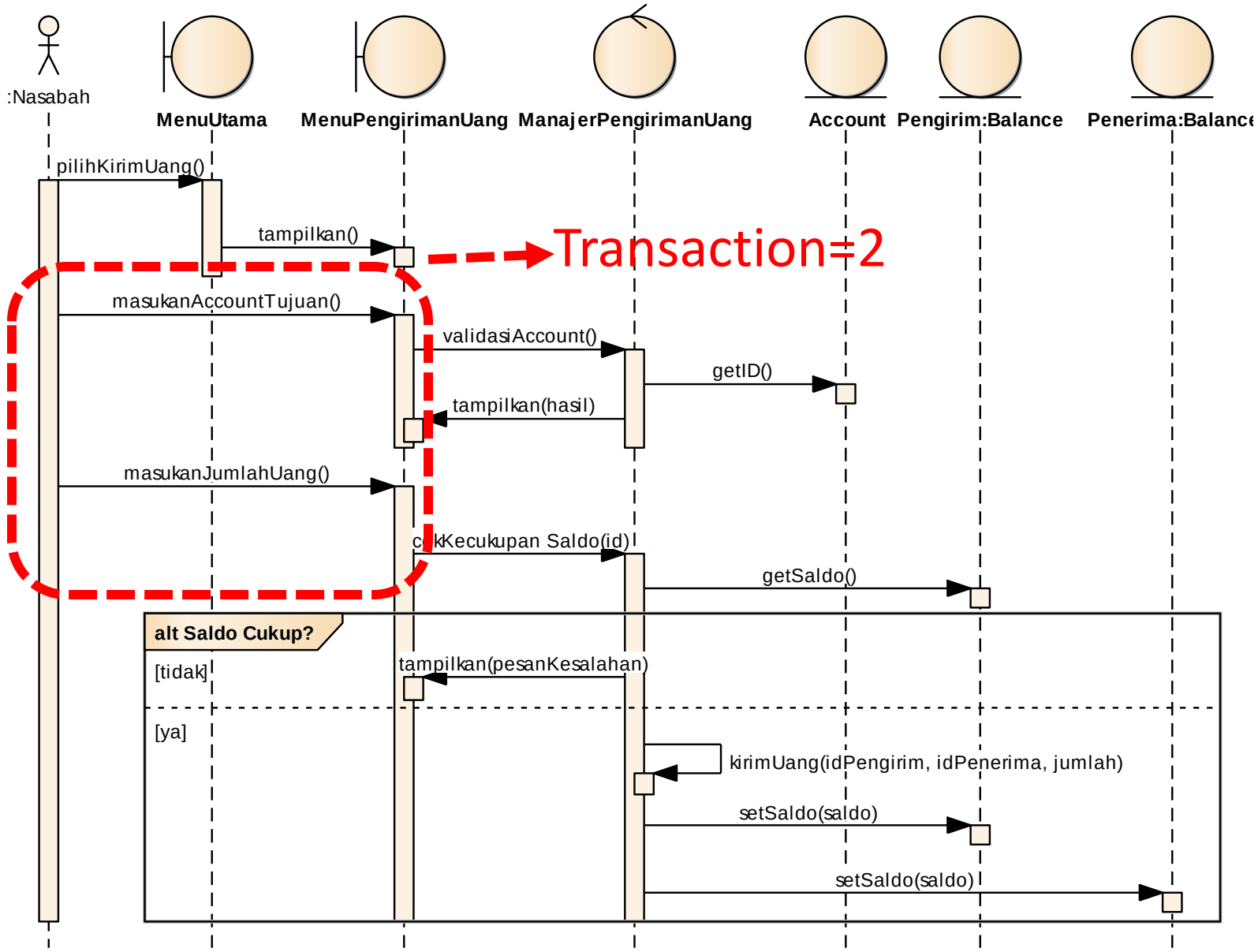
Sistem ATM – Use Case Diagram



Activity Diagram: Mengirim Uang



Sequence Diagram: Mengirim Uang



Technical Complexity Factors (TCF)

Factor Number	Description	Weight
T1	Distributed system	2.0
T2	Response time or throughput performance objectives	1.0
T3	End-user online efficiency	1.0
T4	Complex internal processing	1.0
T5	Reusability of code	1.0
T6	Easy to install	0.5
T7	Ease of use	0.5
T8	Portability	2.0
T9	Ease of change	1.0

$$TCF = 0.6 + (0.01 * TFactor)$$

Environmental Complexity Factors (ECF)

Factor Number	Description	Weight
E1	Familiarity with system development process in use	1.5
E2	Application experience	0.5
E3	Object-oriented experience	1.0
E4	Lead analyst capability	0.5
E5	Motivation	1.0
E6	Requirements stability	2.0
E7	Part time staff	-1.0
E8	Difficulty of programming language	-1.0

$$ECF = 1.4 + (-0.03 * EFactor)$$

Computing Use Case Points

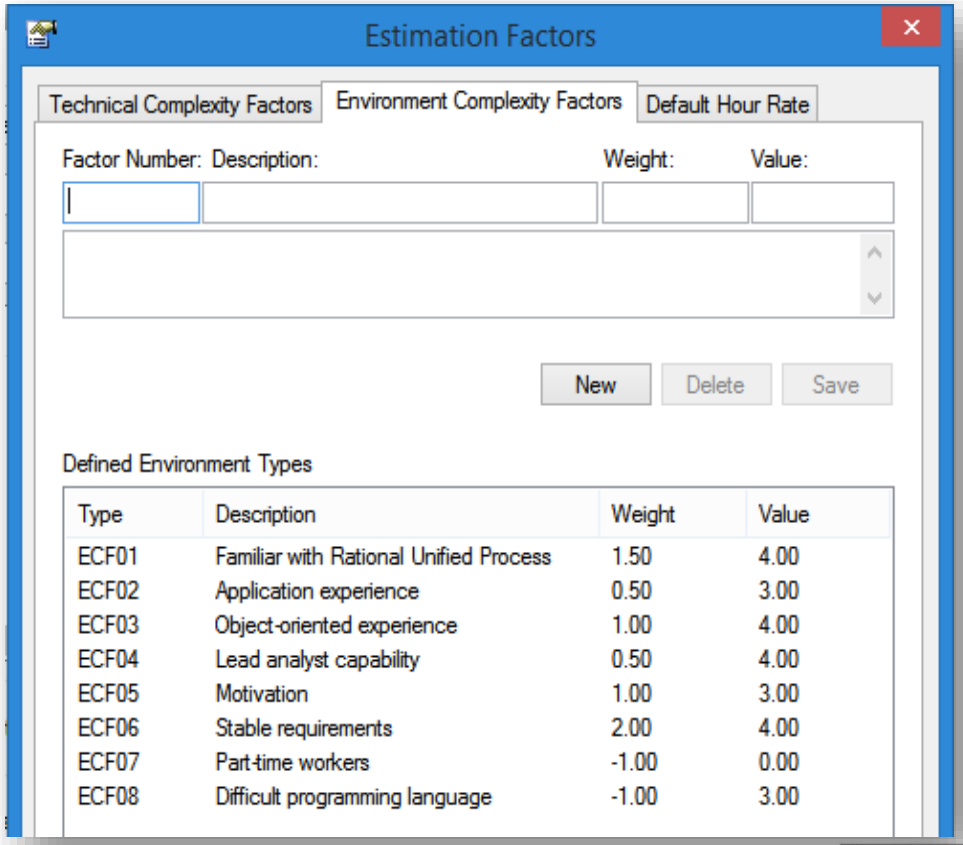
- Adjusted Use Case Points (UCP) = UUCP * TCF * ECF
- Effort in Person Hours = UCP * PHM

Person Hour Multiplier (PHM)

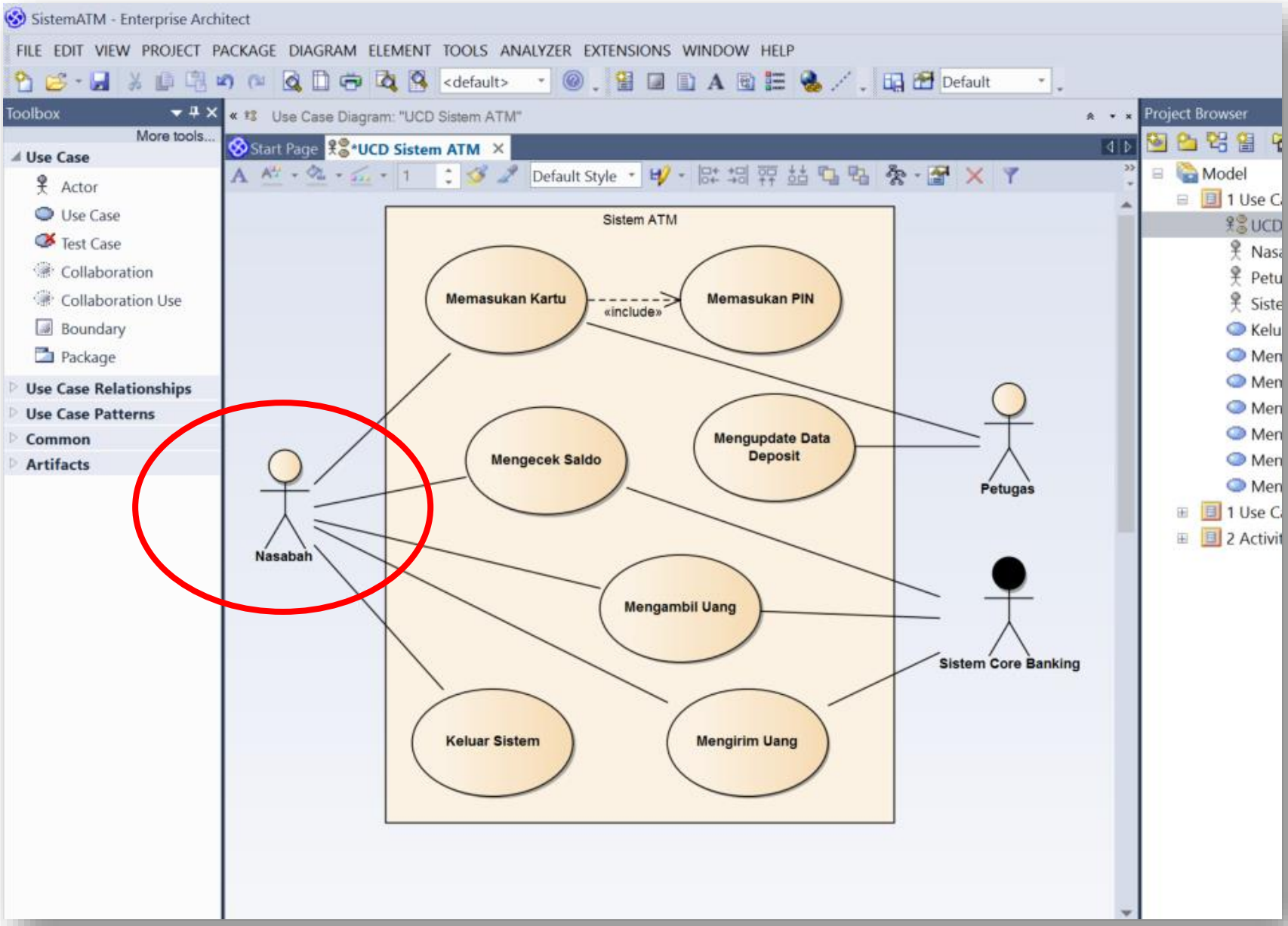
Let F1 = Number of ECF1 to ECF6 that are < 3

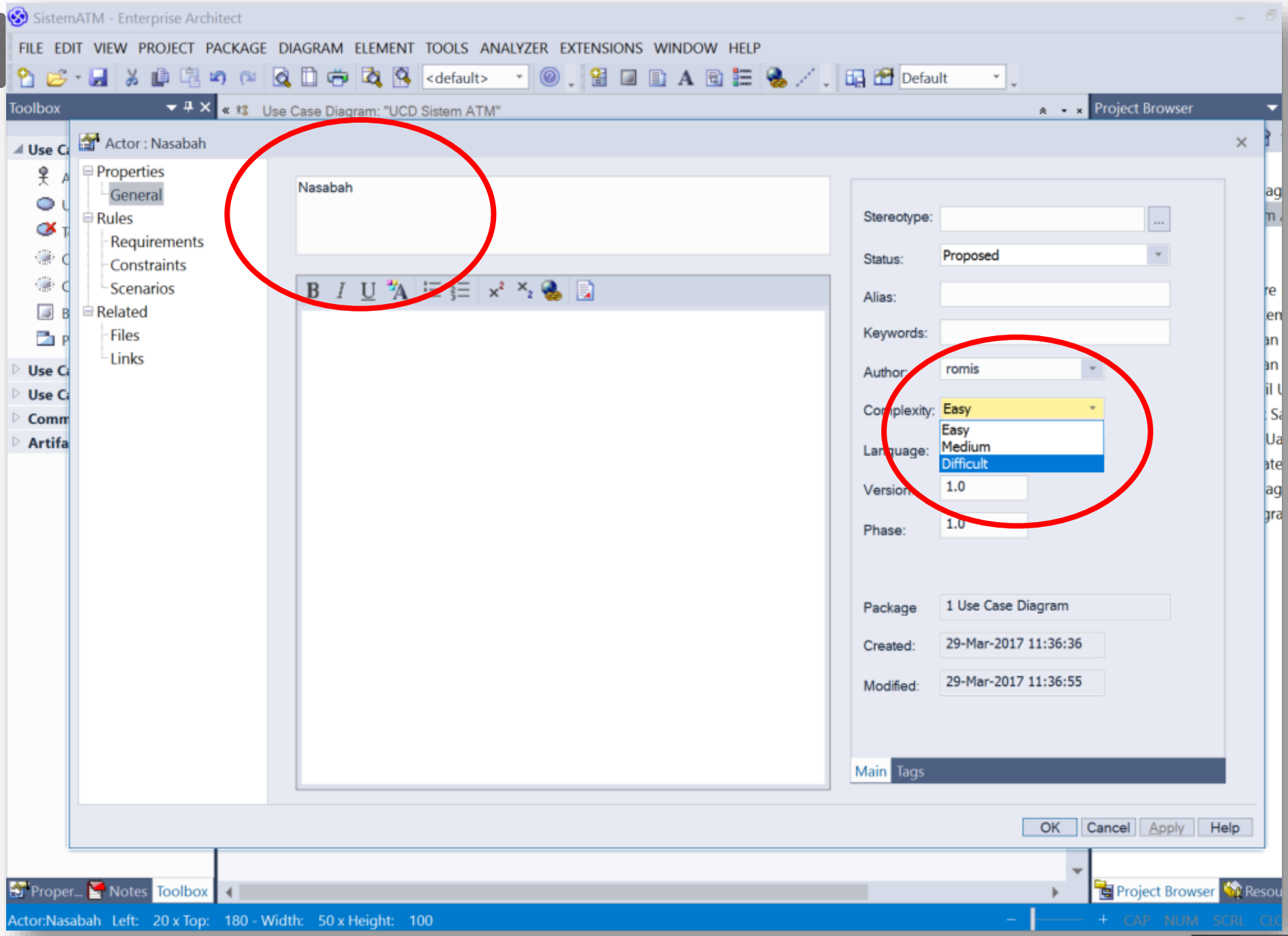
Let F2 = Number of ECF7 and ECF8 that are > 3

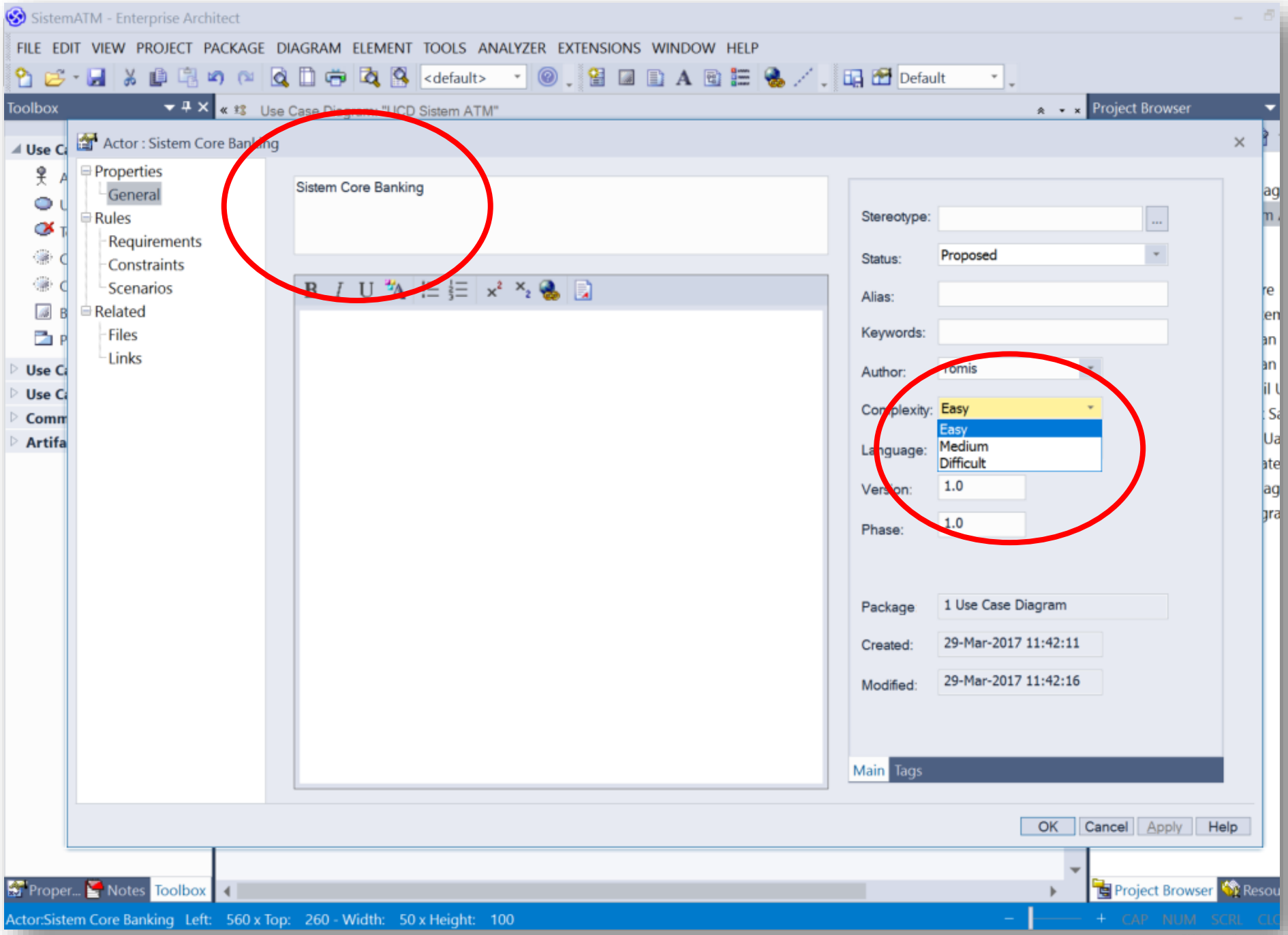
If F1 + F2 <= 2
    PHM = 20
Else if F1 + F2 = 3 or 4
    PHM = 28
Else
    Scrap the project



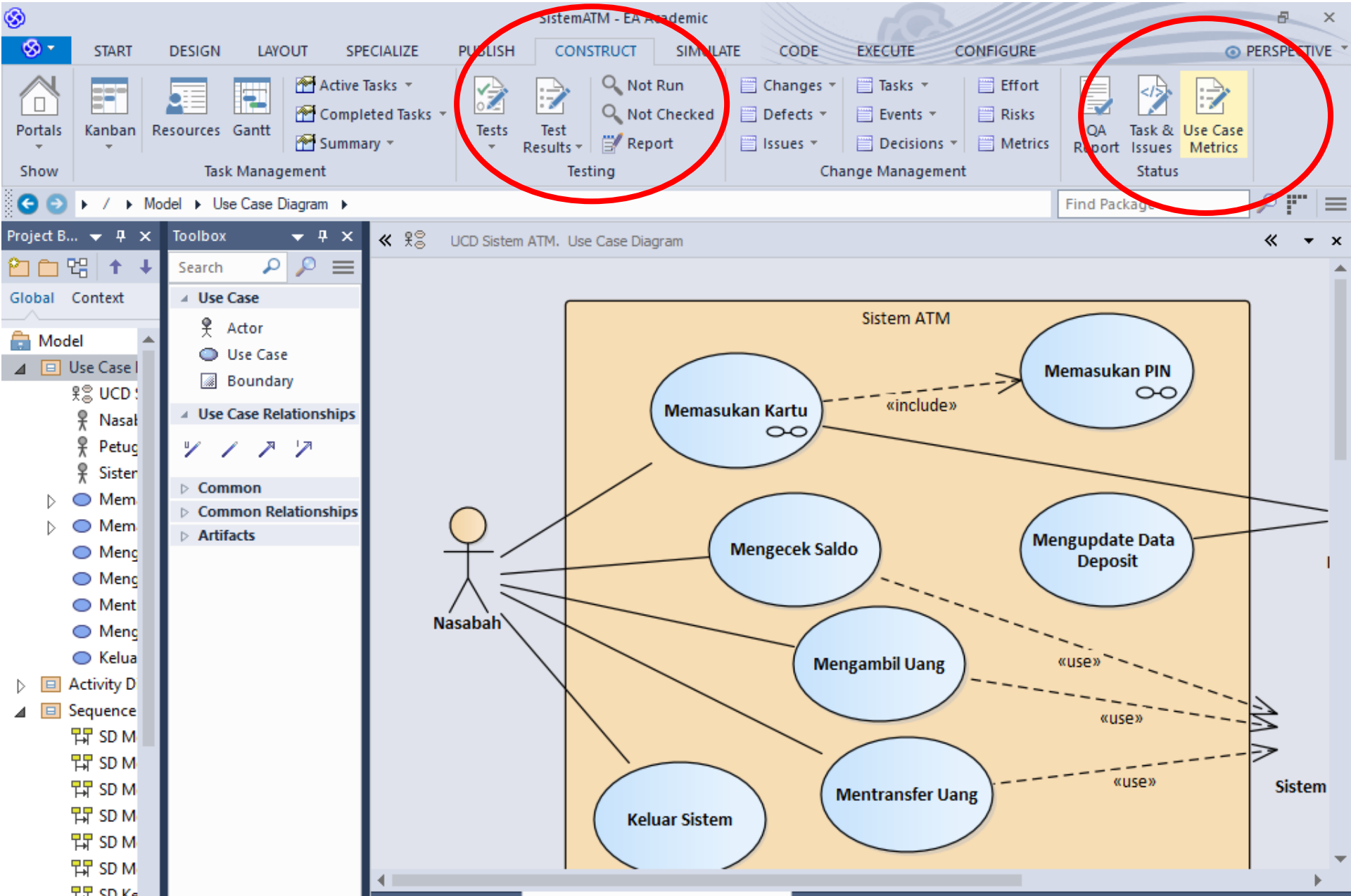
Computing Use Case Points with Sparx EA

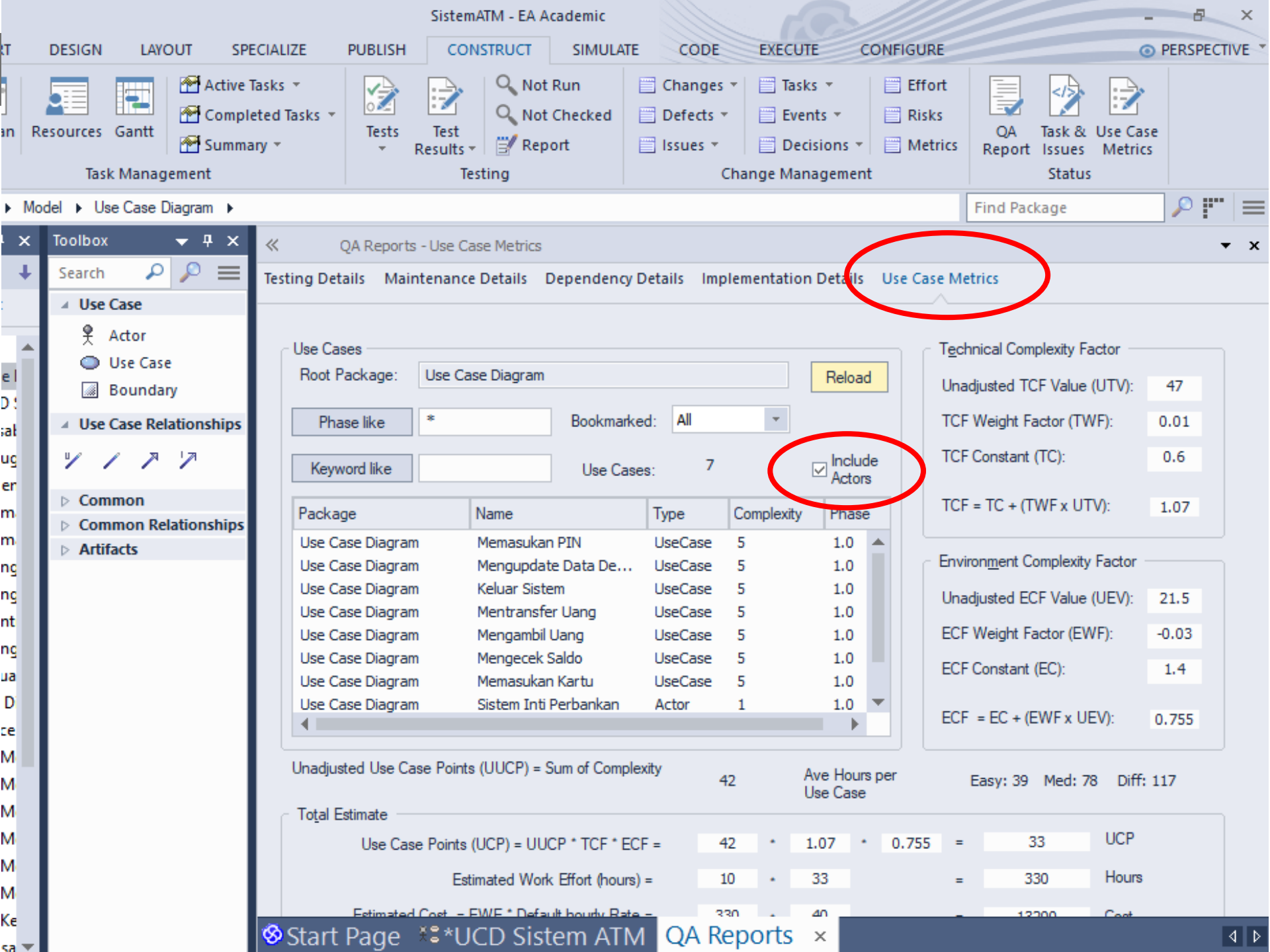


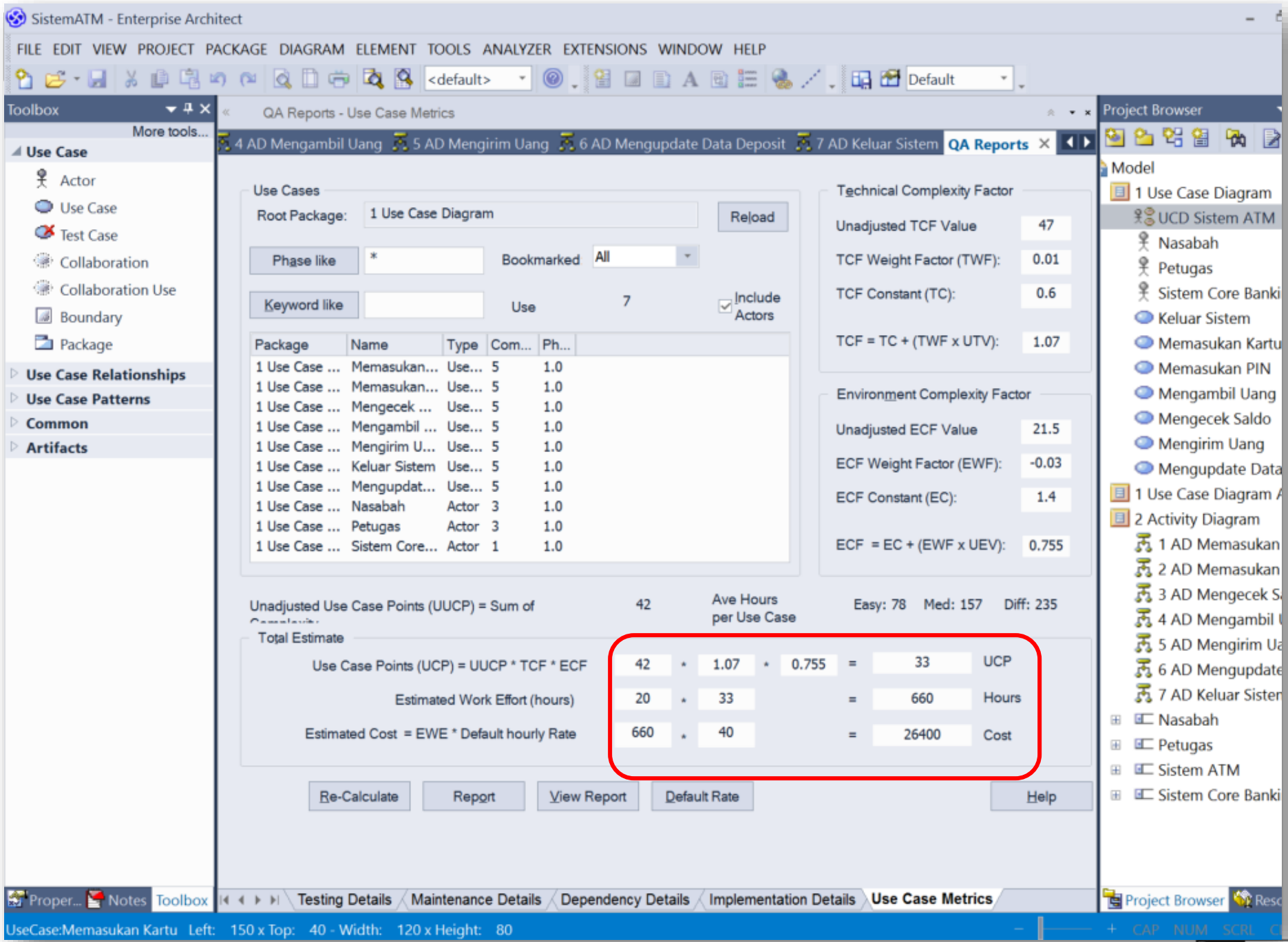

SistemATM - Enterprise Architect
FILE EDIT VIEW PROJECT PACKAGE DIAGRAM ELEMENT TOOLS ANALYZER EXTENSIONS WINDOW HELP
Toolbox
Use Case Diagram: "UCD Sistem ATM"
Project Browser
Actor : Nasabah
Properties
General
Rules
Requirements
Constraints
Scenarios
Related
Files
Links
Nasabah
Stereotype:
Status: Proposed
Alias:
Keywords:
Author: romis
Complexity: Easy
Easy
Medium
Difficult
Language:
Version: 1.0
Phase: 1.0
Package 1 Use Case Diagram
Created: 29-Mar-2017 11:36:36
Modified: 29-Mar-2017 11:36:55
Main Tags
OK Cancel Apply Help
Proper... Notes Toolbox
Project Browser
Actor:Nasabah Left: 20 x Top: 180 - Width: 50 x Height: 100


SistemATM - Enterprise Architect
FILE EDIT VIEW PROJECT PACKAGE DIAGRAM ELEMENT TOOLS ANALYZER EXTENSIONS WINDOW HELP
Toolbox
Project Browser
Actor : Sistem Core Banking
Properties
General
Rules
Requirements
Constraints
Scenarios
Related
Files
Links
Sistem Core Banking
Stereotype:
Status: Proposed
Alias:
Keywords:
Author: Tomis
Complexity: Easy
Easy
Medium
Difficult
Language:
Version: 1.0
Phase: 1.0
Package 1 Use Case Diagram
Created: 29-Mar-2017 11:42:11
Modified: 29-Mar-2017 11:42:16
Main Tags
OK Cancel Apply Help
Actor:Sistem Core Banking Left: 560 x Top: 260 - Width: 50 x Height: 100

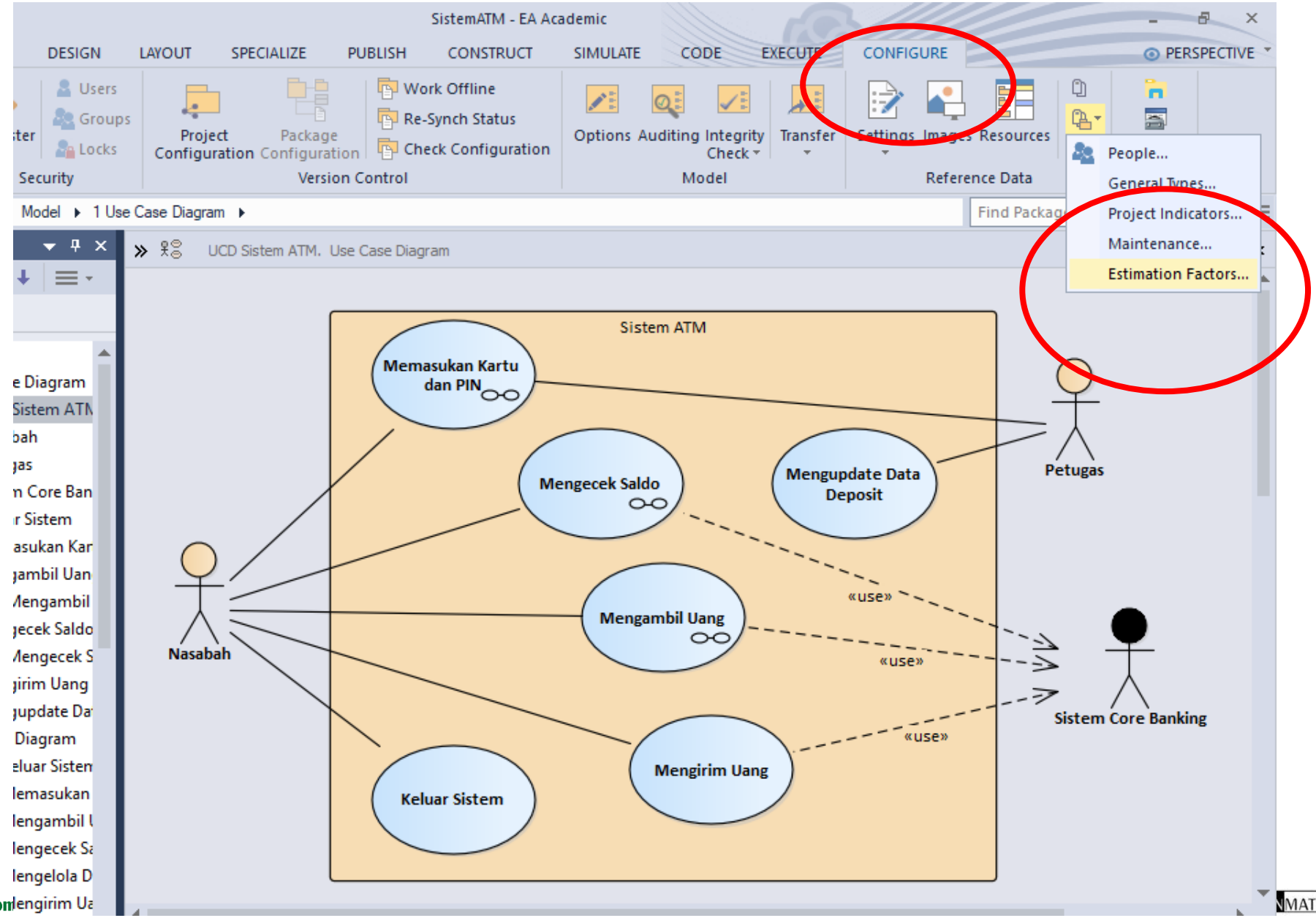
Use Case Points in Sparx EA 14



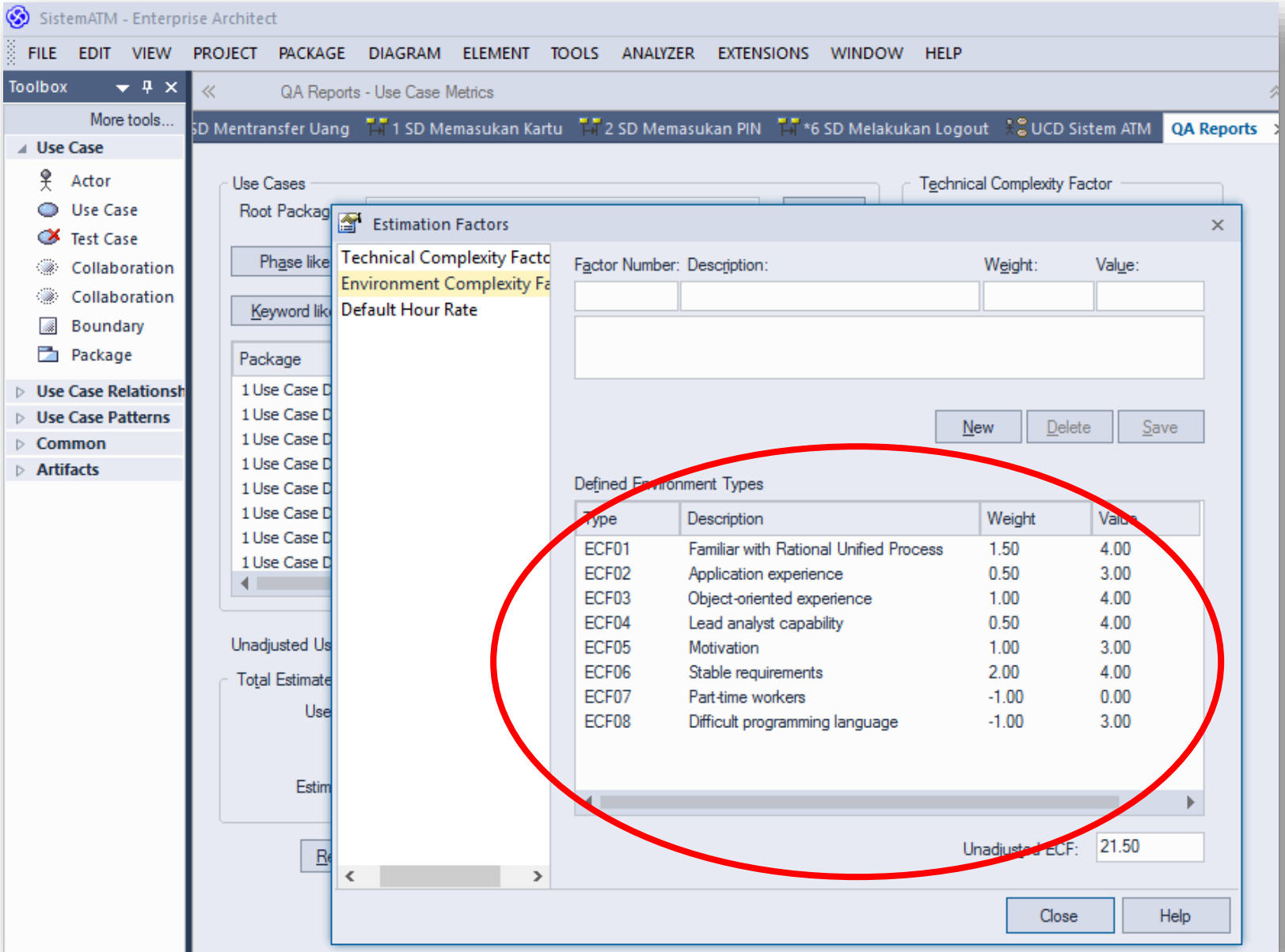

SistemATM - EA Academic
DESIGN
LAYOUT
SPECIALIZE
PUBLISH
CONSTRUCT
SIMULATE
CODE
EXECUTE
CONFIGURE
PERSPECTIVE
Resources
Gantt
Active Tasks
Completed Tasks
Summary
Task Management
Tests
Test Results
Not Run
Not Checked
Report
Testing
Changes
Defects
Issues
Tasks
Events
Decisions
Effort
Risks
Metrics
Change Management
QA Report
Task & Issues
Use Case Metrics
Status
Model
Use Case Diagram
Find Package
Toolbox
Search
Use Case
Actor
Use Case
Boundary
Use Case Relationships
Common
Common Relationships
Artifacts
QA Reports - Use Case Metrics
Testing Details
Maintenance Details
Dependency Details
Implementation Details
Use Case Metrics
Use Cases
Root Package:
Use Case Diagram
Reload
Phase like
*
Bookmarked:
All
Keyword like
Use Cases:
7
Include Actors
Package
Name
Type
Complexity
Phase
Use Case Diagram
Memasukan PIN
UseCase
5
1.0
Use Case Diagram
Mengupdate Data De...
UseCase
5
1.0
Use Case Diagram
Keluar Sistem
UseCase
5
1.0
Use Case Diagram
Mentransfer Uang
UseCase
5
1.0
Use Case Diagram
Mengambil Uang
UseCase
5
1.0
Use Case Diagram
Mengecek Saldo
UseCase
5
1.0
Use Case Diagram
Memasukan Kartu
UseCase
5
1.0
Use Case Diagram
Sistem Inti Perbankan
Actor
1
1.0
Technical Complexity Factor
Unadjusted TCF Value (UTV):
47
TCF Weight Factor (TWF):
0.01
TCF Constant (TC):
0.6
TCF = TC + (TWF x UTV):
1.07
Environment Complexity Factor
Unadjusted ECF Value (UEV):
21.5
ECF Weight Factor (EWF):
-0.03
ECF Constant (EC):
1.4
ECF = EC + (EWF x UEV):
0.755
Unadjusted Use Case Points (UUCP) = Sum of Complexity
42
Ave Hours per Use Case
Easy: 39
Med: 78
Diff: 117
Total Estimate
Use Case Points (UCP) = UUCP * TCF * ECF =
42
1.07
0.755
33
UCP
Estimated Work Effort (hours) =
10
33
330
Hours
Start Page
*UCD Sistem ATM
QA Reports


SistemATM - Enterprise Architect
FILE EDIT VIEW PROJECT PACKAGE DIAGRAM ELEMENT TOOLS ANALYZER EXTENSIONS WINDOW HELP
Toolbox
More tools...
Use Case
Actor
Use Case
Test Case
Collaboration
Collaboration Use
Boundary
Package
Use Case Relationships
Use Case Patterns
Common
Artifacts
QA Reports - Use Case Metrics
4 AD Mengambil Uang
5 AD Mengirim Uang
6 AD Mengupdate Data Deposit
7 AD Keluar Sistem
QA Reports
Use Cases
Root Package: 1 Use Case Diagram
Reload
Phase like
*
Bookmarked
All
Keyword like
Use
7
Include Actors
Package
Name
Type
Com...
Ph...
1 Use Case ... Memasukan... Use... 5 1.0
1 Use Case ... Memasukan... Use... 5 1.0
1 Use Case ... Mengecek ... Use... 5 1.0
1 Use Case ... Mengambil ... Use... 5 1.0
1 Use Case ... Mengirim U... Use... 5 1.0
1 Use Case ... Keluar Sistem Use... 5 1.0
1 Use Case ... Mengupdat... Use... 5 1.0
1 Use Case ... Nasabah Actor 3 1.0
1 Use Case ... Petugas Actor 3 1.0
1 Use Case ... Sistem Core... Actor 1 1.0
Technical Complexity Factor
Unadjusted TCF Value 47
TCF Weight Factor (TWF): 0.01
TCF Constant (TC): 0.6
TCF = TC + (TWF x UTV): 1.07
Environment Complexity Factor
Unadjusted ECF Value 21.5
ECF Weight Factor (EWF): -0.03
ECF Constant (EC): 1.4
ECF = EC + (EWF x UEV): 0.755
Unadjusted Use Case Points (UUCP) = Sum of
42
Ave Hours per Use Case
Easy: 78 Med: 157 Diff: 235
Total Estimate
Use Case Points (UCP) = UUCP * TCF * ECF
42 * 1.07 * 0.755 = 33 UCP
Estimated Work Effort (hours)
20 * 33 = 660 Hours
Estimated Cost = EWE * Default hourly Rate
660 * 40 = 26400 Cost
Re-Calculate
Report
View Report
Default Rate
Help
Project Browser
Model
1 Use Case Diagram
UCD Sistem ATM
Nasabah
Petugas
Sistem Core Banki
Keluar Sistem
Memasukan Kartu
Memasukan PIN
Mengambil Uang
Mengecek Saldo
Mengirim Uang
Mengupdate Data
1 Use Case Diagram A
2 Activity Diagram
1 AD Memasukan
2 AD Memasukan
3 AD Mengecek S
4 AD Mengambil
5 AD Mengirim Ua
6 AD Mengupdate
7 AD Keluar Siste
Nasabah
Petugas
Sistem ATM
Sistem Core Banki
Proper... Notes Toolbox
Testing Details
Maintenance Details
Dependency Details
Implementation Details
Use Case Metrics
Project Browser
UseCase:Memasukan Kartu Left: 150 x Top: 40 - Width: 120 x Height: 80
CAP NUM SCRL

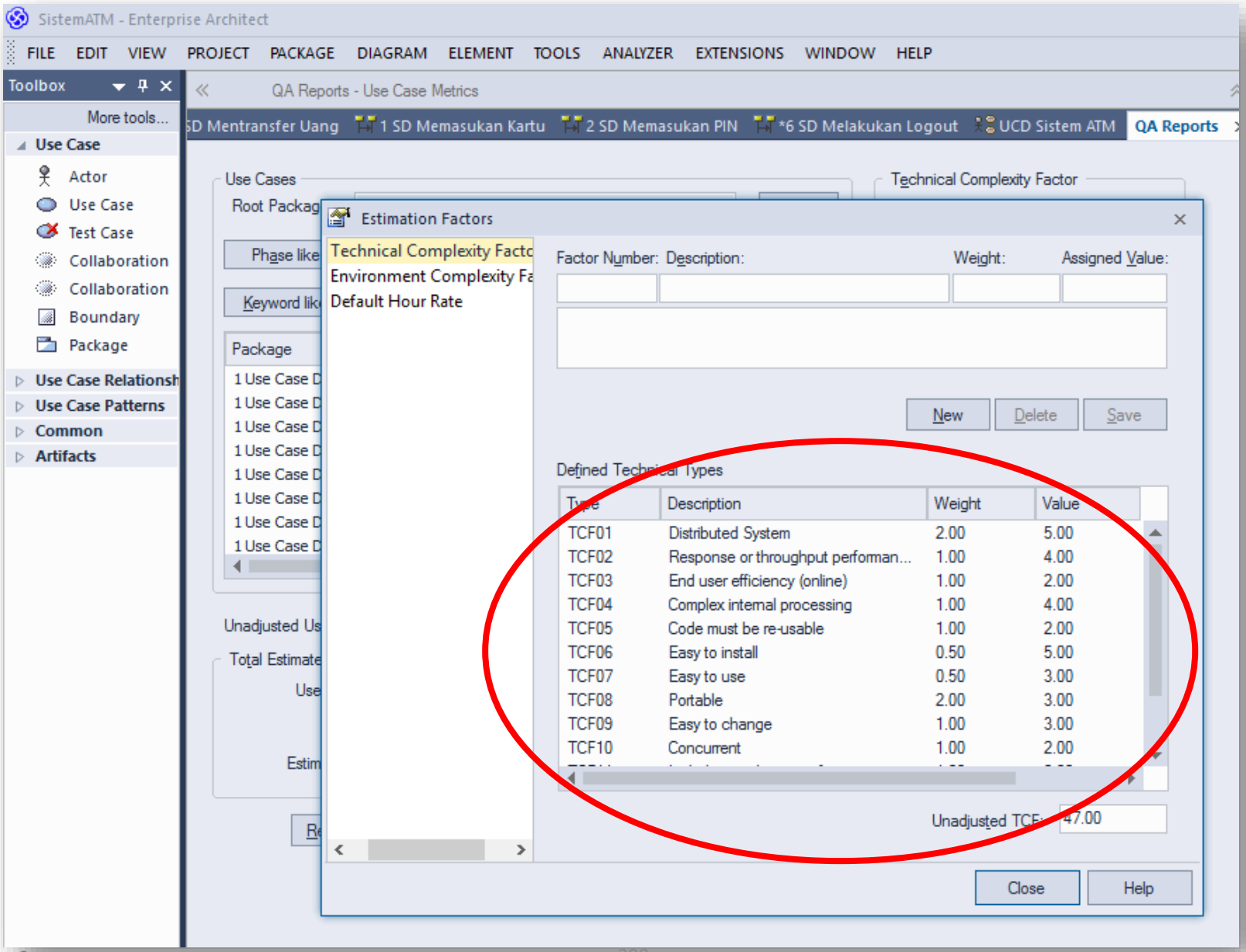
Estimation Factors (Sparx EA 14)



Environment Complexity Factors (ECF)



Technical Complexity Factors (TCF)



Example: Sistem ATM

UCP	= 33	
PHM	= 20	
PH	= 20*33	= 660 PH
	28*33	= 924
PM	= 660/8/22	= 3.75 PM
	= 924/8/22	= 5.25 PM
TIME (M)	= 3.0 * PM $^{1/3}$	
TIME (M)	= 3.0 * 3.75 $^{1/3}$	= 4.6 M
	= 3.0 * 5.25 $^{1/3}$	= 5.21 M

Example: Sistem ERP

UCP = 1517

PHM = 20

PH = 20*1517 = 30340 PH

28*1517

PM = 30340/8/22 = 172 PM

TIME (M) = $3.0 * PM^{1/3}$

TIME (M) = $3.0 * 172^{1/3}$ = 14 M

Example: MusicPedia

Technical Feasibility

Use Case Points

Tahap 1 - Menghitung Person Hours (PH)				
Use Case Points (UCP)		Person Hours Multiplier (PHM)	Person Hours (PH)	
51		20	1020	
51		28	1428	
Tahap 2 - Menghitung Person Month (PM)				
PHM	Person Hours (PH)	Lama Bekerja Perhari	Jumlah Bekerja Sebulan	Person Months (PM)
20	1020	8	22	5.80
	1020	10	26	3.92
28	1428	8	22	8.11
	1428	10	26	5.49
Tahap 3 - Menghitung Time (Month)				
PHM	Formula Penghitung Waktu		Jumlah Bekerja Sebulan	Waktu dalam Bulan (M)
20	3 * PM^(1/3)		22	5.39
			26	4.73
28			22	6.03
			26	5.29

Rangkuman Systems Planning

- Project software dimulai ketika ada seseorang yang melihat peluang menciptakan business value dengan menggunakan software dan teknologi informasi, dan dibuatlah System Request
  - System Request: Business Needs, Business Requirement, Business Value

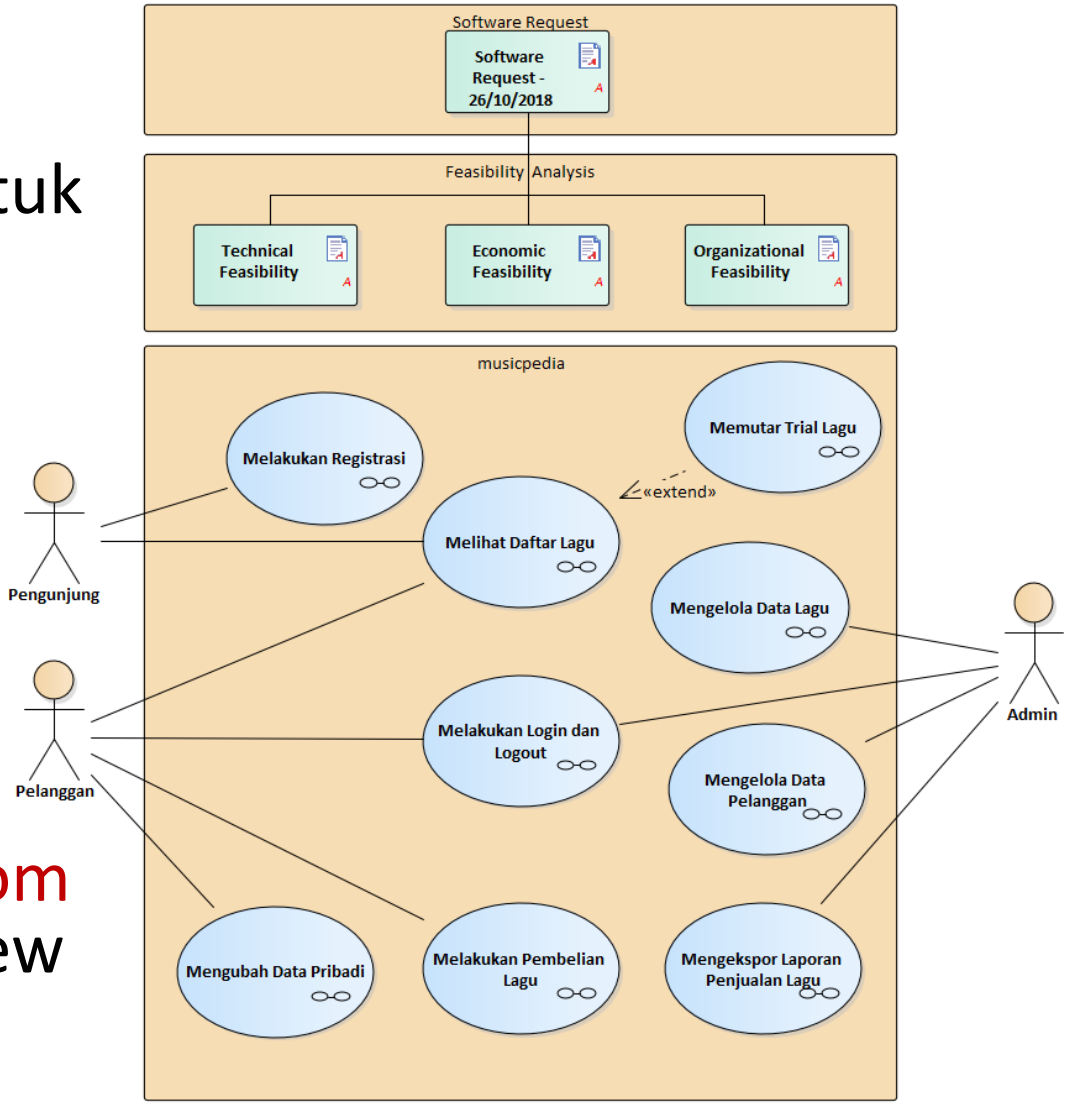
- System Request kemudian dianalisis kelayakannya (Feasibility Analysis) untuk menentukan apakah akan diteruskan projectnya atau tidak
  - Feasibility Analysis: technical, economic, and organizational

- Di dalam Technical Feasibility ada proses estimasi usaha pengembangan software (Software Effort Estimation)
  - Software Effort Estimation: simply method, function point and use case point

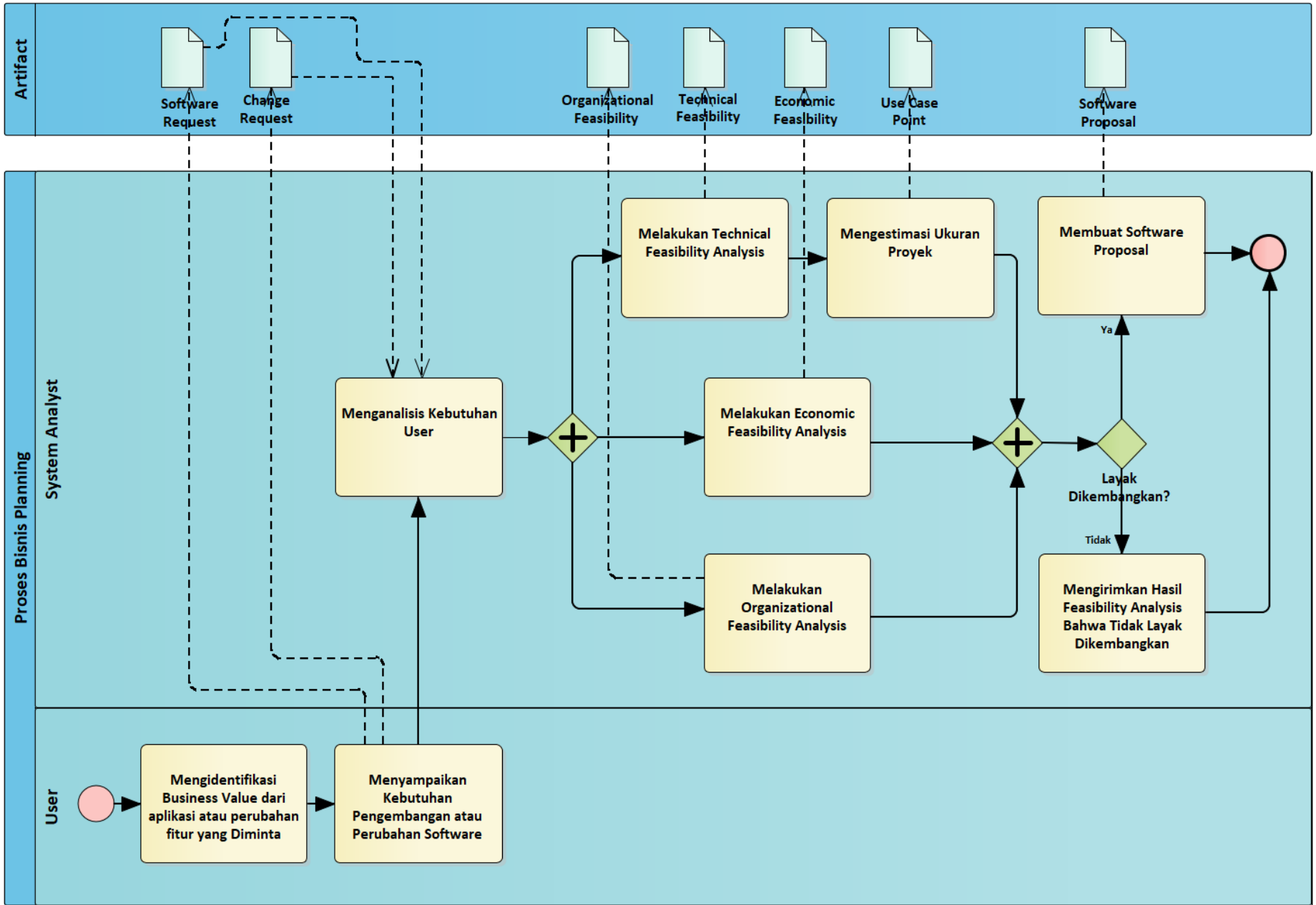
Exercise: Selesaikan Fase Planning

1. Selesaikan seluruh tahapan planning untuk System Request kita
   - System Request
   - Feasibility Analysis (+ Use Case Points)
   - Business Requirements (Use Case Diagram)

2. Kirim file EAPX ke romi@brainmatics.com untuk dilakukan review bersama



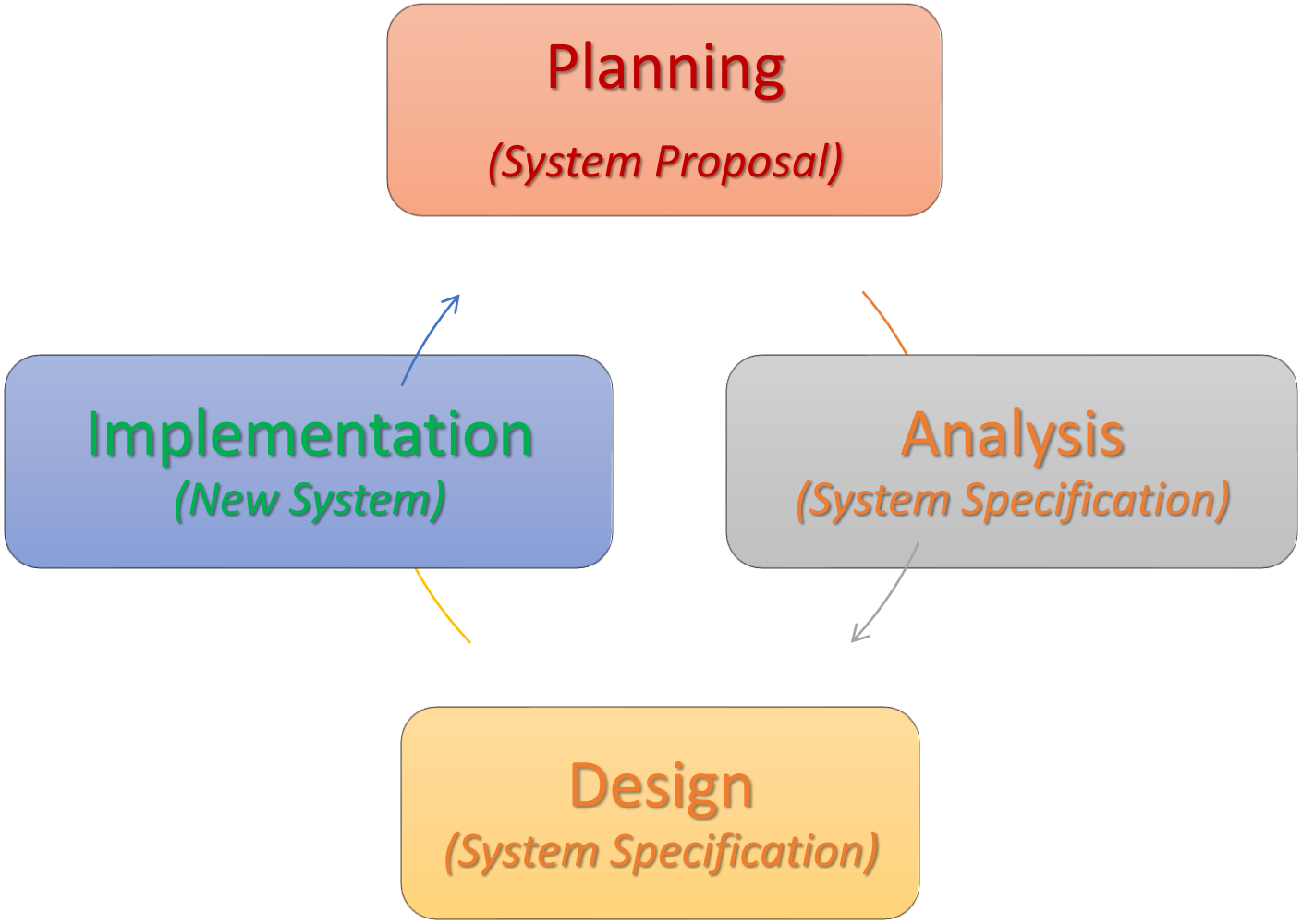
Systems Planning



3. Systems Analysis

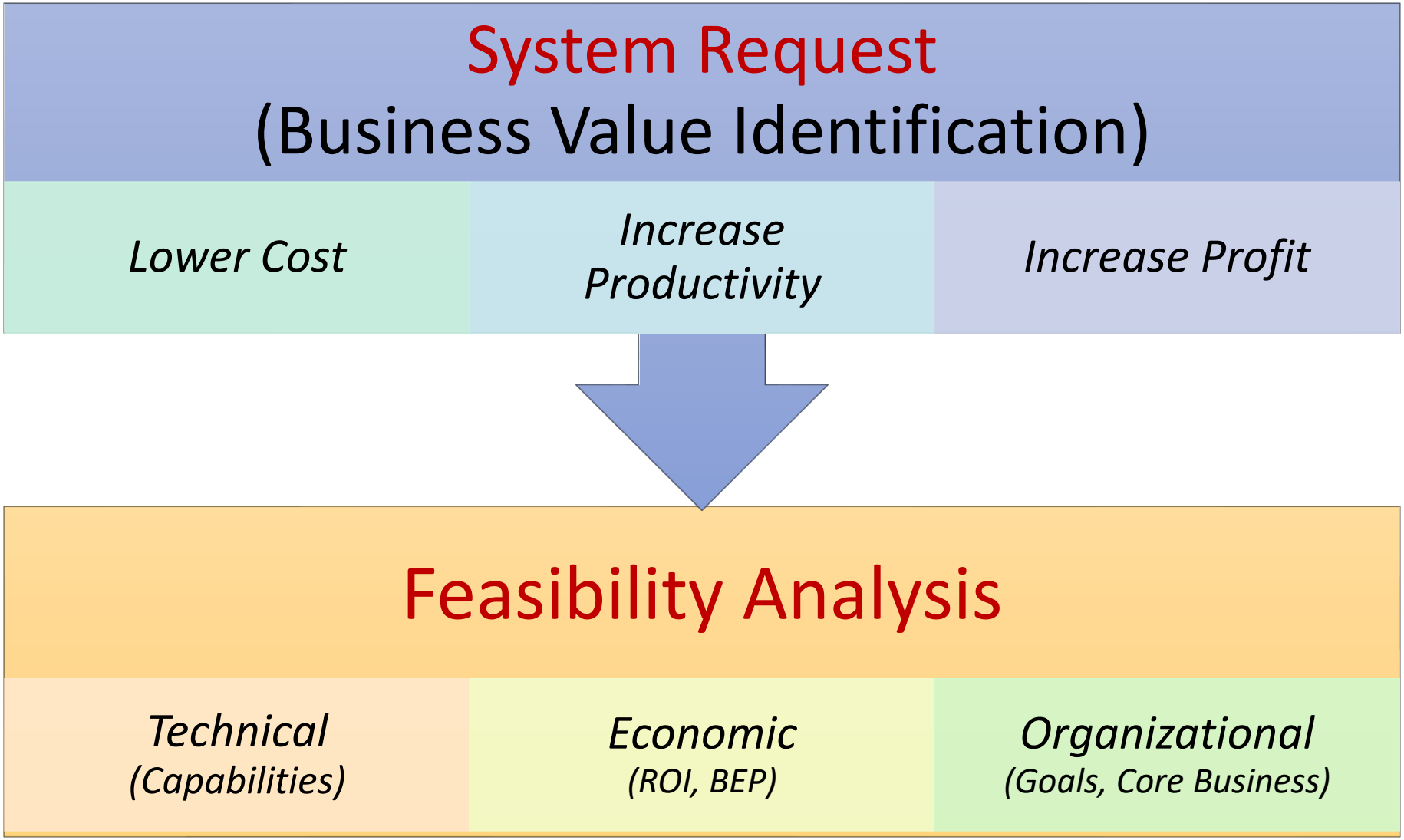
3.1 Analisis Kebutuhan Software
3.2 Identifikasi Proses Bisnis dengan Use Case Diagram
3.3 Pemodelan Proses Bisnis dengan AD atau BPMN
3.4 Realisasi Proses Bisnis dengan Sequence Diagram

Siklus Pengembangan Software

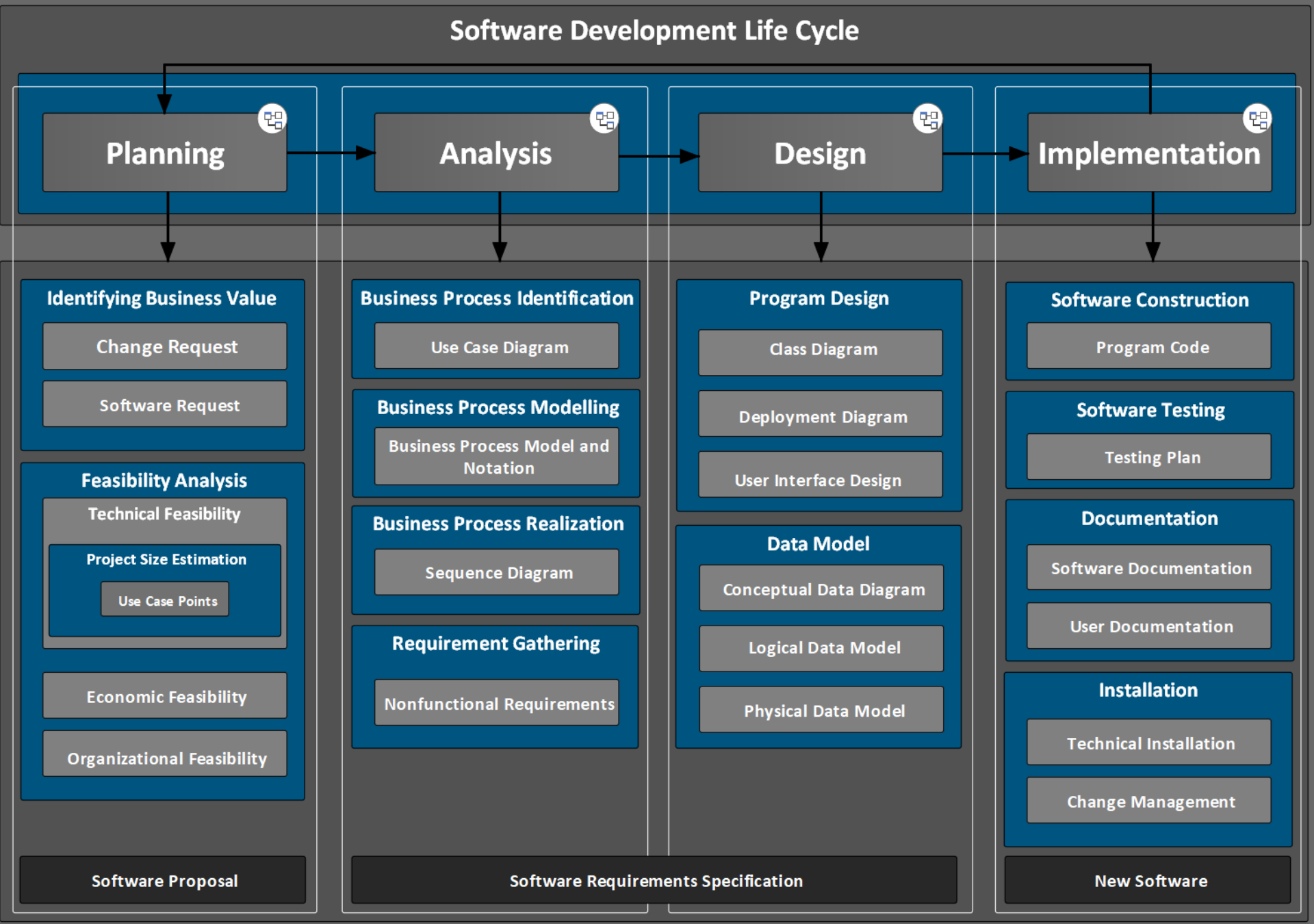


(Tilley, 2012) *(Dennis, 2016)* *(Valacich, 2017)*

Planning


System Request
(Business Value Identification)
Lower Cost
Increase Productivity
Increase Profit
Feasibility Analysis
Technical
(Capabilities)
Economic
(ROI, BEP)
Organizational
(Goals, Core Business)

Application Development Governance



3.1 Requirement Gathering

3.1.1 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

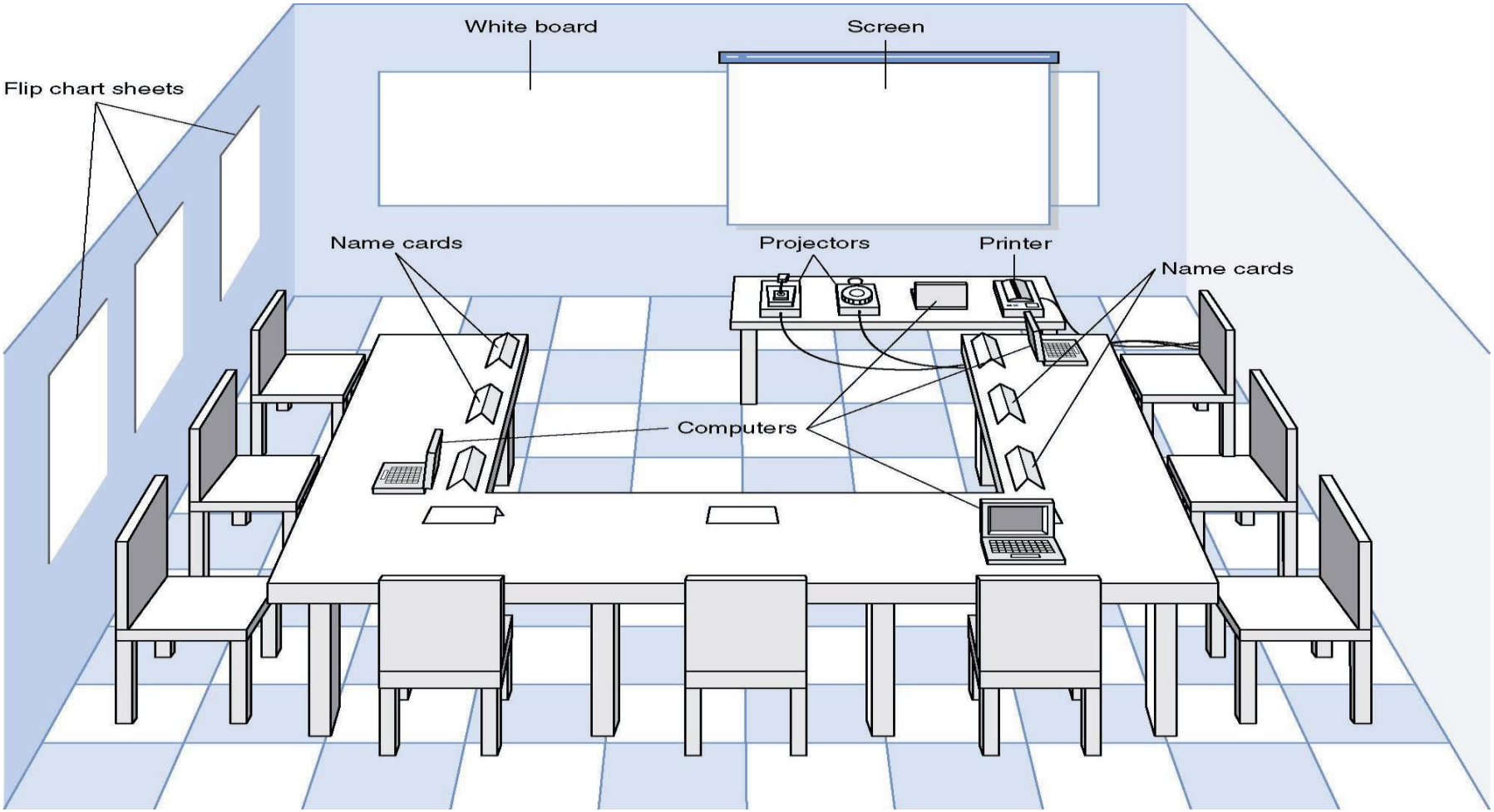
2. Interviews

- 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

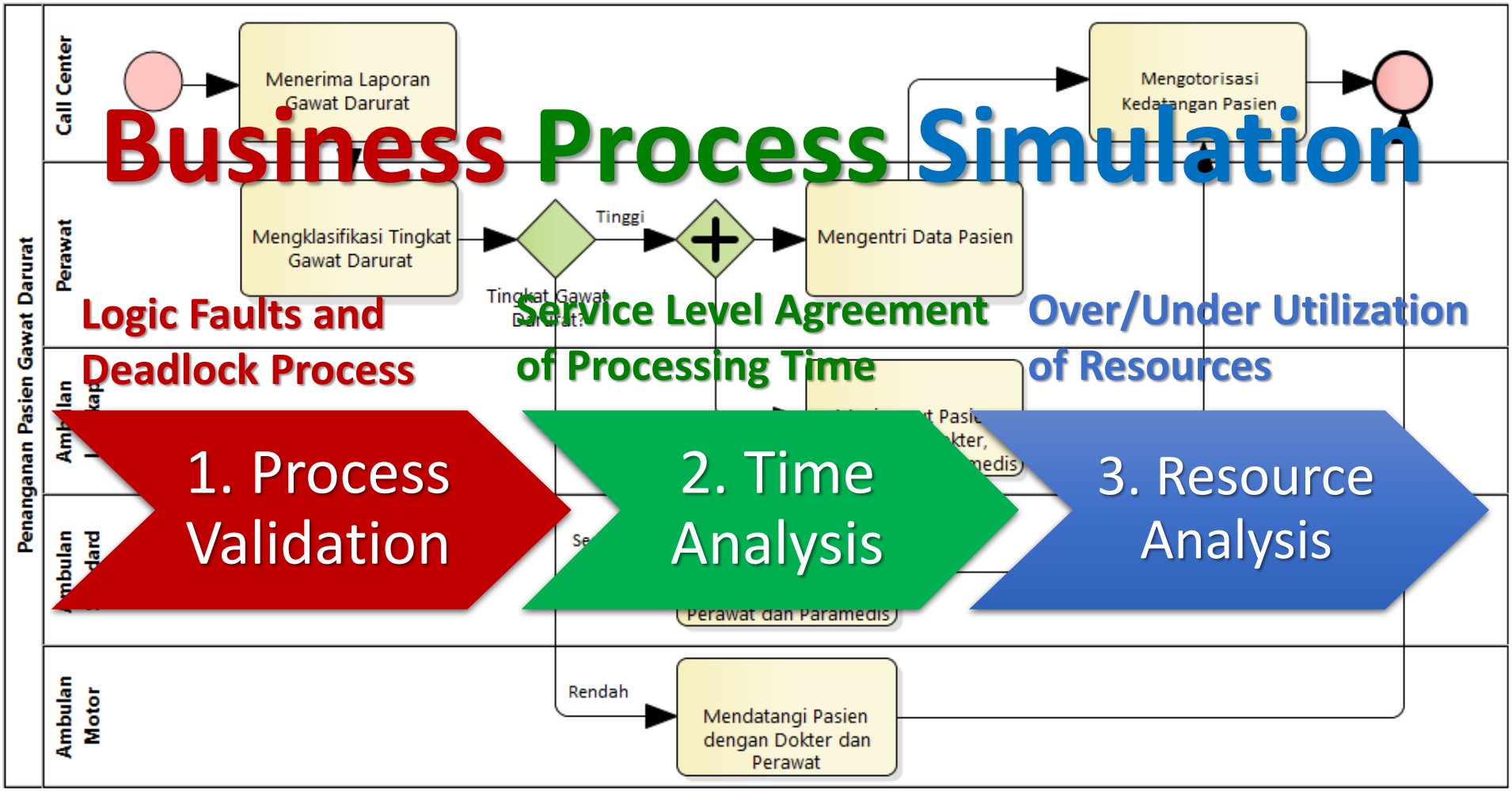
2. BPI (Business Process Improvement)
   - Change what the users do, not just how efficiently they do it

3. BPR (Business Process Reengineering)
   - Throw away everything, start with a blank page

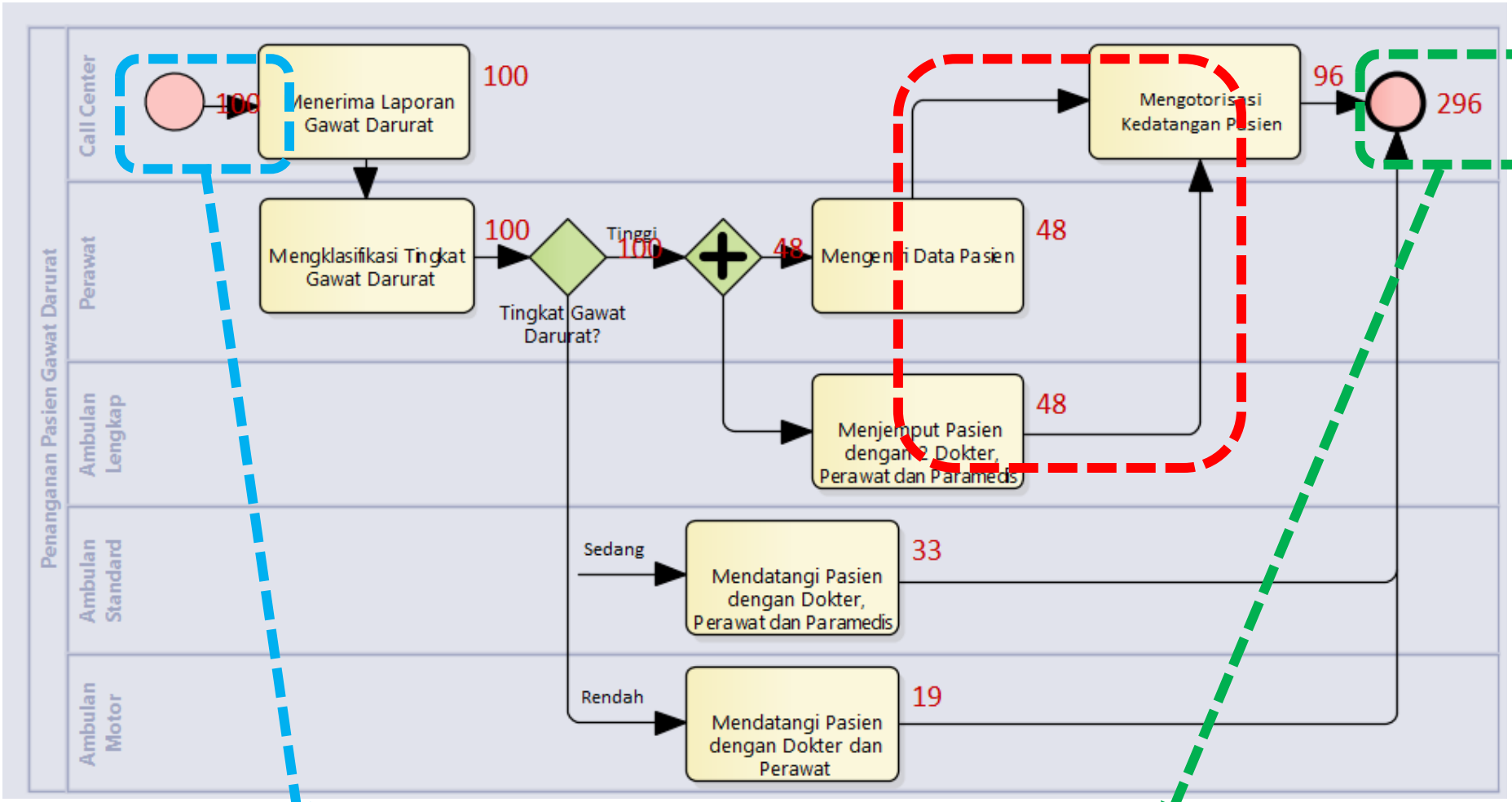
Business Process Analysis Strategies Comparison

	Business Process Automation	Business Process Improvement	Business Process Reeingineering
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

Business Process Simulation: Penanganan Pasien Gawat Darurat



Validasi Proses Bisnis

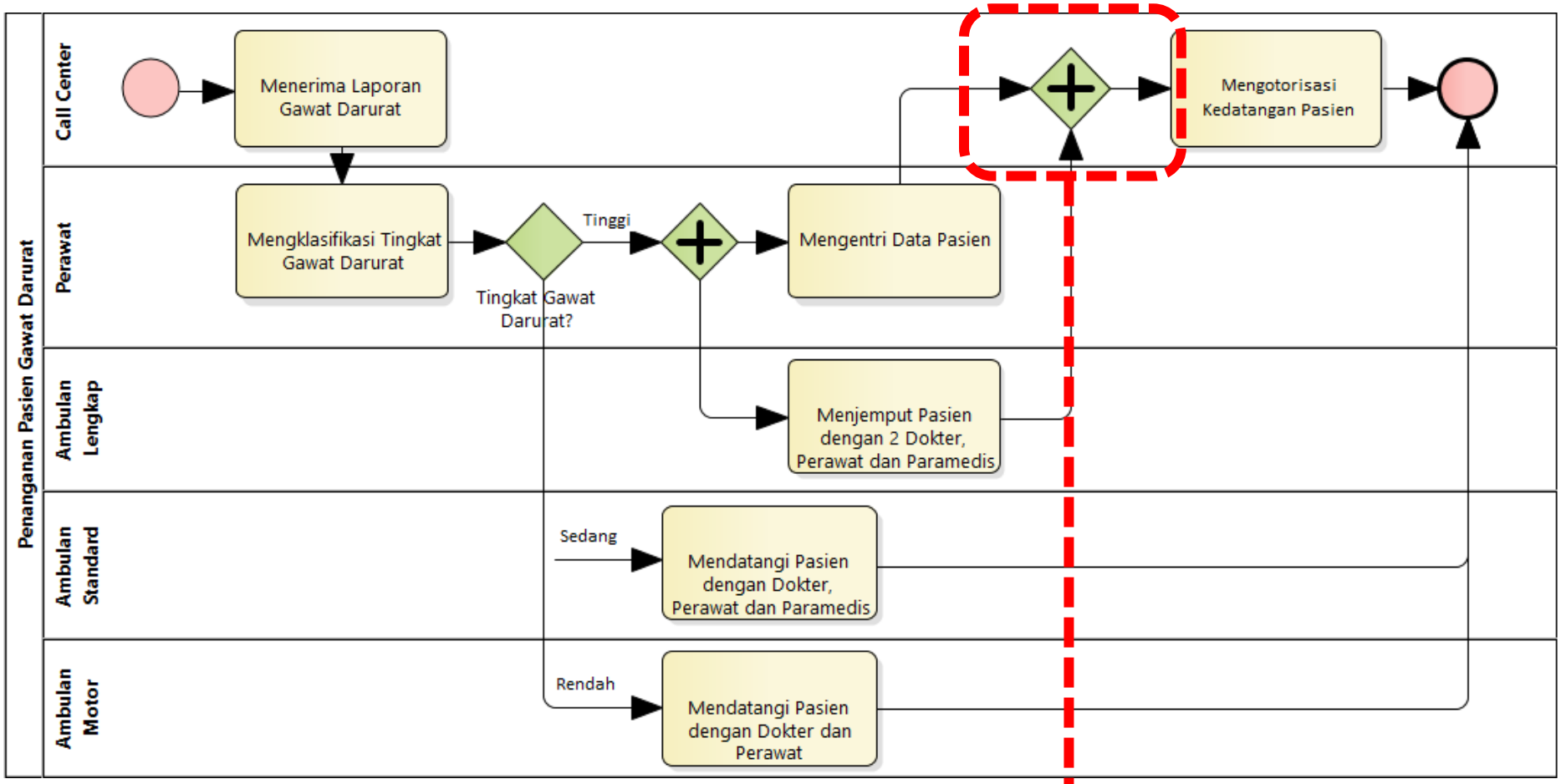


Menangani 100 Orang Pasien

Seperti Menangani 296 Orang Pasien

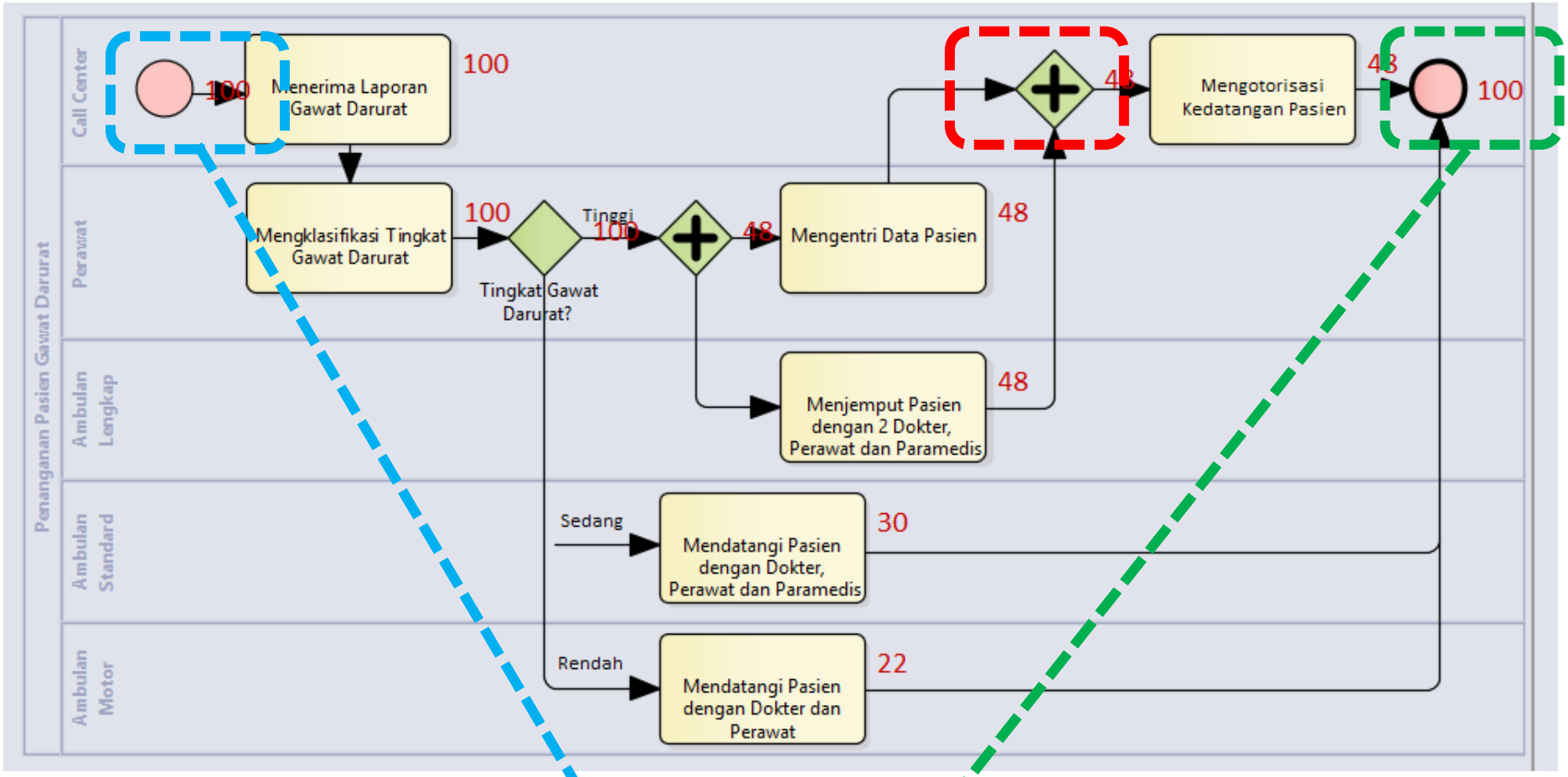
PROSES BISNIS INI TIDAK EFISIEN!

Validasi Proses Bisnis



Perlu Sinkronisasi Pekerjaan,
Sebelum Otorisasi Kedatangan Dilakukan

Validasi Proses Bisnis

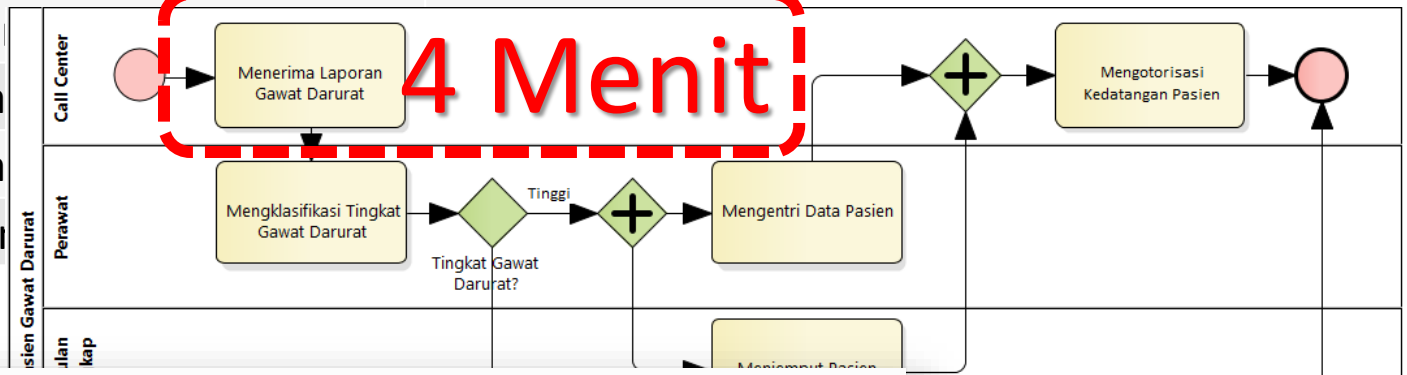


Proses Bisnis Terbukti Valid dan

100 Pasien Tertangani dengan Efisien dan Efektif!

Analisis Waktu

Aktifitas	Waktu (Menit)
Menerima Laporan Gawat Darurat	4
Mengklasifikasi Tingkat Gawat Darurat	5
Mengentri Data Pasien	11
Menjemput Pasien denga	
Mendatangi Pasien denga	
Mendatangi Pasien denga	
Mengotorisasi Kedatanga	



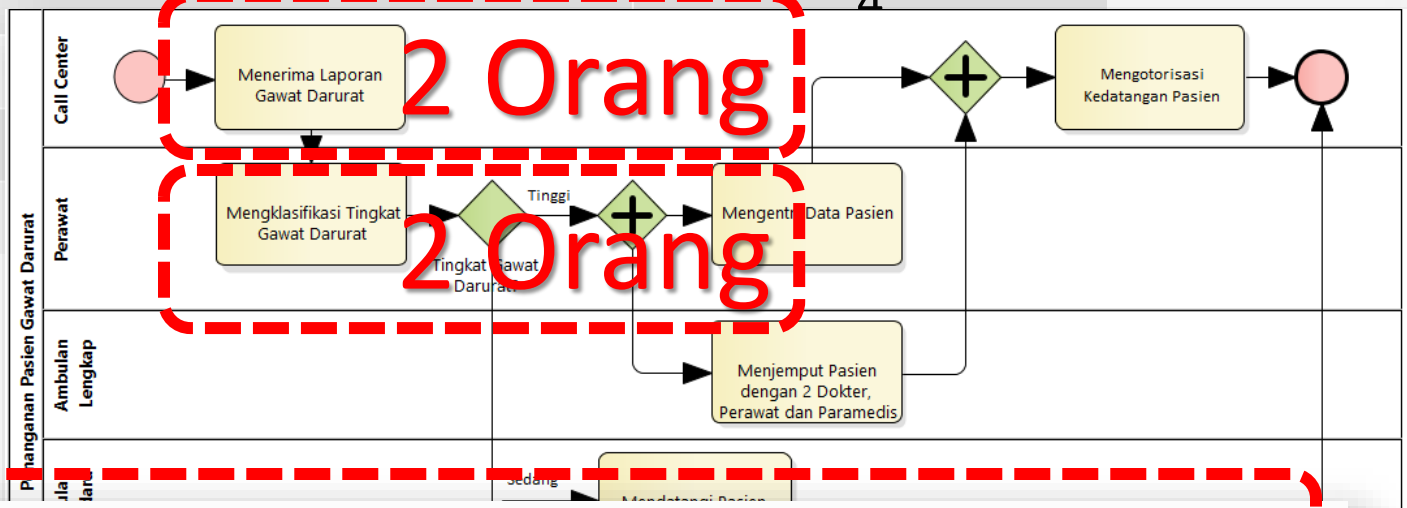
4 Menit

Penanganan Pasien Gawat Darurat	Average Time	249.63
Penanganan Pasien Gawat Darurat	Mean Of Processing Time	17.73
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat	Number Of Processes Started	100
Penanganan Pasien Gawat Darurat	Standard Deviation Time	131.97

Perlu Waktu 17.73 Menit untuk Menangani 1 Pasien

Analisis Sumber Daya Manusia

Sumber Daya	Kuantitas
Call center	2
Perawat	2
Ambulan Lengkap	4
Ambulan Standar	
Ambulan Motor	
Resepsionis	

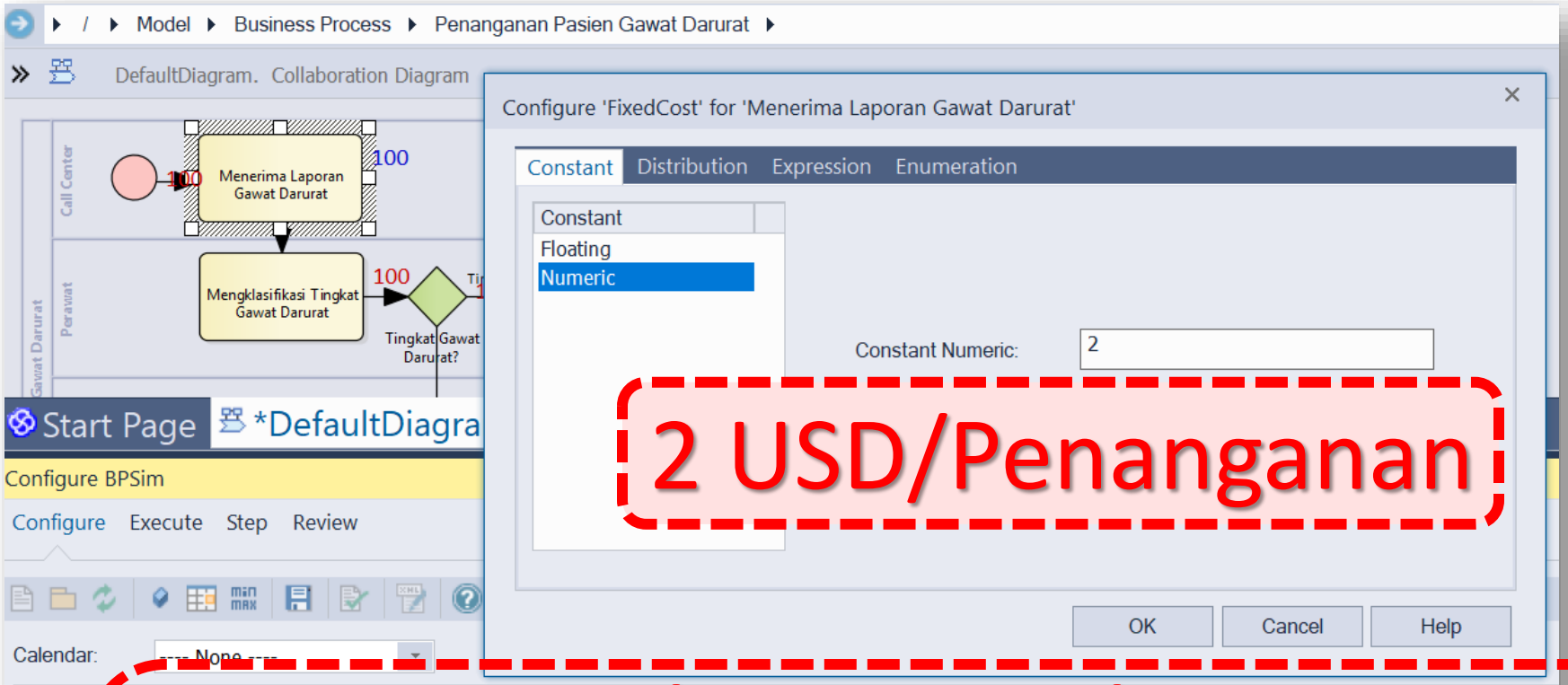


Overutilization

Underutilization

Perawat	Degree Of Utilisation	99%
Paramedis	Degree Of Utilisation	98.34%
Call Center	Degree Of Utilisation	65.56%
Dokter	Degree Of Utilisation	50.28%
Supir	Degree Of Utilisation	33.22%
Ambulan Standard	Degree Of Utilisation	33.22%
Ambulan Motor	Degree Of Utilisation	17.05%

Analisis Biaya



2 USD/Penanganan

Total Biaya Perhari

Menerima Laporan Gawat Darurat	Total Completion Cost	200
Mengentri Data Pasien	Total Completion Cost	48
Mengklasifikasi Tingkat Gawat Darurat	Total Completion Cost	100
Mengotorisasi Kedatangan Pasien	Total Completion Cost	48

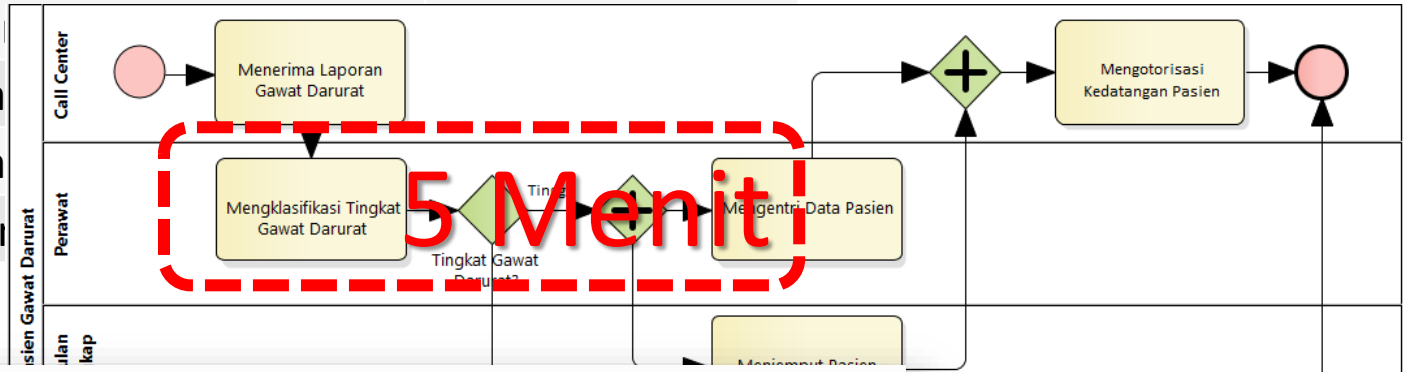
Business Process Automation

- Dikembangkan Sistem Cerdas yang membantu perawat dengan Mengautomasi Klasifikasi Tingkat Gawat Darurat

- Pekerjaan "Mengklasifikasi Tingkat Gawat Darurat" yang sebelumnya perlu waktu 5 menit
  - Dengan sistem cerdas menjadi hanya 1 menit

Business Process Simulation

Aktifitas	Waktu (Menit)
Menerima Laporan Gawat Darurat	4
Mengklasifikasi Tingkat Gawat Darurat	5
Mengentri Data Pasien	11
Menjemput Pasien denga	
Mendatangi Pasien denga	
Mendatangi Pasien denga	
Mengotorisasi Kedatanga	

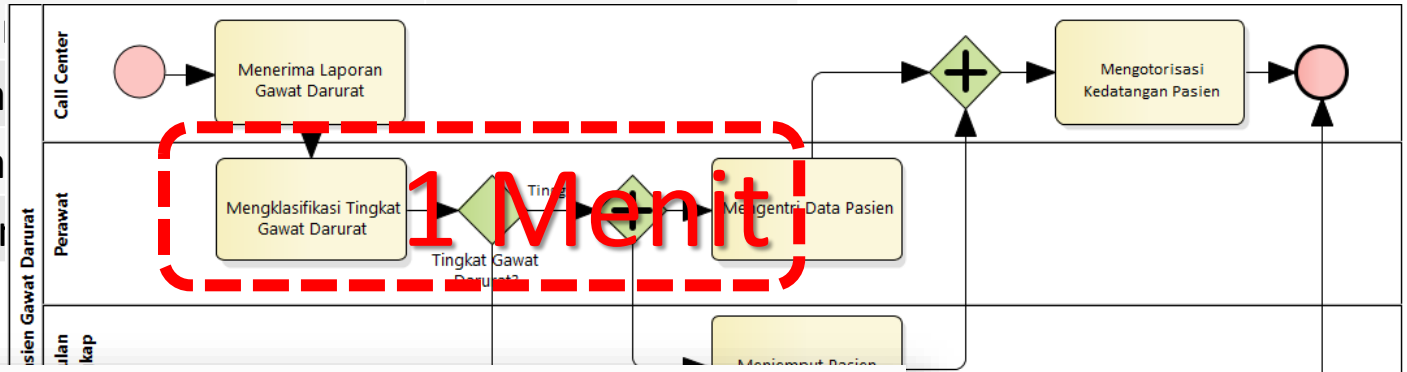


Penanganan Pasien Gawat Darurat	Average Time	249.63
Penanganan Pasien Gawat Darurat	Mean Of Processing Time	17.73
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat	Number Of Processes Started	100
Penanganan Pasien Gawat Darurat	Standard Deviation Time	131.97

Perlu Waktu 17.73 Menit untuk Menangani 1 Pasien

Business Process Automation

Aktifitas	Waktu (Menit)
Menerima Laporan Gawat Darurat	4
Mengklasifikasi Tingkat Gawat Darurat	1
Mengentri Data Pasien	11
Menjemput Pasien denga	
Mendatangi Pasien denga	
Mendatangi Pasien denga	
Mengotorisasi Kedatangai	



Penanganan Pasien Gawat Darurat	Average Time	249.63
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat		
Penanganan Pasien Gawat Darurat	Number Of Processes Started	100
Penanganan Pasien Gawat Darurat	Standard Deviation Time	131.97

Hanya Perlu Waktu 8 Menit untuk Menangani 1 Pasien

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, hmmmm, 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.

3.1.2 Paradigma Analysis dan Design

Evolusi Paradigma Analysis dan Design

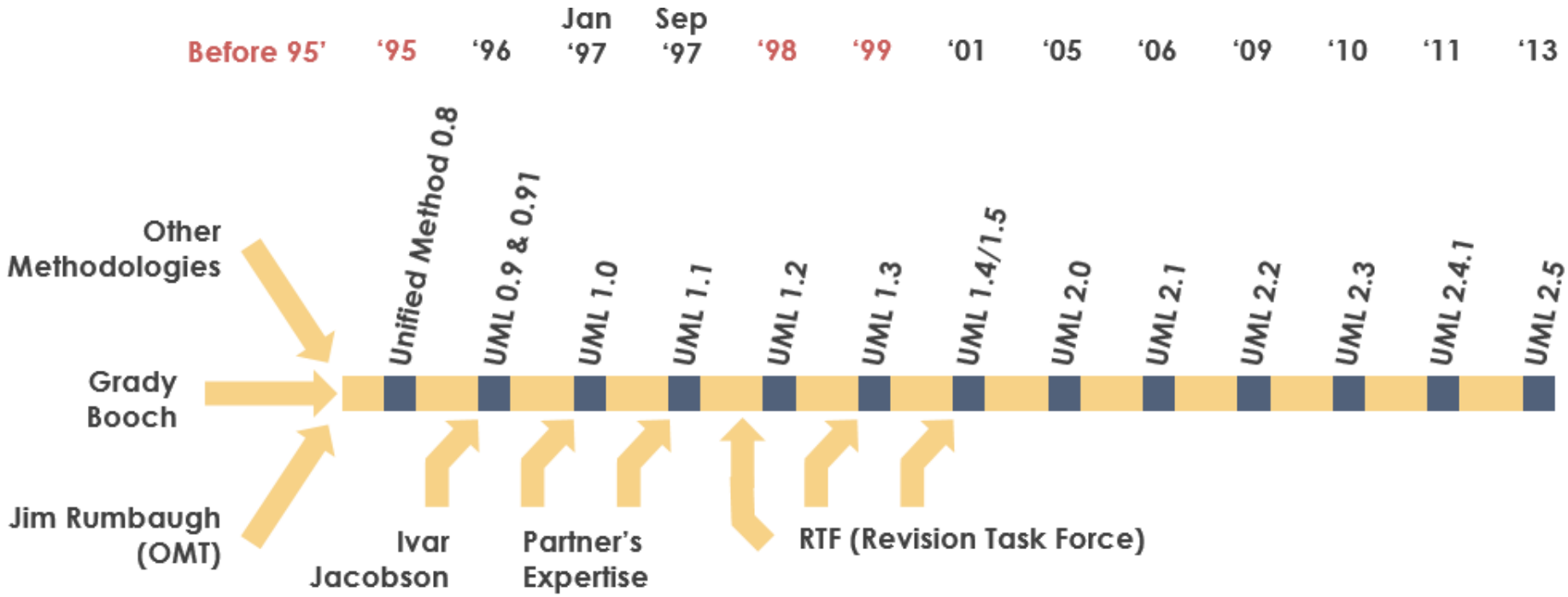
	Paradigm	Diagrams
1	Process-oriented Paradigm	Flowchart
2	Data-oriented Paradigm	DFD
3	Object-oriented Paradigm (data + process)	UML

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

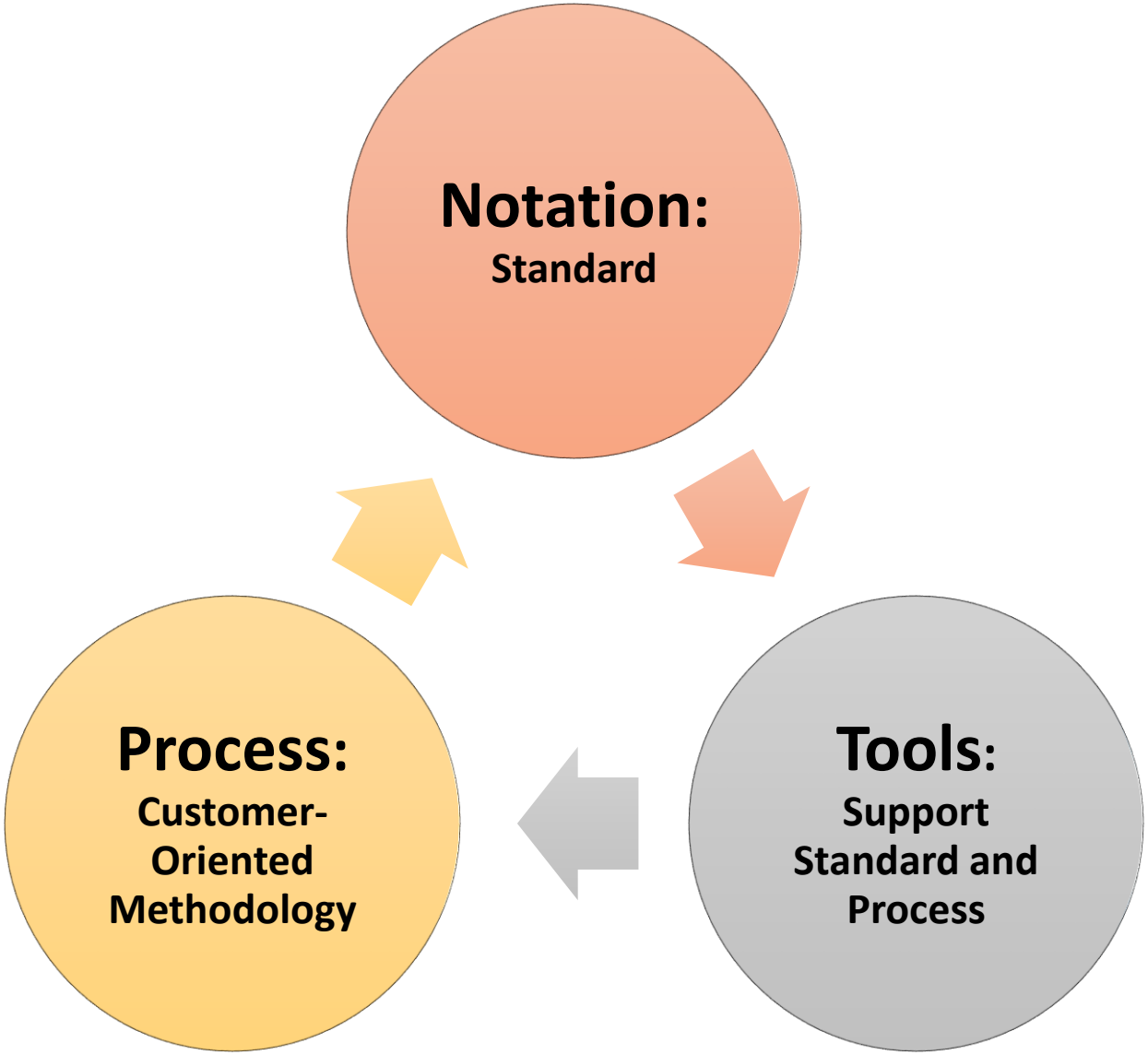
Sejarah UML

Booch, Jacobson, Rumbaugh



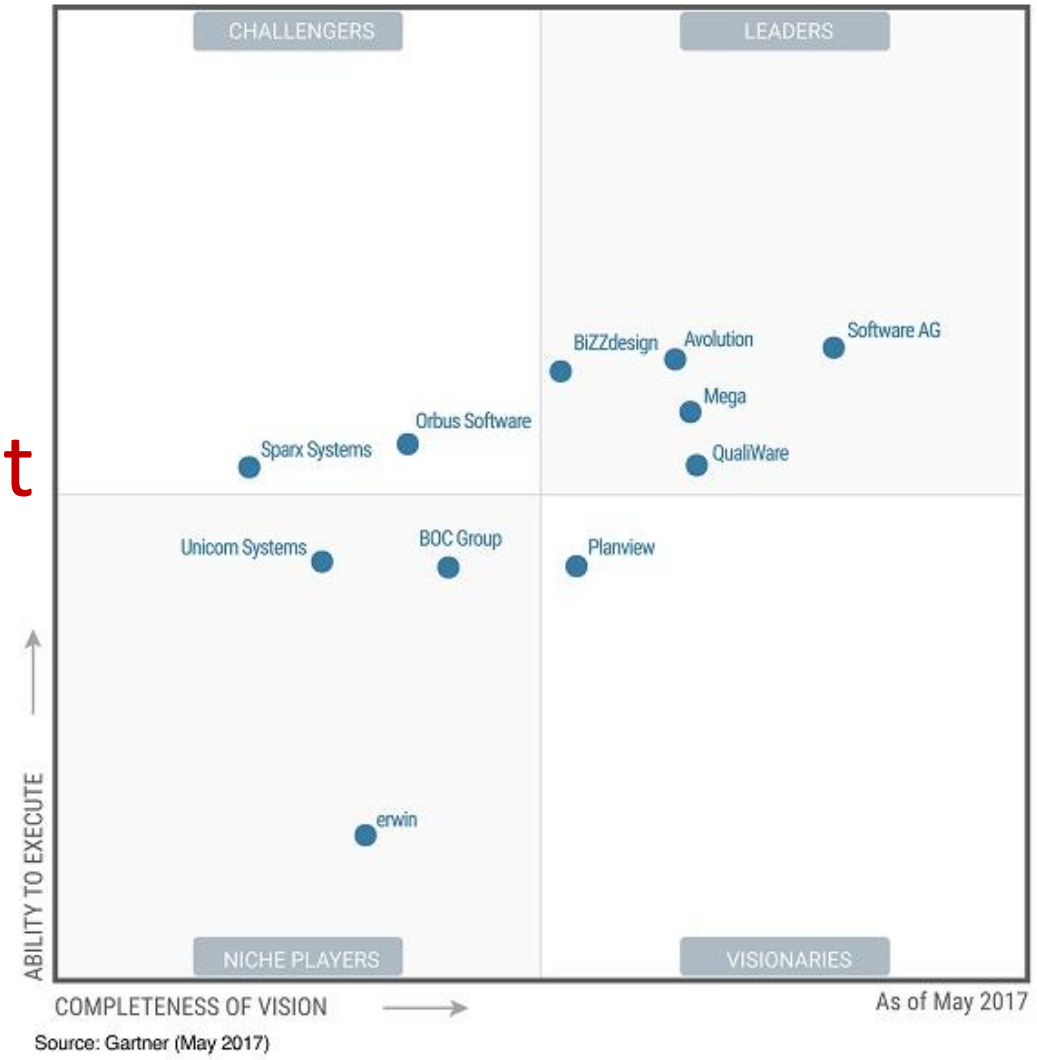
Before 95' - Fragmentation ▶ 95' - Unification ▶ 98' - Standardization ▶ 99' - Industrialization

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?

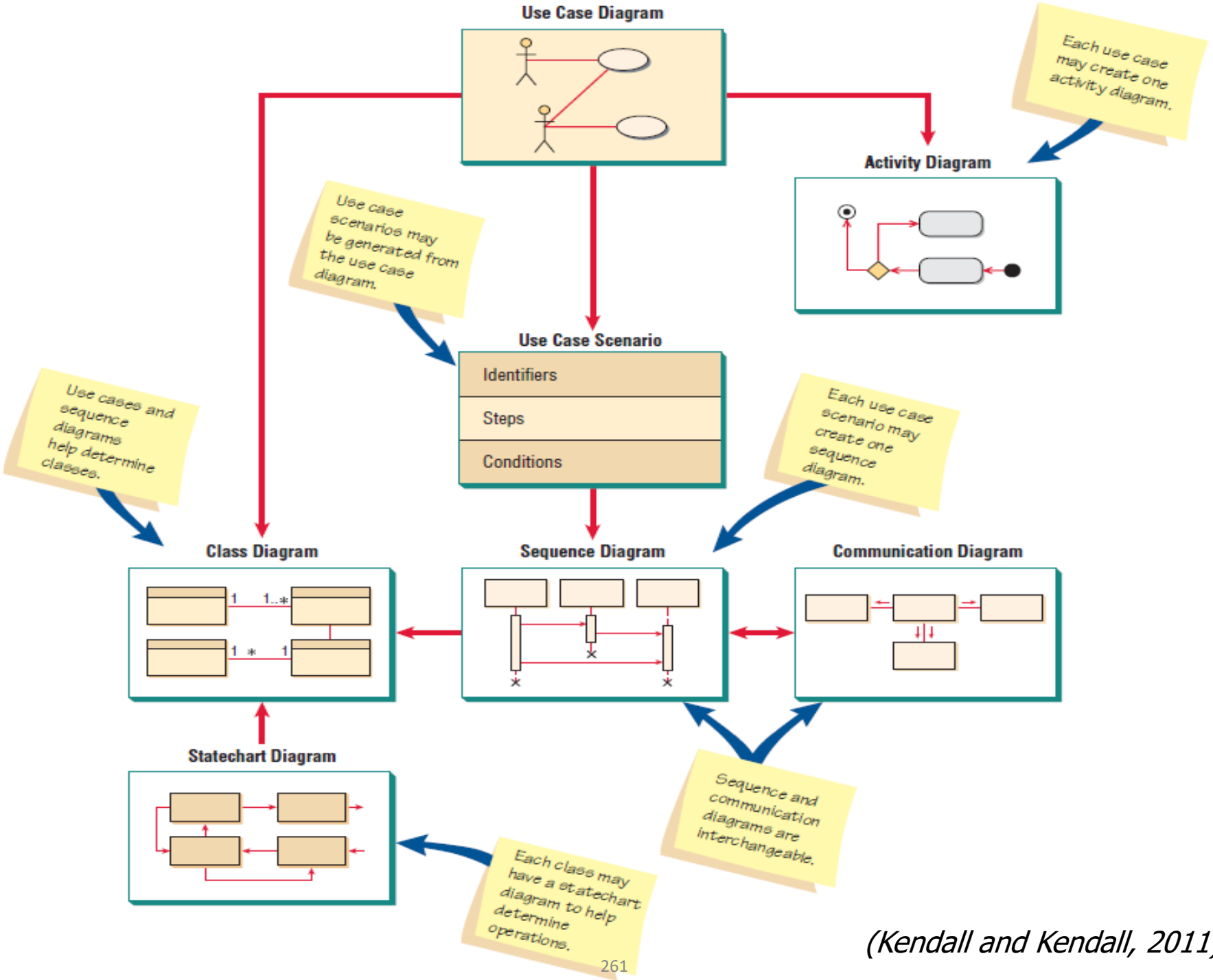
2. UML is too complex, difficult to understand quickly
   - Which UML diagrams should we use?

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

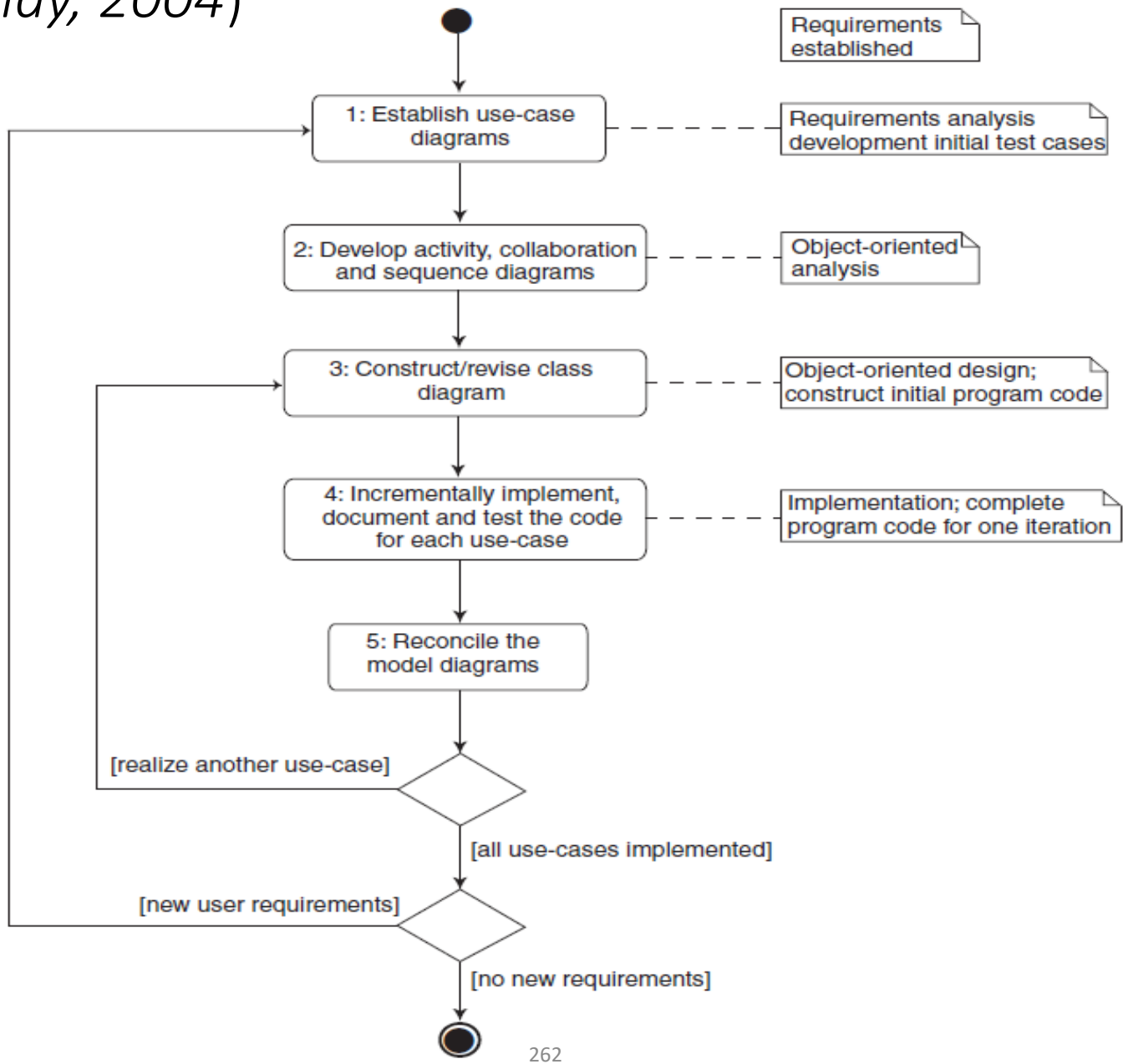
UML based Software Analysis and Design (*Kendal, 2011*)

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



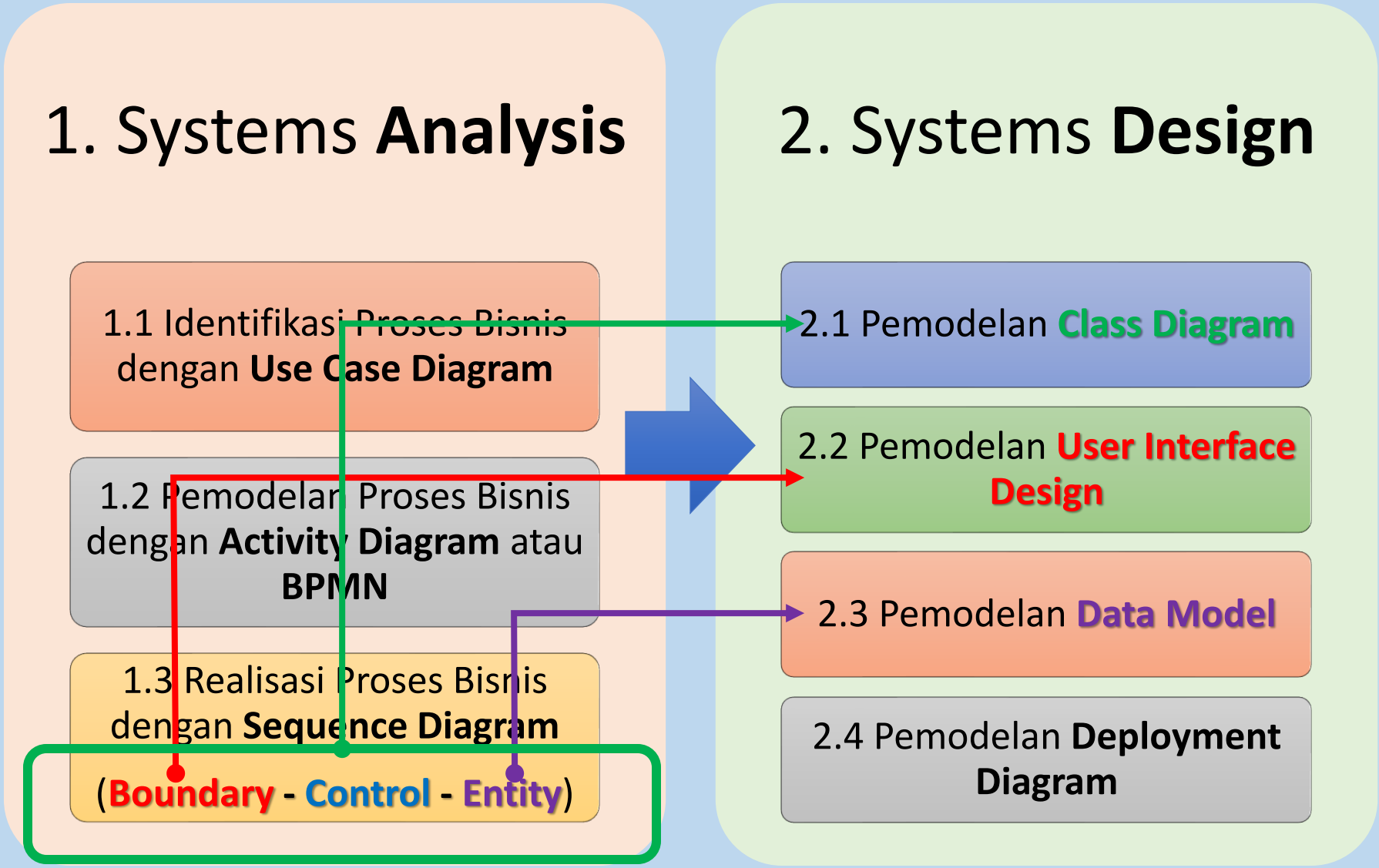
(Kendall and Kendall, 2011)

UML based Software Analysis and Design
(*Barclay, 2004*)

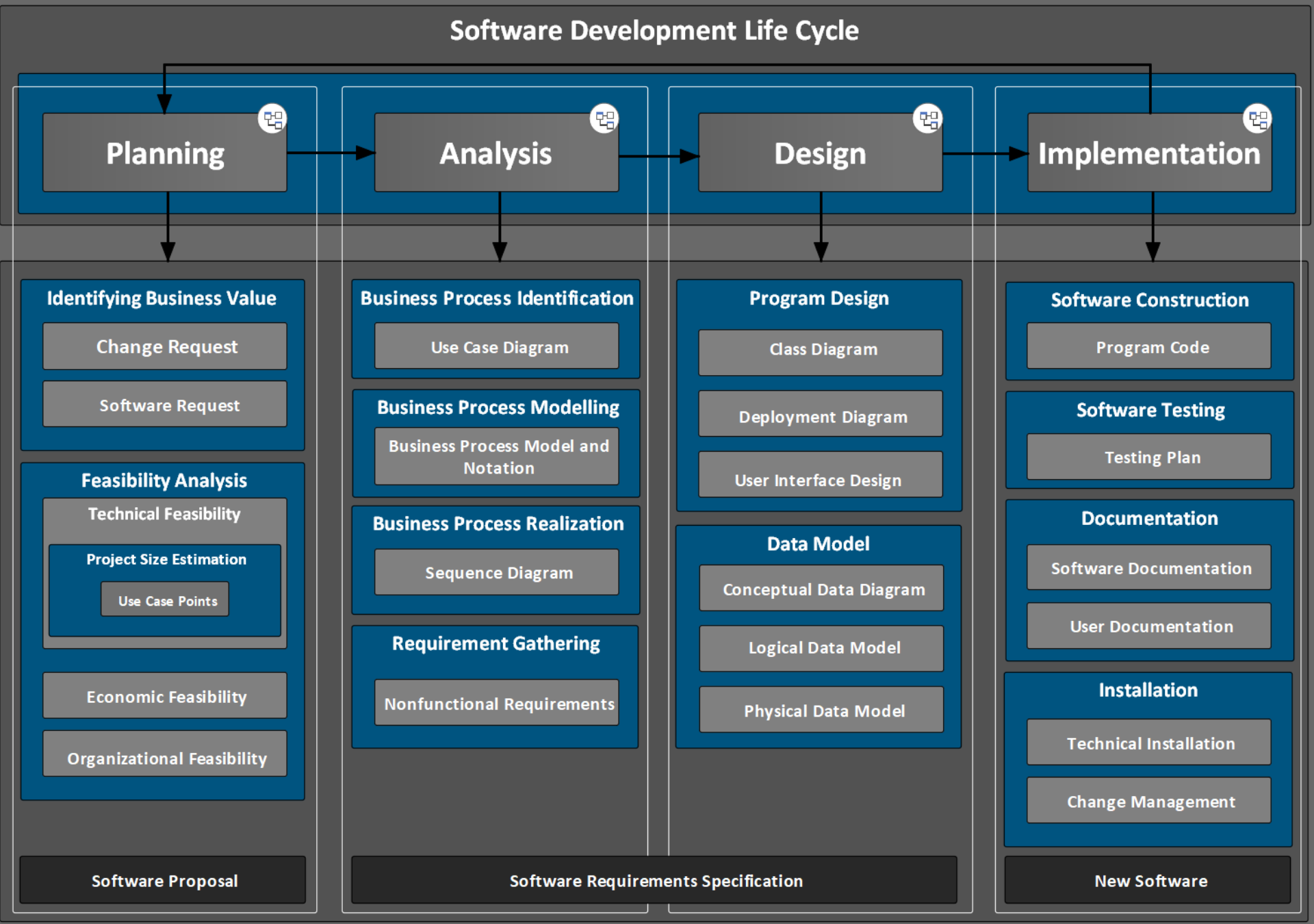


UML based Software Analysis and Design

(Wahono, 2009)

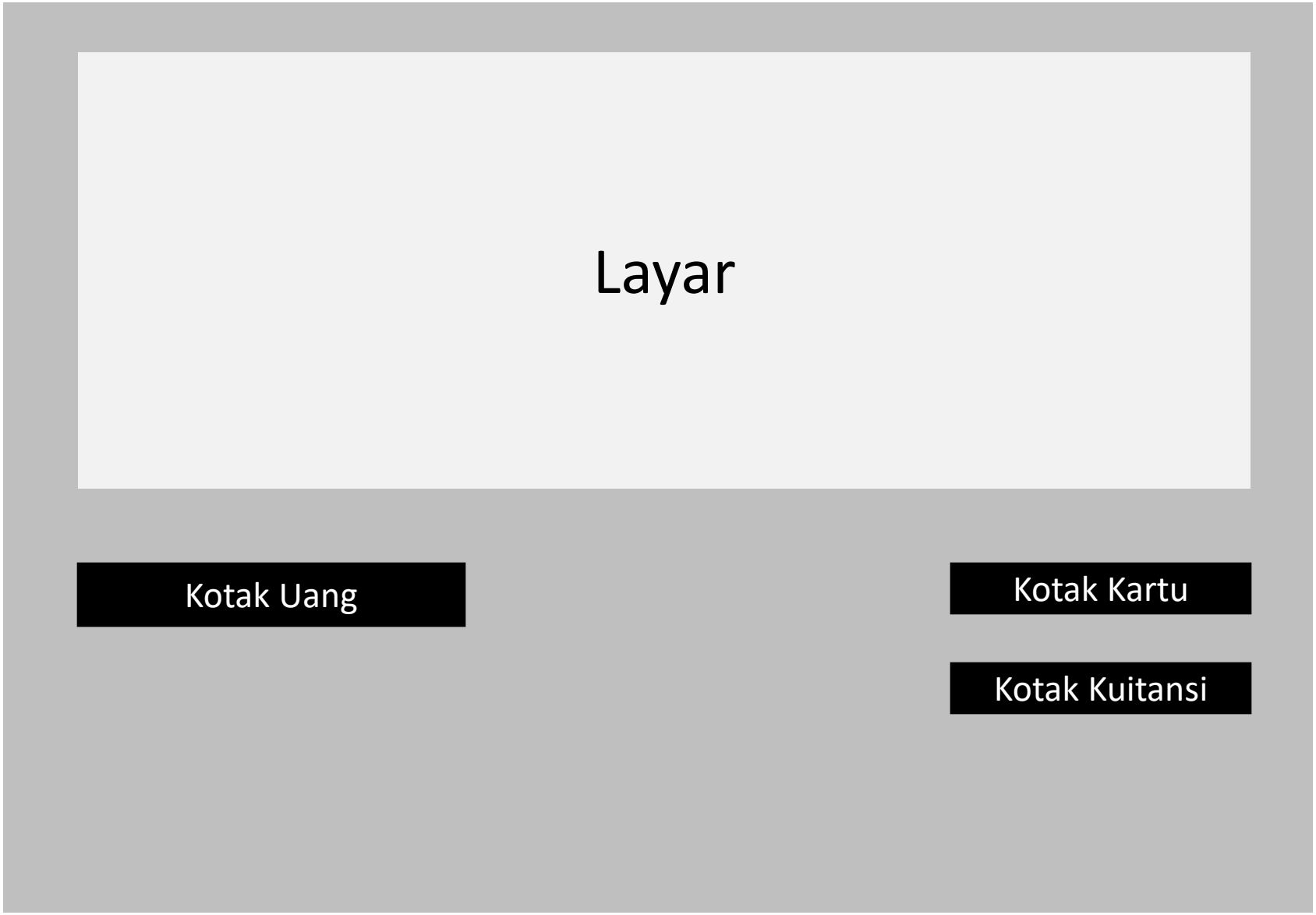


Application Development Governance



Case Study: ATM System

ATM System



Menu PIN

Masukkan PIN:

Kotak Uang

Kotak Kartu

Kotak Kuitansi

Menu Utama

1. Mengecek Saldo
2. Mentransfer Uang
3. Mengambil Uang
4. Logout

Kotak Uang

Kotak Kartu

Kotak Kuitansi

Menu Pengecekan Saldo

1. Saldo anda adalah …

Kotak Uang

Kotak Kartu

Kotak Kuitansi

Menu Pengiriman Uang

1. No Account Penerima:

Kotak Uang

Kotak Kartu

Kotak Kuitansi

Menu Pengiriman Uang

1. Jumlah uang yang dikirim:

Kotak Uang

Kotak Kartu

Kotak Kuitansi

Menu Pengiriman Uang

1. Uang berhasil terkirim

Kotak Uang

Kotak Kartu

Kotak Kuitansi

Menu Pengambilan Uang

1. Jumlah uang yang diambil:

Kotak Uang

Kotak Kartu

Kotak Kuitansi

Menu Pengambilan Uang

Uang berhasil diambil

Kotak Uang

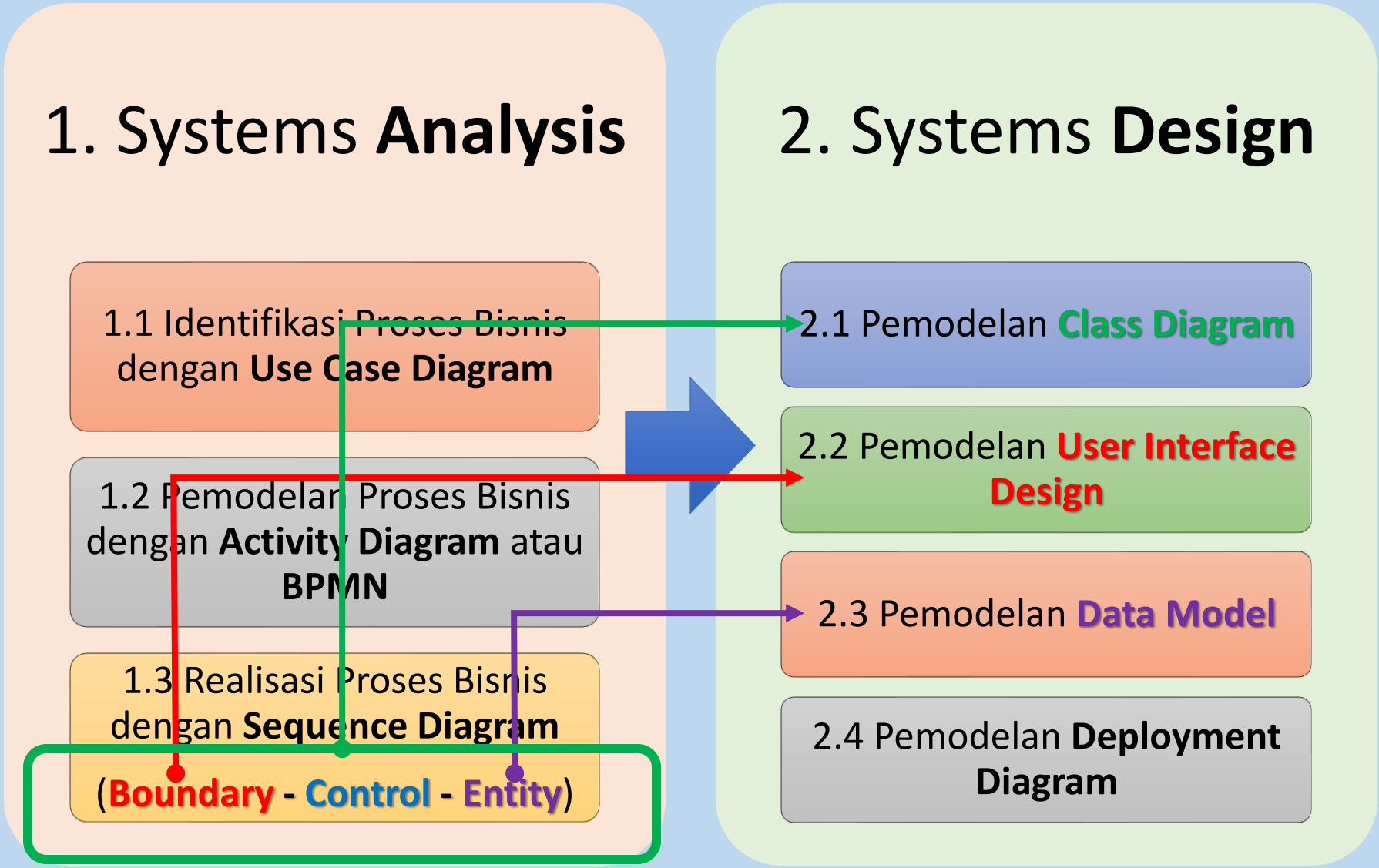
Kotak Kartu

Kotak Kuitansi

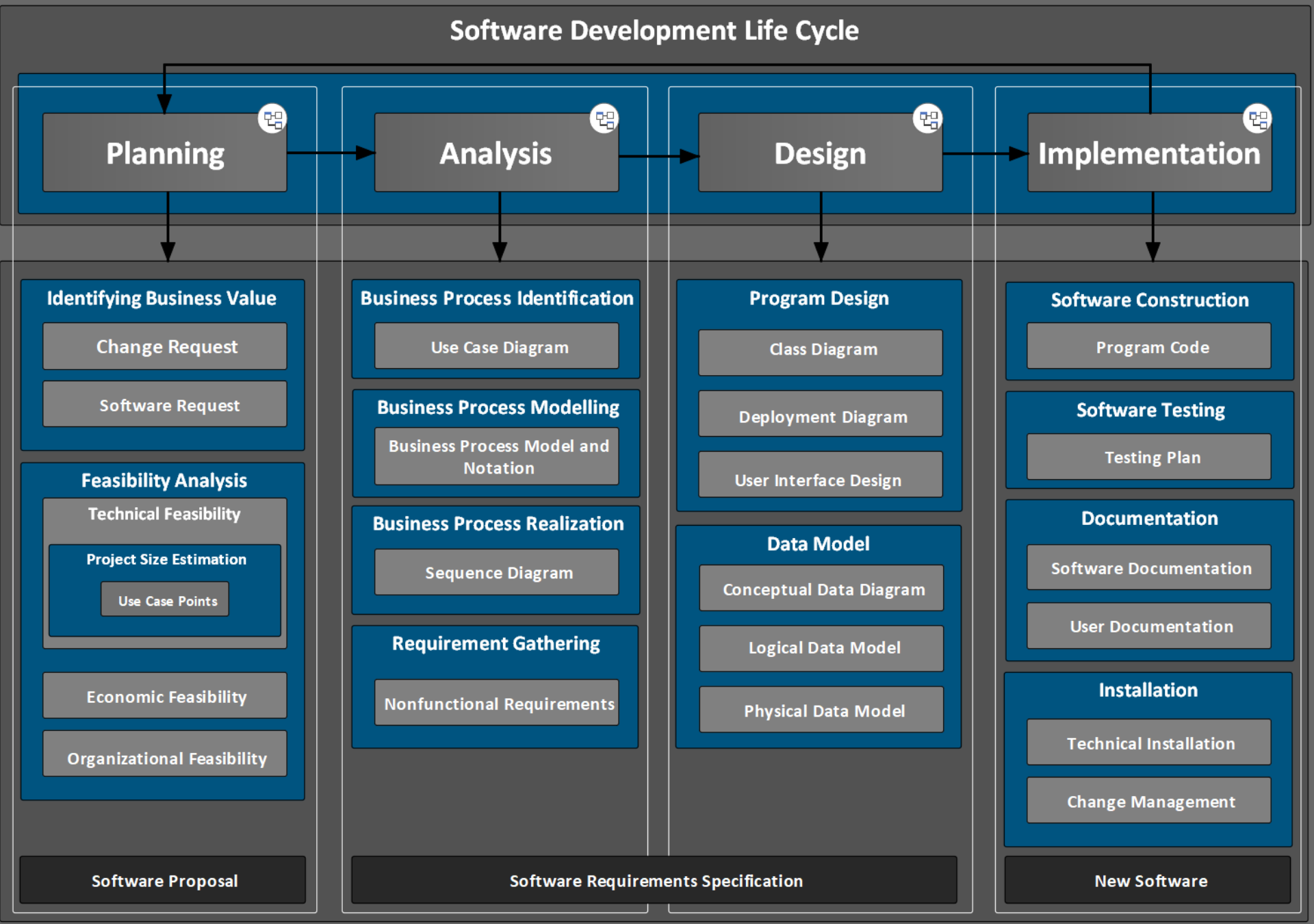
3.2 Identifikasi Proses Bisnis dengan Use Case Diagram

UML based Software Analysis and Design

(Wahono, 2009)



Application Development Governance

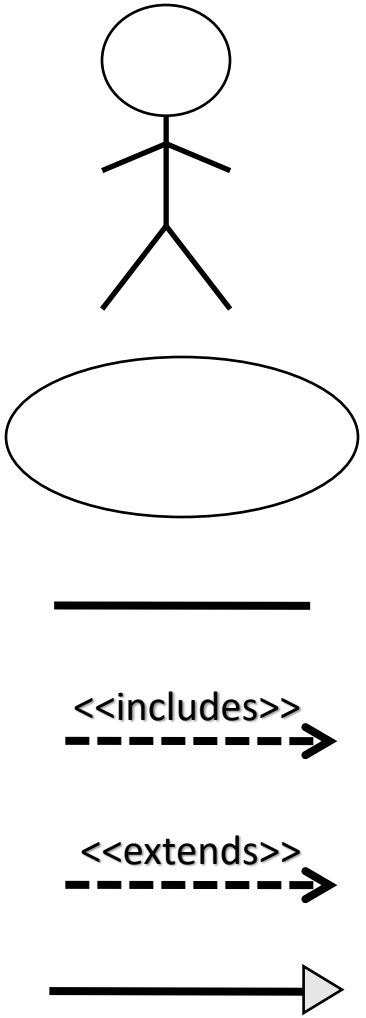


Use Case Diagram

- 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

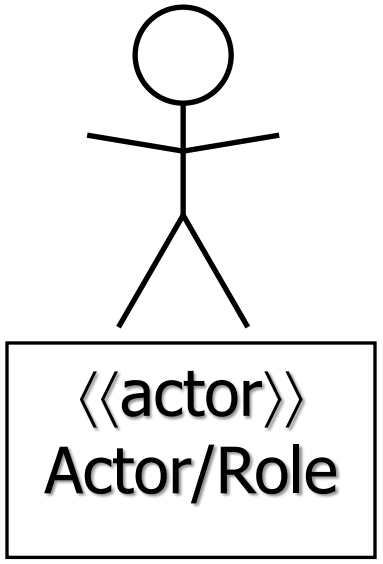
- A major piece of system functionality
- Can extend other Use Cases
- Placed inside system boundary
- Labeled with descriptive verb - noun phrase

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

Actor

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

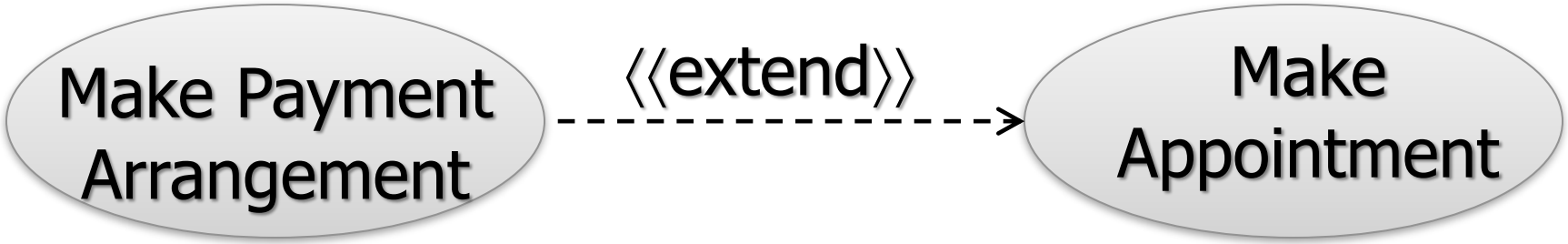


Association Relationship

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

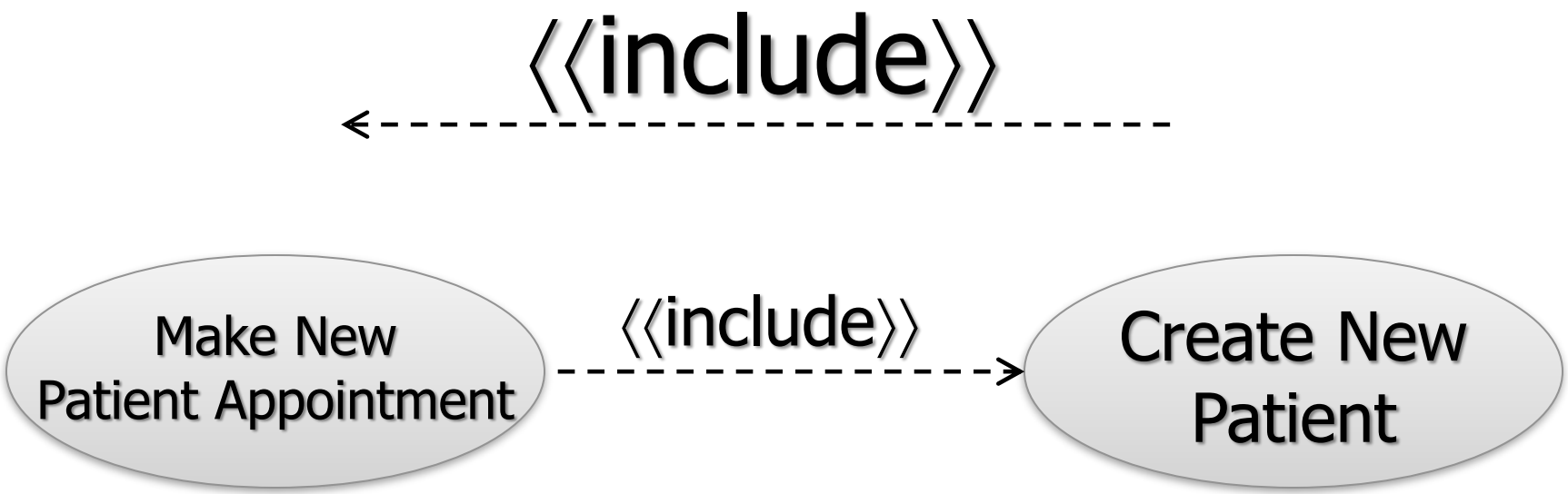
Extends Relationship

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



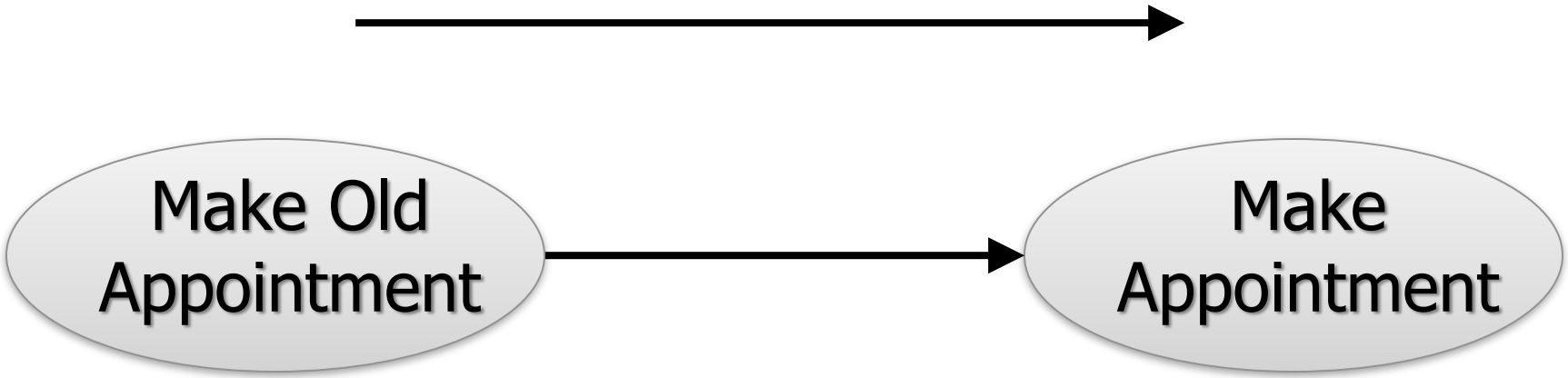
Include Relationship

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

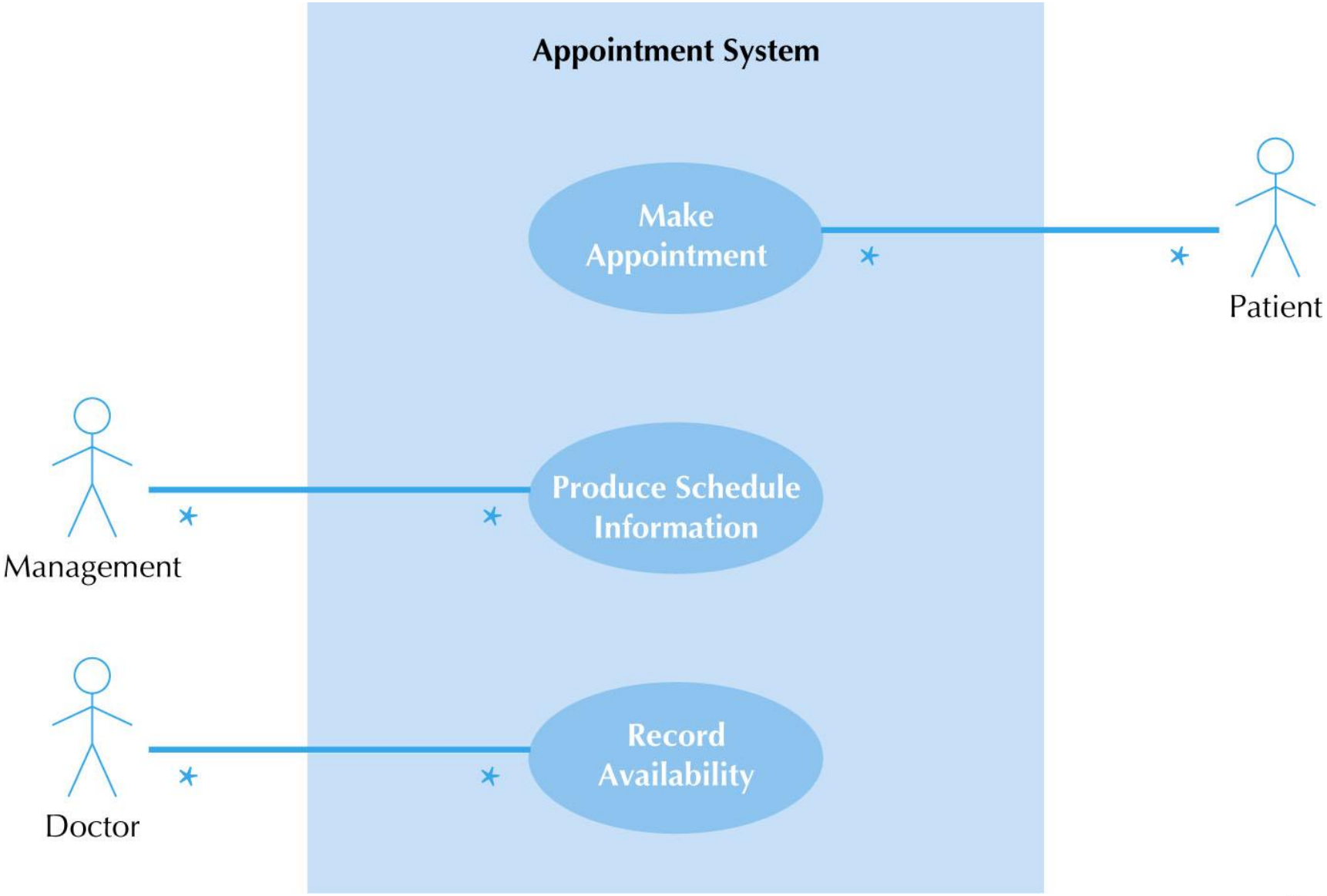


Generalization Relationship

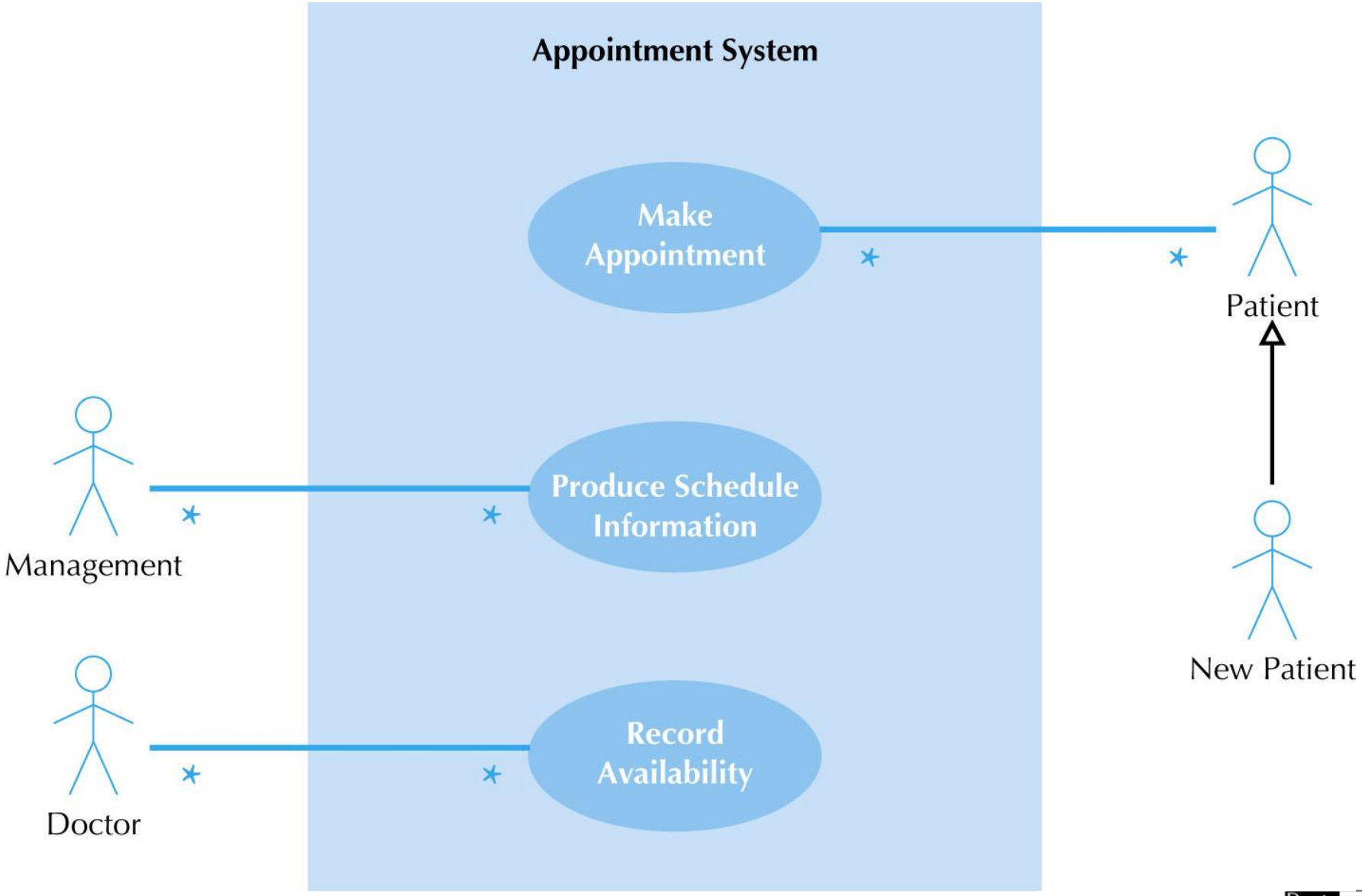
- A specialized Use Case to a more generalized Use Case
- Arrow points from specialized to general Use Case



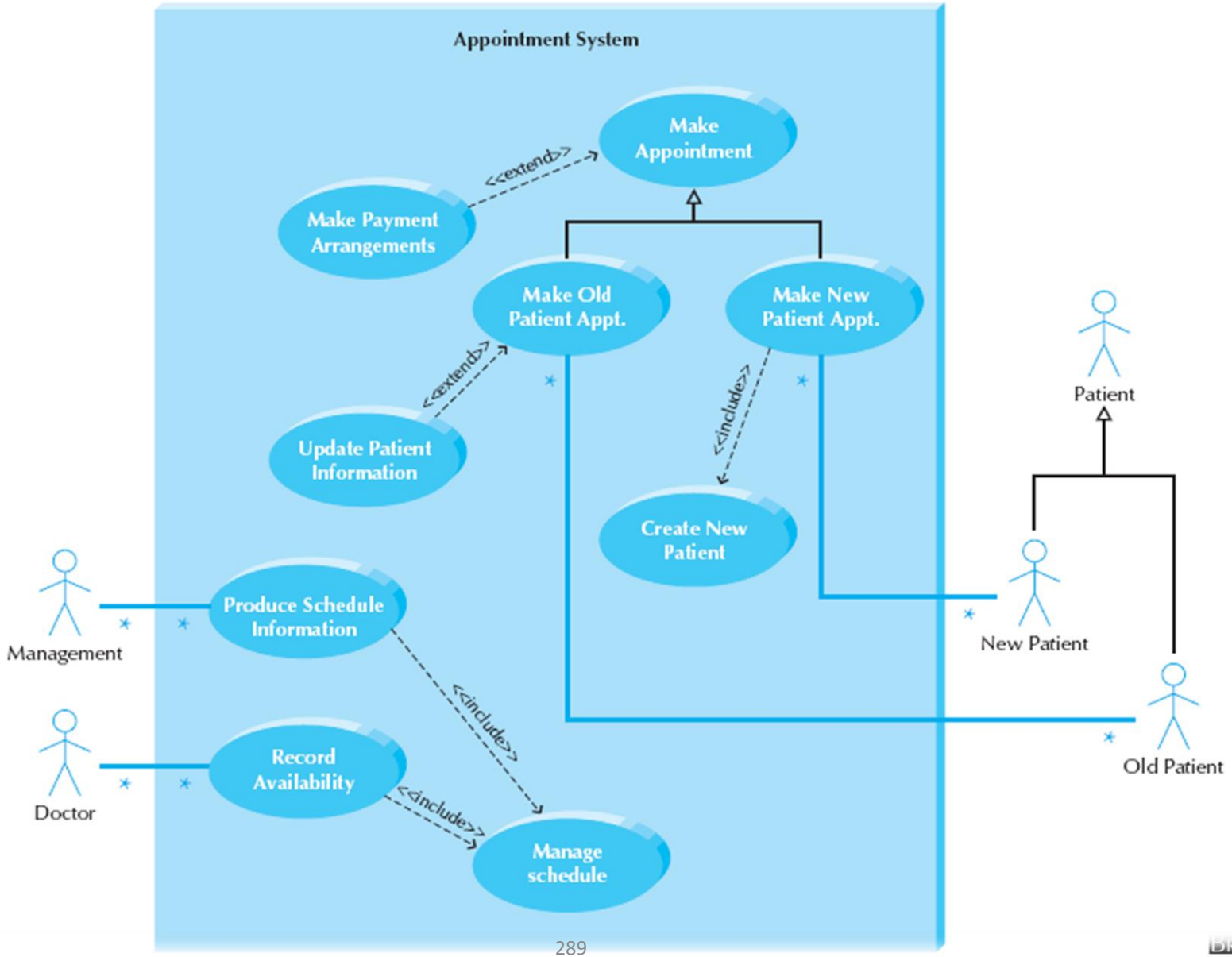
Use Case Diagram for Appointment System



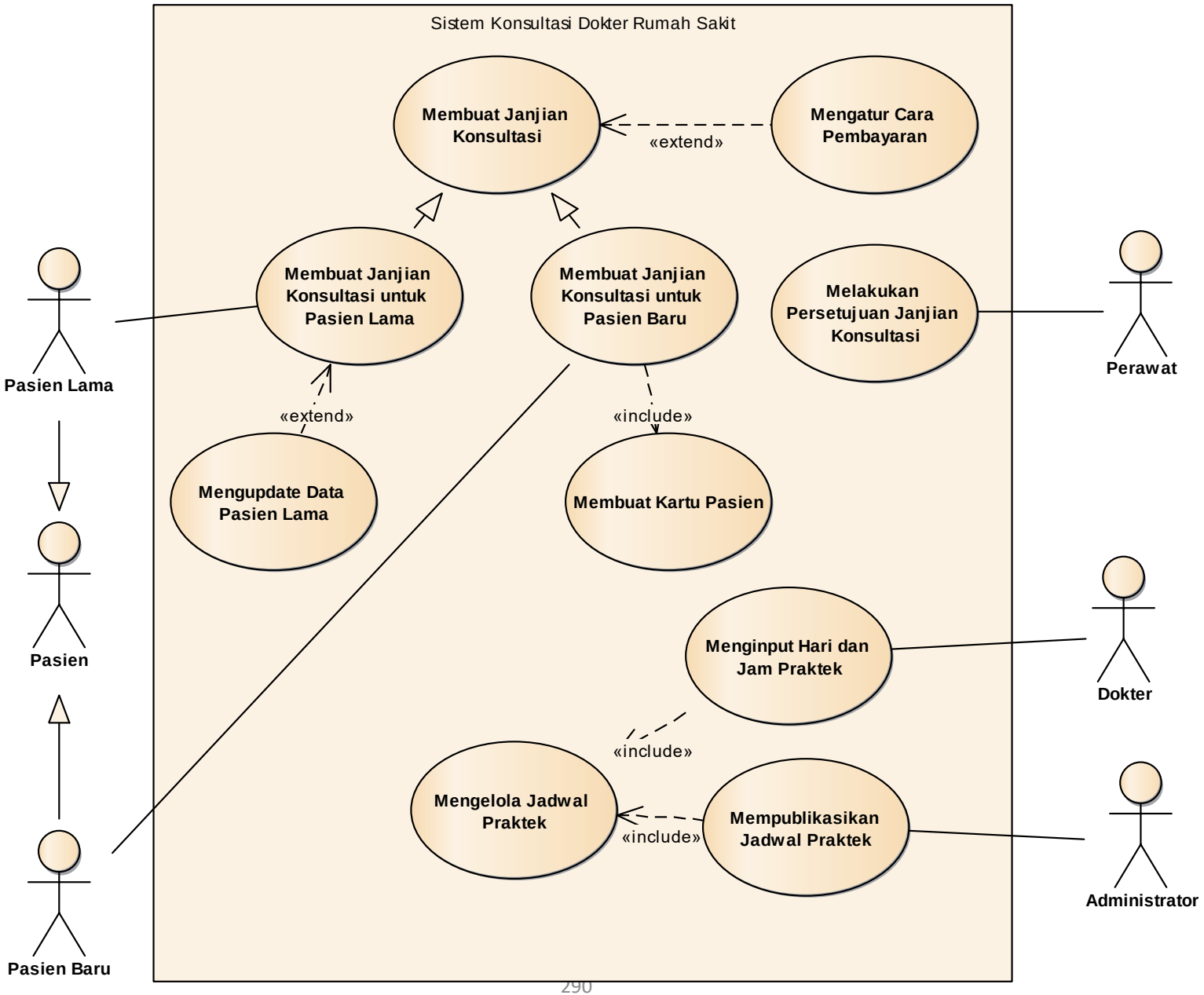
Use Case Diagram with Specialized Actor



Extend and Include Relationships

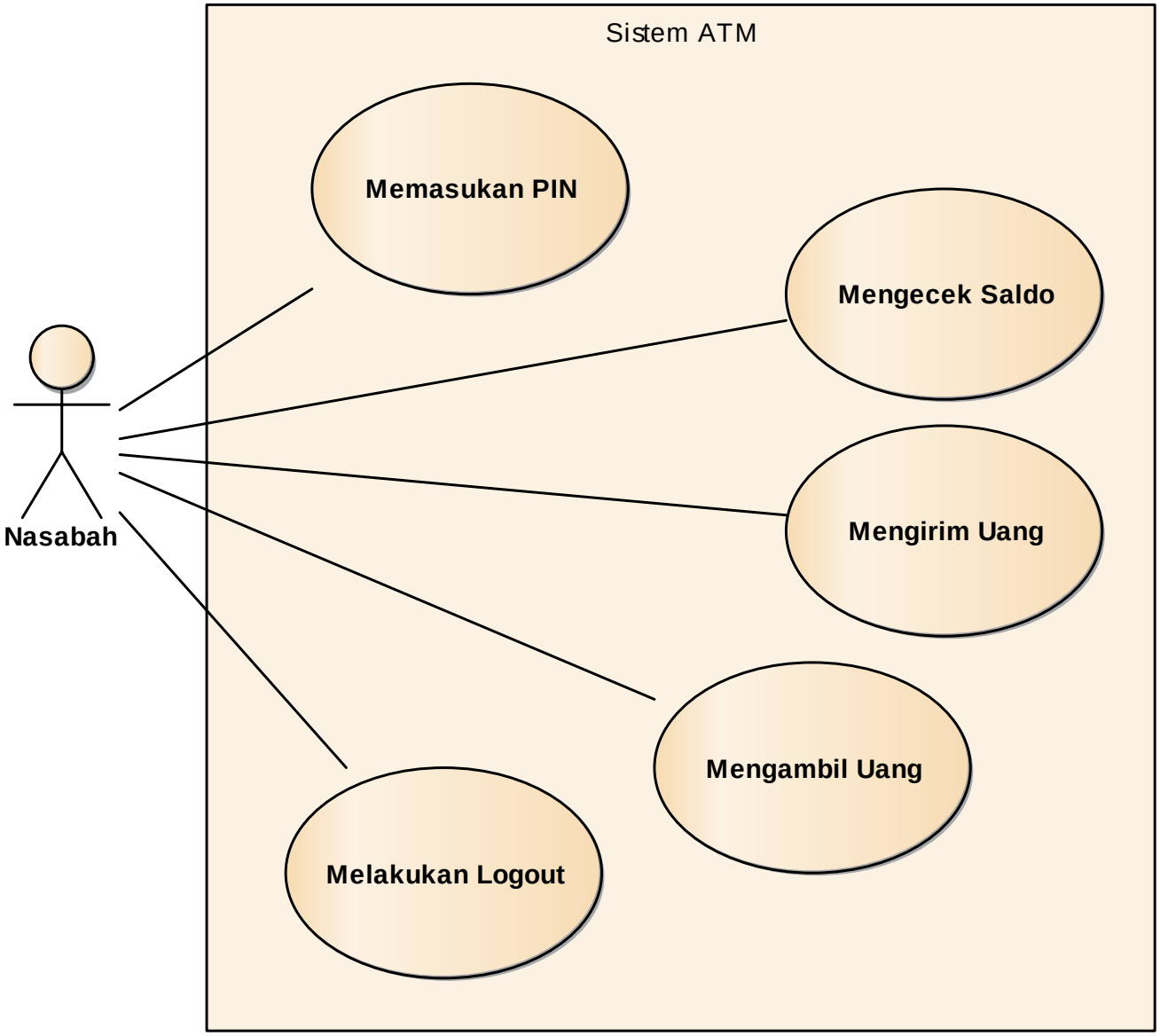


Sistem Konsultasi Dokter Rumah Sakit

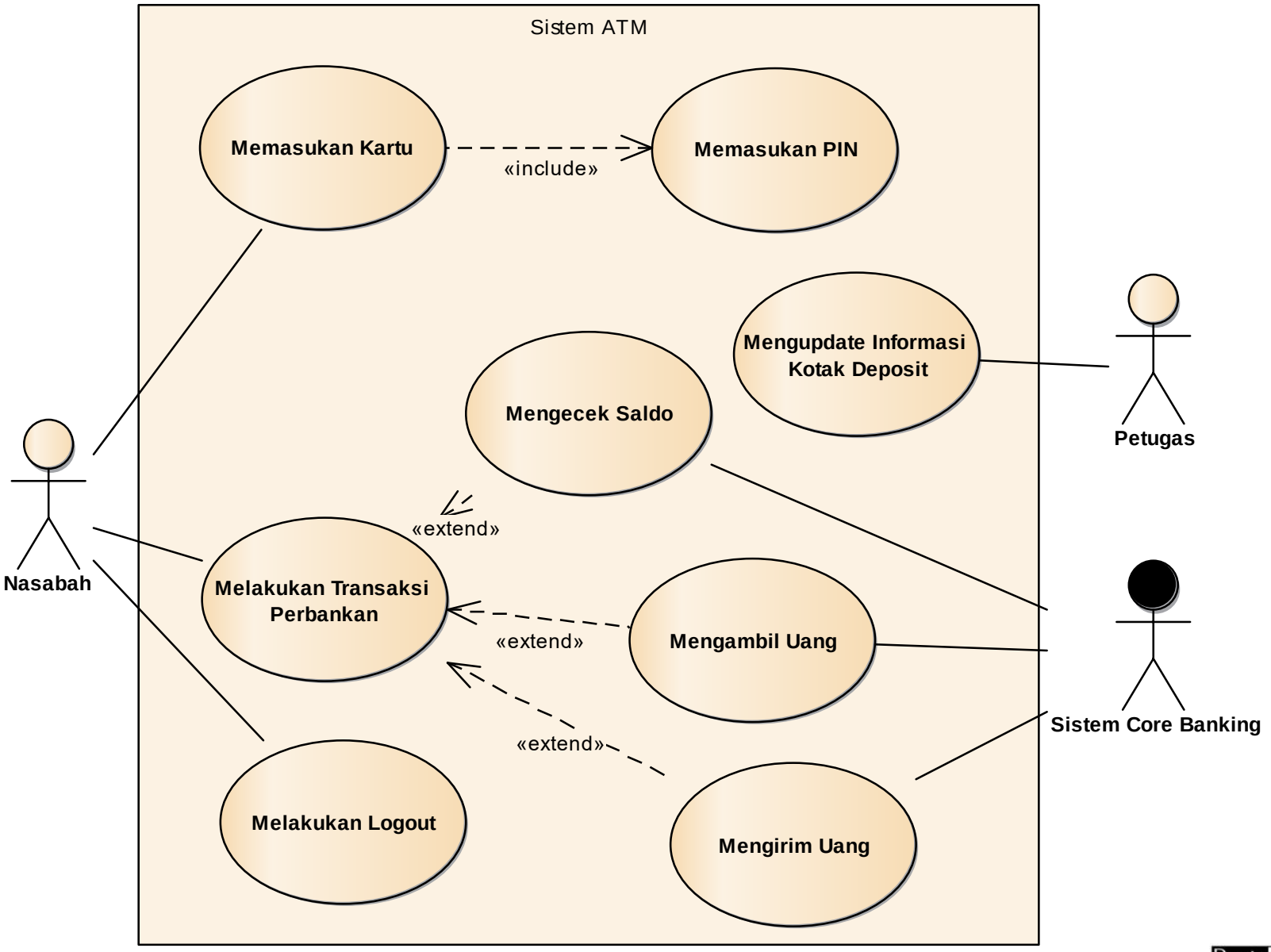


Studi Kasus: Use Case Diagram Sistem ATM

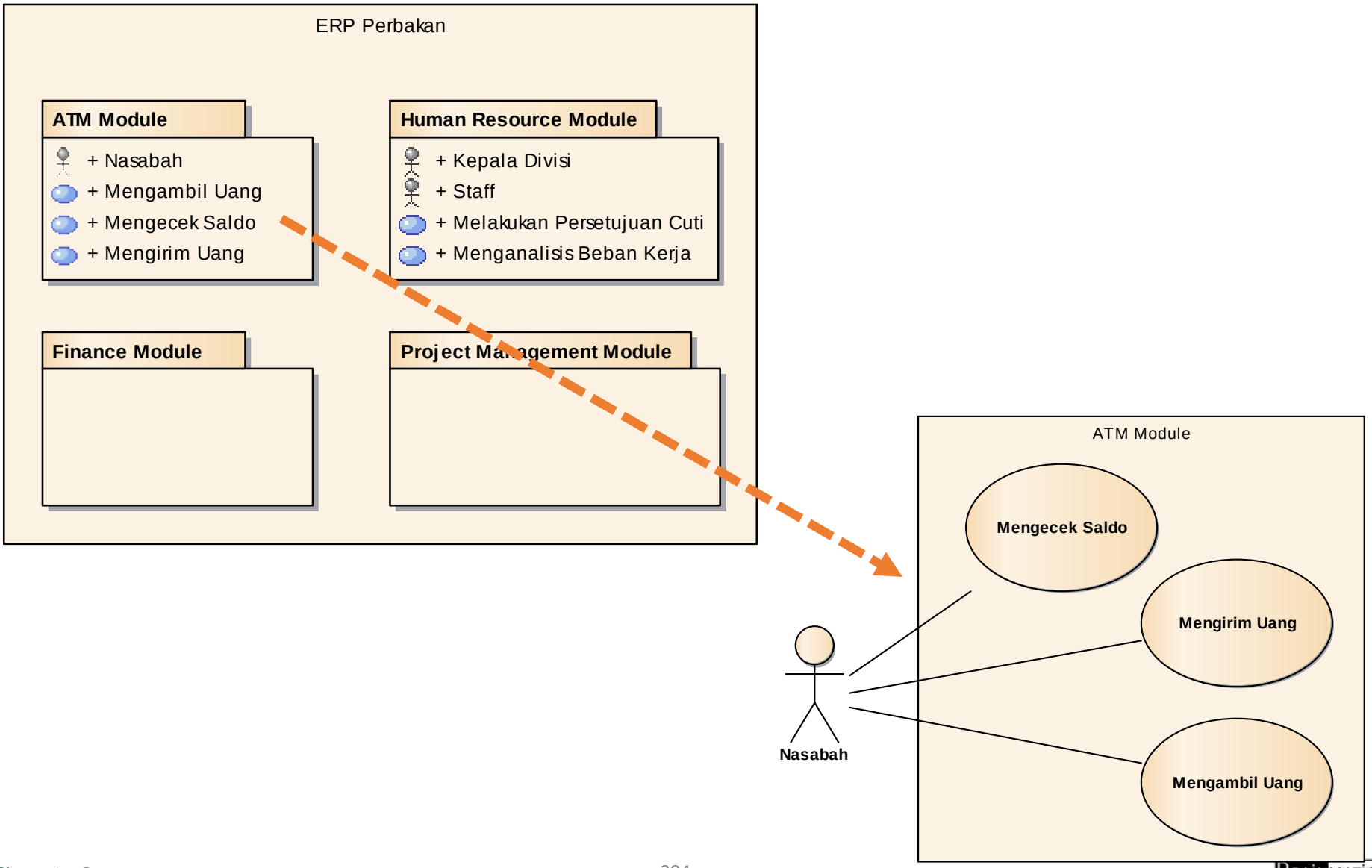
Use Case Diagram Sistem ATM
(versi Sederhana)



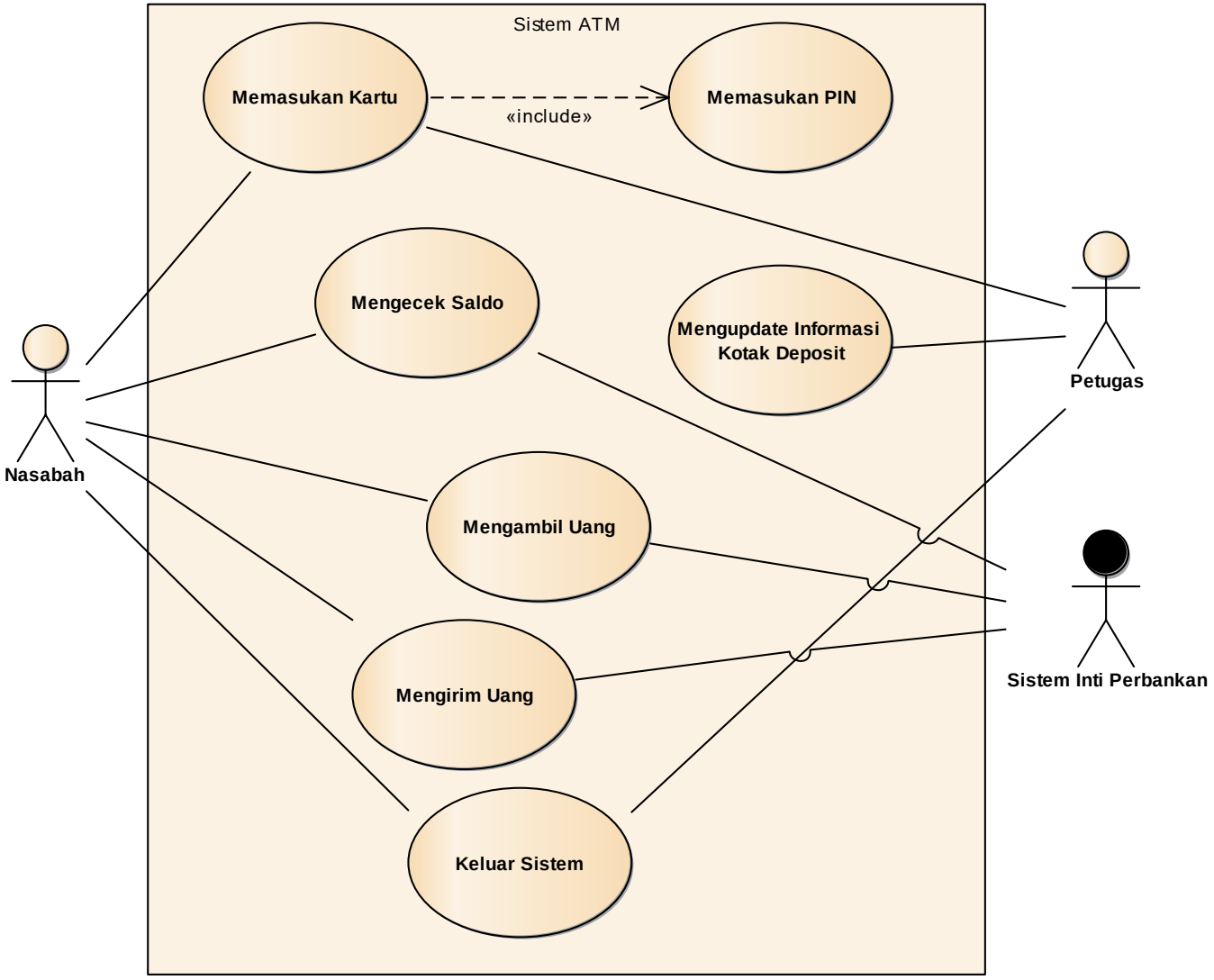
Use Case Diagram Sistem ATM
(Versi Include dan Extends)

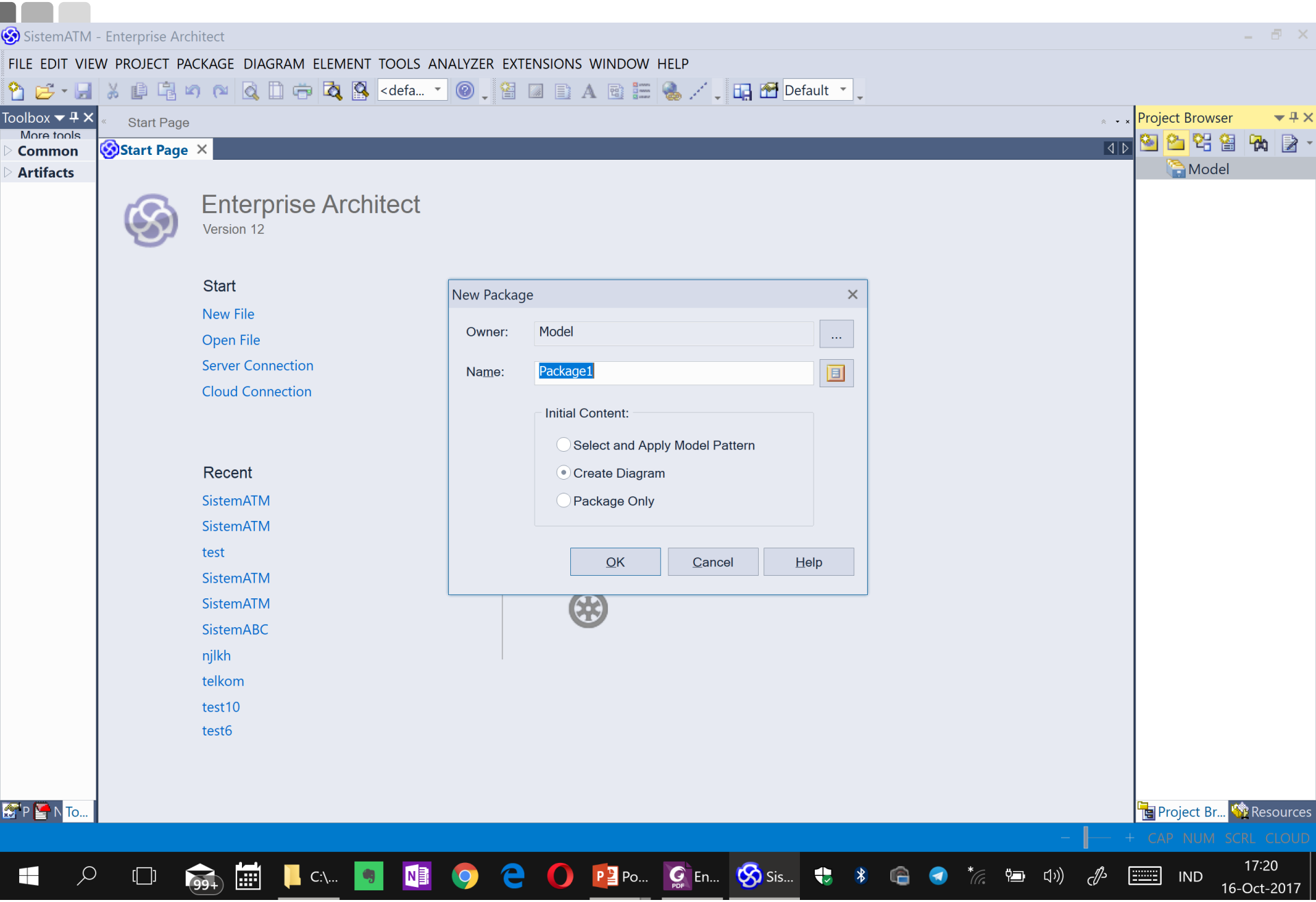


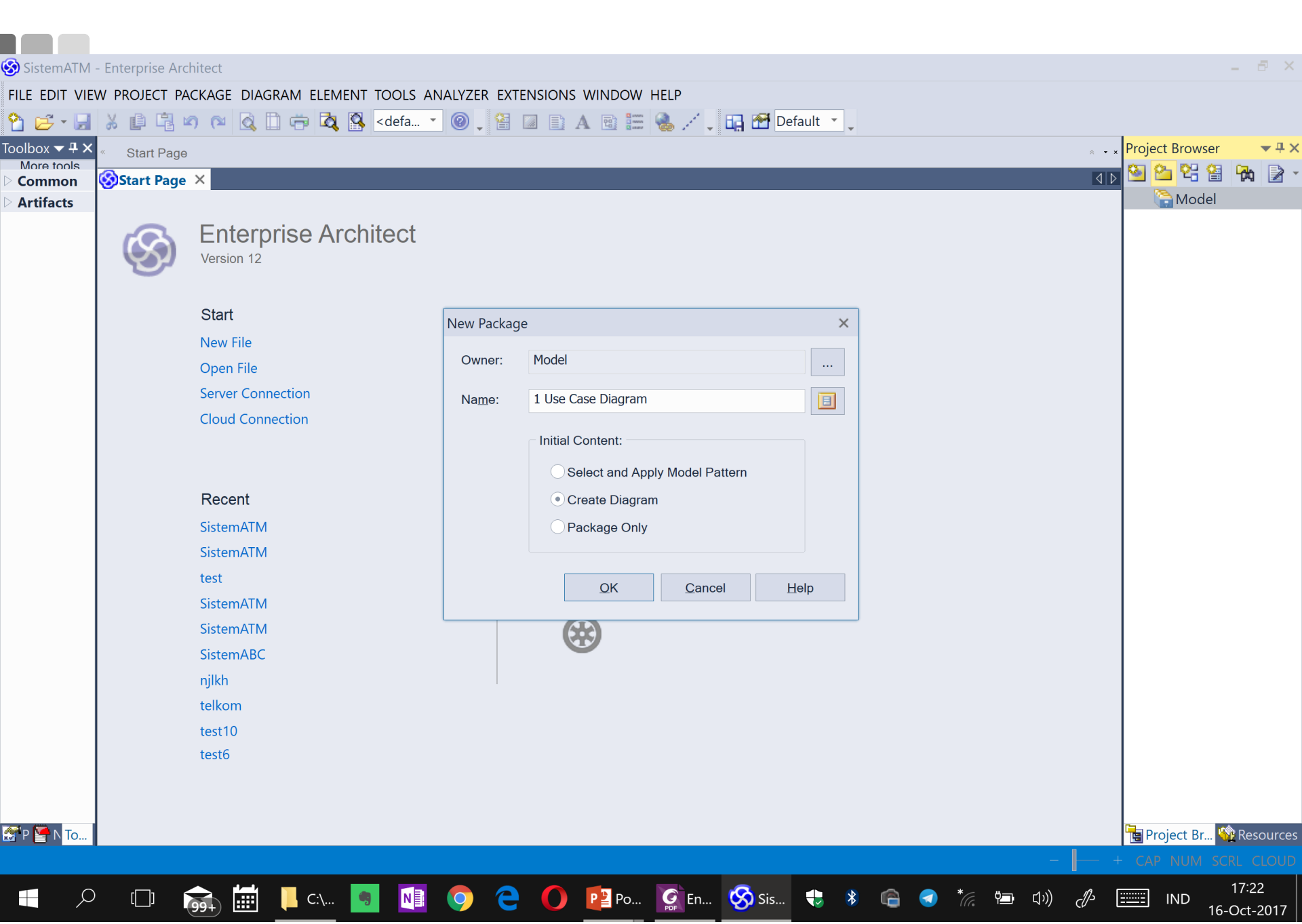
Use Case Diagram ERP Perbankan
(Sistem Lebih Kompleks)

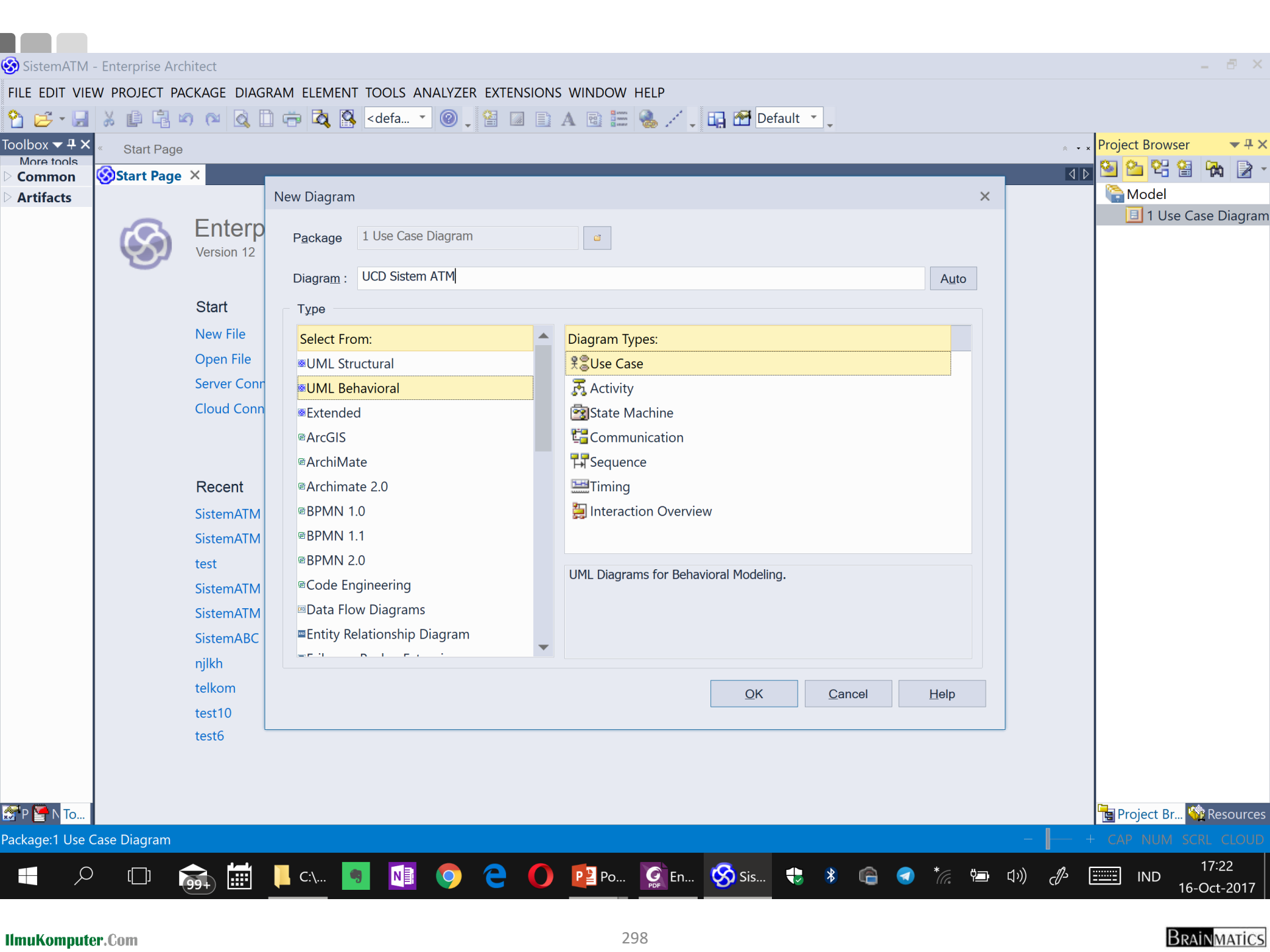


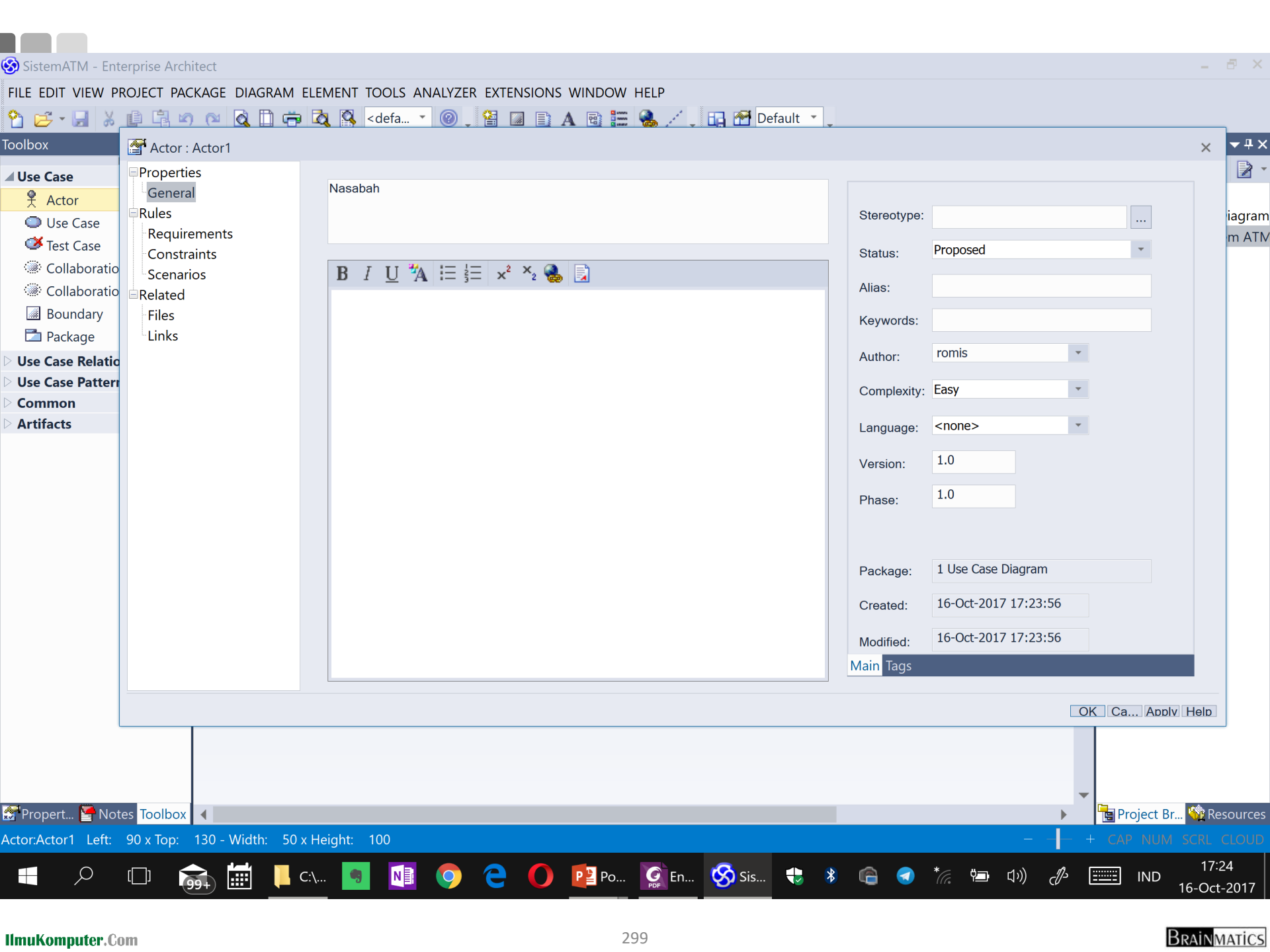
Use Case Diagram Sistem ATM
(Versi Normal)




SistemATM - Enterprise Architect
FILE EDIT VIEW PROJECT PACKAGE DIAGRAM ELEMENT TOOLS ANALYZER EXTENSIONS WINDOW HELP
Toolbox
More tools
Common
Artifacts
Start Page
Enterprise Architect
Version 12
Start
New File
Open File
Server Connection
Cloud Connection
Recent
SistemATM
SistemATM
test
SistemATM
SistemATM
SistemABC
njlkh
telkom
test10
test6
New Package
Owner:
Model
Name:
Package1
Initial Content:
Select and Apply Model Pattern
Create Diagram
Package Only
OK
Cancel
Help
Project Browser
Model
Project Br...
Resources
CAP NUM SCRL CLOUD
IND
17:20
16-Oct-2017


SistemATM - Enterprise Architect
FILE EDIT VIEW PROJECT PACKAGE DIAGRAM ELEMENT TOOLS ANALYZER EXTENSIONS WINDOW HELP
Toolbox
More tools
Common
Artifacts
Start Page
Enterprise Architect
Version 12
Start
New File
Open File
Server Connection
Cloud Connection
Recent
SistemATM
SistemATM
test
SistemATM
SistemATM
SistemABC
njlkh
telkom
test10
test6
New Package
Owner:
Model
Name:
1 Use Case Diagram
Initial Content:
Select and Apply Model Pattern
Create Diagram
Package Only
OK
Cancel
Help
Project Browser
Model
Project Br...
Resources
CAP NUM SCRL CLOUD
IND
17:22
16-Oct-2017


SistemATM - Enterprise Architect
FILE EDIT VIEW PROJECT PACKAGE DIAGRAM ELEMENT TOOLS ANALYZER EXTENSIONS WINDOW HELP
Toolbox
More tools
Common
Artifacts
Start Page
Start Page
Enterp
Version 12
Start
New File
Open File
Server Conn
Cloud Conn
Recent
SistemATM
SistemATM
test
SistemATM
SistemATM
SistemABC
njlkh
telkom
test10
test6
New Diagram
Package
1 Use Case Diagram
Diagram :
UCD Sistem ATM
Auto
Type
Select From:
UML Structural
UML Behavioral
Extended
ArcGIS
ArchiMate
Archimate 2.0
BPMN 1.0
BPMN 1.1
BPMN 2.0
Code Engineering
Data Flow Diagrams
Entity Relationship Diagram
Diagram Types:
Use Case
Activity
State Machine
Communication
Sequence
Timing
Interaction Overview
UML Diagrams for Behavioral Modeling.
OK
Cancel
Help
Project Browser
Model
1 Use Case Diagram
Project Br...
Resources
Package:1 Use Case Diagram
CAP NUM SCRL CLOUD
IND
17:22
16-Oct-2017


SistemATM - Enterprise Architect
FILE EDIT VIEW PROJECT PACKAGE DIAGRAM ELEMENT TOOLS ANALYZER EXTENSIONS WINDOW HELP
Toolbox
Use Case
Actor
Use Case
Test Case
Collaboratio
Collaboratio
Boundary
Package
Use Case Relatio
Use Case Patter
Common
Artifacts
Actor : Actor1
Properties
General
Rules
Requirements
Constraints
Scenarios
Related
Files
Links
Nasabah
Stereotype:
Status: Proposed
Alias:
Keywords:
Author: romis
Complexity: Easy
Language: <none>
Version: 1.0
Phase: 1.0
Package: 1 Use Case Diagram
Created: 16-Oct-2017 17:23:56
Modified: 16-Oct-2017 17:23:56
Main Tags
OK Ca... Apply Help
Propert... Notes Toolbox
Project Br... Resources
Actor:Actor1 Left: 90 x Top: 130 - Width: 50 x Height: 100
CAP NUM SCRL CLOUD
IND
17:24
16-Oct-2017

SistemATM - Enterprise Architect

FILE EDIT VIEW PROJECT PACKAGE DIAGRAM ELEMENT TOOLS ANALYZER EXTENSIONS WINDOW HELP

Toolbox

More tools

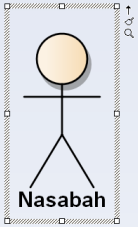
Use Case
- Actor
- Use Case
- Test Case
- Collaboration
- Collaboration Use
- Boundary
- Package

Use case and Actor elements for Usecase diagrams

Use Case Relationships
Use Case Patterns
Common
Artifacts

Use Case Diagram: "UCD Sistem ATM"

Start Page | *UCD Sistem ATM



Project Browser

Model
1 Use Case Diagram
UCD Sistem ATM
Nasabah

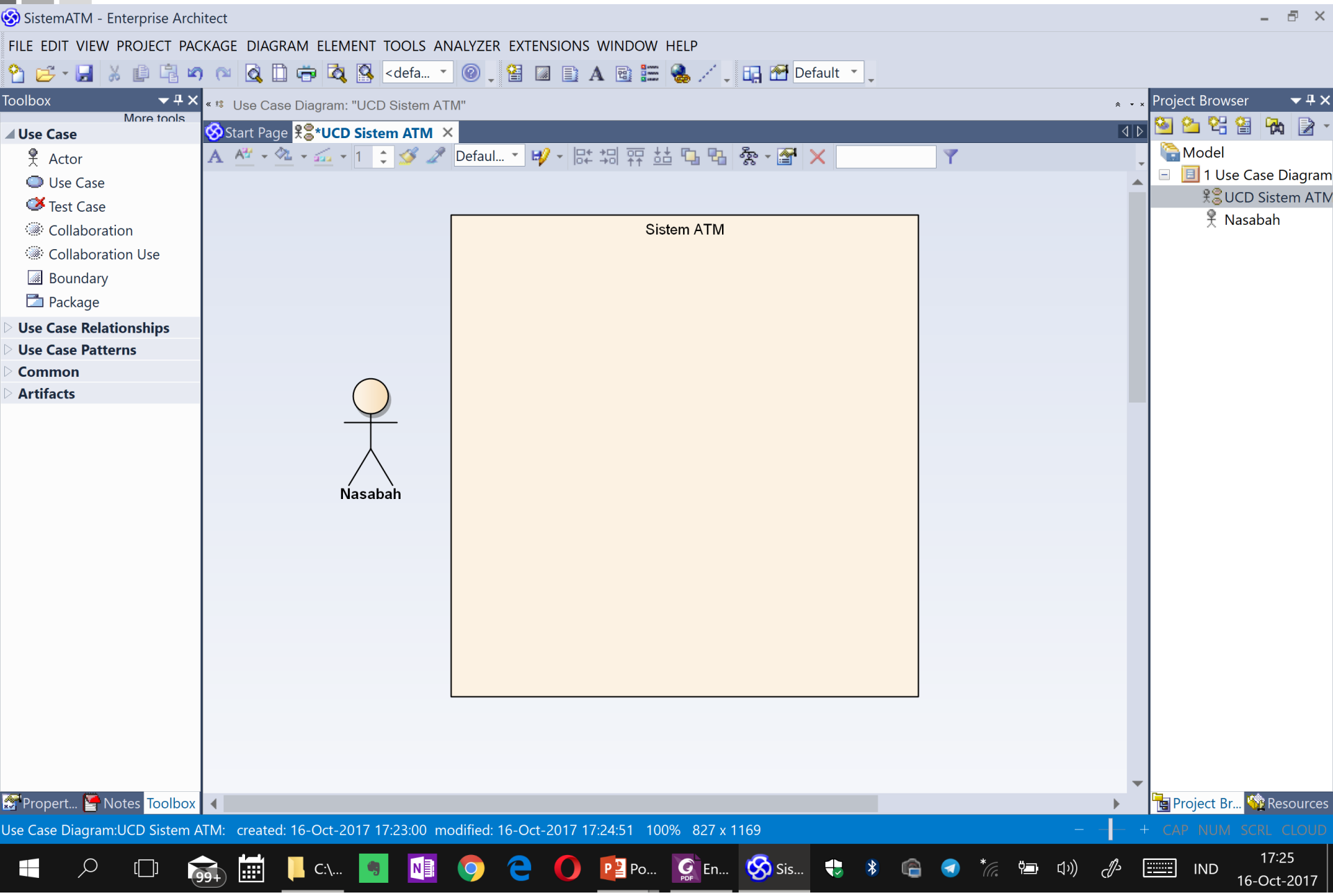
Propert... Notes Toolbox

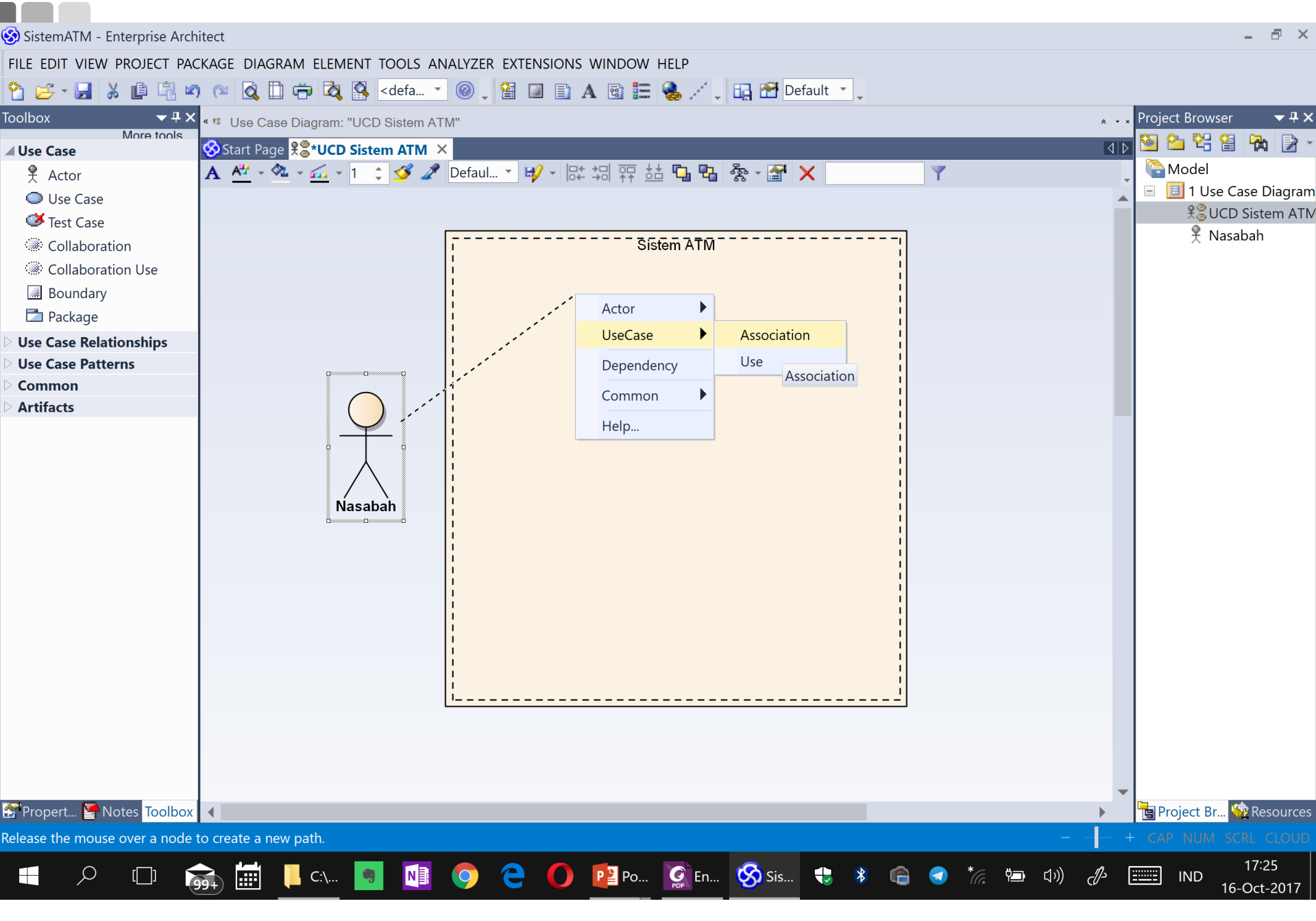
Project Br... Resources

Actor:Nasabah Left: 90 x Top: 130 - Width: 50 x Height: 100

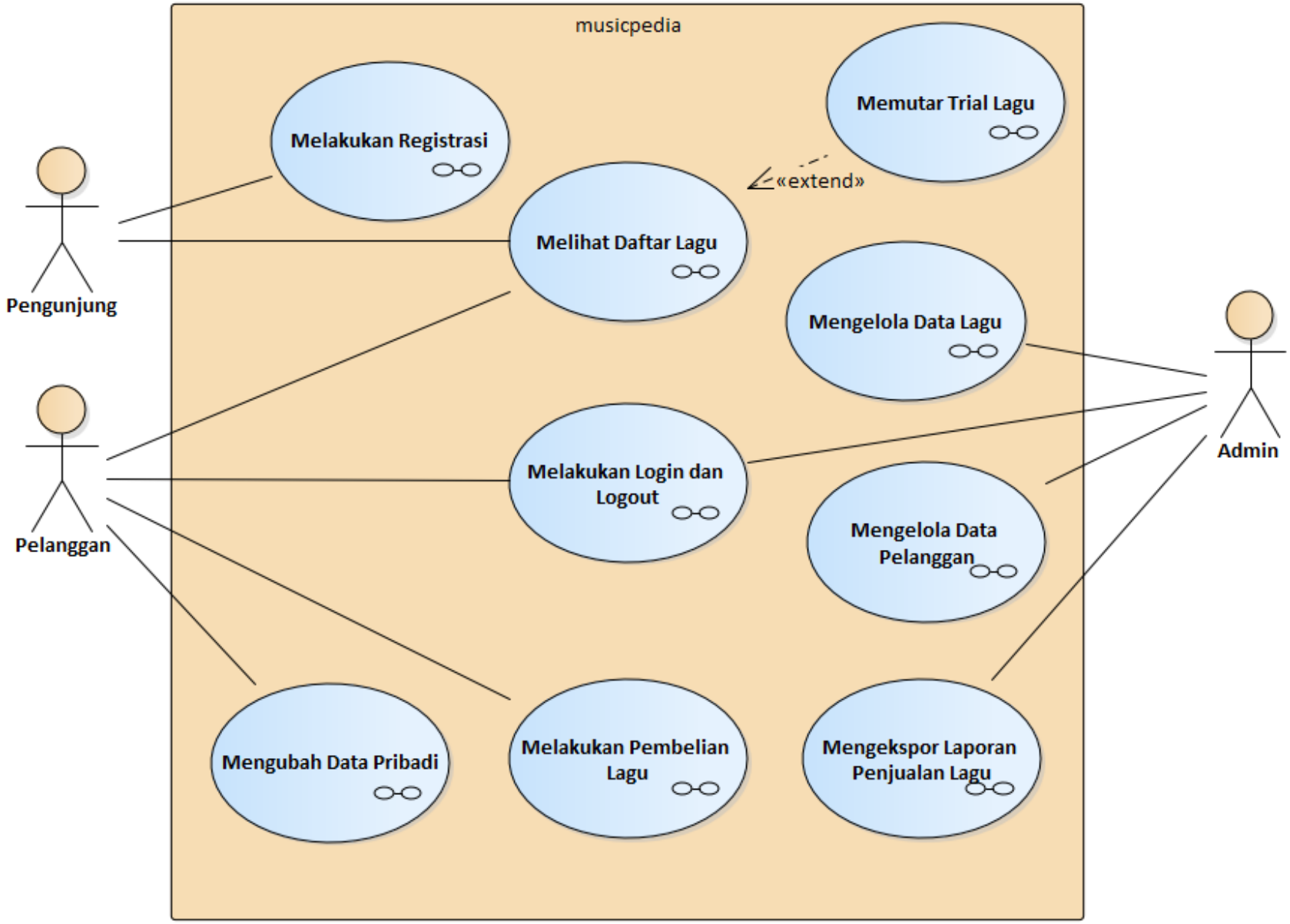
CAP NUM SCRL CLOUD

IND 17:24 16-Oct-2017


SistemATM - Enterprise Architect
FILE EDIT VIEW PROJECT PACKAGE DIAGRAM ELEMENT TOOLS ANALYZER EXTENSIONS WINDOW HELP
Toolbox
Use Case
Actor
Use Case
Test Case
Collaboration
Collaboration Use
Boundary
Package
Use Case Relationships
Use Case Patterns
Common
Artifacts
Use Case Diagram: "UCD Sistem ATM"
Start Page
*UCD Sistem ATM
Sistem ATM
Nasabah
Project Browser
Model
1 Use Case Diagram
UCD Sistem ATM
Nasabah
Use Case Diagram:UCD Sistem ATM: created: 16-Oct-2017 17:23:00 modified: 16-Oct-2017 17:24:51 100% 827 x 1169
17:25
16-Oct-2017


SistemATM - Enterprise Architect
FILE EDIT VIEW PROJECT PACKAGE DIAGRAM ELEMENT TOOLS ANALYZER EXTENSIONS WINDOW HELP
Toolbox
Use Case
Actor
Use Case
Test Case
Collaboration
Collaboration Use
Boundary
Package
Use Case Relationships
Use Case Patterns
Common
Artifacts
Use Case Diagram: "UCD Sistem ATM"
Start Page
*UCD Sistem ATM
Sistem ATM
Actor
UseCase
Dependency
Common
Help...
Association
Use
Association
Nasabah
Project Browser
Model
1 Use Case Diagram
UCD Sistem ATM
Nasabah
Release the mouse over a node to create a new path.
IND
17:25
16-Oct-2017

Use Case Diagram MusicPedia



Exercise: Business Process Identification

1. Buat project baru di Sparx EA, dengan nama file adalah nama project sesuai *System Request*
2. Lihat kembali dan rapikan *System Request* yang sudah anda buat
3. Lakukan business process identification dengan membuatkan Use Case Diagram untuk *System Request* tersebut

4. Kirim file EAPX ke romi@brainmatics.com untuk dilakukan review bersama

Exercise: Systems Analysis and Design

- Lakukan sistem analysis and design yang menghasilkan diagram:
    1. Use Case Diagram

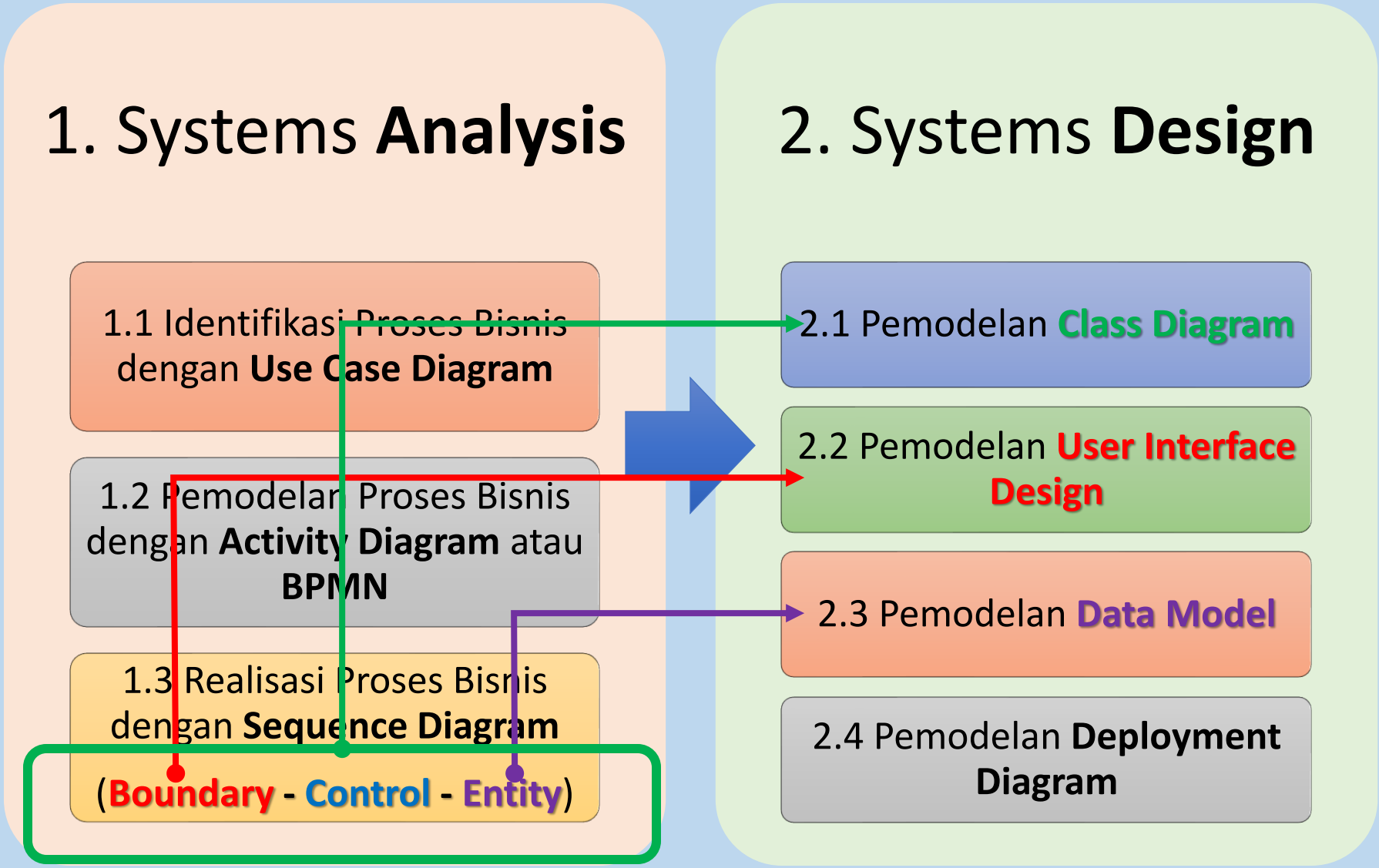
- Pilih salah satu aplikasi di bawah:
    1. Aplikasi Rental Mobil
    2. Aplikasi Pengelolaan Klinik
    3. Aplikasi Pengelolaan Apotik
    4. Aplikasi Pengelolaan Service Mobil
    5. Aplikasi Penjualan Motor
    6. Aplikasi Pengelolaan Perpustakaan
    7. Aplikasi Penjualan Buku Online
    8. Aplikasi Penjualan Tiket Kereta Online
    9. Aplikasi Manajemen Universitas Online
    10. Aplikasi Penjualan Laptop Online
    11. Aplikasi Perpustakaan Digital
    12. Aplikasi Pengelolaan Project Software

3.3 Pemodelan Proses Bisnis dengan AD atau BPMN

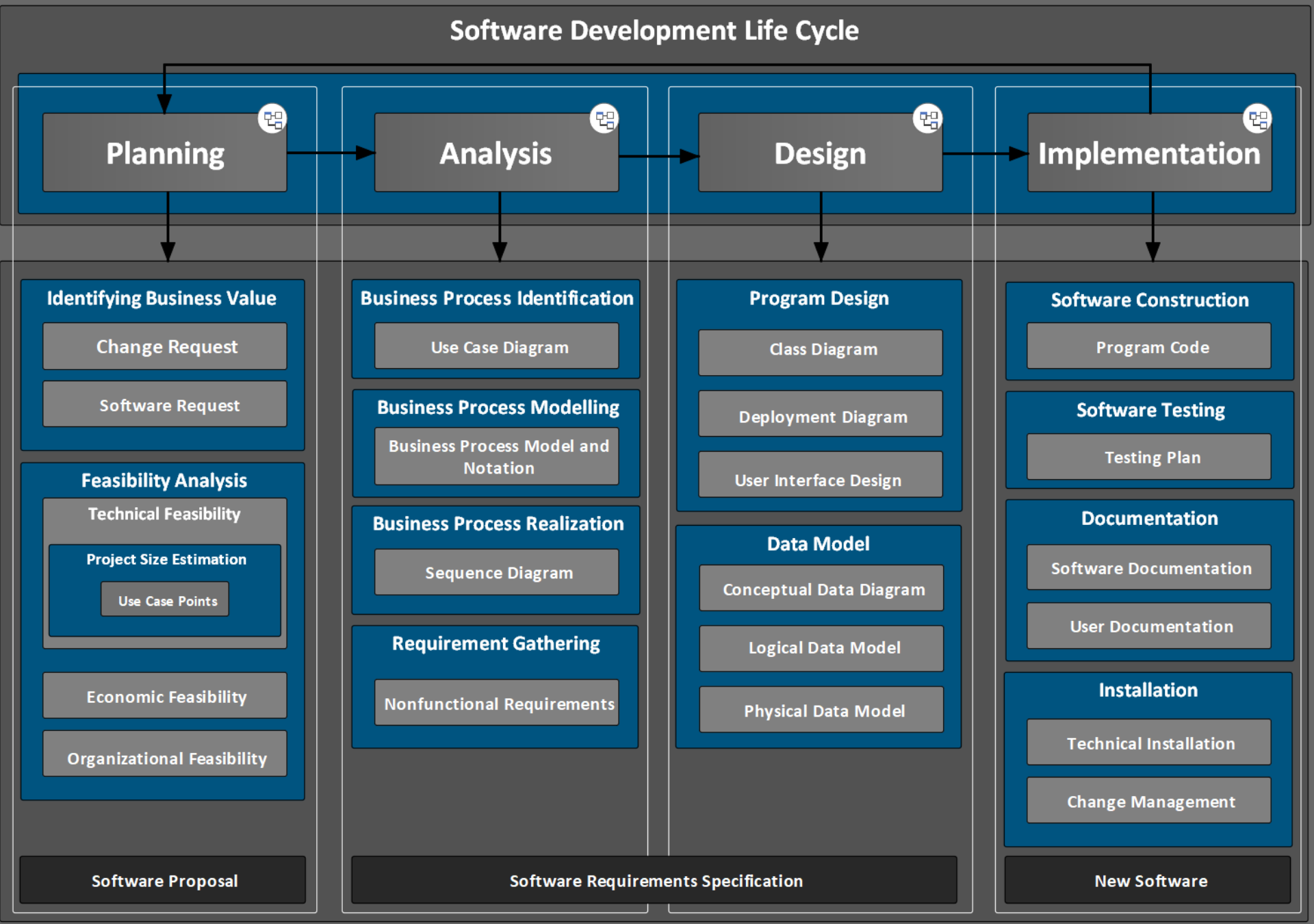
- Activity Diagram
- Business Process Model and Notation

UML based Software Analysis and Design

(Wahono, 2009)

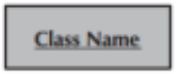
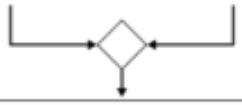


Application Development Governance



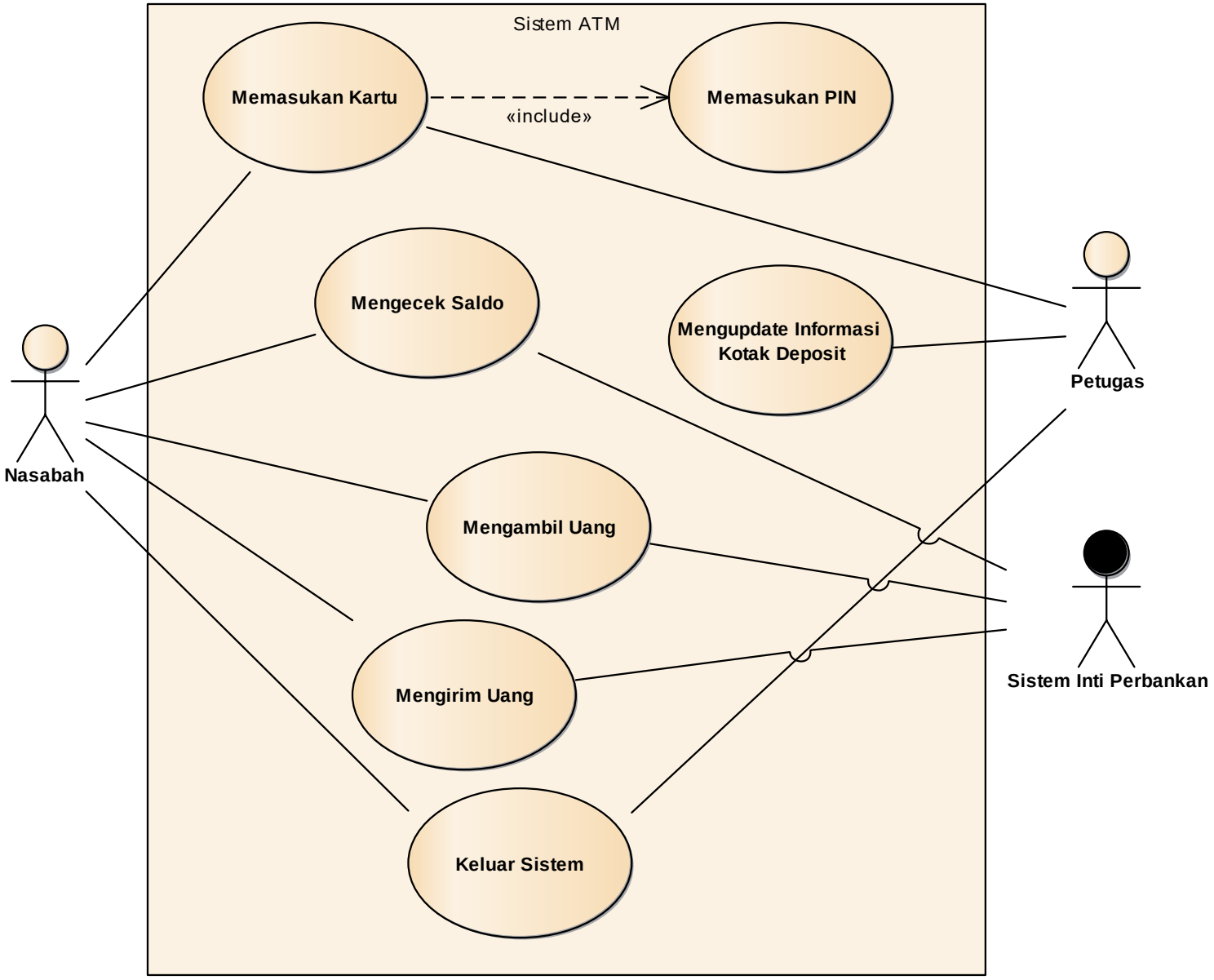
3.3.1 Activity Diagram

Activity Diagram Syntax

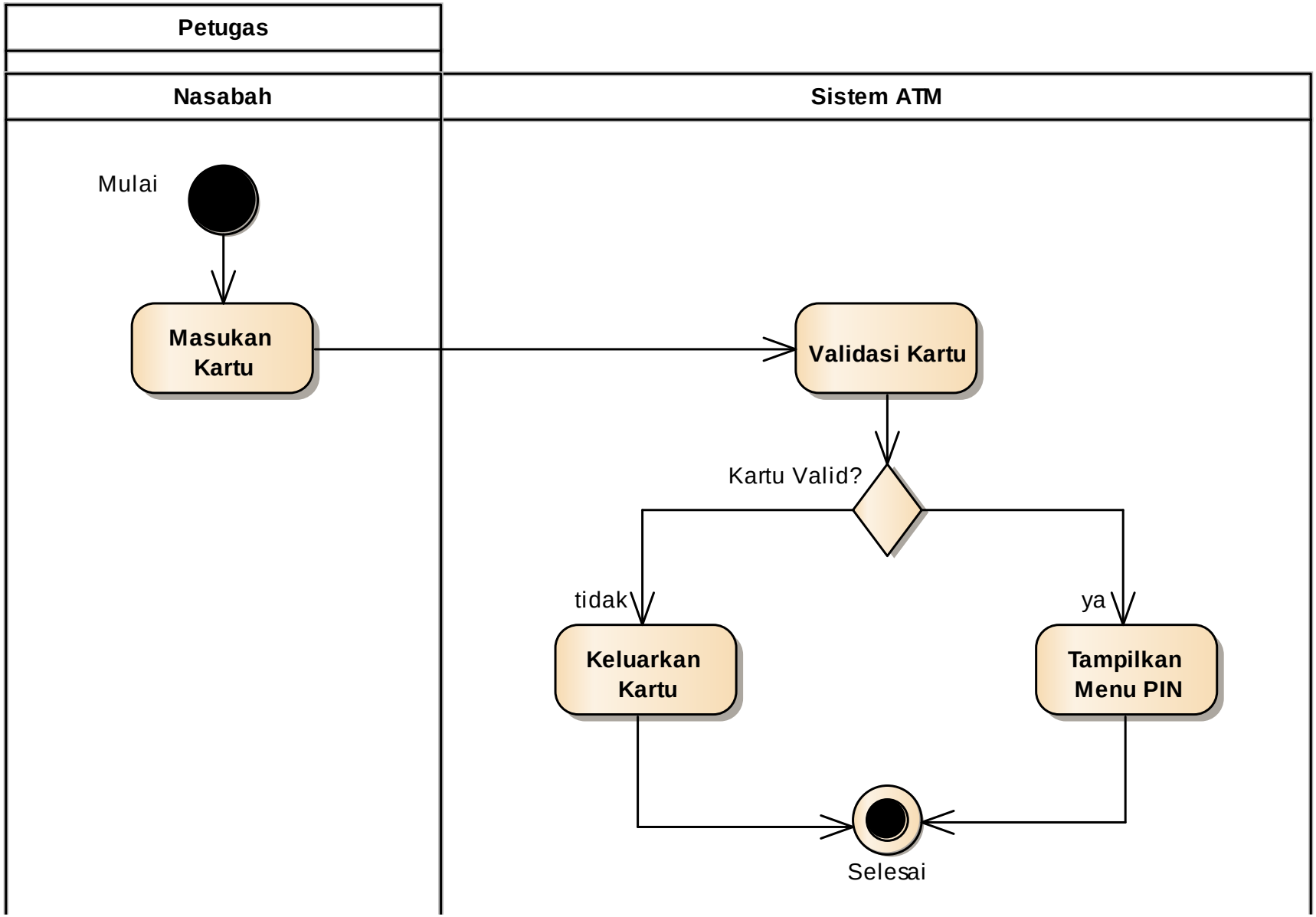
An action: ■ Is a simple, nondecomposable piece of behavior. ■ Is labeled by its name.	
An activity: ■ Is used to represent a set of actions. ■ Is labeled by its name.	
An object node: ■ Is used to represent an object that is connected to a set of object flows. ■ Is labeled by its class name.	
A control flow: ■ Shows the sequence of execution.	
An object flow: ■ Shows the flow of an object from one activity (or action) to another activity (or action).	
An initial node: ■ Portrays the beginning of a set of actions or activities.	
A final-activity node: ■ Is used to stop all control flows and object flows in an activity (or action).	
A final-flow node: ■ Is used to stop a specific control flow or object flow.	
A decision node: ■ Is used to represent a test condition to ensure that the control flow or object flow only goes down one path. ■ Is labeled with the decision criteria to continue down the specific path.	
A merge node: ■ Is used to bring back together different decision paths that were created using a decision node.	
A fork node: Is used to split behavior into a set of parallel or concurrent flows of activities (or actions)	
A join node: Is used to bring back together a set of parallel or concurrent flows of activities (or actions)	
A swimlane: Is used to break up an activity diagram into rows and columns to assign the individual activities (or actions) to the individuals or objects that are responsible for executing the activity (or action) Is labeled with the name of the individual or object responsible	

Studi Kasus: Activity Diagram Sistem ATM

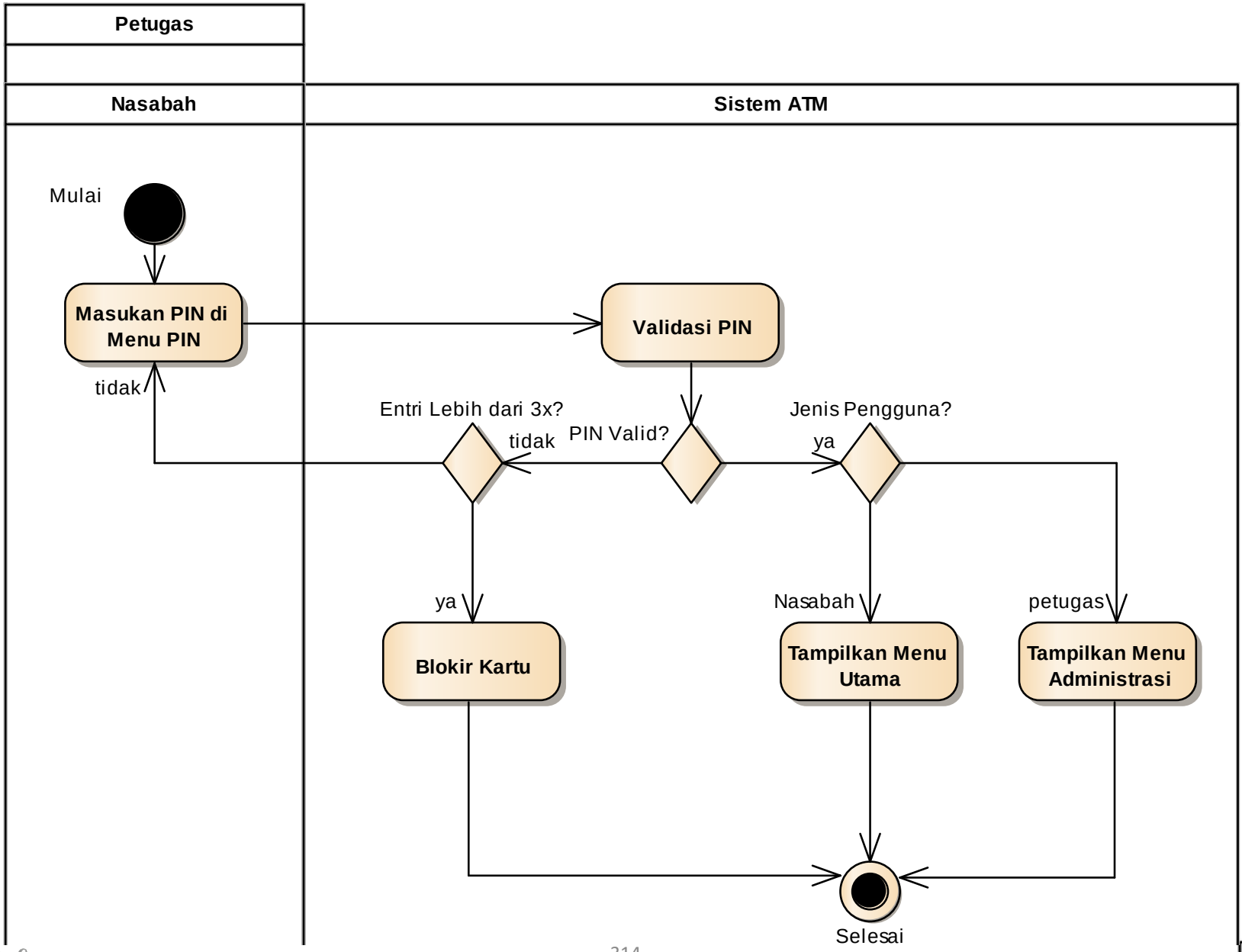
Use Case Diagram Sistem ATM



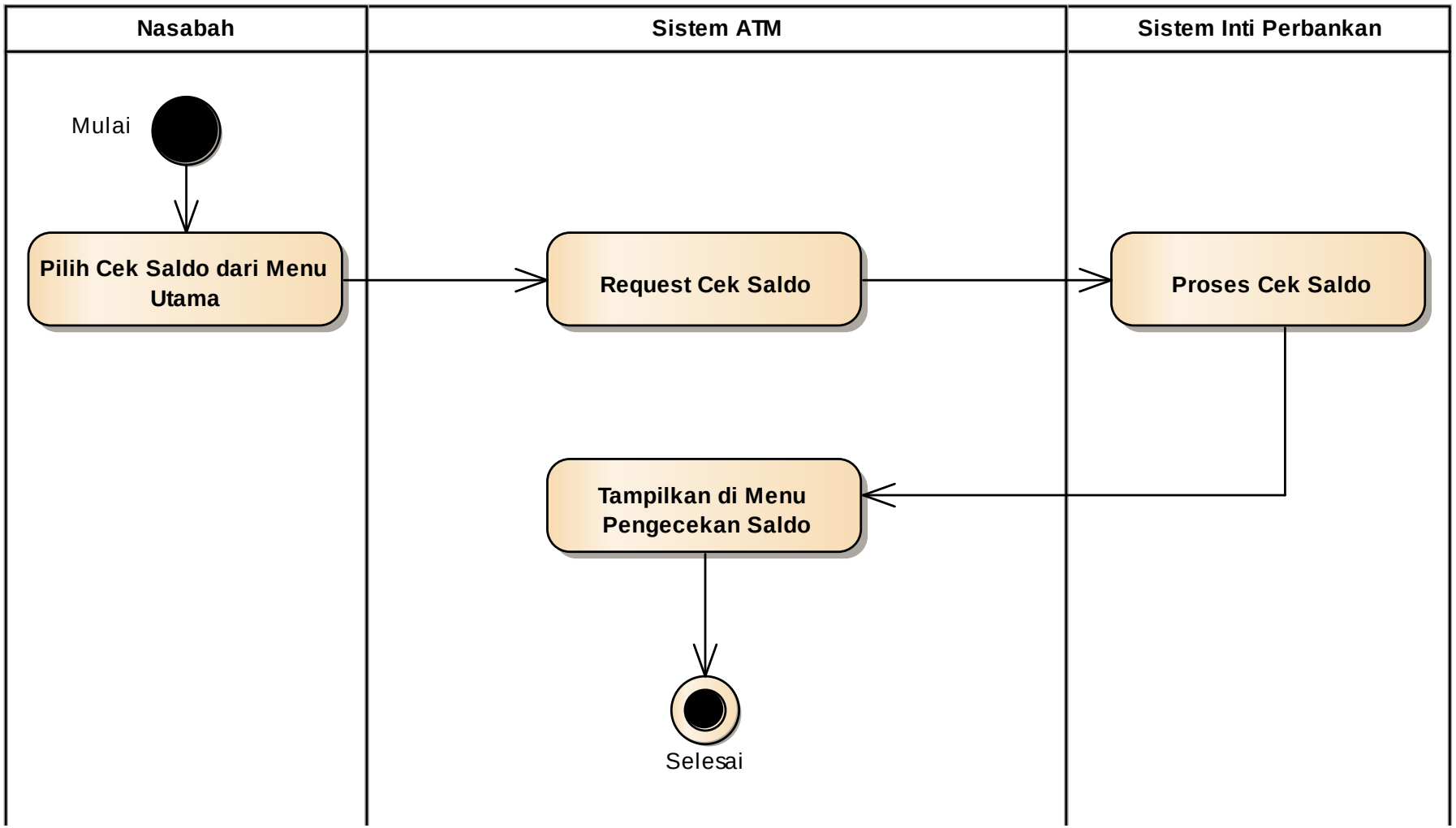
Activity Diagram: Memasukkan Kartu



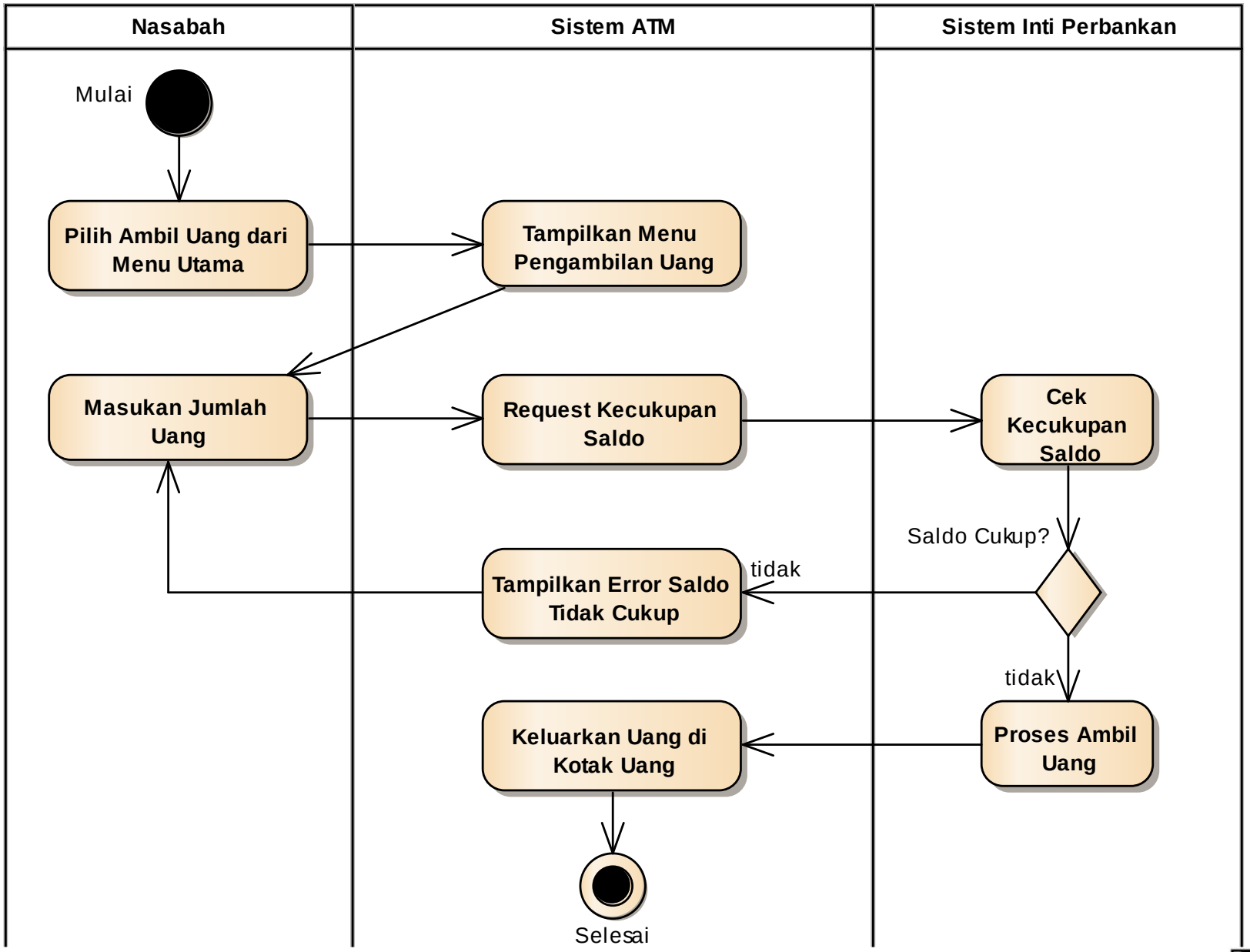
Activity Diagram: Memasukkan PIN



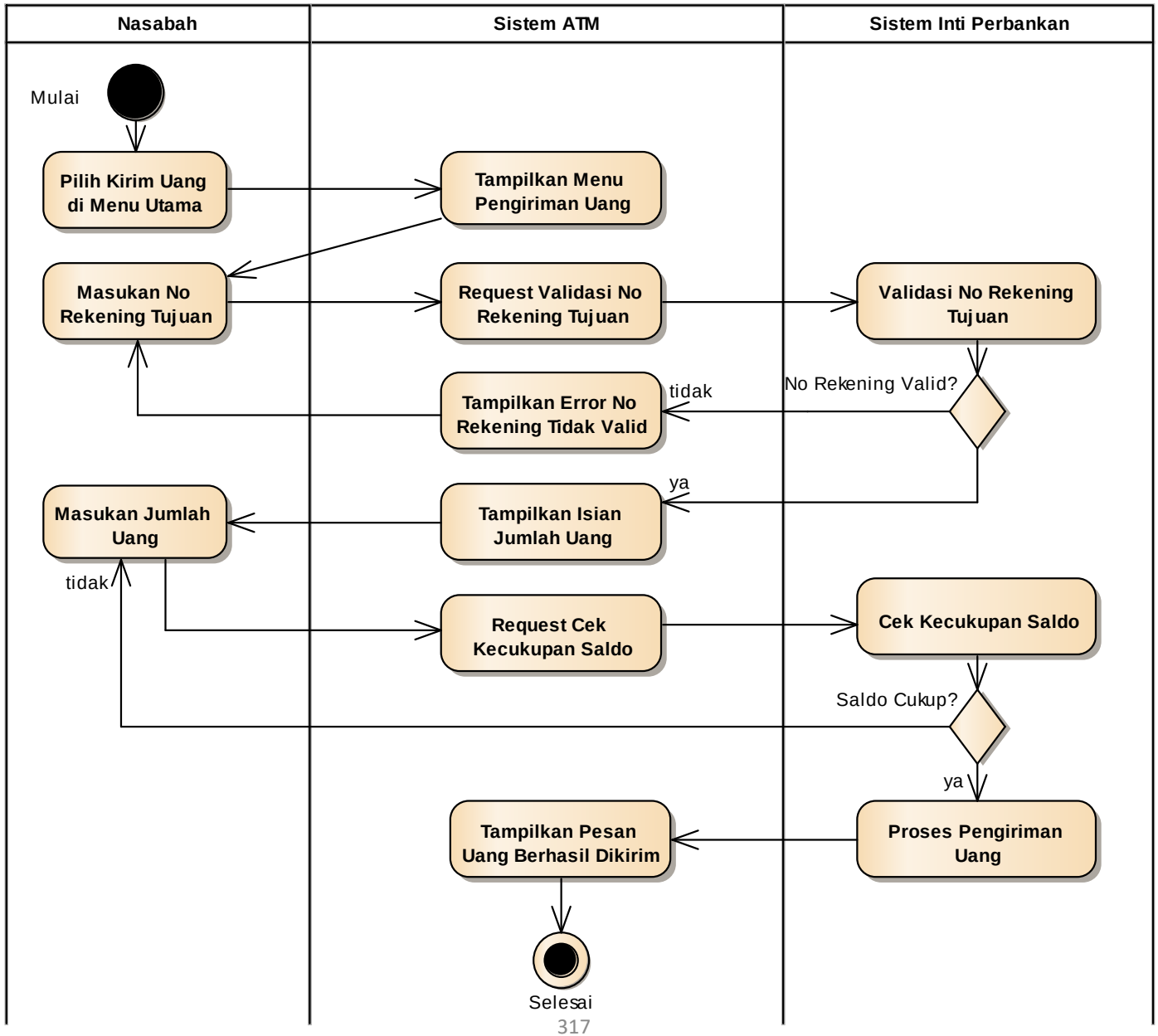
Activity Diagram: Mengecek Saldo



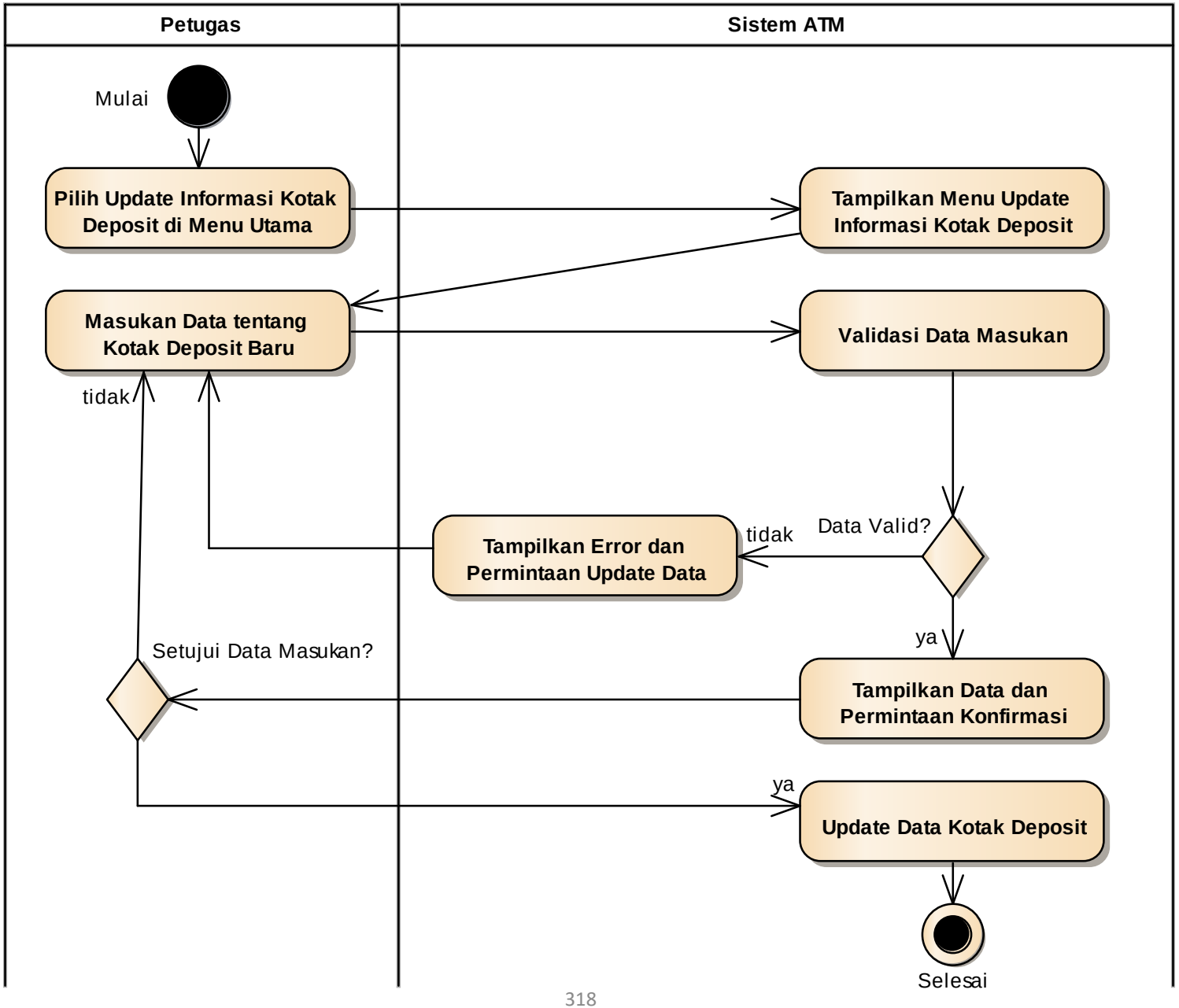
Activity Diagram: Mengambil Uang



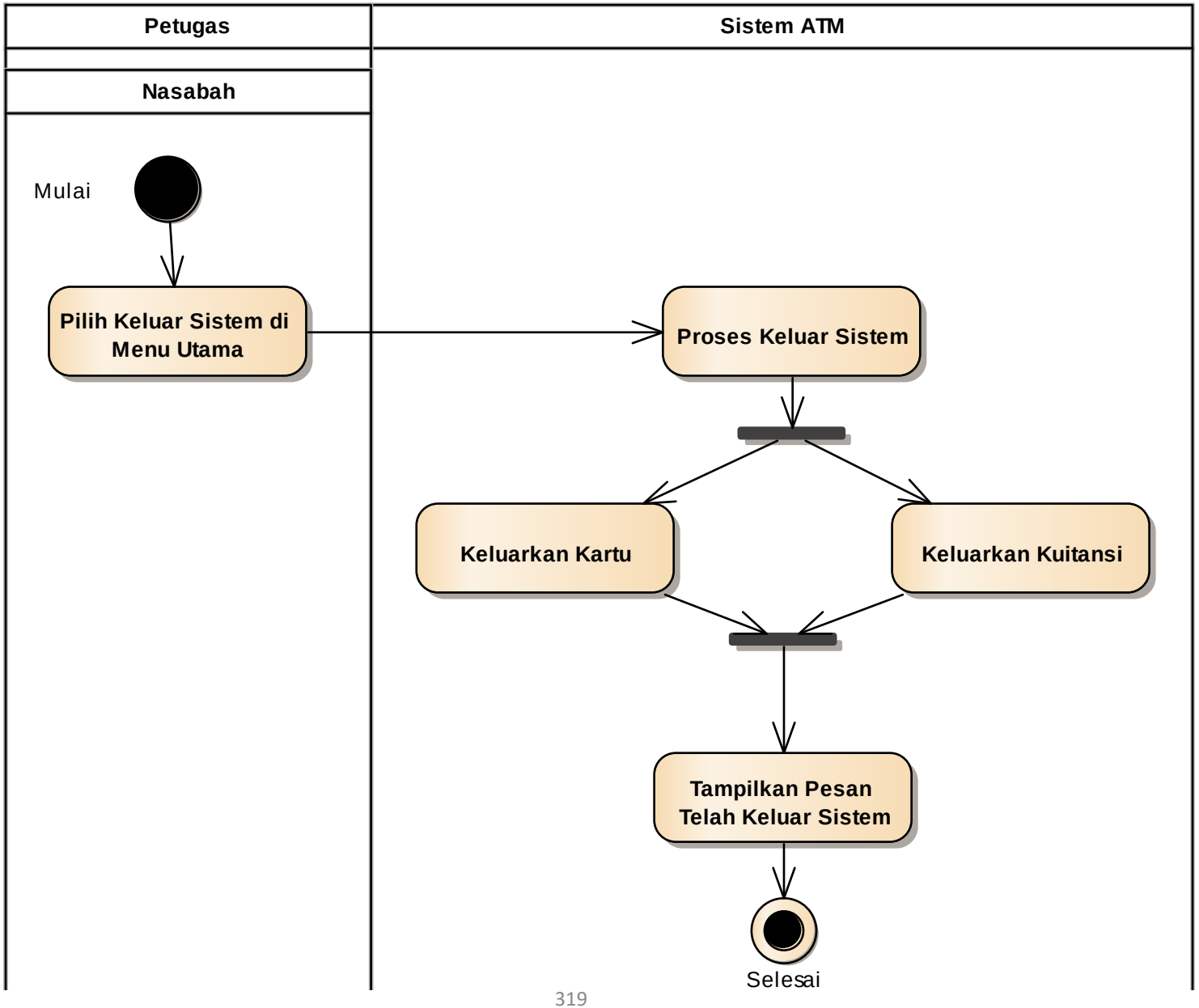
Activity Diagram: Mengirim Uang



Activity Diagram: Mengupdate Informasi Kotak Deposit



Activity Diagram: Keluar Sistem



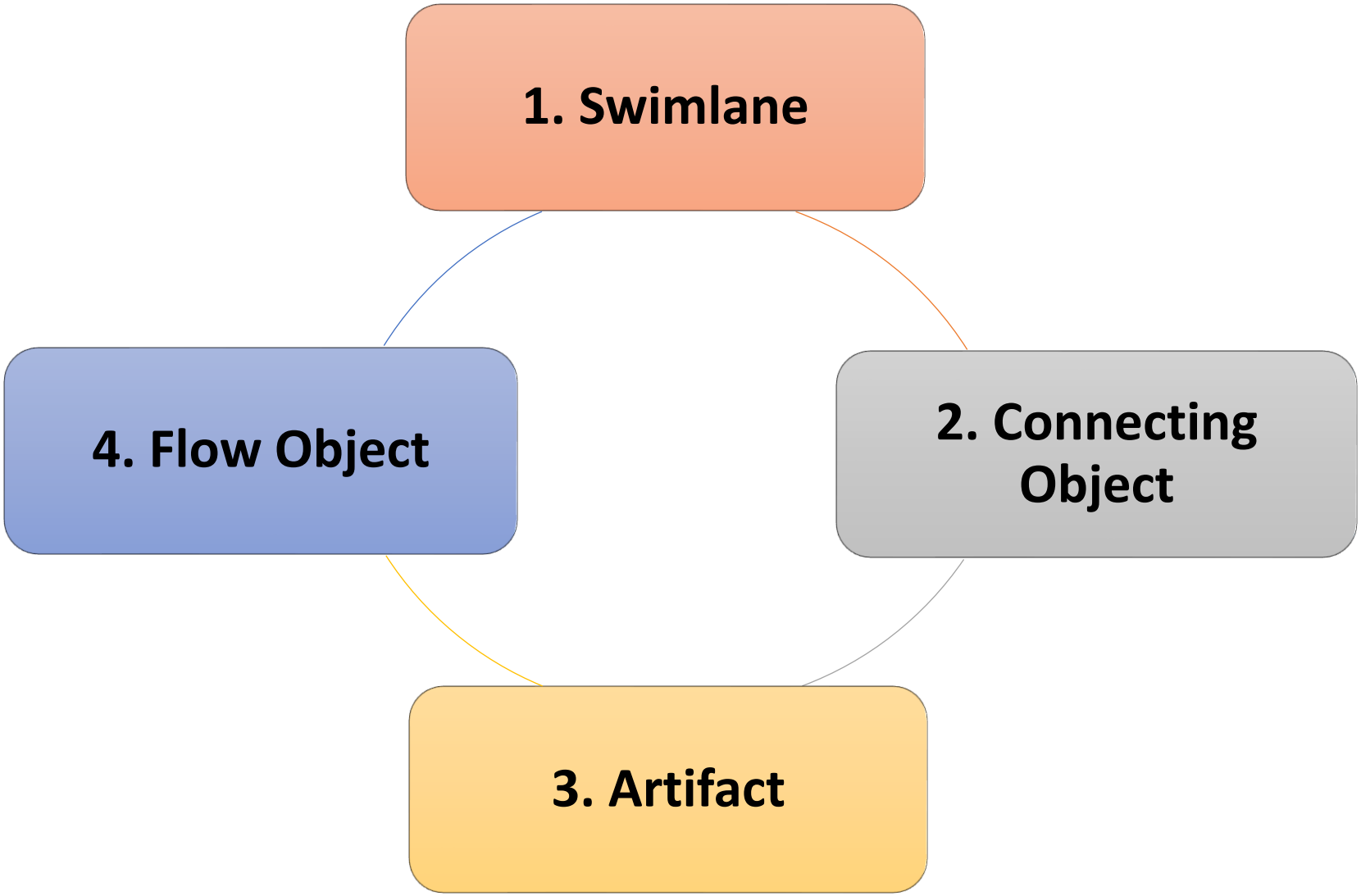
Exercise: Business Process Modeling

1. Lihat kembali System Request dan Use Case Diagram yang sudah anda buat
2. Lakukan business process modeling dengan membuatkan Activity Diagram untuk setiap Use Case yang dibuat

3. Kirim file EAPX ke romi@brainmatics.com untuk dilakukan review bersama

3.3.2 BPMN

Elemen BPMN



Elemen dan Notasi BPMN (Bizagi)

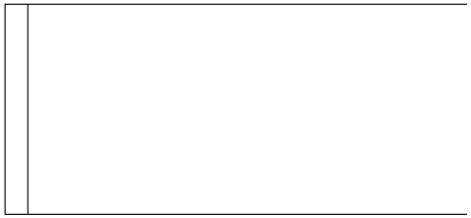
ELEMEN	DESKRIPSI	NAMA NOTASI
Swimlane	Mekanisme untuk mengatur dan memisahkan peran atau penanggungjawab dari suatu proses	**Pool**
		Lane
Connecting Object	Konektor dari obyek yang mengalir pada suatu proses	**Sequence Flow**
		Message Flow
		Association
Artifact	Informasi tambahan dalam suatu proses	**Annotation**
		Group
		Data Object
		Data Store
Flow Object	Obyek yang mengalir pada suatu proses	**Event**
		Activity
		Gateway

Elemen dan Notasi BPMN

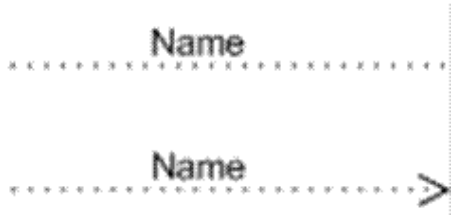
(EA Template using Sparx EA (Wahono, 2017))

Elemen	Deskripsi	Nama Notasi
Swimlane	Mekanisme untuk mengatur dan memisahkan peran atau penanggungjawab dari suatu proses	**Pool**
		Lane
Connecting Object	Konektor dari obyek yang mengalir pada suatu proses	**Sequence Flow**
		Dependency
Artifact	Informasi tambahan dalam suatu proses	**Class (Data)**
		Package (Aplikasi)
		Requirement (KPI)
		Risk (Risiko)
Flow Object	Obyek yang mengalir pada suatu proses	**Event**
		Activity
		Gateway

Swimlane

Nama Notasi	Deskripsi	Notasi
Pool	Kontainer dari satu proses	
Lane	Partisi dari suatu proses, yang menunjukkan sub organisasi, jabatan, peran atau penanggungjawab	

Connecting Object (Bizagi)

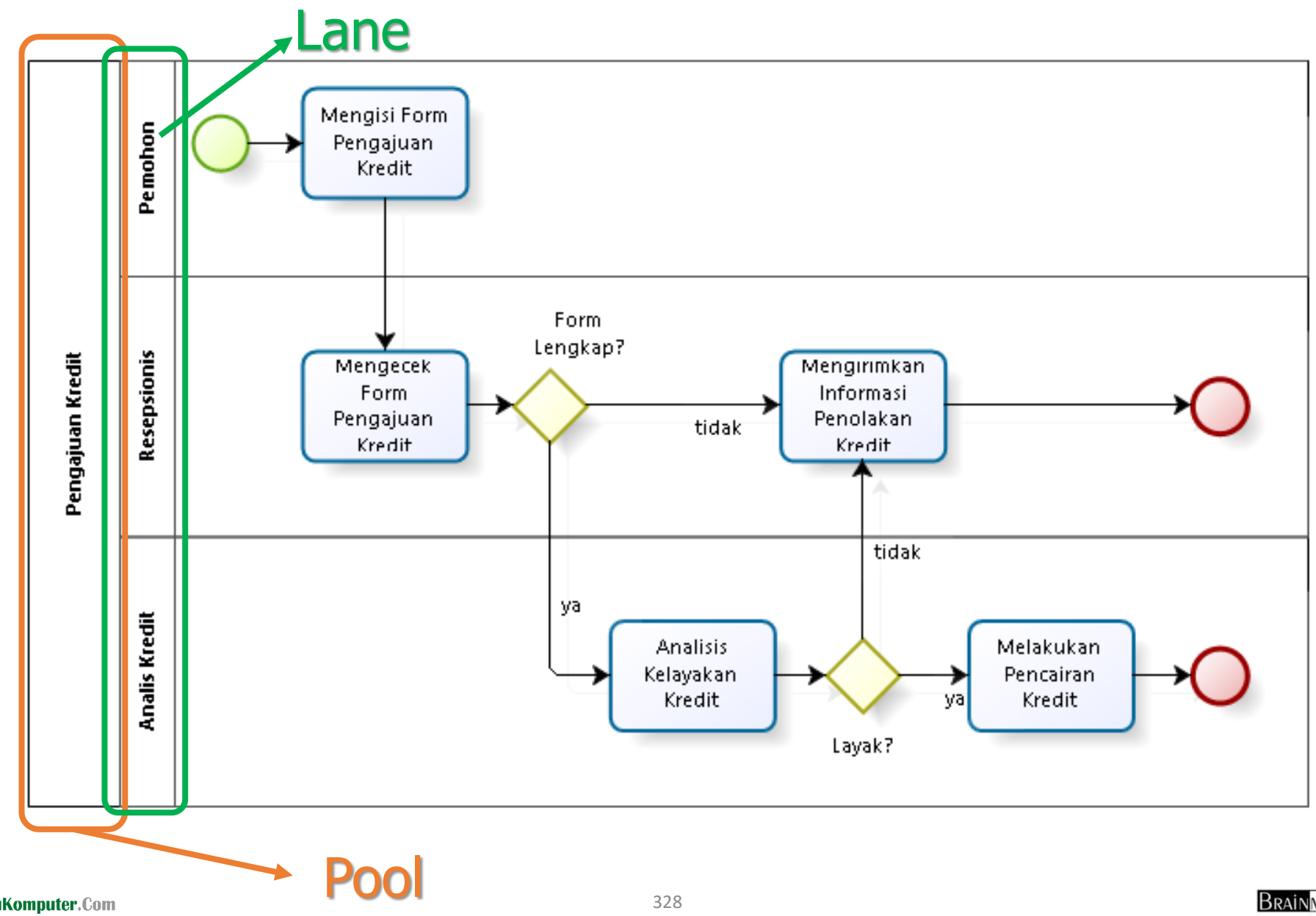
NAMA NOTASI	DESKRIPSI	NOTASI
Sequence Flow	Konektor yang menghubungkan antar obyek yang mengalir dalam satu proses (satu pool)	
Message Flow	Konektor yang menghubungkan antar obyek yang mengalir antar proses (beda pool)	
Association	Konektor yang menghubungkan obyek yang mengalir ke artifact	

Connecting Object

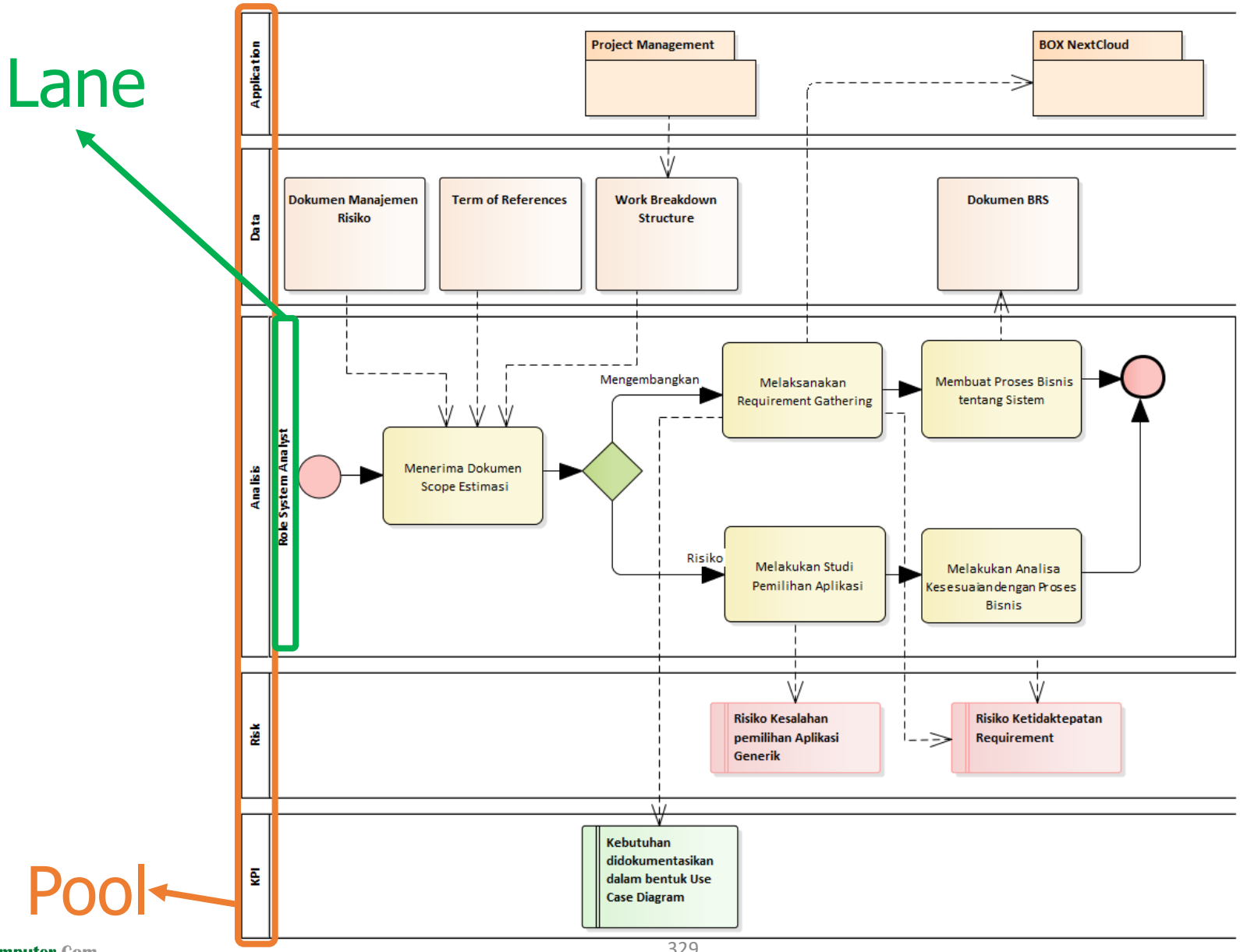
(EA Template using Sparx EA (Wahono, 2017))

Nama Notasi	Deskripsi	Notasi
Sequence Flow	Konektor yang menghubungkan antar obyek yang mengalir dalam satu proses (satu pool)	
Dependency	Konektor yang menghubungkan obyek yang mengalir ke artifact	

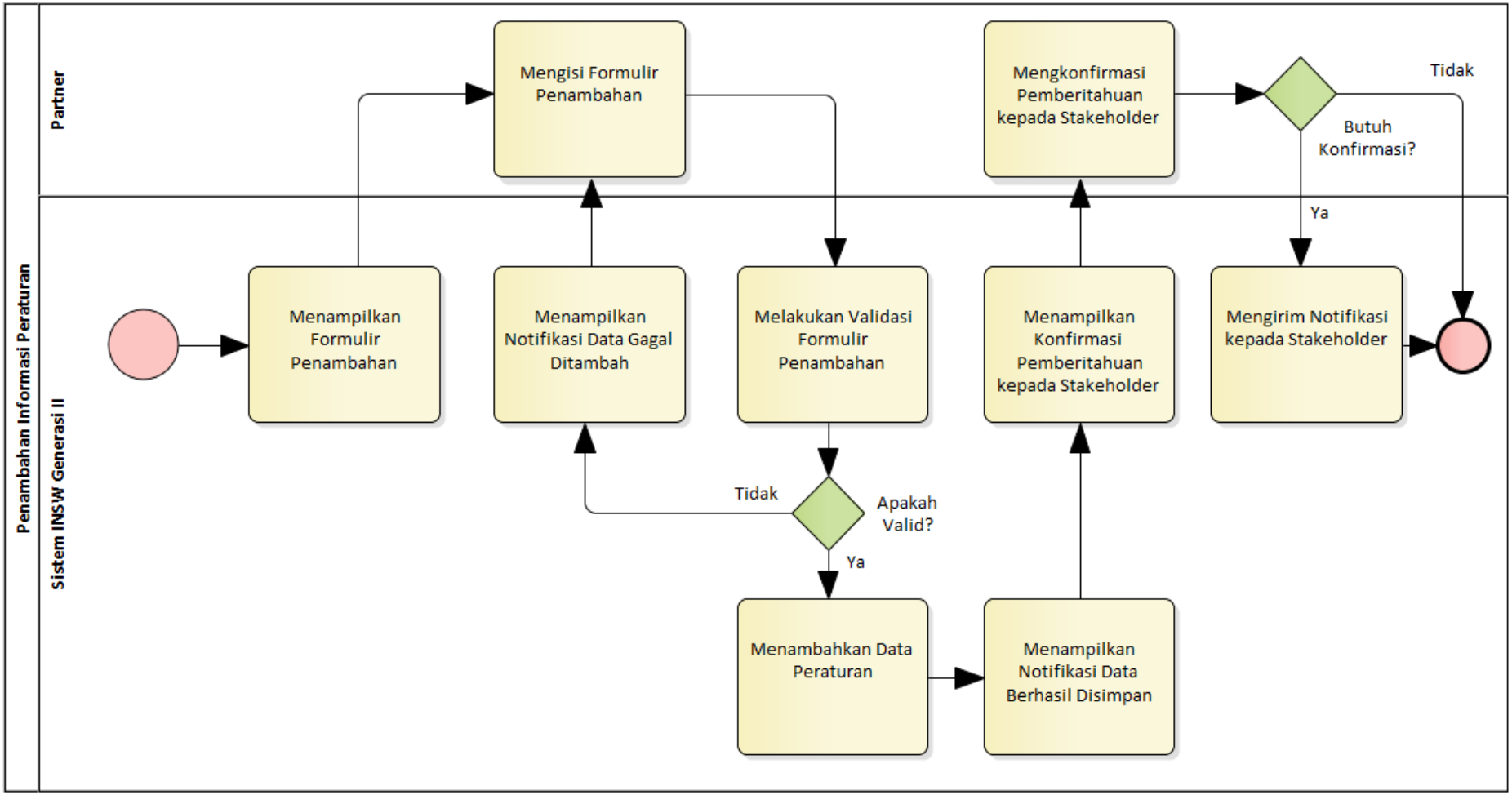
Swimlane - Proses Bisnis Organisasi (Bizagi)



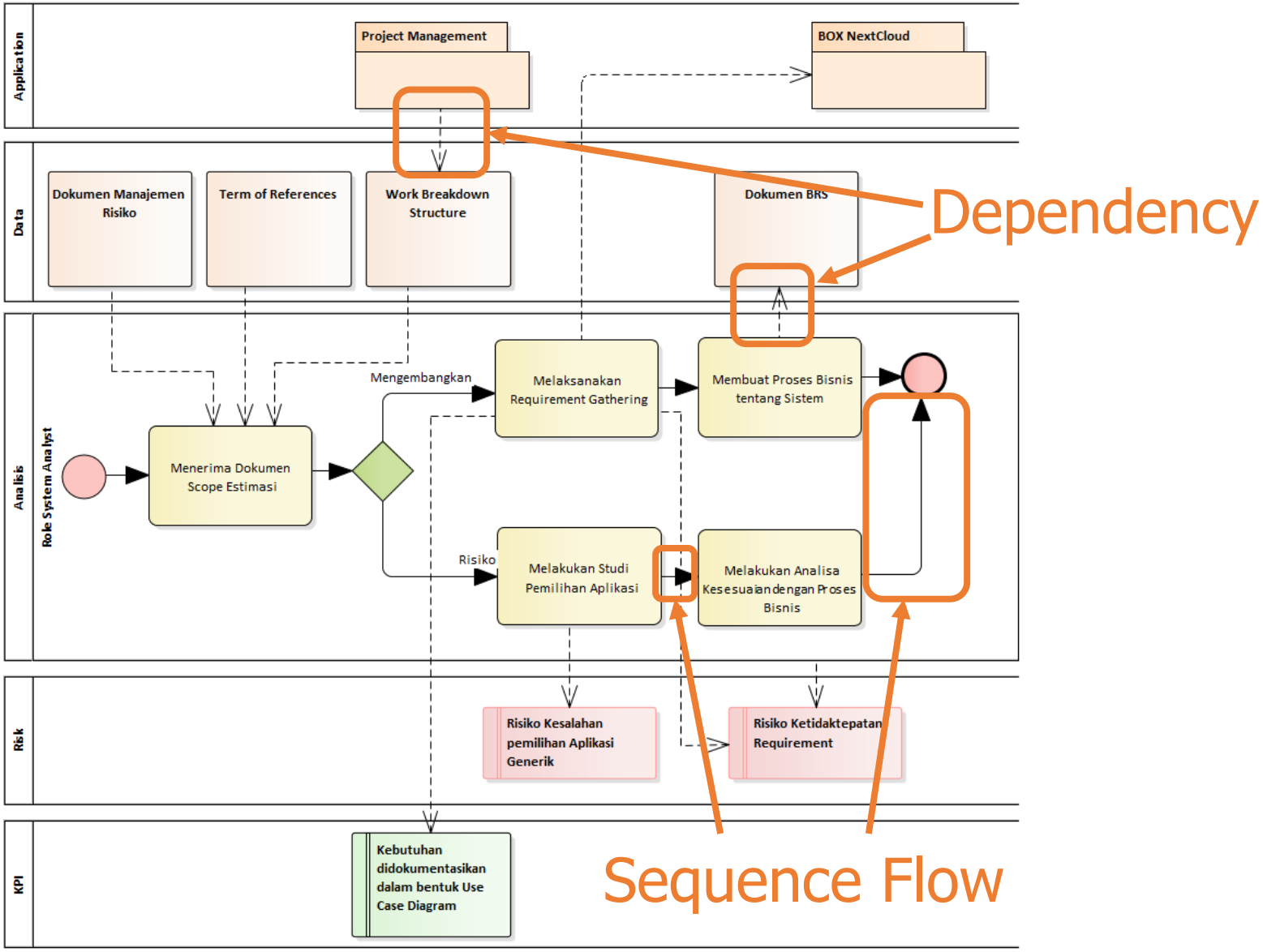
Swimlane - Proses Bisnis Organisasi



Swimlane - Proses Bisnis Sistem



Sequence Flow dan Dependency

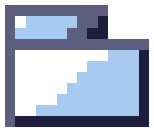
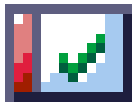
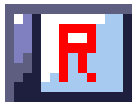


Artifact (Bizagi)

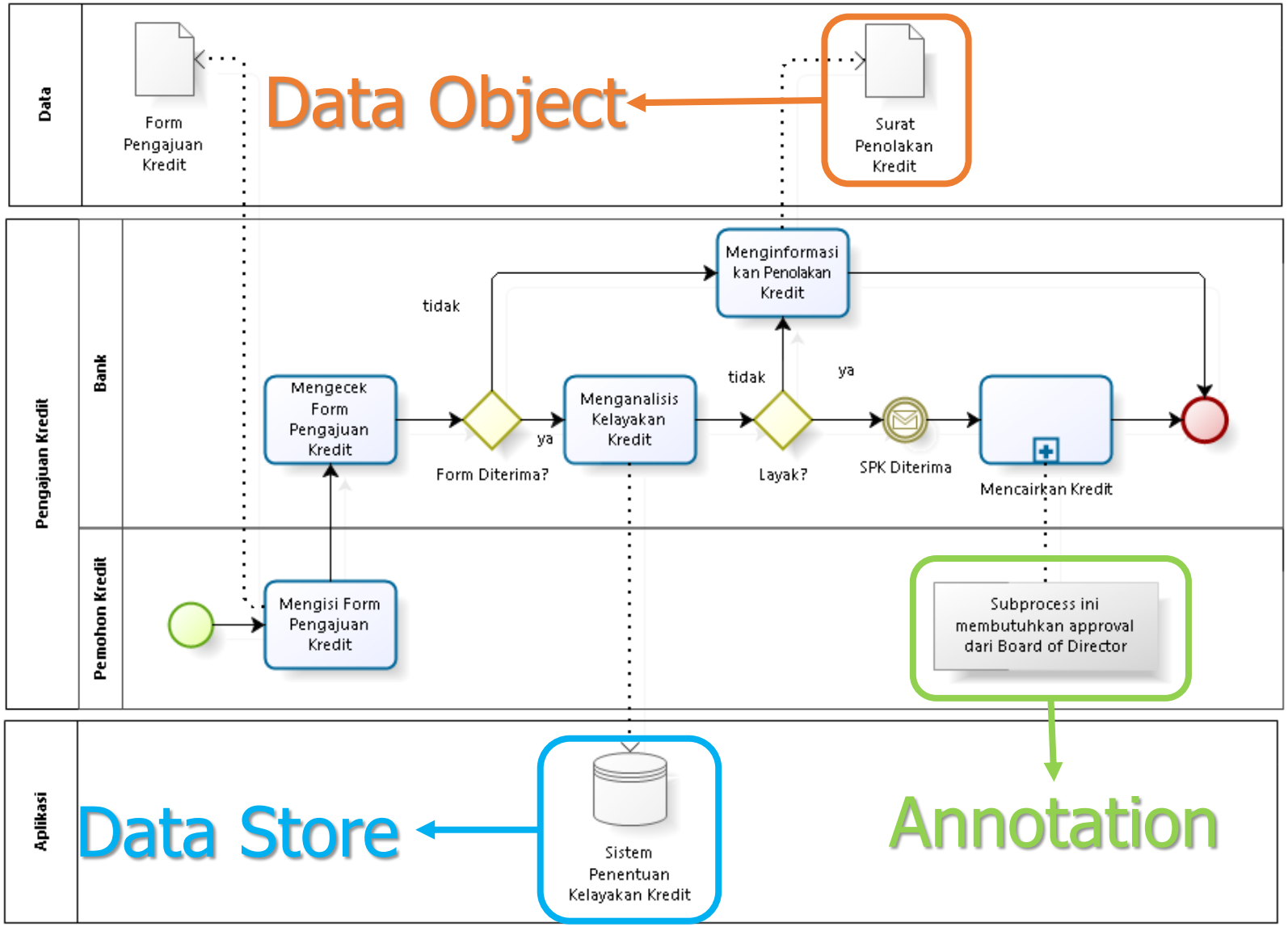
NAMA NOTASI	DESKRIPSI	NOTASI
Annotation	Penjelasan dari suatu obyek yang mengalir	
Group	Pengelompokan dari beberapa obyek yang mengalir	
Data Object	File dan dokumen yang digunakan dan dihasilkan oleh suatu aktifitas	
Data Store	Sistem dan aplikasi yang digunakan dan dihasilkan oleh suatu aktifitas	

Artifact

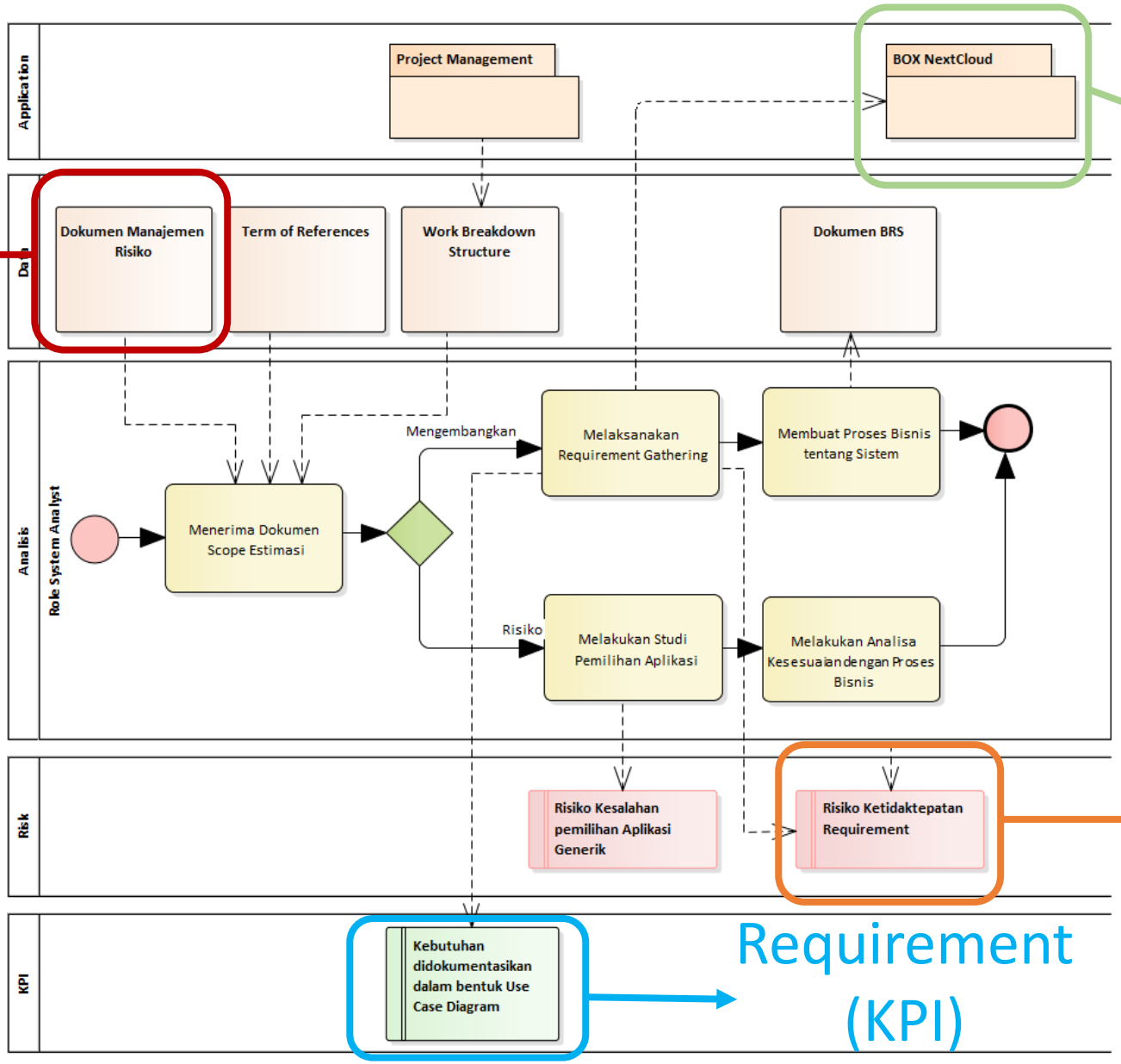
(EA Template using Sparx EA (Wahono, 2017))

Nama Notasi	Deskripsi	Notasi
Class (Data)	File dan dokumen yang digunakan dan dihasilkan oleh suatu aktifitas	
Package (Aplikasi)	Sistem dan aplikasi yang digunakan dan dihasilkan oleh suatu aktifitas	
Requirement (KPI)	KPI yang dijadikan target dari suatu task	
Risk (Risiko)	Risiko yang muncul dari suatu task	

Annotation, Data Object dan Data Store (Bizagi)



Class, Package, Requirement & Risk



Class (Data)

Package (Aplikasi)

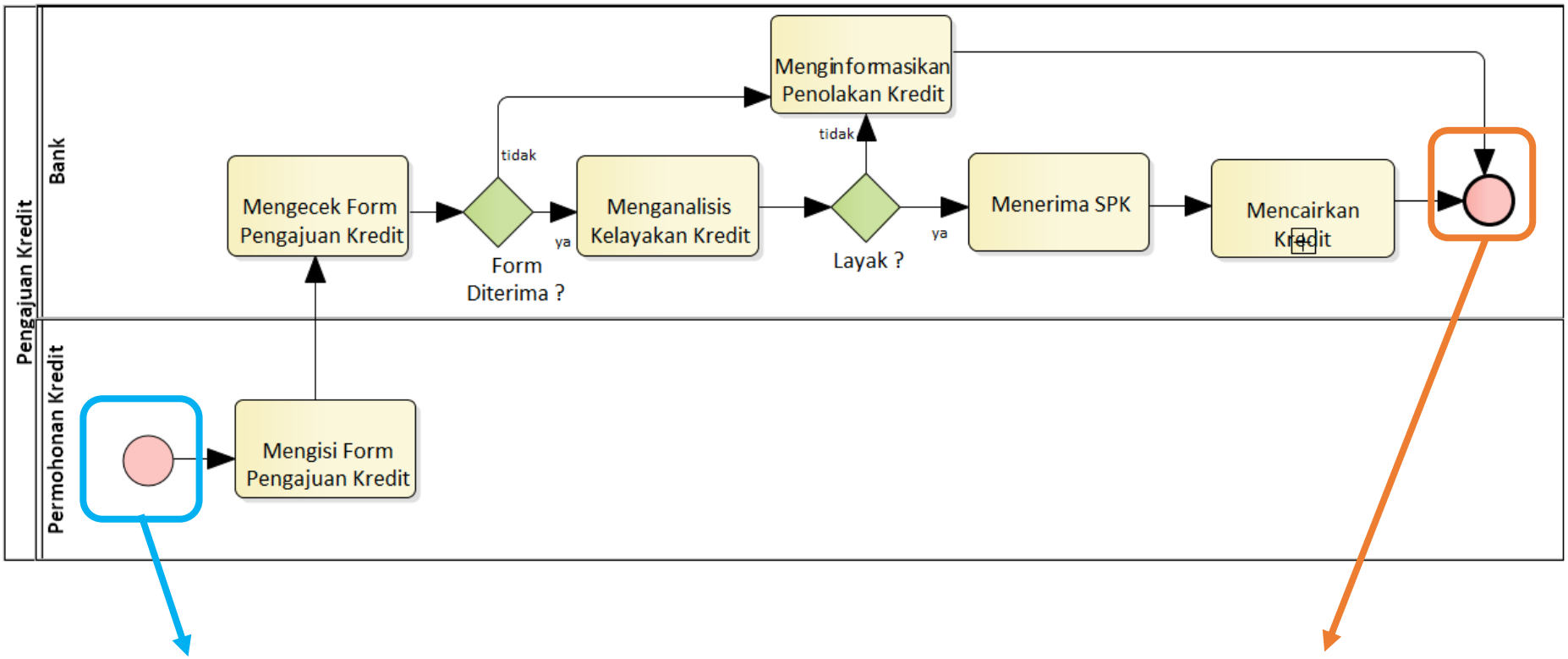
Risk (Risiko)

Requirement (KPI)

Flow Object

Nama Notasi	Deskripsi	Notasi
Event	Suatu kejadian dan sifatnya pasif (*Something that happened*)	
Activities	Kegiatan yang secara aktif dilakukan (*Something to do*)	
Gateway	Pemecah dari beberapa aktifitas	

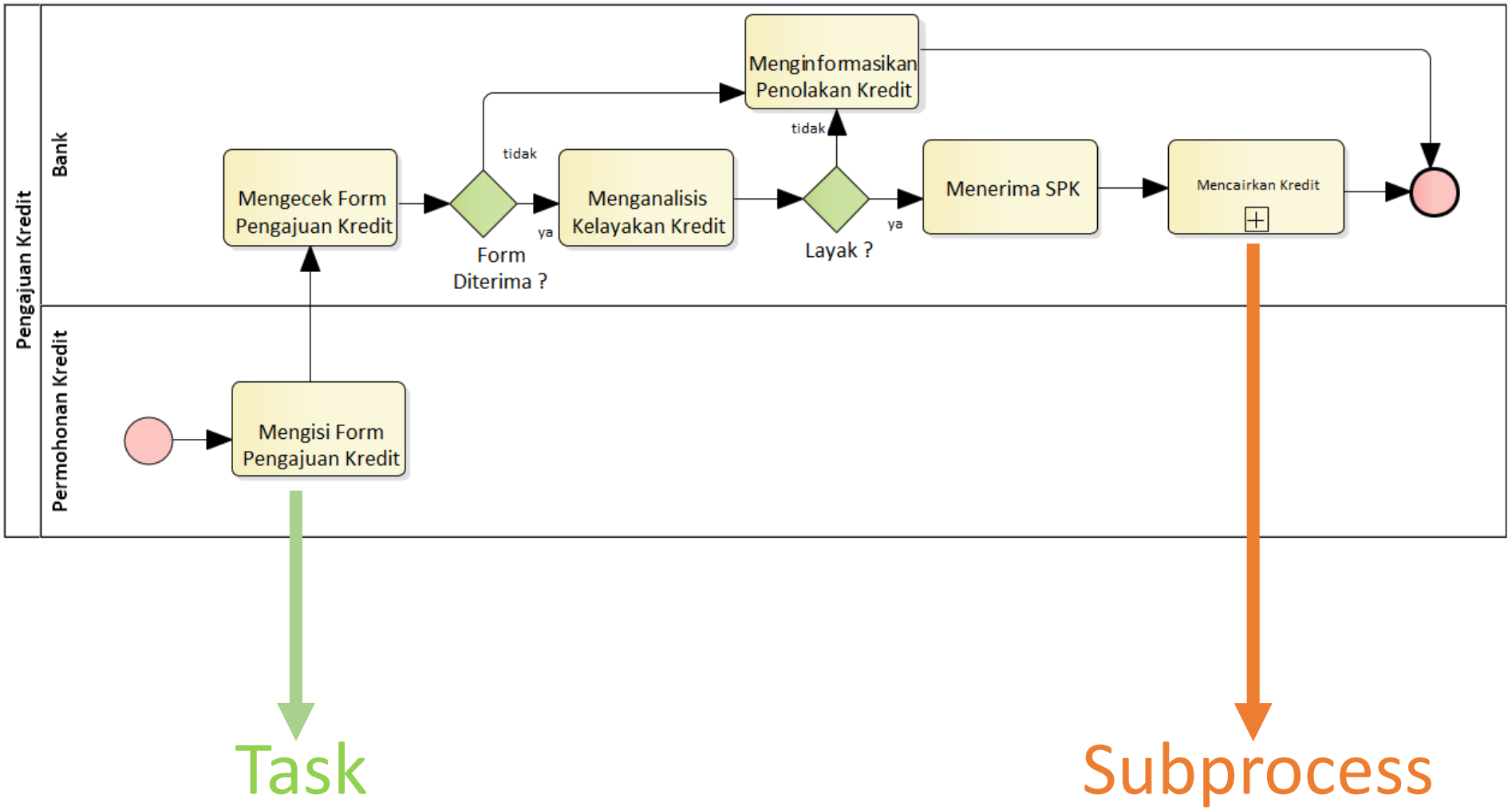
Event



Start Event

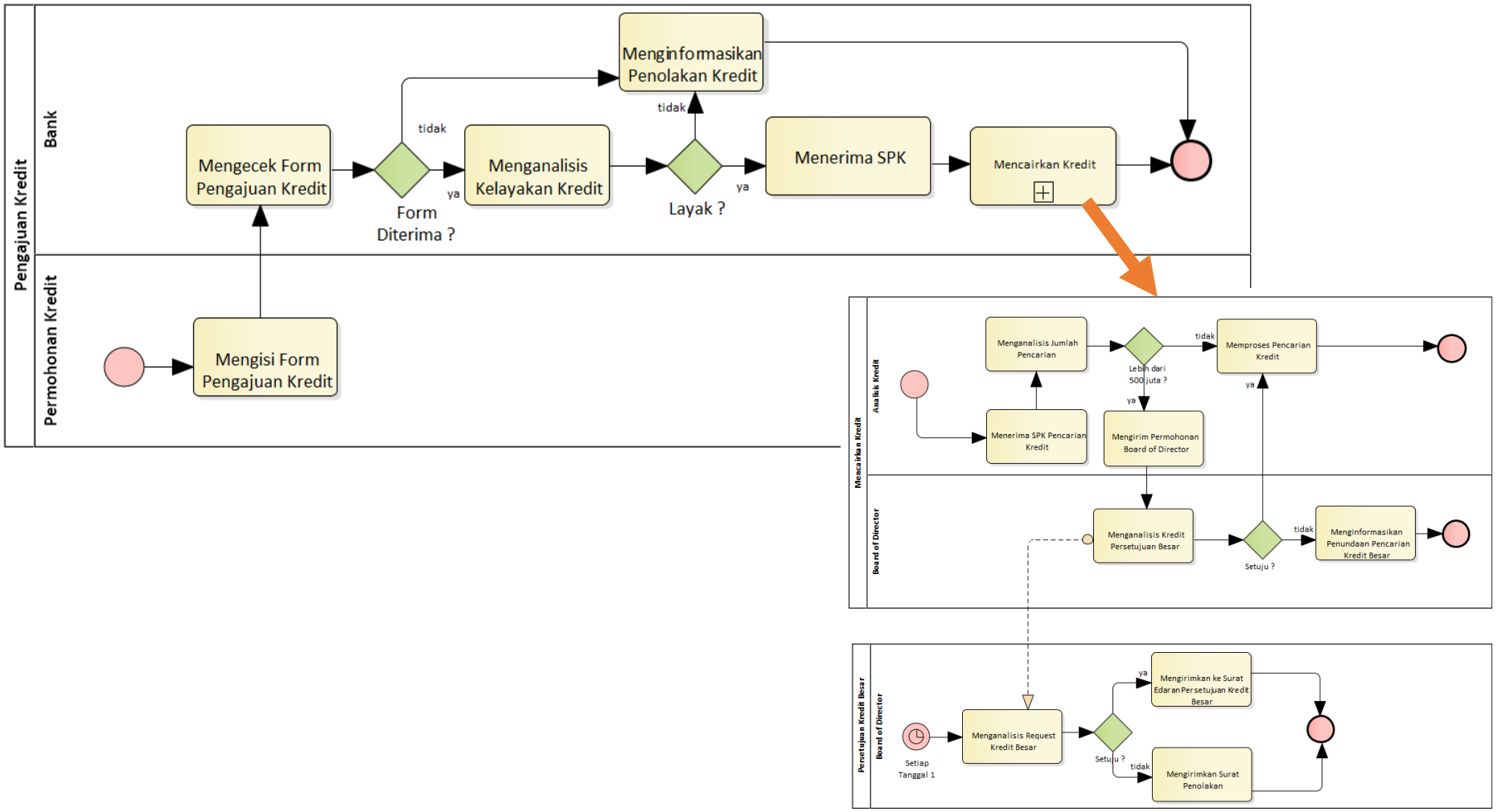
End Event

Activity


Pengajuan Kredit
Bank
Permohonan Kredit
Menginformasikan Penolakan Kredit
tidak
tidak
Mengecek Form Pengajuan Kredit
Form Diterima ?
ya
Menganalisis Kelayakan Kredit
Layak ?
ya
Menerima SPK
Mencairkan Kredit
Mengisi Form Pengajuan Kredit
Task
Subprocess

Subprocess

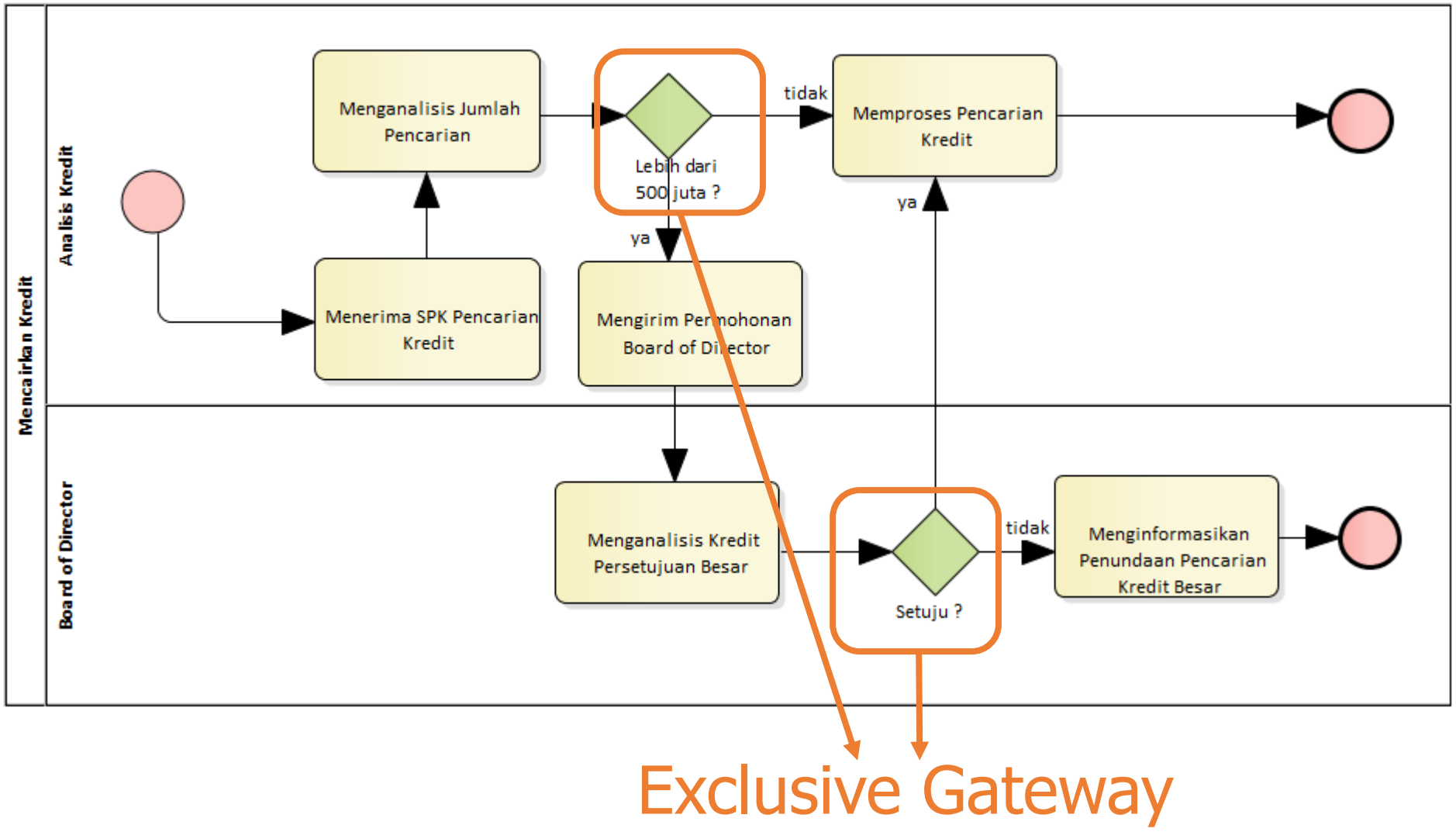
Leveling pada proses bisnis menggunakan BPMN menggunakan subprocess



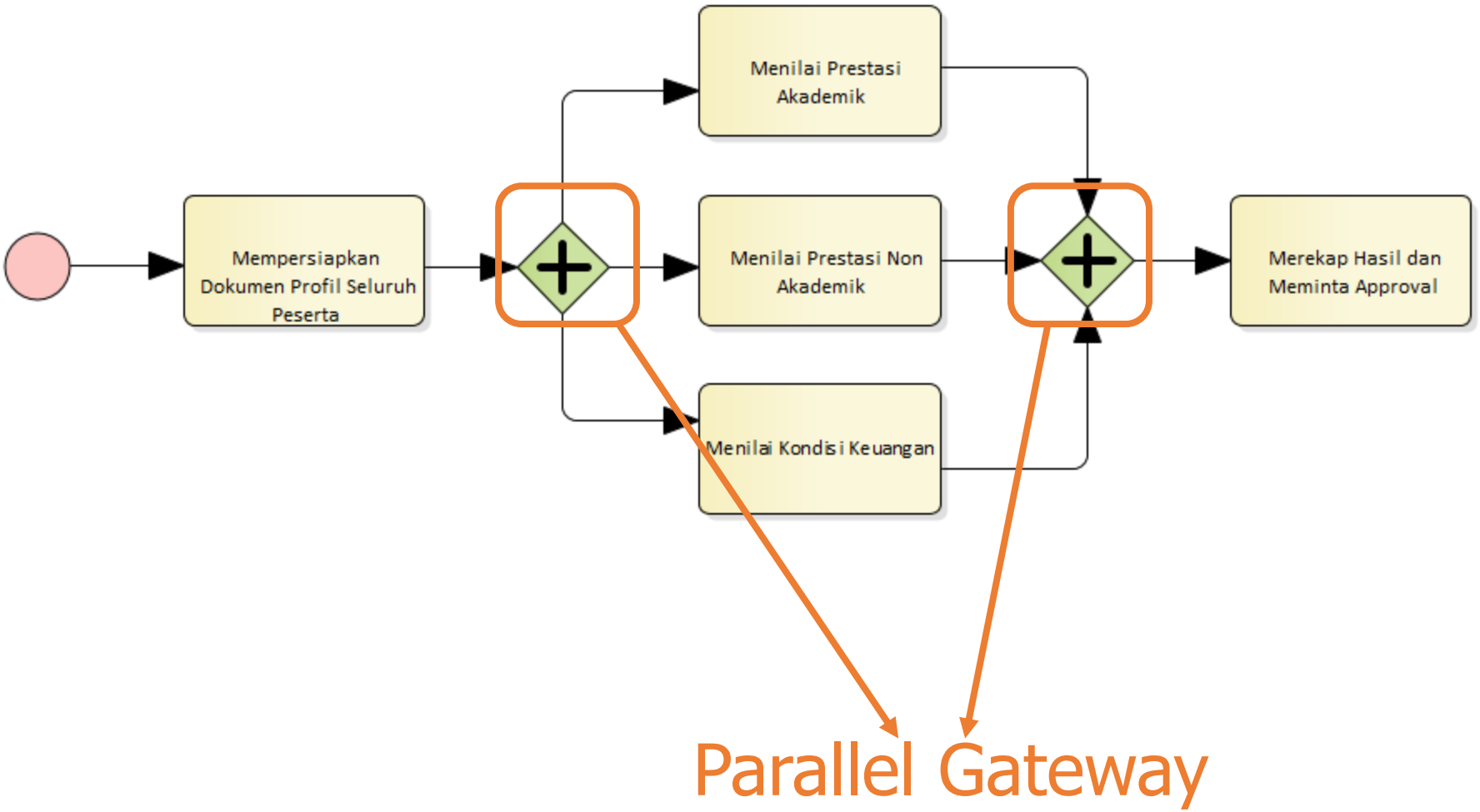
Gateway

Nama Notasi	Deskripsi	Notasi
Exclusive Gateway	Pilih salah satu	
Parallel Gateway	Kegiatan bersamaan (paralel) dalam satu waktu	

Exclusive Gateway



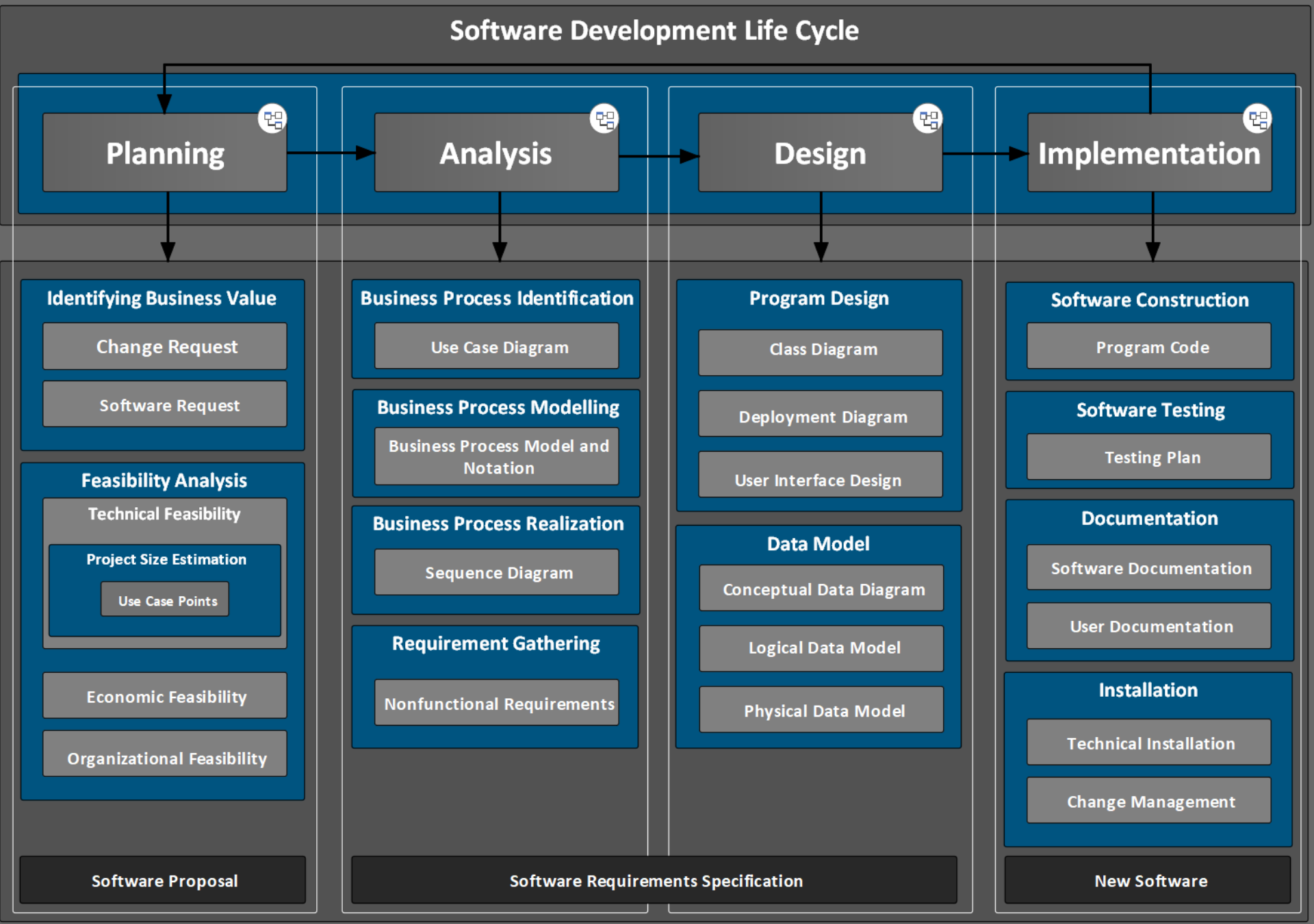
Parallel Gateway

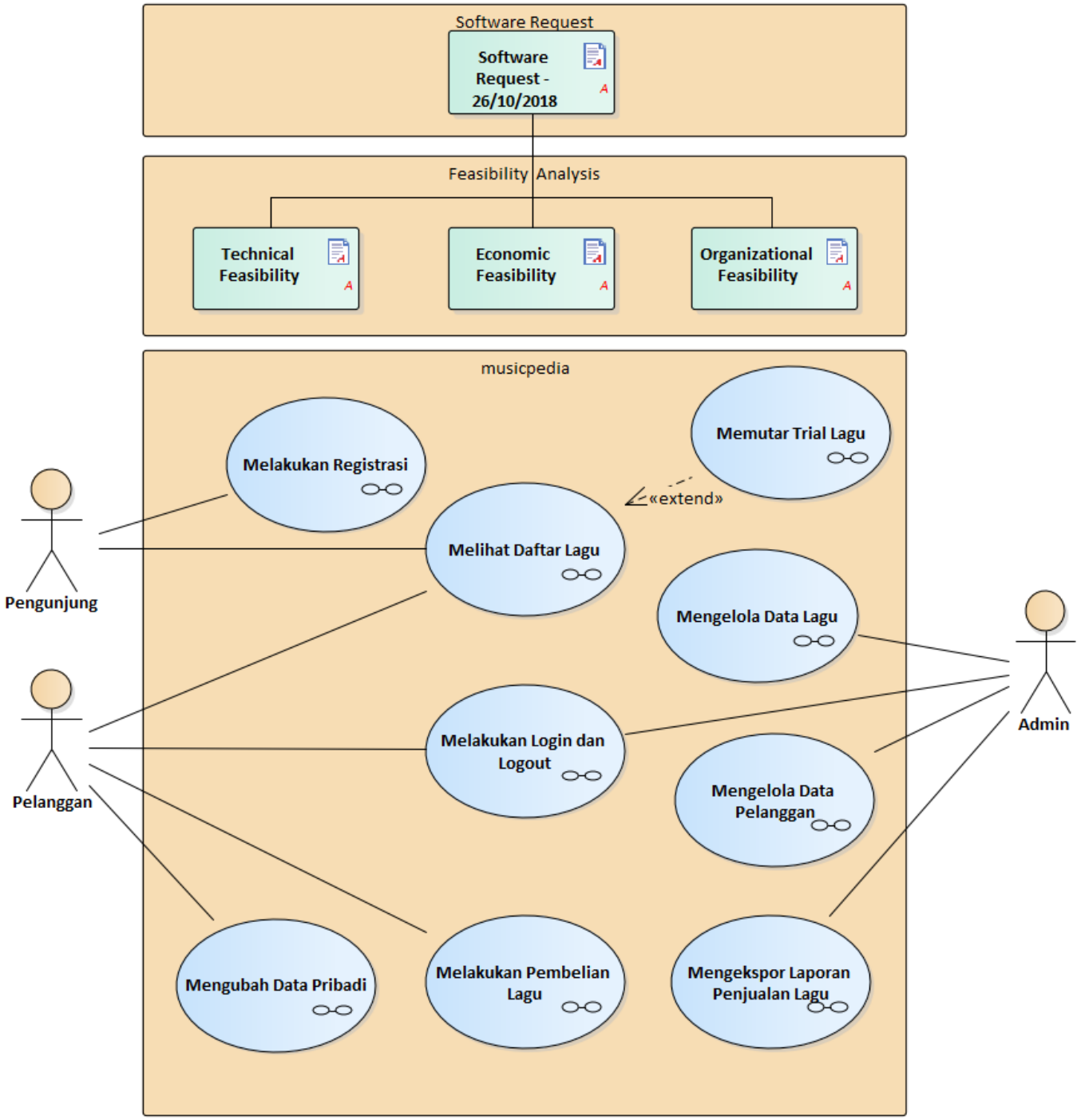


Studi Kasus: BPMN MusicPedia

User Interface Diagram langsung digambarkan (prototyping) untuk mempermudah pengguna memahami konteks dari alur proses berjalannya software

Application Development Governance





Technical Feasibility

musicpedia				
Date: 26 Oktober 2018				
Penjelasan isian	1. Sangat Kurang	2. Kurang	3. Baik	4. Sangat Baik

Kefamiliaran dengan Aplikasi	1	2	3	4
Pengguna familiar terhadap pengoperasian aplikasi ini.	☐	☐	☑	☐
Pengembang familiar terhadap pengembangan aplikasi ini.	☐	☐	☐	☑
Kefamiliaran dengan Teknologi	**1**	**2**	**3**	**4**
Pengguna familiar dengan teknologi pendukung aplikasi.	☐	☑	☐	☐
Pengembang familiar mengembangkan aplikasi dengan platform, bahasa pemrograman dan tool IDE yang dipilih.	☐	☐	☐	☑
Ukuran Proyek				
Jumlah pengembang yang dibutuhkan.	7 Man/Month			
Waktu yang dibutuhkan dalam mengembangkan aplikasi ini.	6 Month			
Kompatibilitas	**1**	**2**	**3**	**4**
Kebutuhan pengguna terhadap kompatibilitas aplikasi untuk terintegrasi dengan aplikasi lain.	☐	☐	☑	☐
Kompatibilitas aplikasi terhadap teknologi yang ada pada organisasi.	☐	☐	☐	☑

Secara analisi kelayakan teknis, apakah aplikasi layak dikembangkan sesuai kriteria di atas?	☑ Layak	☐ Tidak Layak

Technical Feasibility

Use Case Points

Tahap 1 - Menghitung Person Hours (PH)				
Use Case Points (UCP)		Person Hours Multiplier (PHM)	Person Hours (PH)	
51		20	1020	
51		28	1428	
Tahap 2 - Menghitung Person Month (PM)				
PHM	Person Hours (PH)	Lama Bekerja Perhari	Jumlah Bekerja Sebulan	Person Months (PM)
20	1020	8	22	5.80
	1020	10	26	3.92
28	1428	8	22	8.11
	1428	10	26	5.49
Tahap 3 - Menghitung Time (Month)				
PHM	Formula Penghitung Waktu		Jumlah Bekerja Sebulan	Waktu dalam Bulan (M)
20	3 * PM^(1/3)		22	5.39
			26	4.73
28			22	6.03
			26	5.29

Economic Feasibility

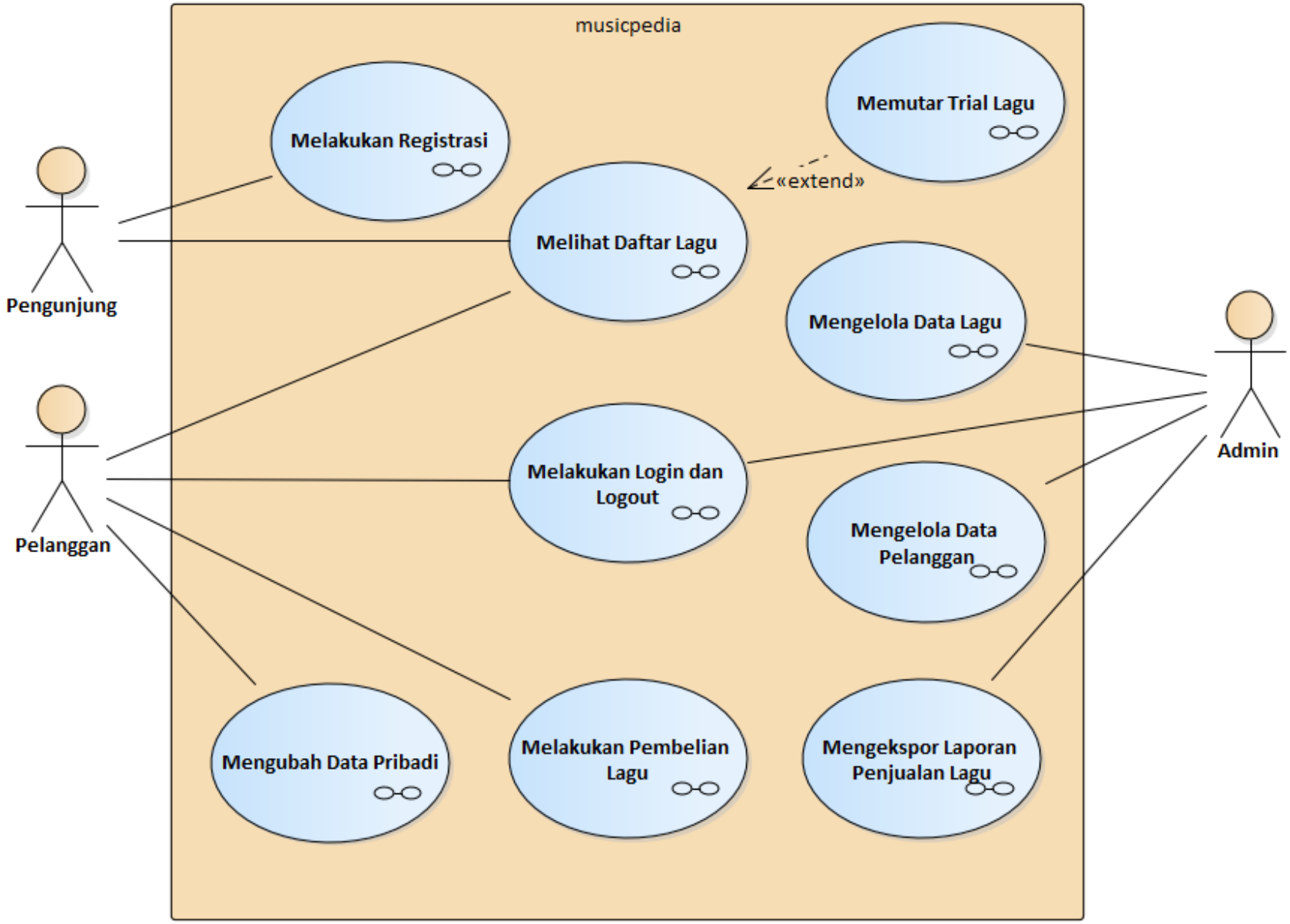
Cost-Benefit Analysis

Tahun	2019	2020	2021	2022
Peningkatan Pendapatan Penjualan Lagu		400,000,000	400,000,000	400,000,000
Pengurangan Biaya Sewa Ruangan		120,000,000	120,000,000	120,000,000
Pengurangan Biaya Komunikasi		6,000,000	6,000,000	6,000,000
Total Benefits	**0**	**526,000,000**	**526,000,000**	**526,000,000**
PV of Benefits	**0**	**468,138,127**	**441,639,743**	**416,641,267**
PV of All Benefits	**0**	**468,138,127**	**909,777,870**	**884,779,394**
Honor Tim (Analysis, Design and Implementation)	250,000,000	120,000,000	120,000,000	120,000,000
Total Development Costs	**250,000,000**	**120,000,000**	**120,000,000**	**120,000,000**
Honor Pengelola Web	72,000,000	72,000,000	72,000,000	72,000,000
Biaya Lisensi Software	10,000,000	10,000,000	10,000,000	10,000,000
Hardware upgrades	50,000,000	50,000,000	50,000,000	50,000,000
Biaya Komunikasi	1,000,000	1,000,000	1,000,000	1,000,000
Biaya Marketing	50,000,000	50,000,000	50,000,000	50,000,000
Total Operational Costs	**183,000,000**	**183,000,000**	**183,000,000**	**183,000,000**
Total Costs	**433,000,000**	**303,000,000**	**303,000,000**	**303,000,000**
PV of Costs	**408,490,566**	**269,668,921**	**153,650,329**	**144,953,140**
PV of all Costs	**408,490,566**	**678,159,487**	**831,809,816**	**976,762,957**
Total Project Costs Less Benefits	**-433,000,000**	**223,000,000**	**223,000,000**	**223,000,000**
Yearly NPV	**-408,490,566**	**198,469,206**	**187,235,100**	**176,636,887**
Cumulative NPV	**-408,490,566**	**-210,021,360**	**-22,786,260**	**153,850,627**
Return on Investment (ROI)	**-100.00%**	**-0.309693168**	**-0.027393593**	**0.15751071**
Break-even Point (BEP)				**3.129000574**

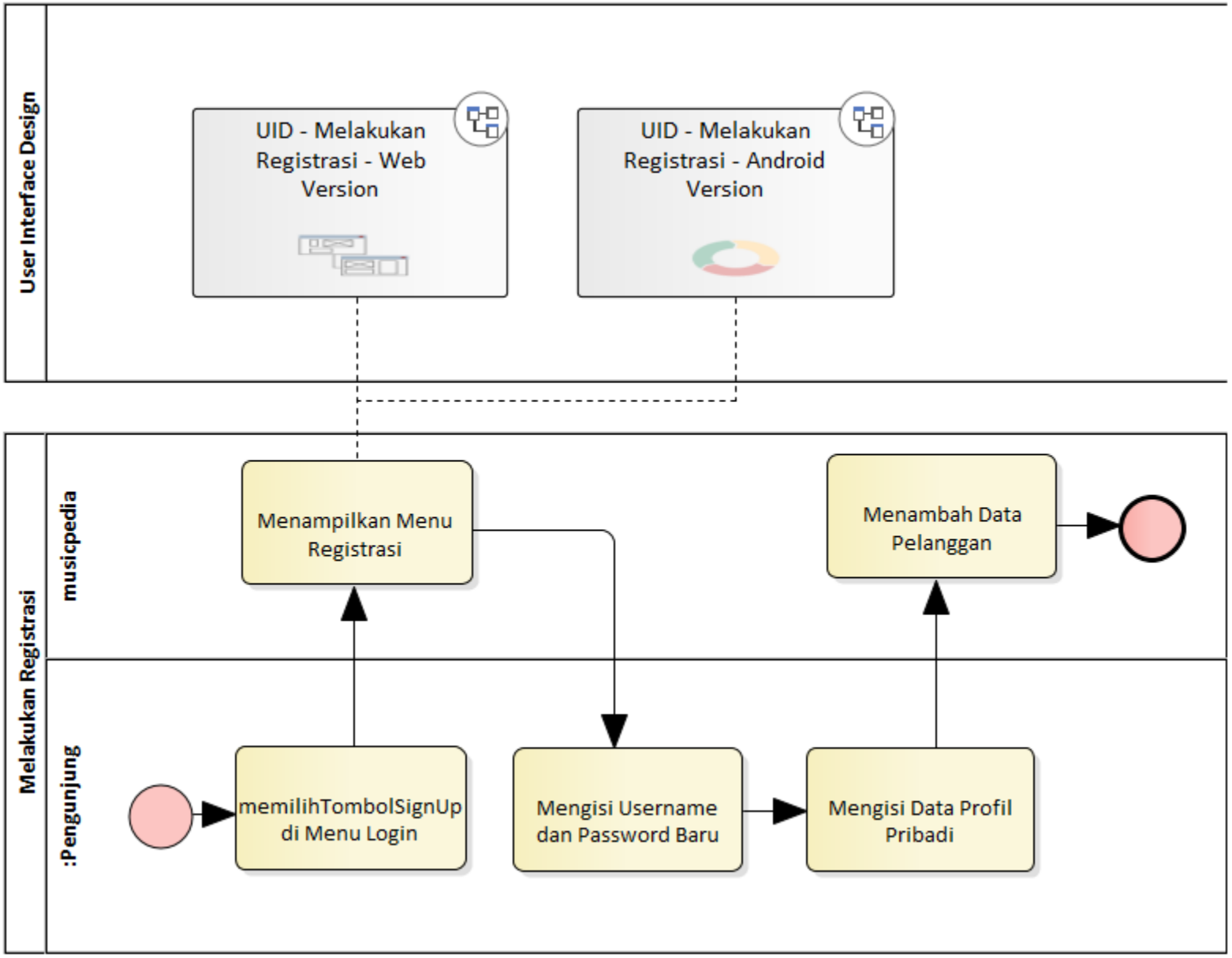
Organizational Feasibility

musicpedia			
Date	**26 Oktober 2018**		
Anggota Tim			
User/Product Owner	Wahyu Utomo		
Project Manager	Haris Dermawan		
System Analyst	Risa Dhani Horasman Purba		
Business Analyst	Mulyana		
Programmer	Achmad Fatkarrofiqi		
Tester	Januar Sapareza		
Apakah aplikasi ini mendukung visi dan misi organisasi?			
Ya			
Apakah aplikasi ini sesuai dengan tugas, fungsi dan KPI unit kerja anda?			
Ya			
Apakah aplikasi ini selaras dengan proses bisnis unit kerja anda?			
Ya			
Secara analisis kelayakan organisasi, apakah aplikasi layak dikembangkan sesuai kriteria di atas?		☑ Layak	☐ Tidak Layak

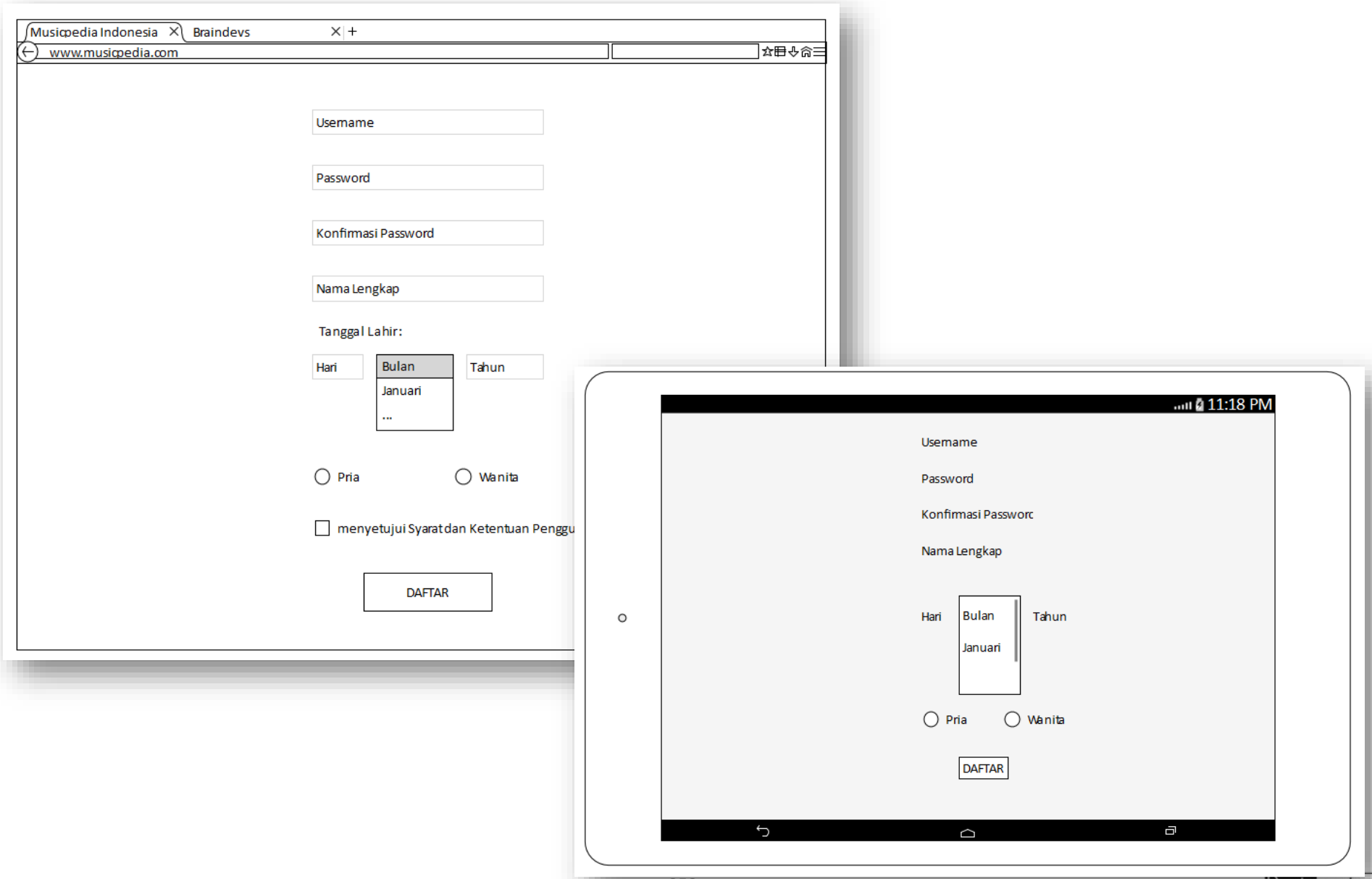
Use Case Diagram MusicPedia



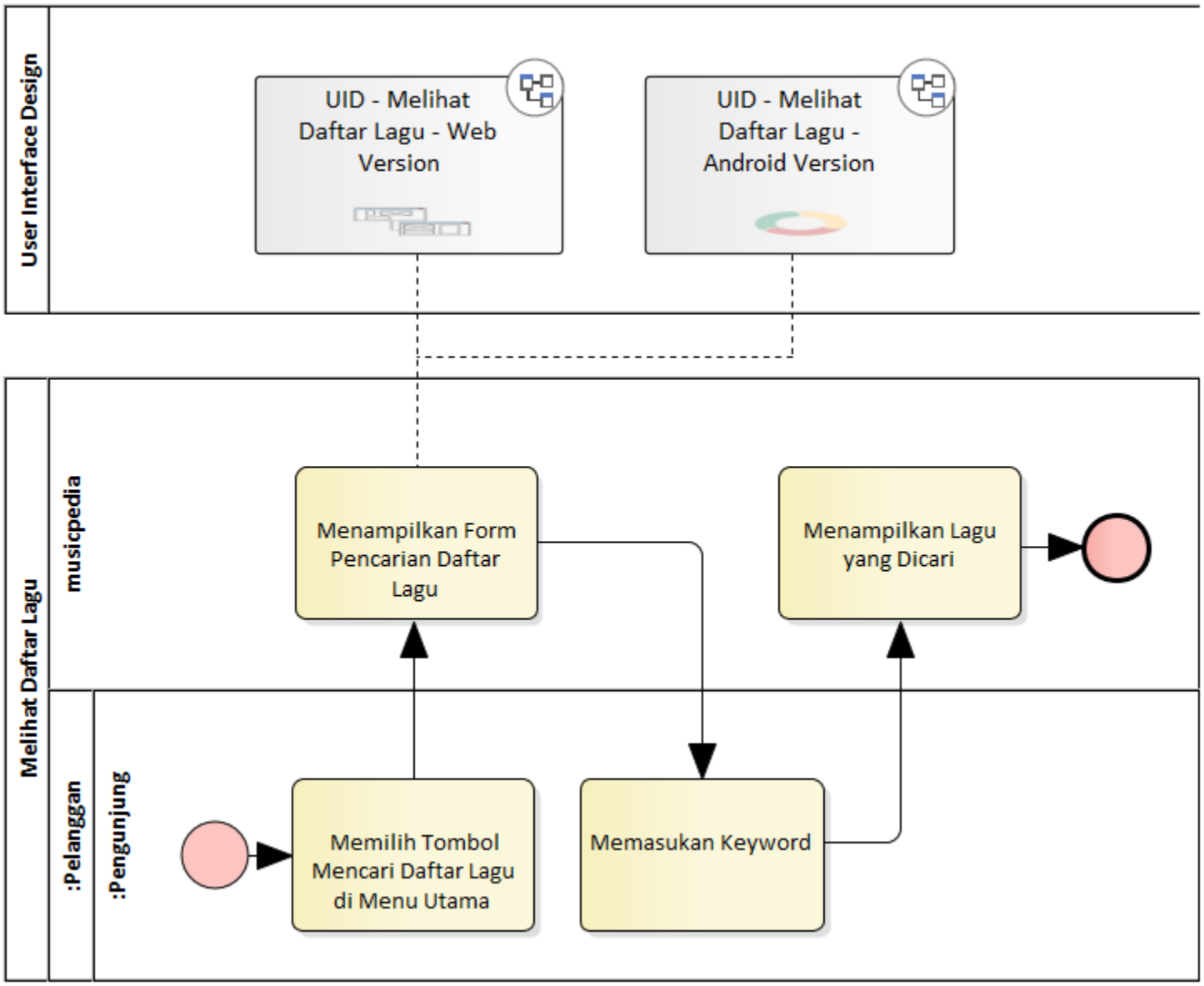
BPMN Melakukan Registrasi



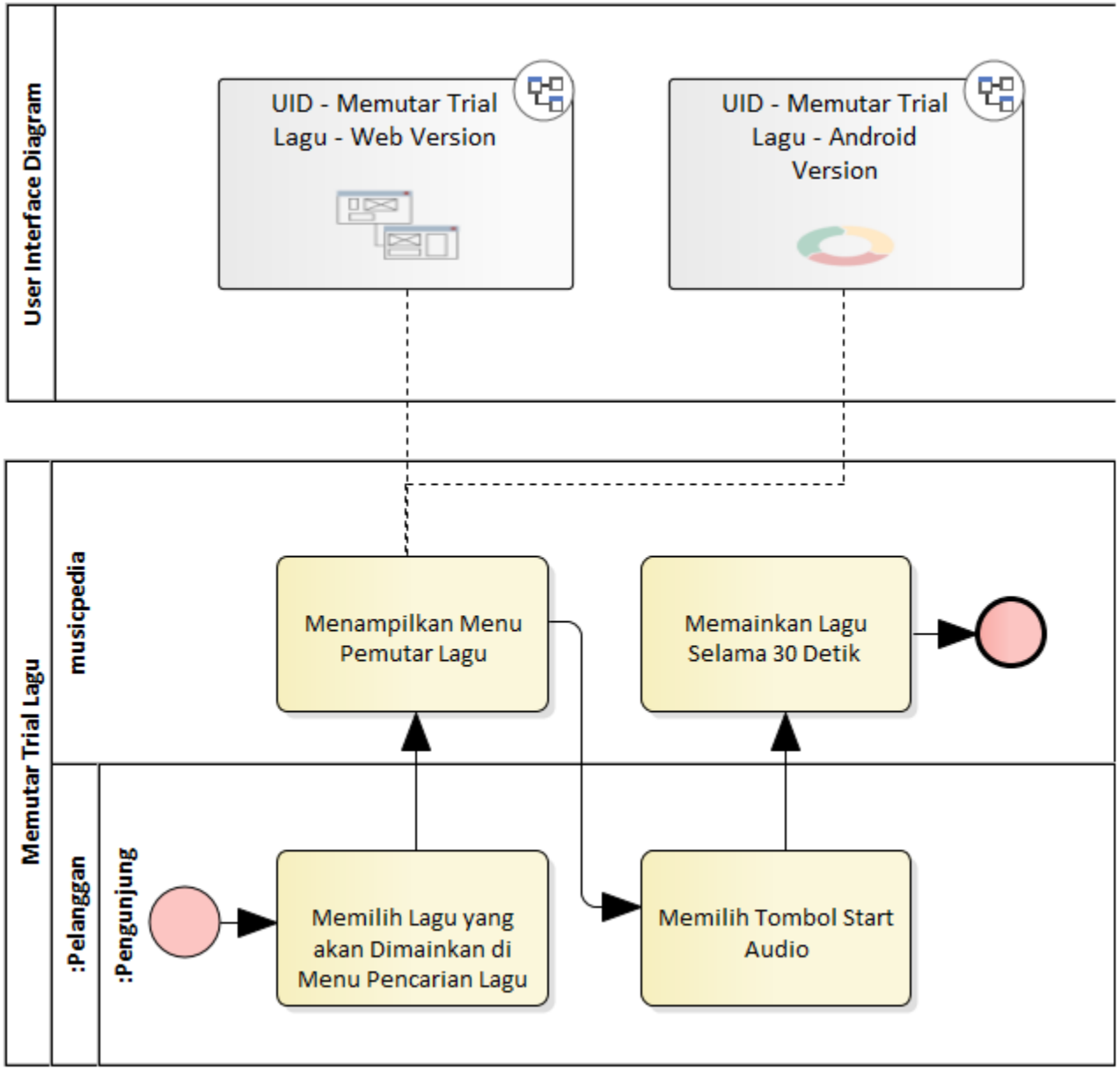
User Interface Design Melakukan Registrasi (versi Web dan versi Android)



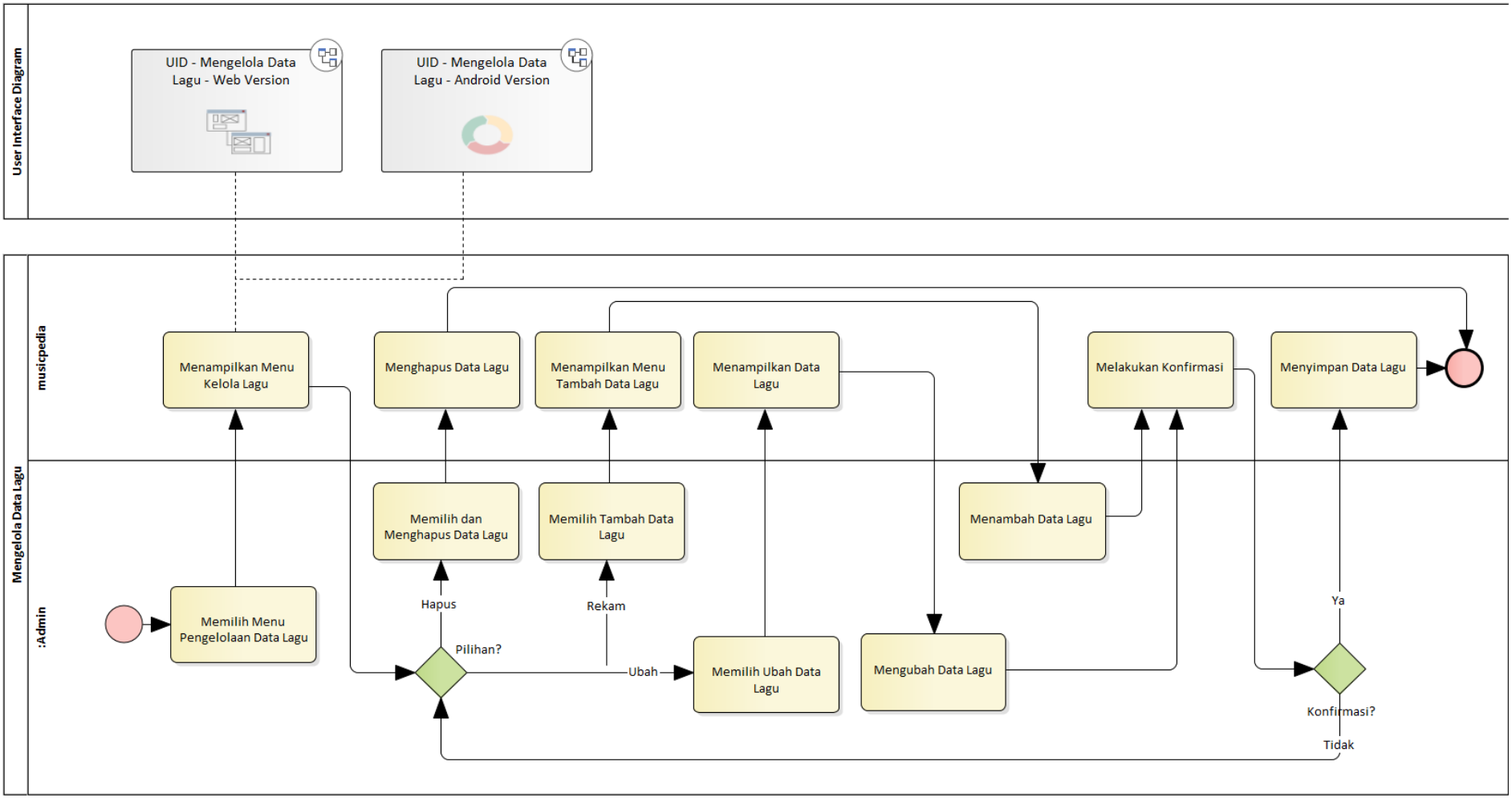
BPMN Melihat Daftar Lagu



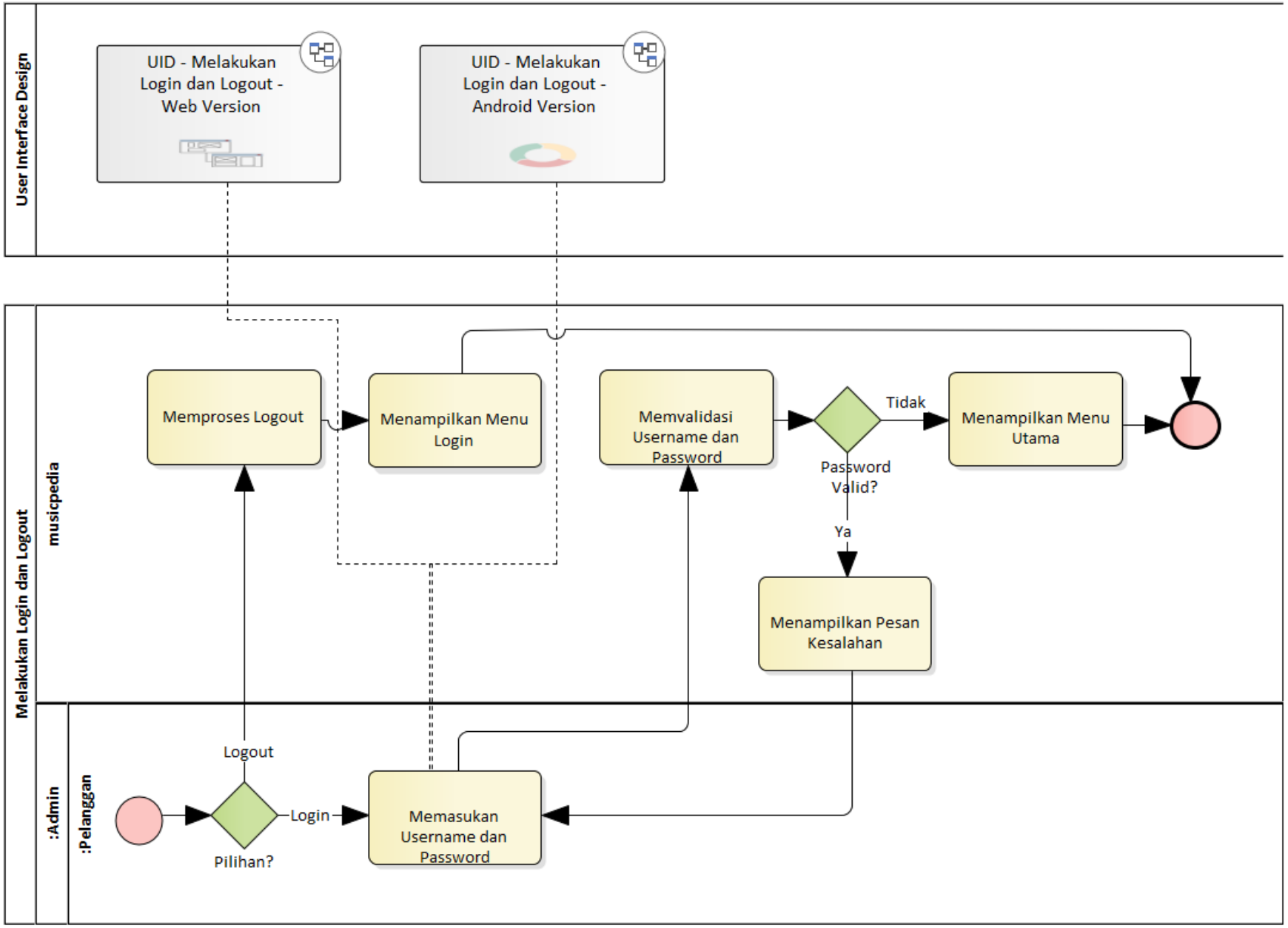
BPMN Memutar Trial Lagu



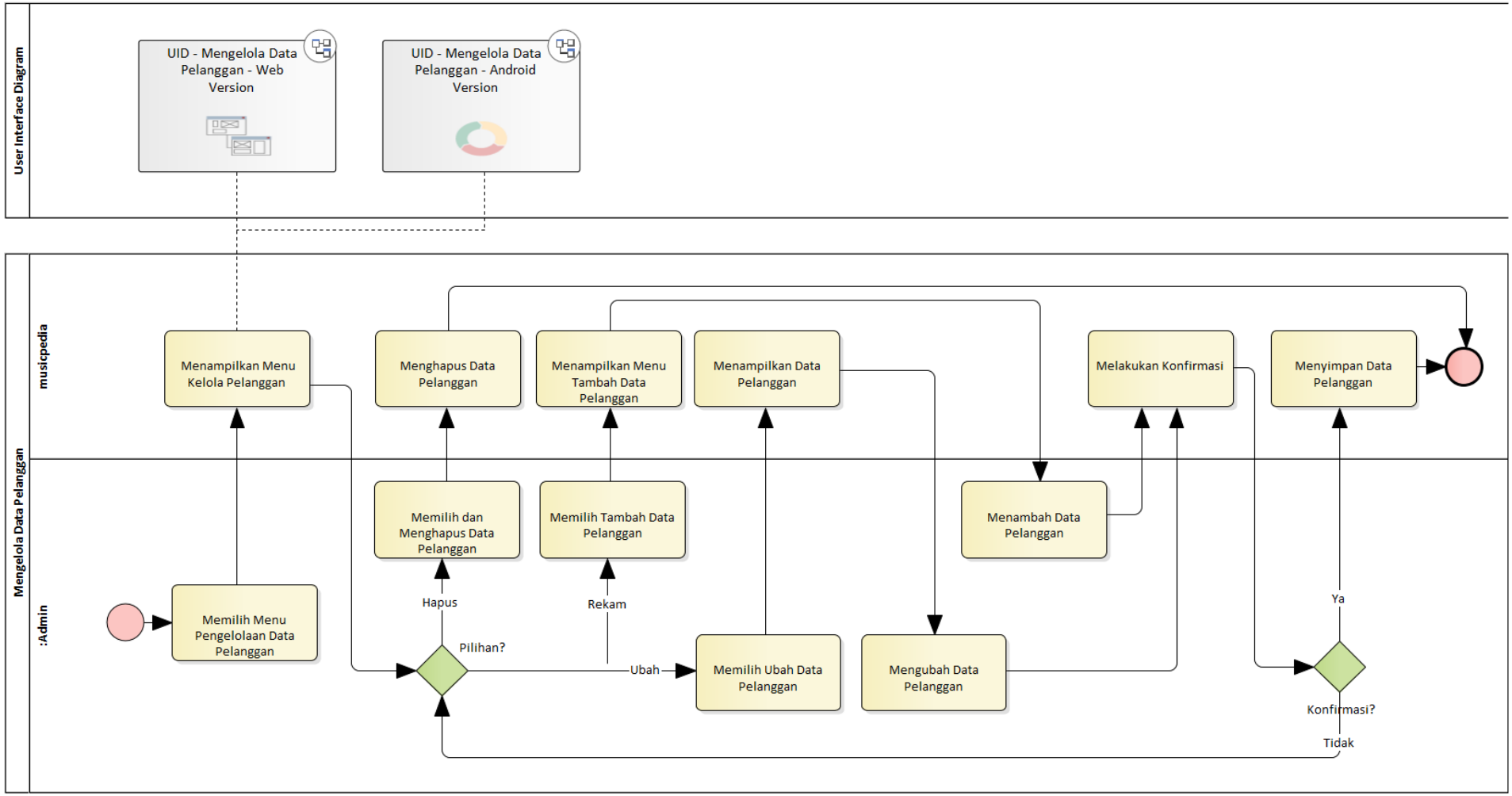
BPMN Mengelola Data Lagu



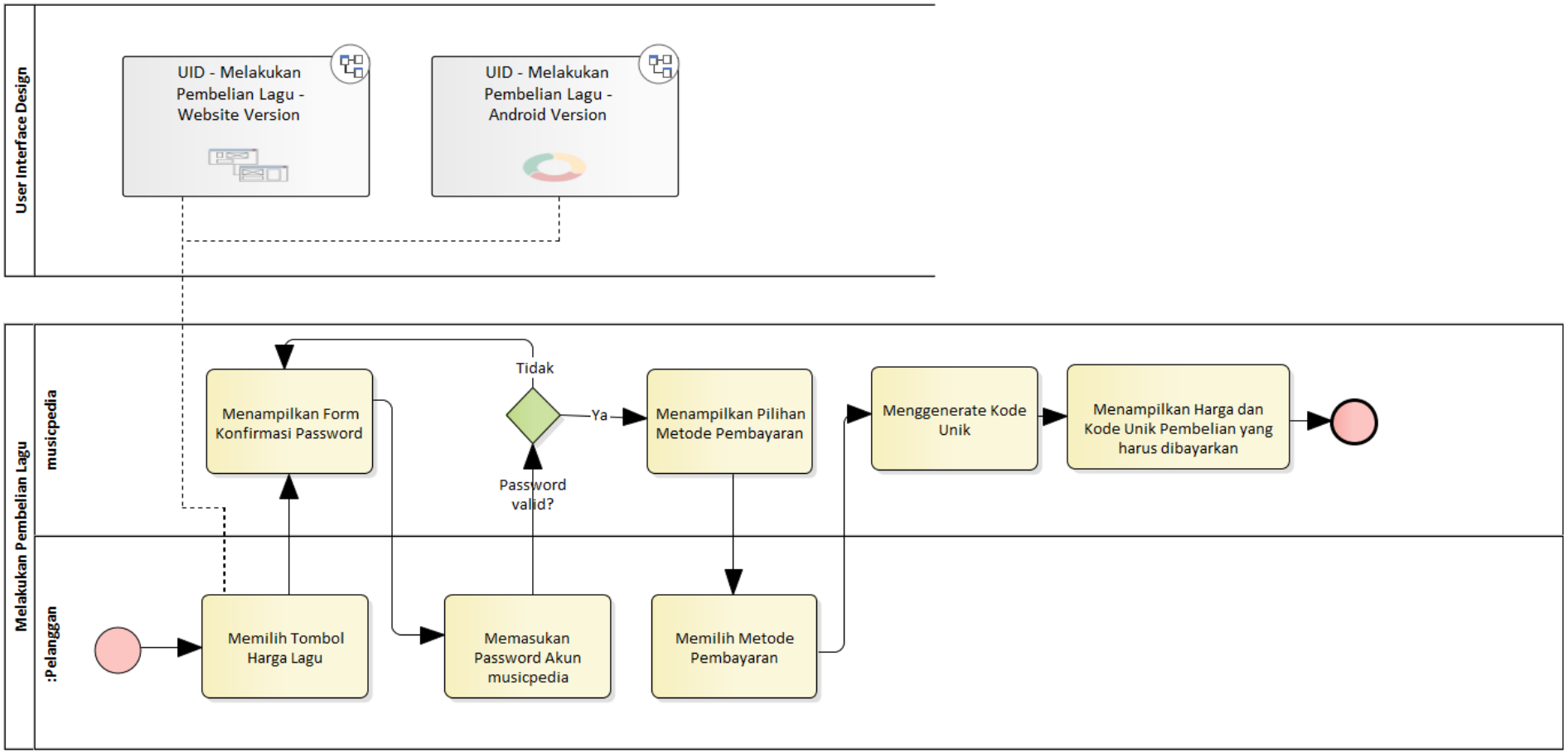
BPMN Melakukan Login dan Logout



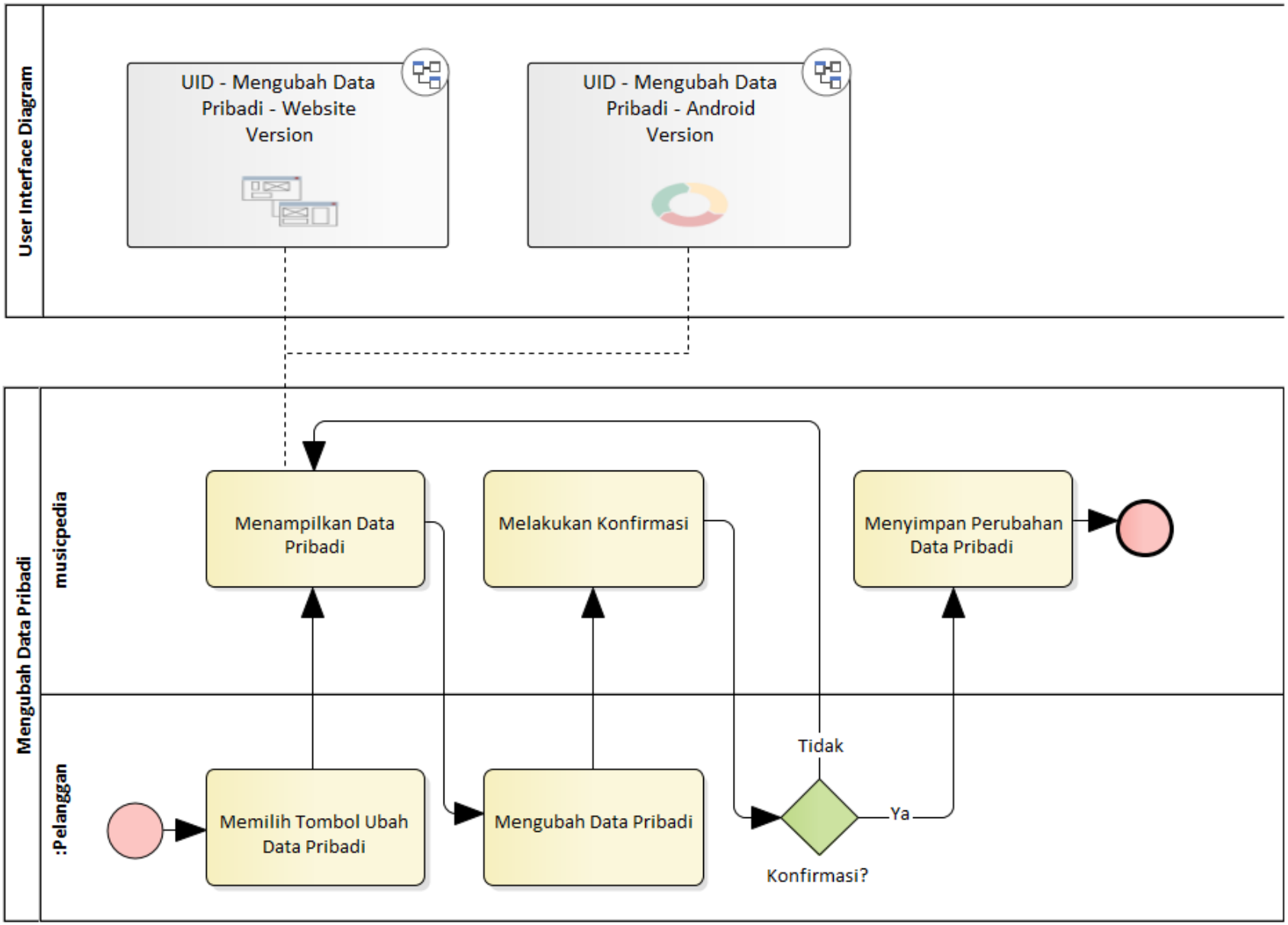
BPMN Mengelola Data Pelanggan



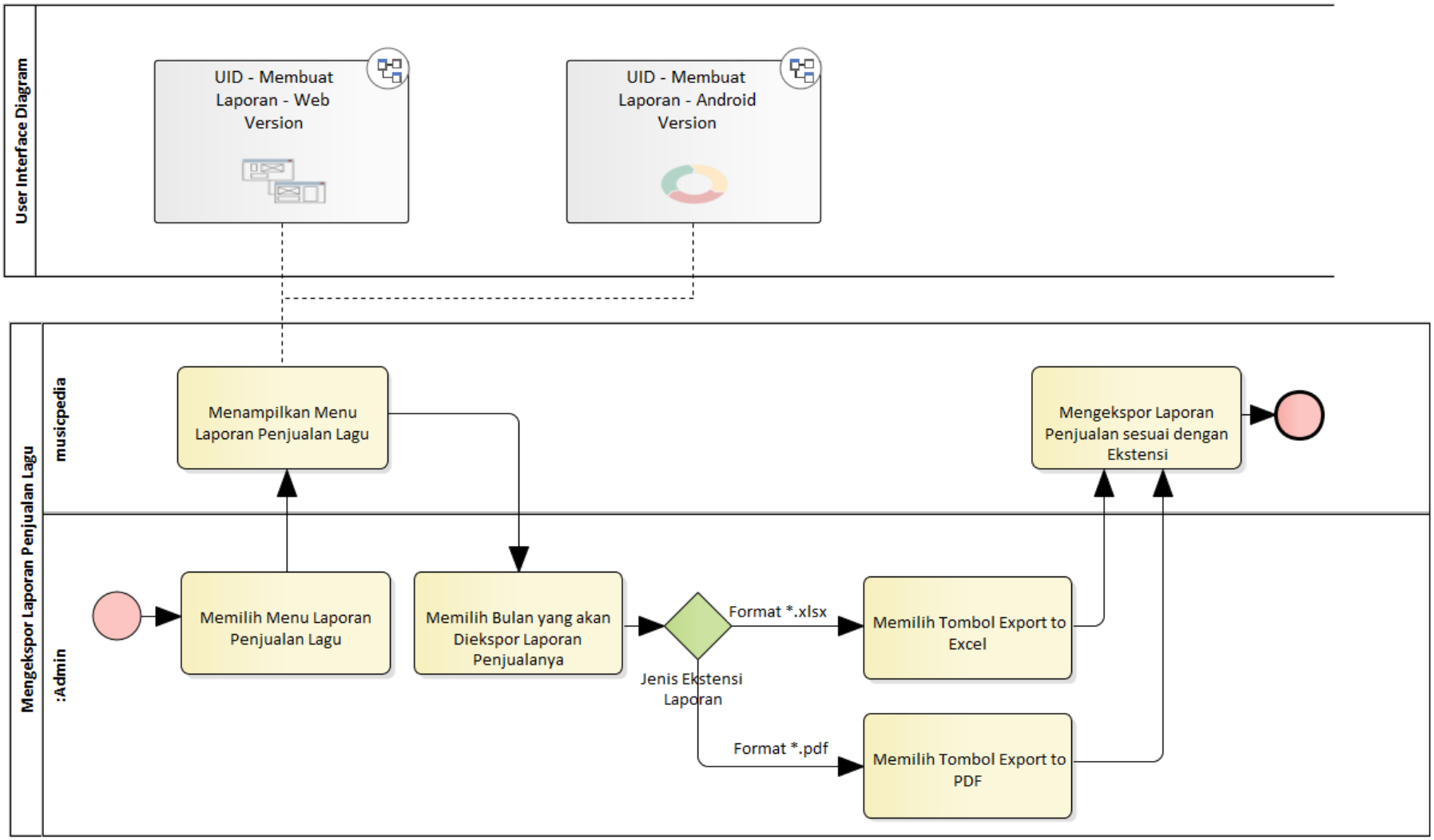
BPMN Melakukan Pembelian lagu



BPMN Mengubah Data Pribadi



BPMN Mengekspor Laporan Penjualan Lagu



Exercise: Business Process Modeling

1. Lihat kembali System Request dan Use Case Diagram yang sudah anda buat
2. Lakukan business process modeling dengan membuatkan BPMN untuk setiap Use Case yang dibuat termasuk User Interface Design apabila diperlukan

3. Kirim file EAPX ke romi@brainmatics.com untuk dilakukan review bersama

Exercise: Systems Analysis and Design

- Lakukan tahapan sistem analysis dan gambarkan dengan:
  1. Use Case Diagram
  2. Activity Diagram **or BPMN**

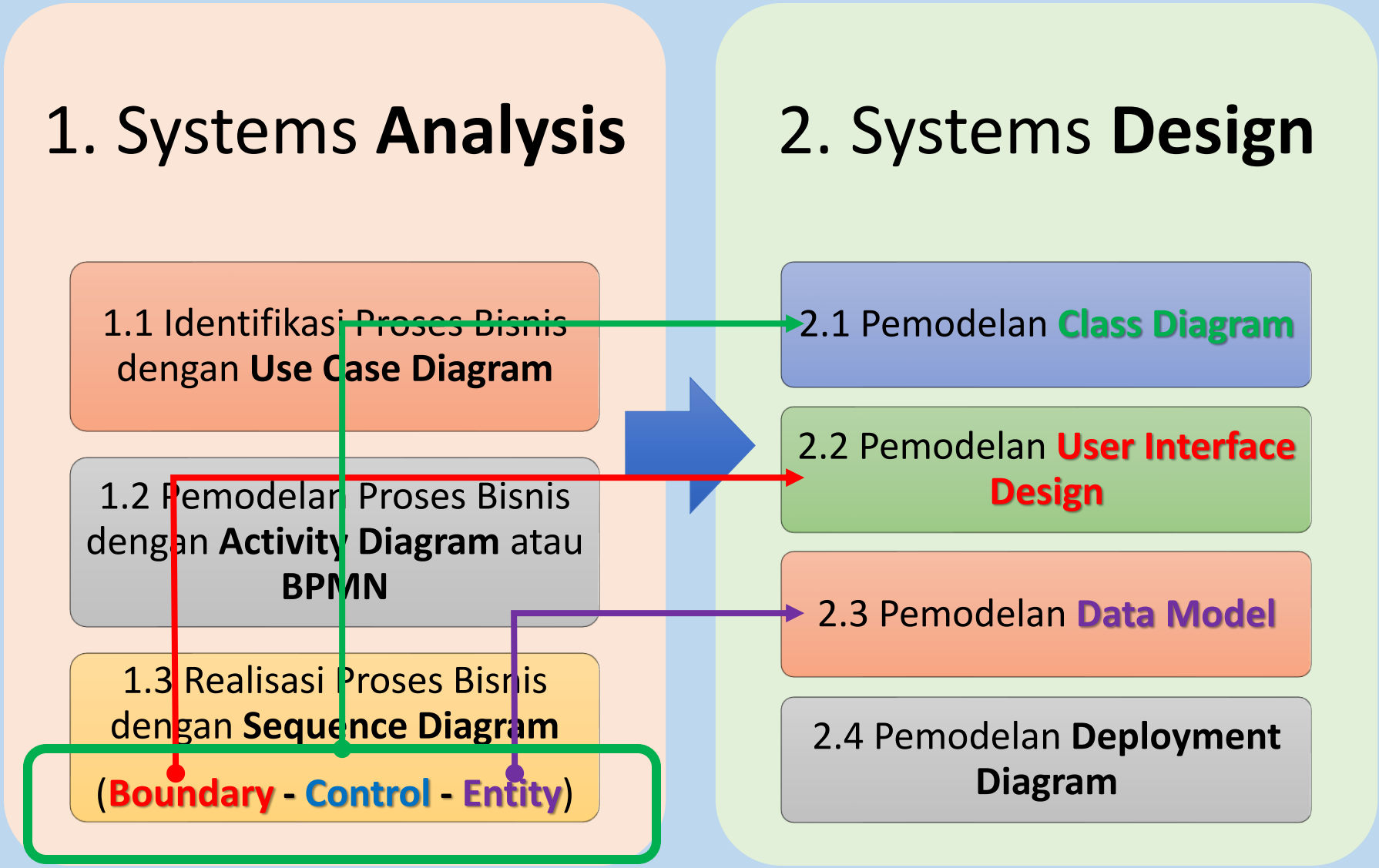
- Pilih salah satu aplikasi di bawah:

1. Aplikasi Rental Mobil
2. Aplikasi Pengelolaan Klinik
3. Aplikasi Pengelolaan Apotik
4. Aplikasi Pengelolaan Service Mobil
5. Aplikasi Penjualan Motor
6. Aplikasi Pengelolaan Perpustakaan
7. Aplikasi Penjualan Buku Online
8. Aplikasi Penjualan Tiket Kereta Online
9. Aplikasi Manajemen Universitas Online
10. Aplikasi Penjualan Laptop Online
11. Aplikasi Perpustakaan Digital
12. Aplikasi Pengelolaan Project Software
13. Aplikasi Pemesanan Taxi

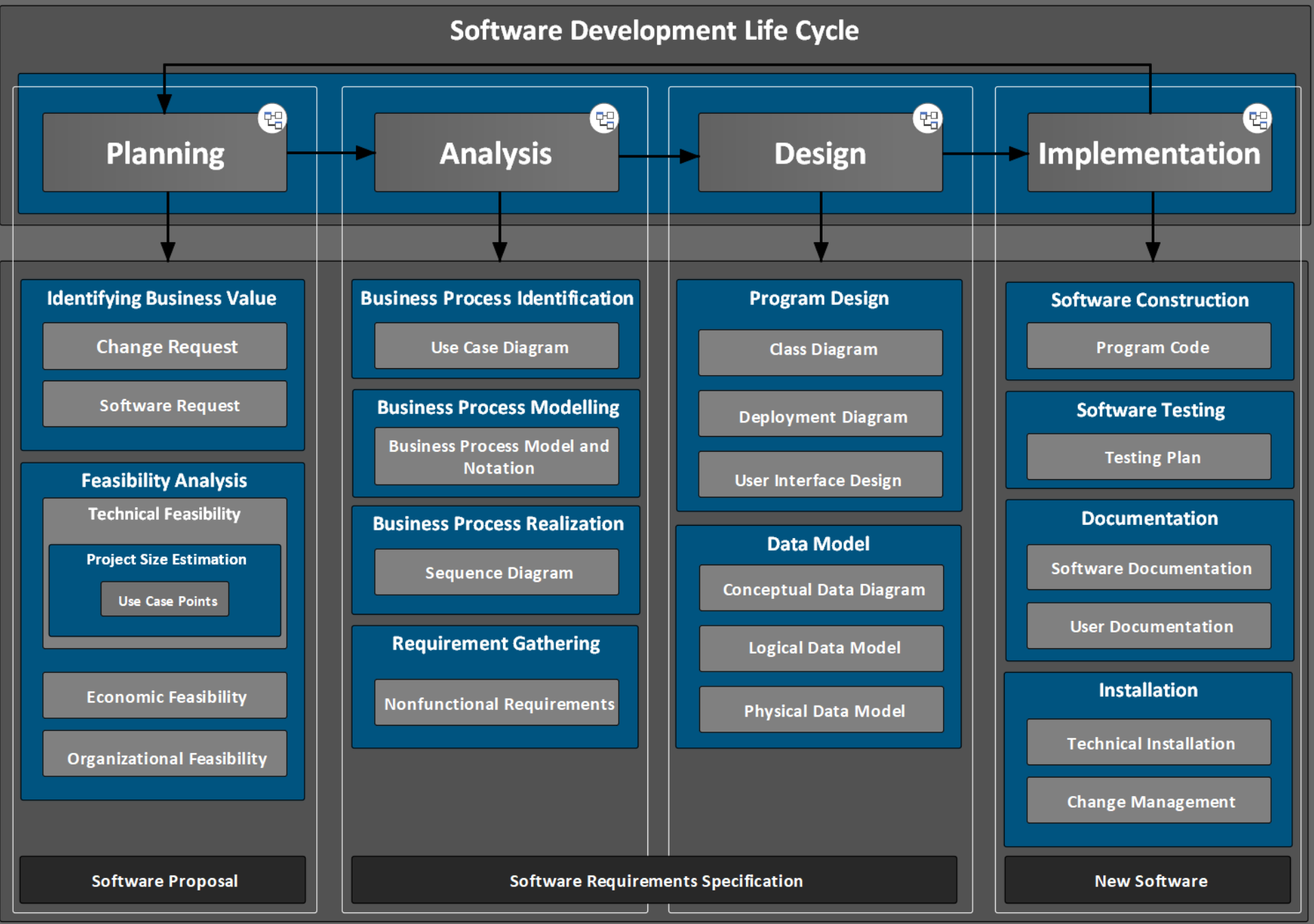
3.4 Realisasi Proses Bisnis dengan Sequence Diagram

UML based Software Analysis and Design

(Wahono, 2009)



Application Development Governance

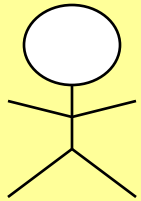


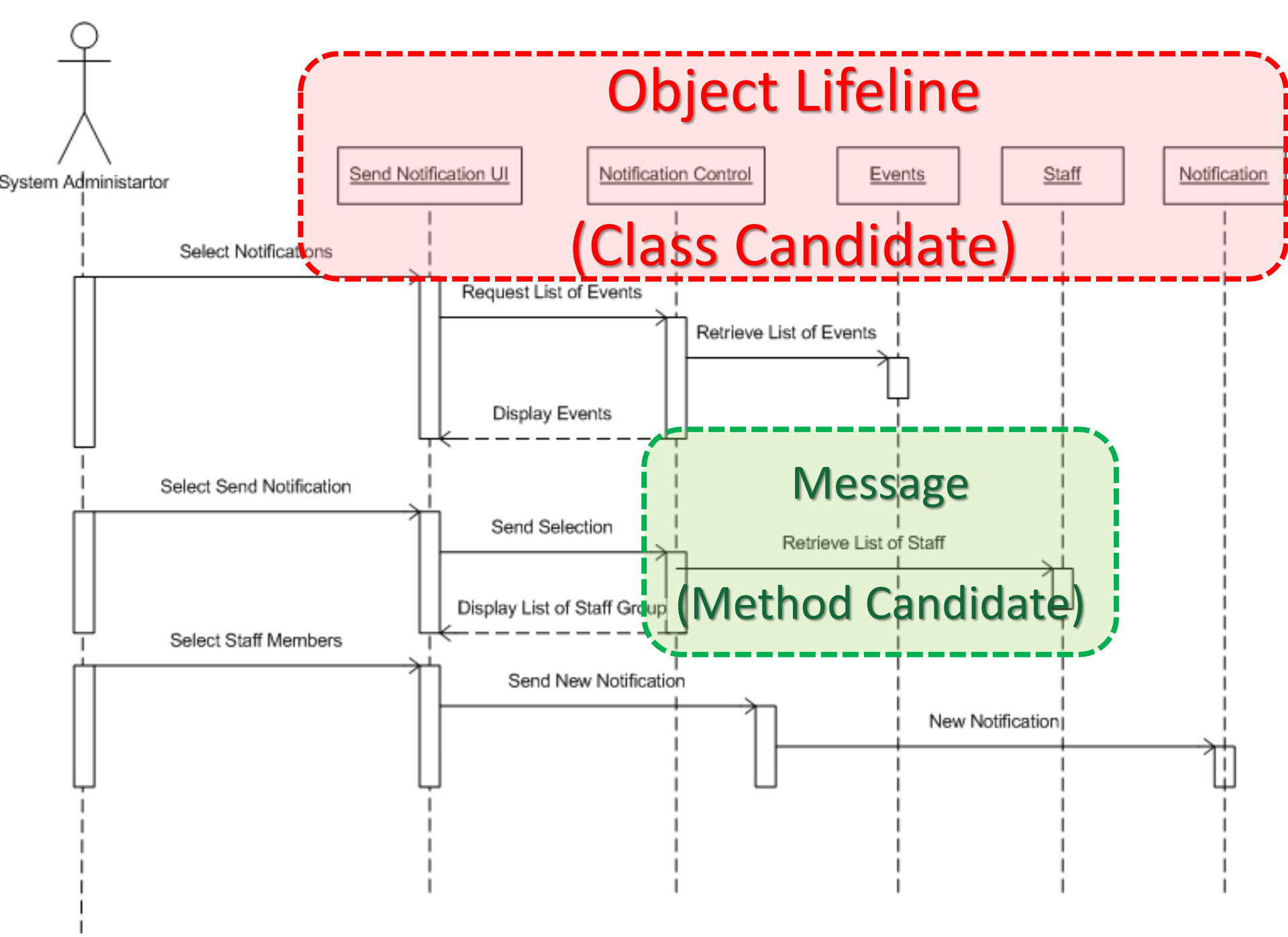
Sequence Diagram

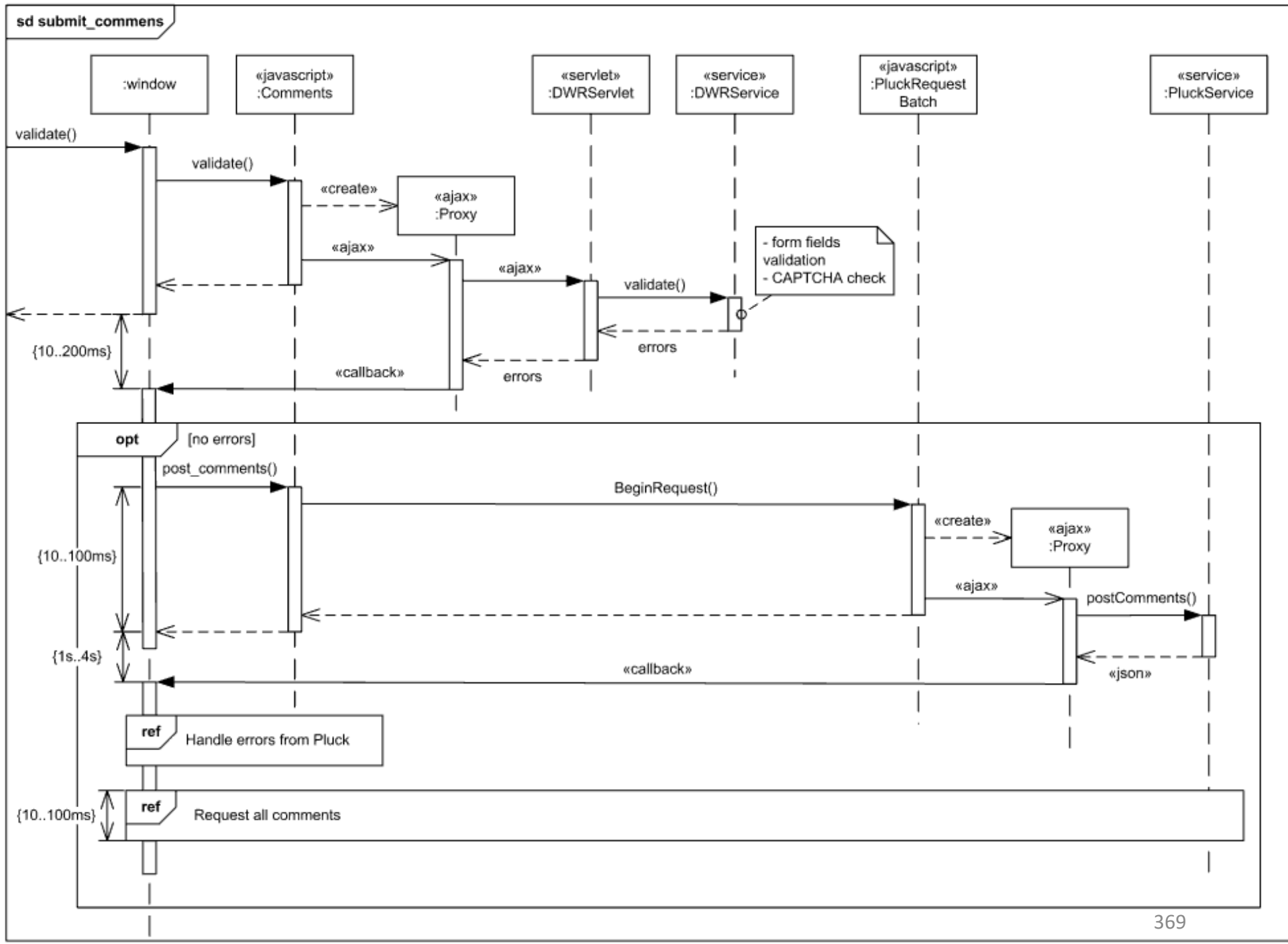
- Sequence diagram menggambarkan interaksi antar *object* (*class*) dalam sofware pada suatu sekuensial waktu
  - Interaksi object (class) berdasarkan alur proses bekerjanya software, dimana interaksi tersebut menggambarkan pesan yang dikirimkan secara sekuensial antara object (class)
  - Garis vertikal (lifeline) menunjukkan object (class), garis horizontal menunjukkan pesan yang mengalir antara object (class) tersebut

- Sequence diagram adalah alat komunikasi System Analyst dengan Programmer, menggambarkan alur proses bekerjanya software sekaligus dengan komposisi software akan seperti apa

Sequence Diagram Syntax

AN ACTOR	
AN OBJECT	anObject:aClass
A LIFELINE	
A FOCUS OF CONTROL	
A MESSAGE	aMessage()
OBJECT DESTRUCTION	X

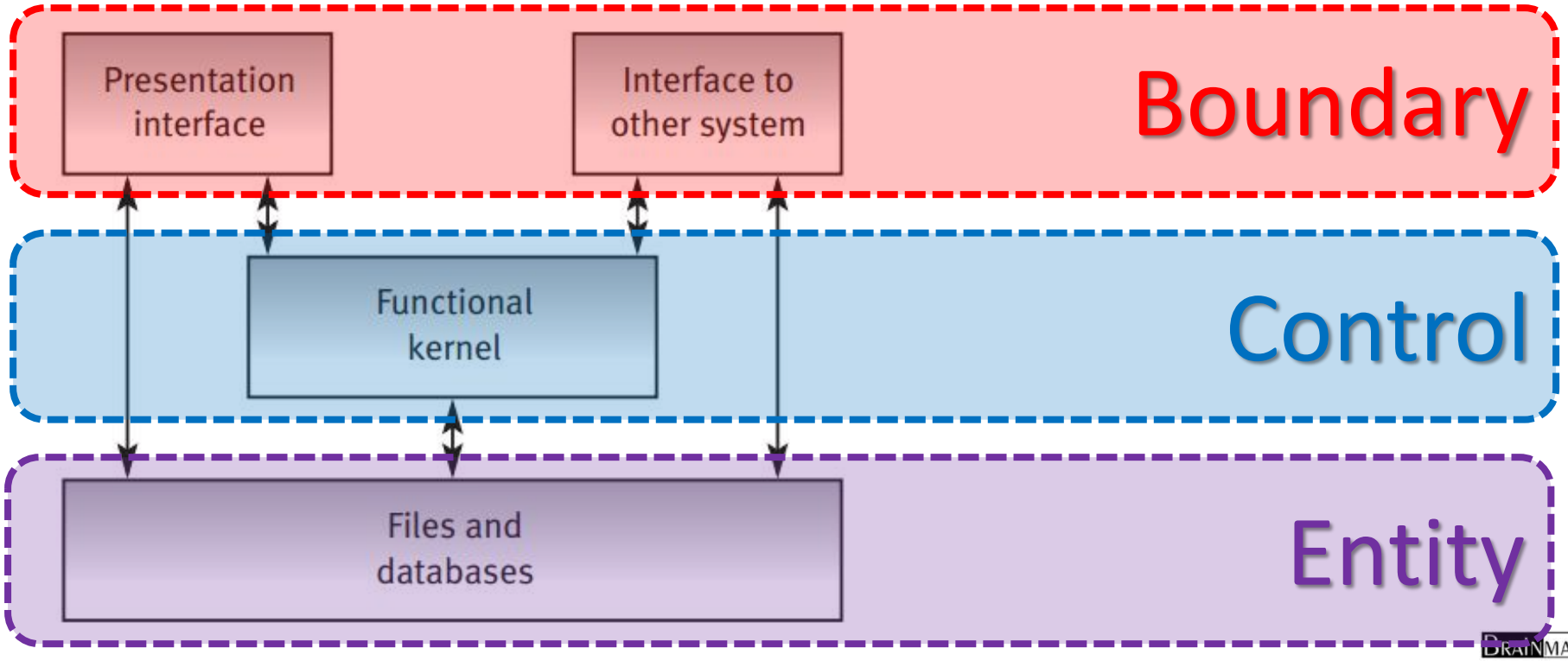

Object Lifeline
(Class Candidate)
System Administartor
Send Notification UI
Notification Control
Events
Staff
Notification
Select Notifications
Request List of Events
Retrieve List of Events
Display Events
Message
Select Send Notification
Send Selection
Retrieve List of Staff
Display List of Staff Group
(Method Candidate)
Select Staff Members
Send New Notification
New Notification



Denert's Law (1991)

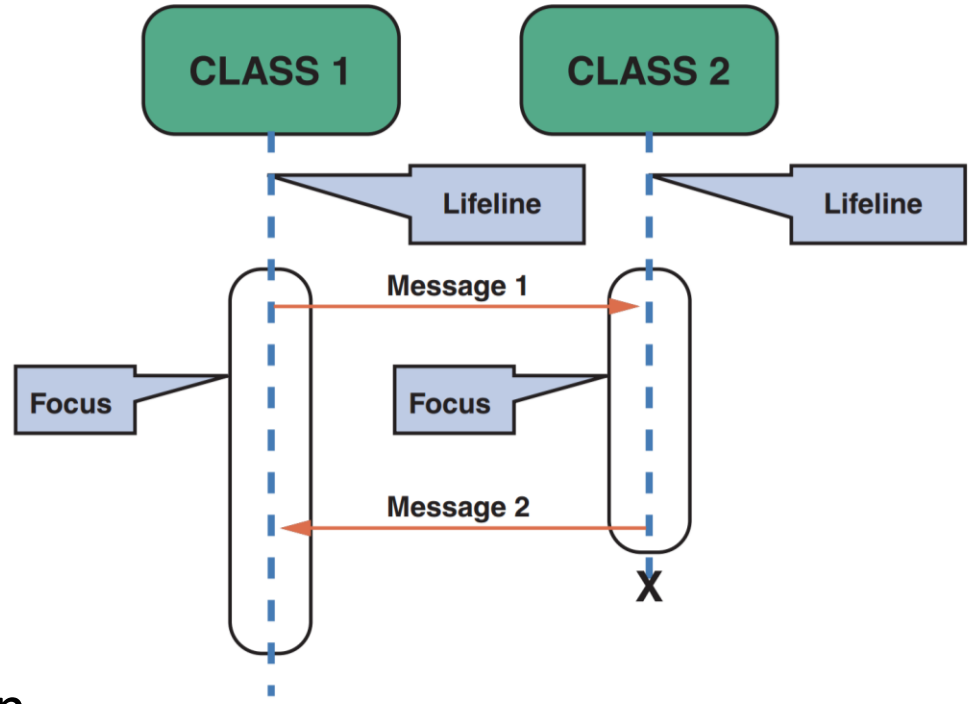
Separation of concerns leads to standard architectures

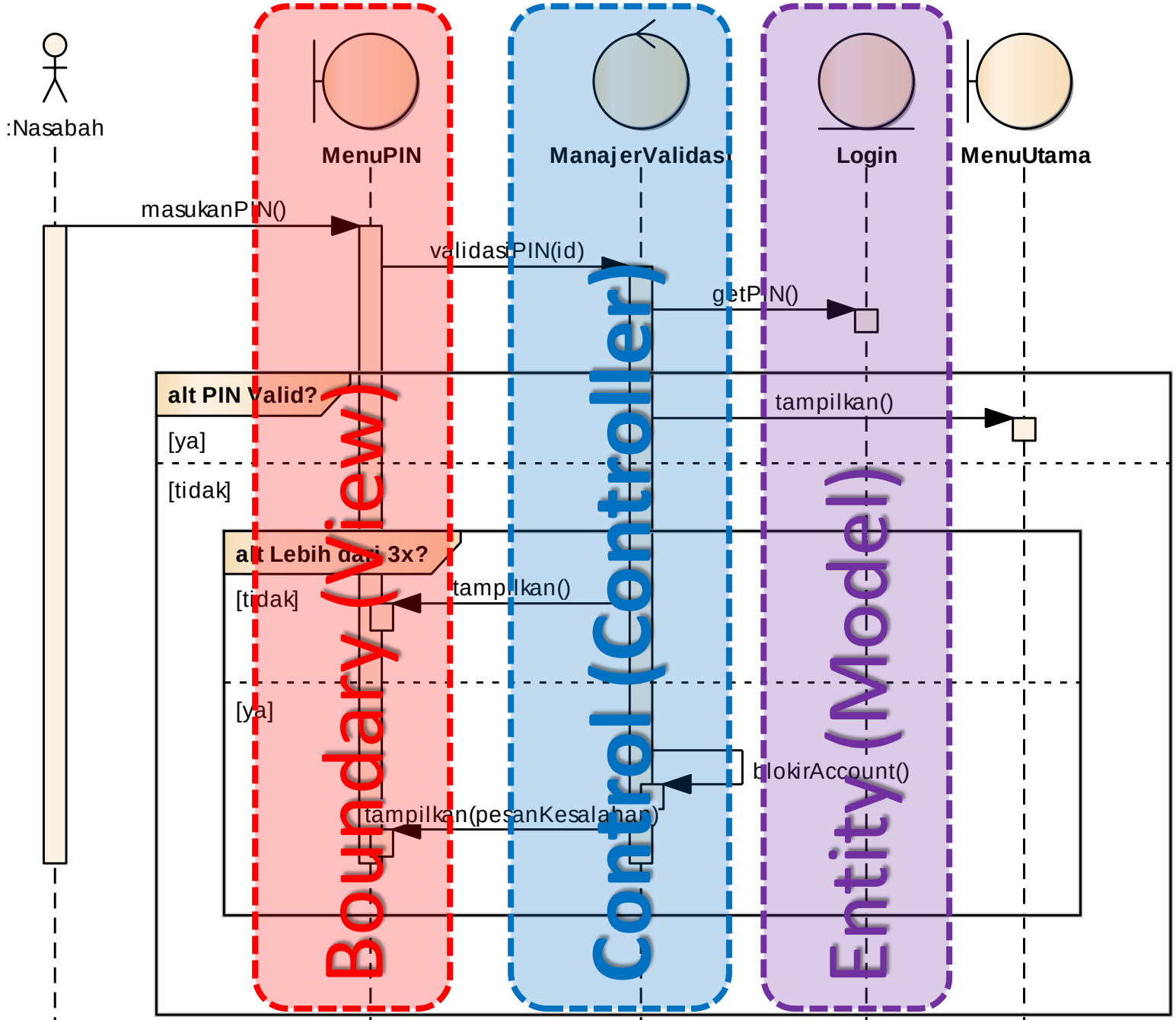
(Endres, 2003) *[L9]*

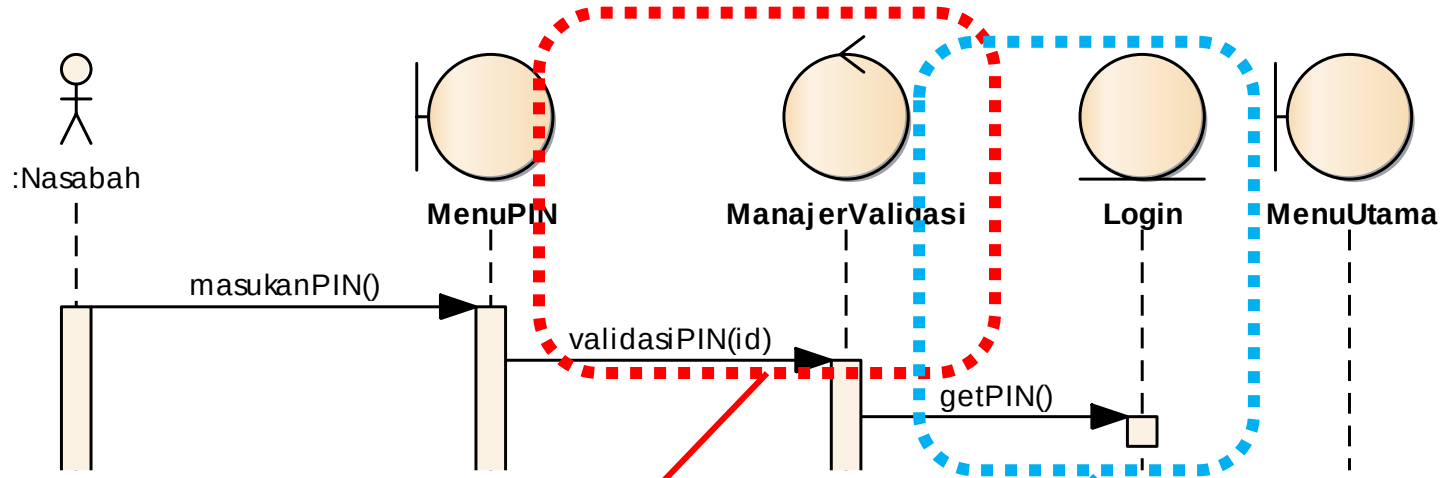


Sequence Diagram berbasis Arsitektur Boundary – Control – Entity

1. Boundary Class:
   - Class yang berinteraksi dengan aktor langsung (user interface)
   - Form, input, UI, dsb
2. Control Class:
   - Class yang berhubungan dengan pemrosesan, penghitungan, kalkulasi, komputasi, query, dst
3. Entity Class:
   - Class yang berhubungan dengan data, penyimpanan data/file




:Nasabah
MenuPIN
ManajerValidasi
Login
MenuUtama
masukanPIN()
validasiPIN(id)
getPIN()
alt PIN Valid?
[ya]
[tidak]
tampilkan()
alt Lebih dari 3x?
[tidak]
tampilkan()
[ya]
blokirAccount()
tampilkan(pesanKesalahan)
Boundary (View)
Control (Controller)
Entity (Model)




```
public class ManajerValidasi {

        private Login m_Login;

        public int validasiKartu(){

        }

        public int validasiPIN(){
            int pin = m_Login.getPIN();
            if(pin.equals(pinuser)){
                        MenuUtama.show()};
        }

        public void blokirKartu(){

        }
}
```

```
public class Login {

        public int getPIN(){

        }

}
```

Sequence Diagram

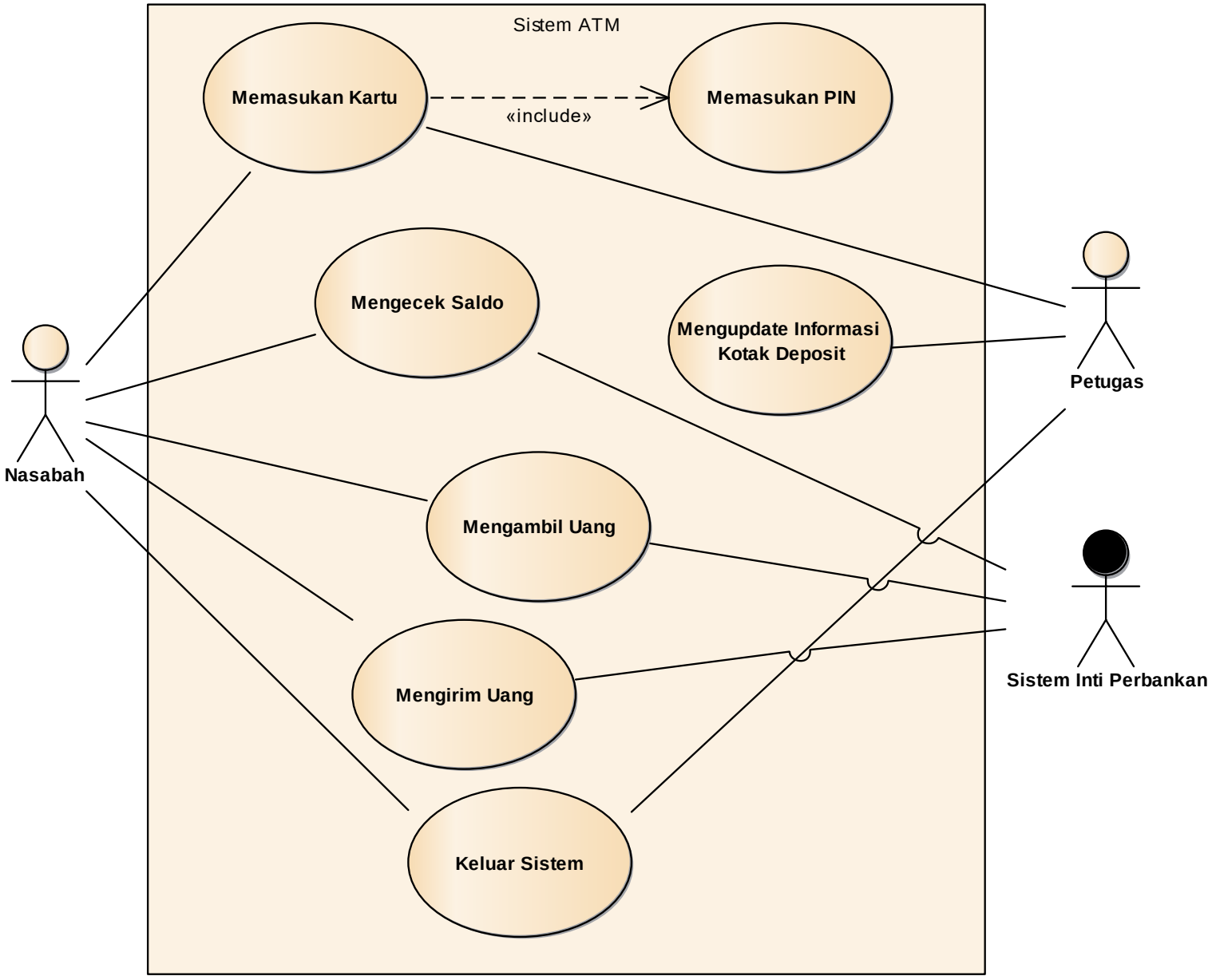
- Sequence Diagram dibuat untuk setiap Use Case yang dibuat

- Dimulai dari menarik Actor yang ada di Use Case Diagram, dilanjutkan dengan membuat sequence detail dari alur proses berjalannya Use Case dengan message yang mengalir didalamnya

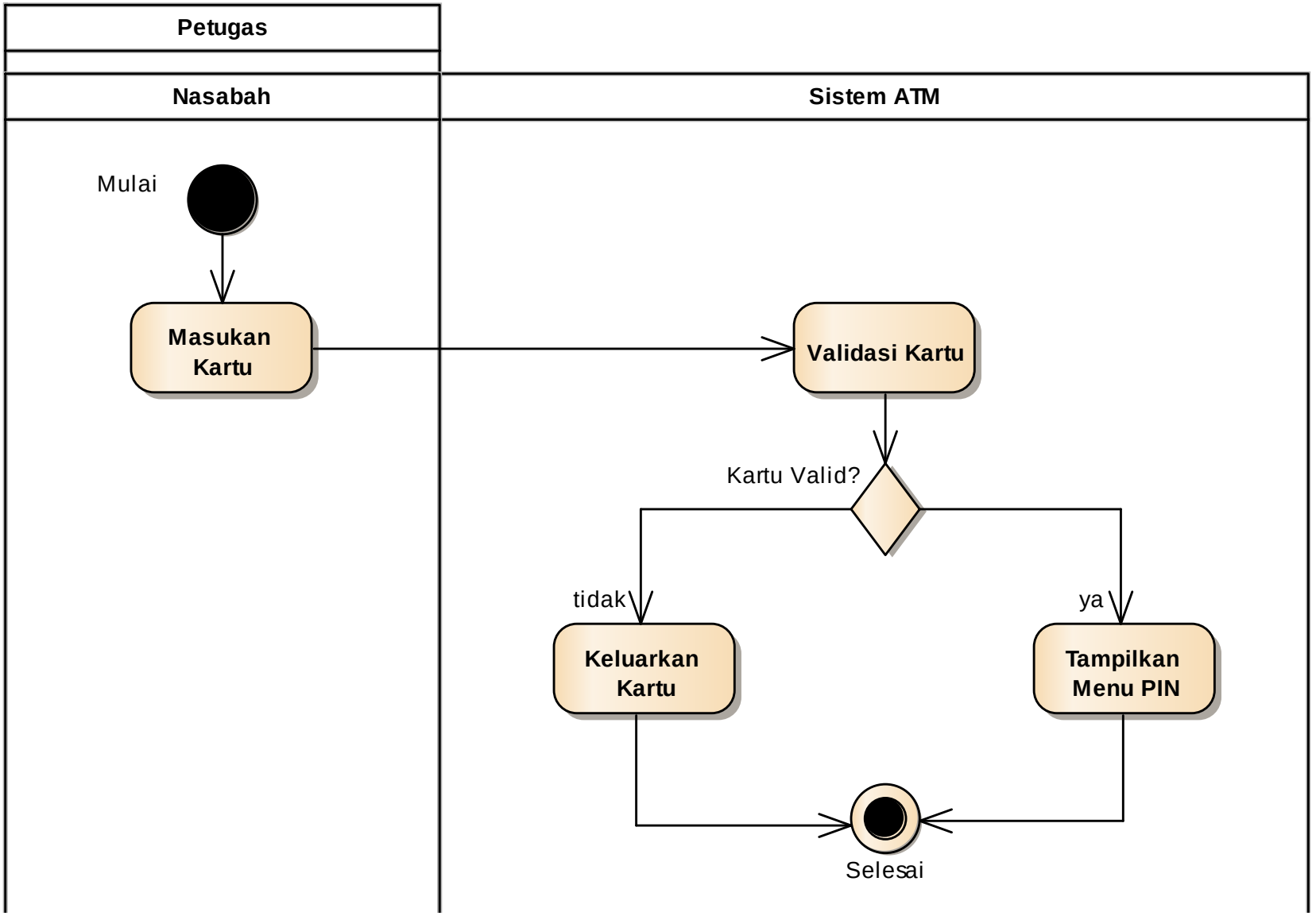
- Catatan: Objek dari Lifeline di Sequence Diagram akan menjadi kandidat Class, karena itulah harus mengikuti arsitektur Boundary – Control – Class

Studi Kasus: Sequence Diagram Sistem ATM

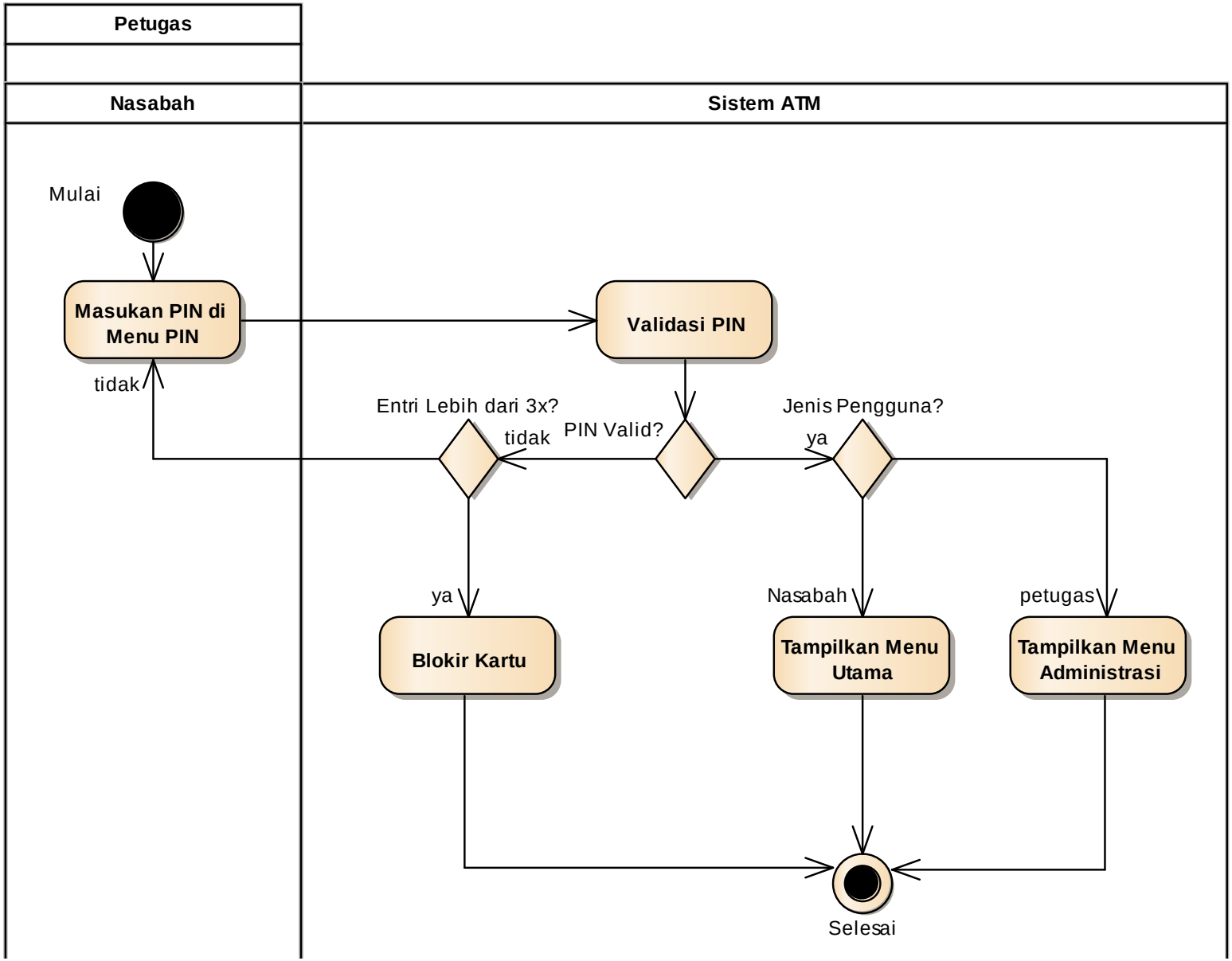
Use Case Diagram Sistem ATM



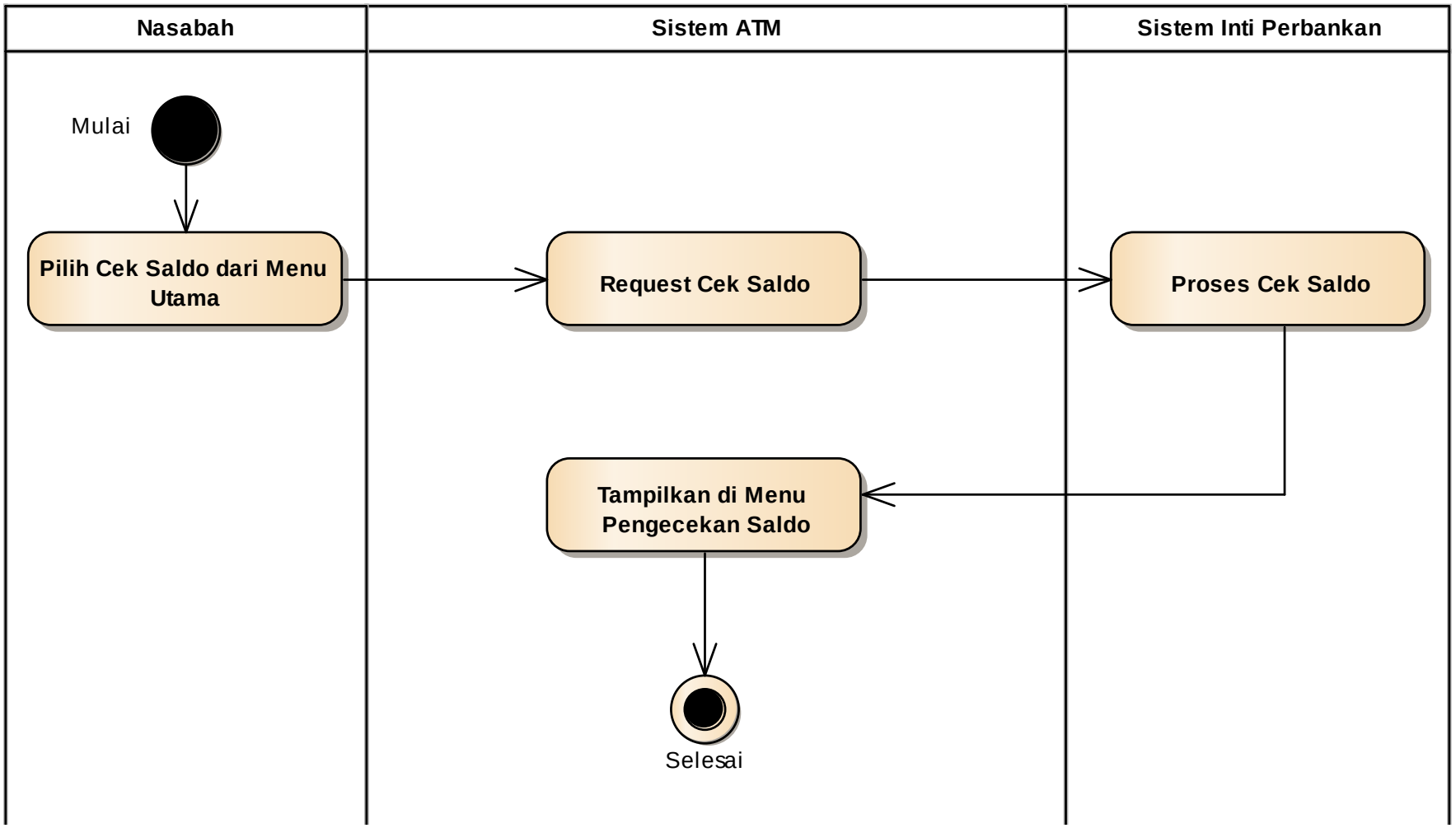
Activity Diagram: Memasukkan Kartu



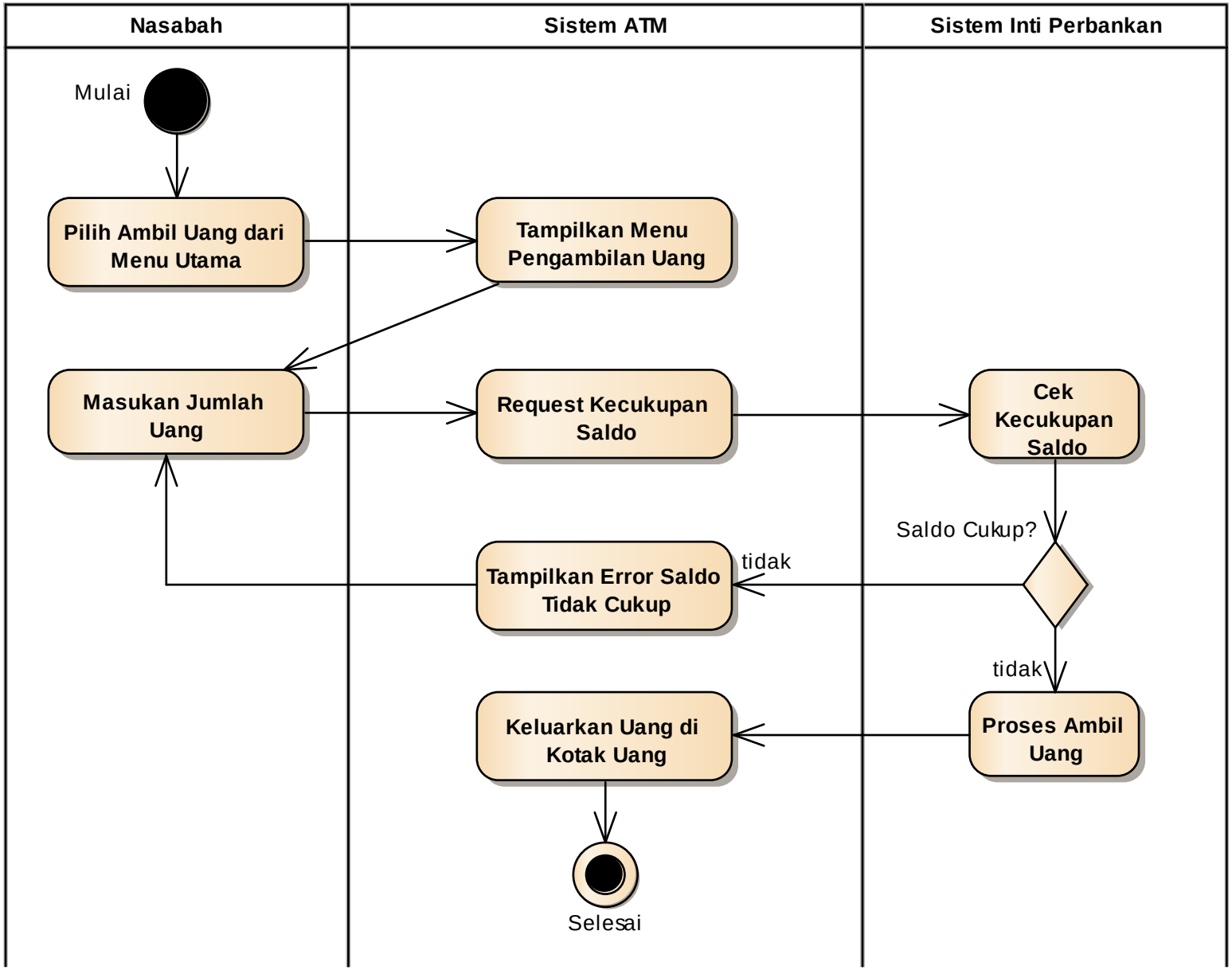
Activity Diagram: Memasukkan PIN



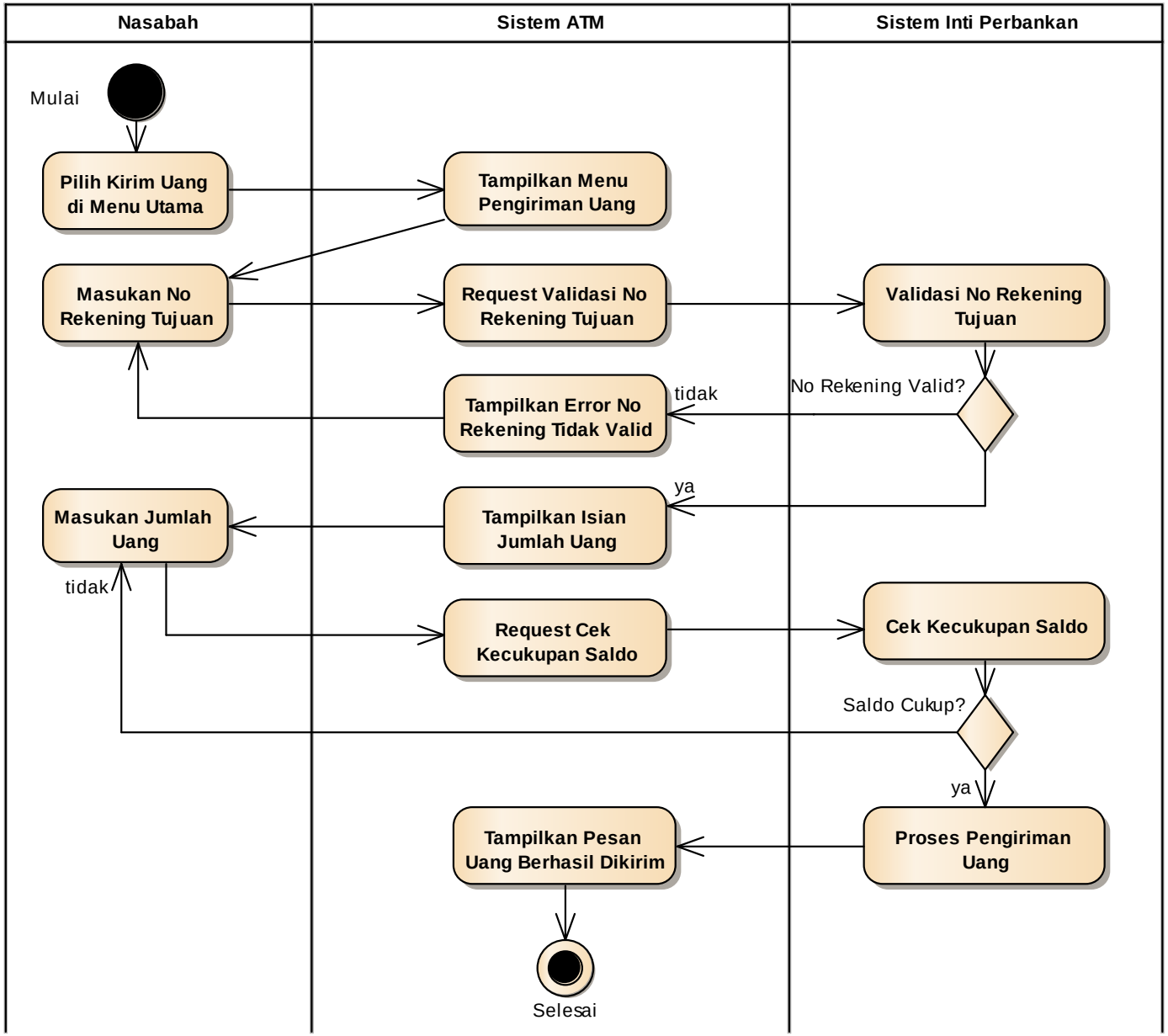
Activity Diagram: Mengecek Saldo



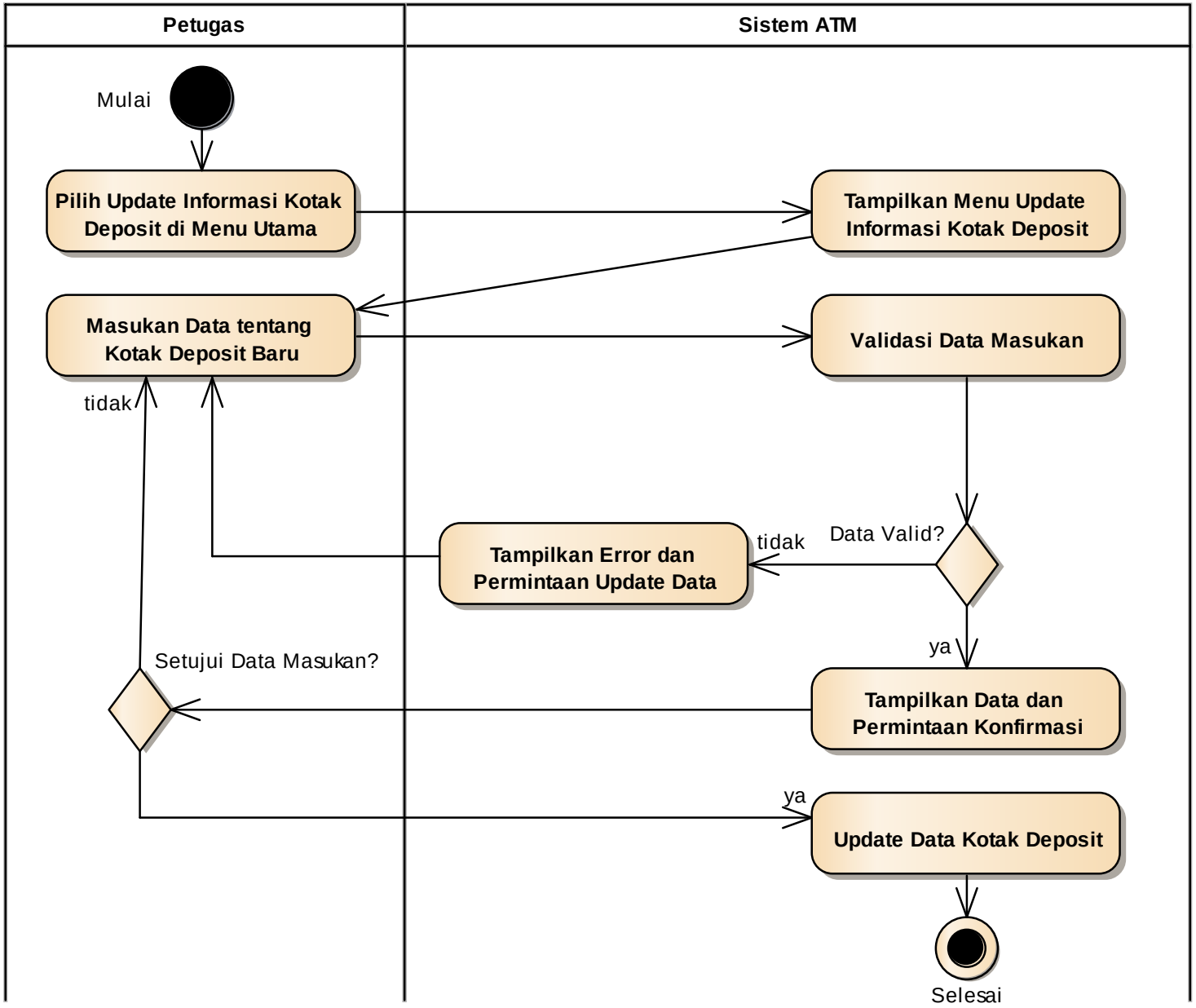
Activity Diagram: Mengambil Uang



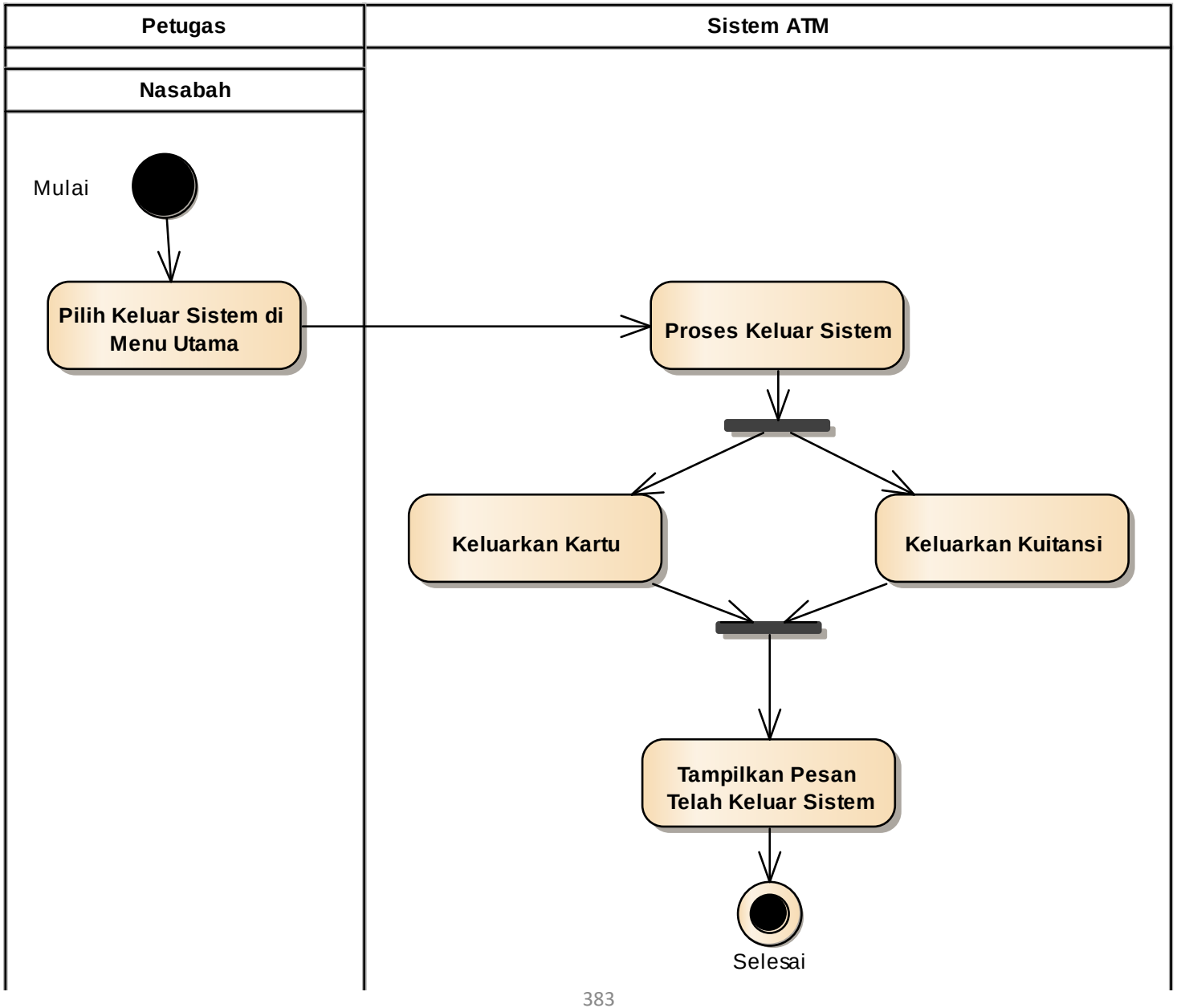
Activity Diagram: Mengirim Uang



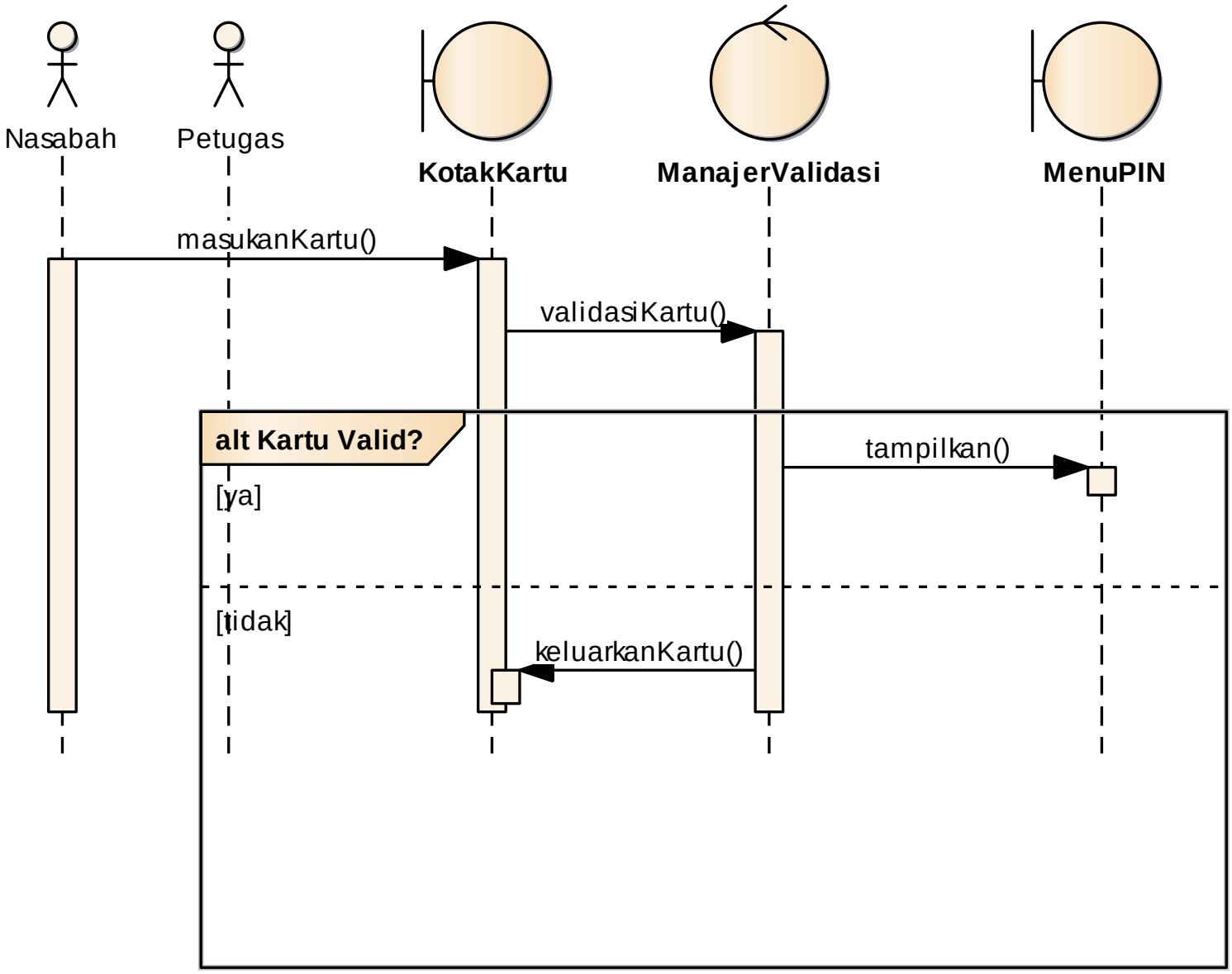
Activity Diagram: Mengupdate Informasi Kotak Deposit



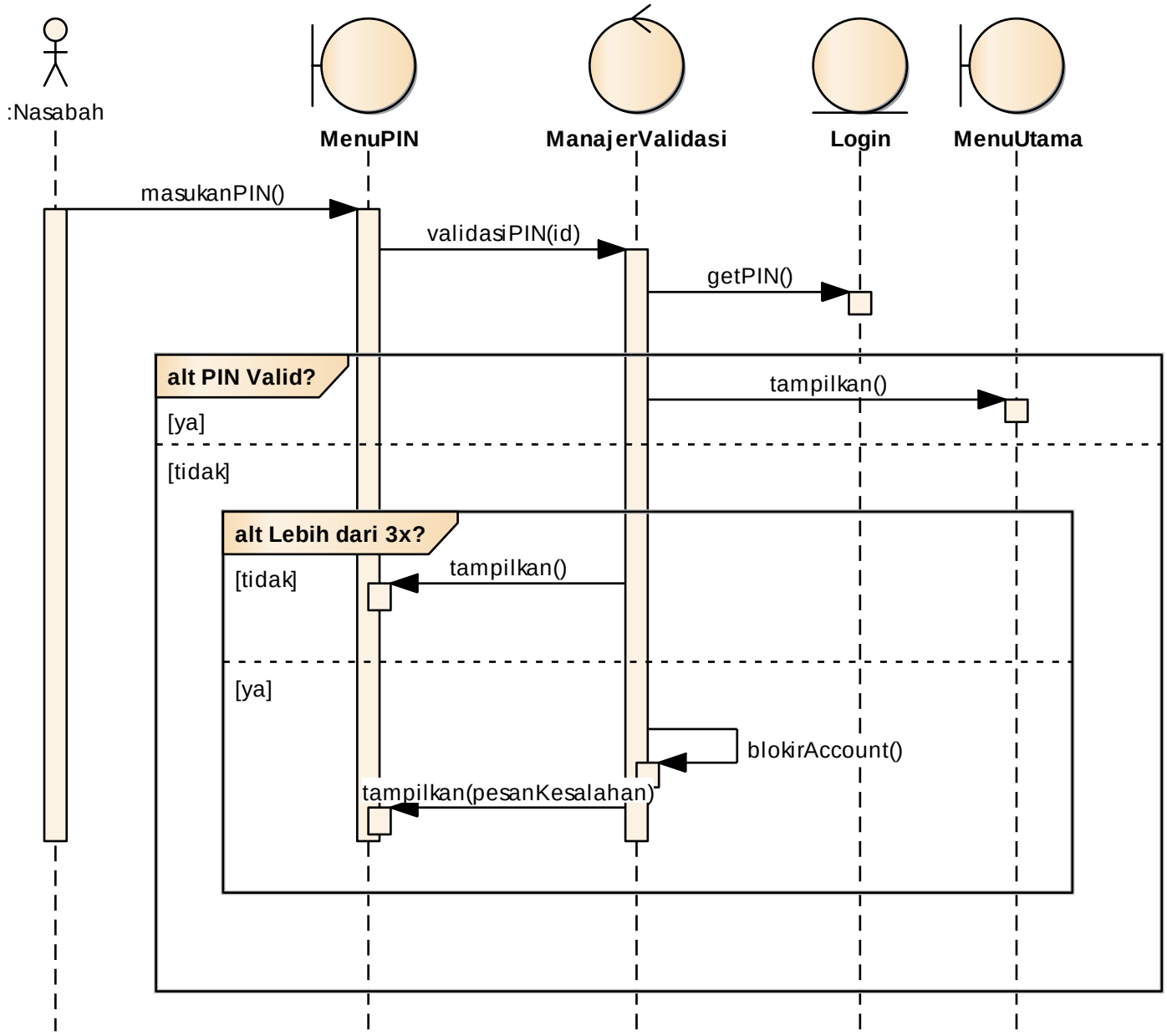
Activity Diagram: Keluar Sistem



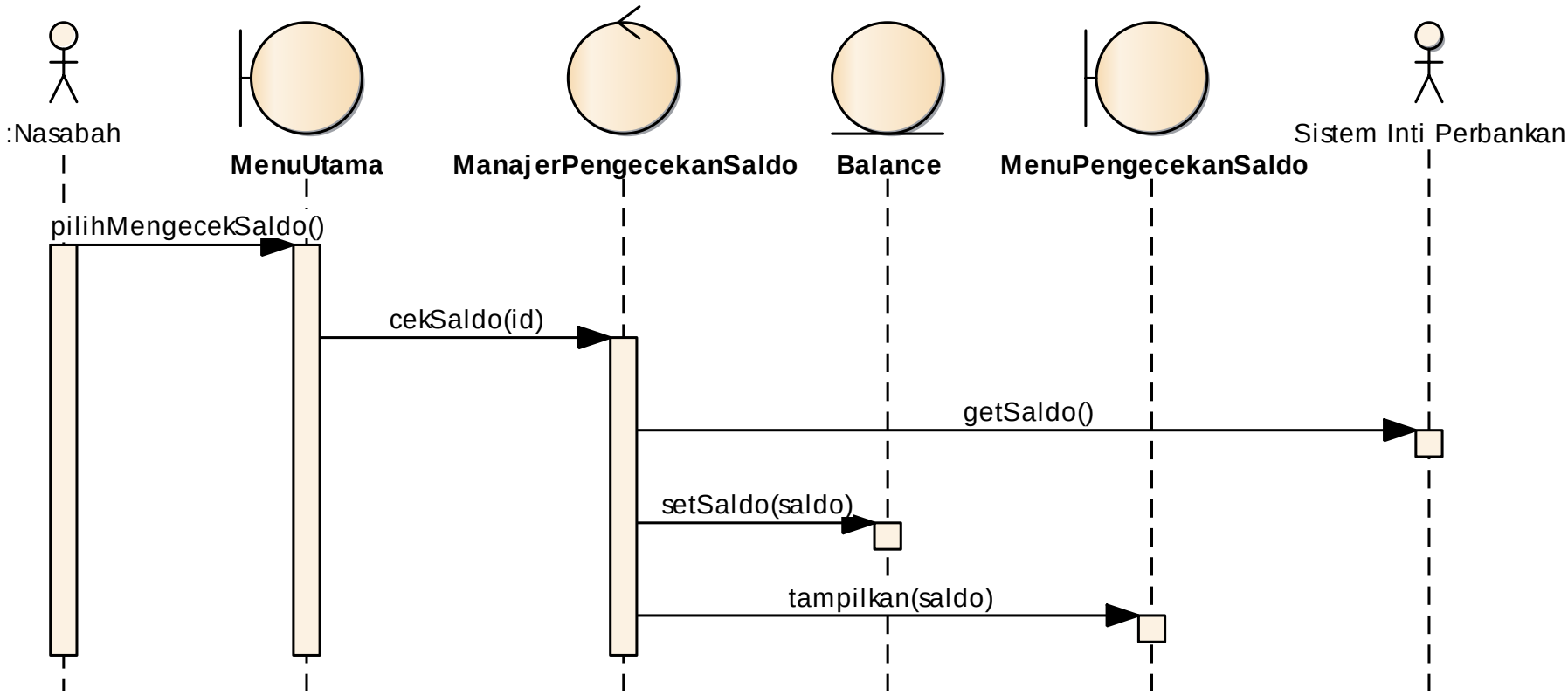
Sequence Diagram: Memasukkan Kartu



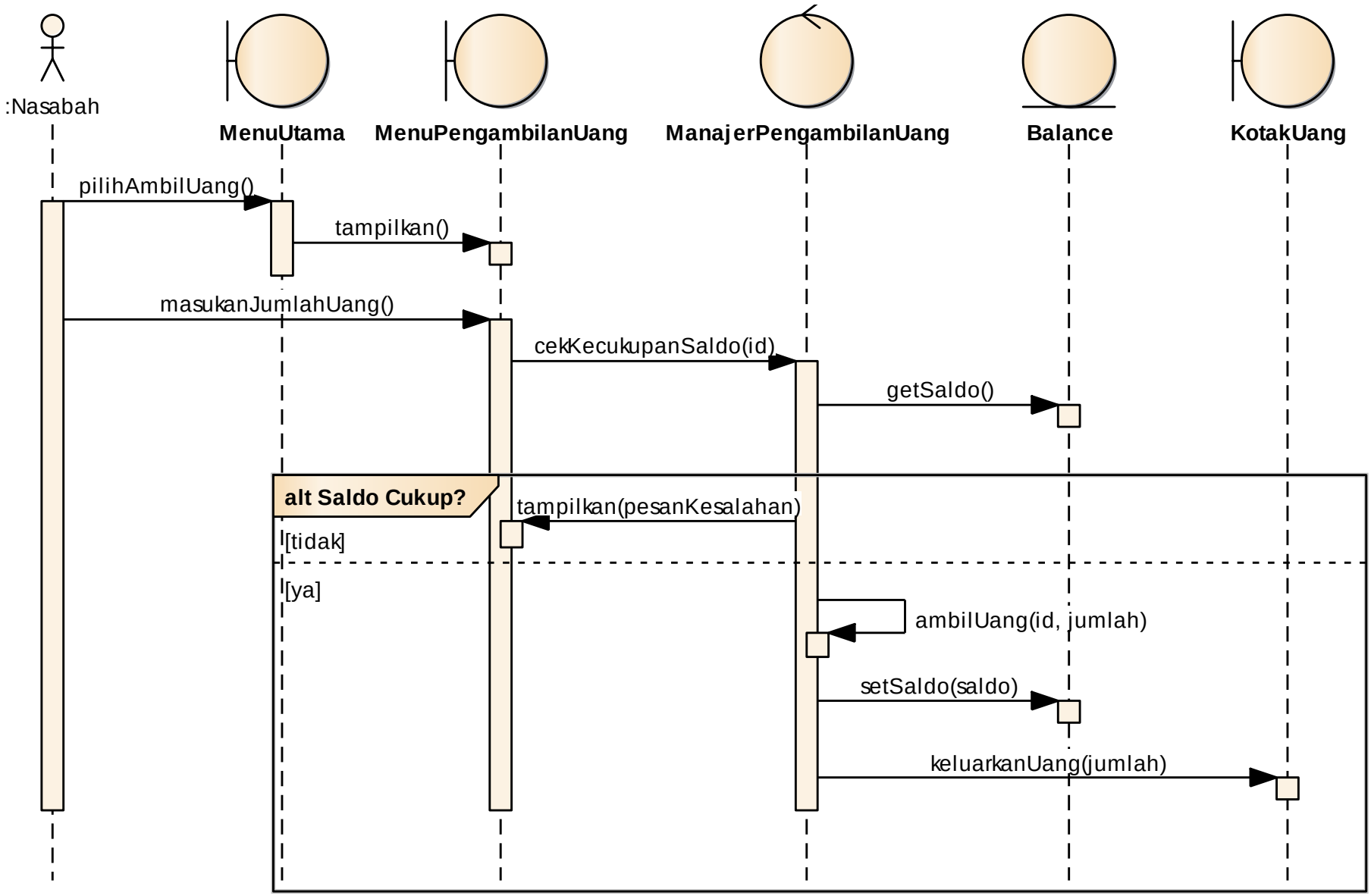
Sequence Diagram: Memasukkan PIN



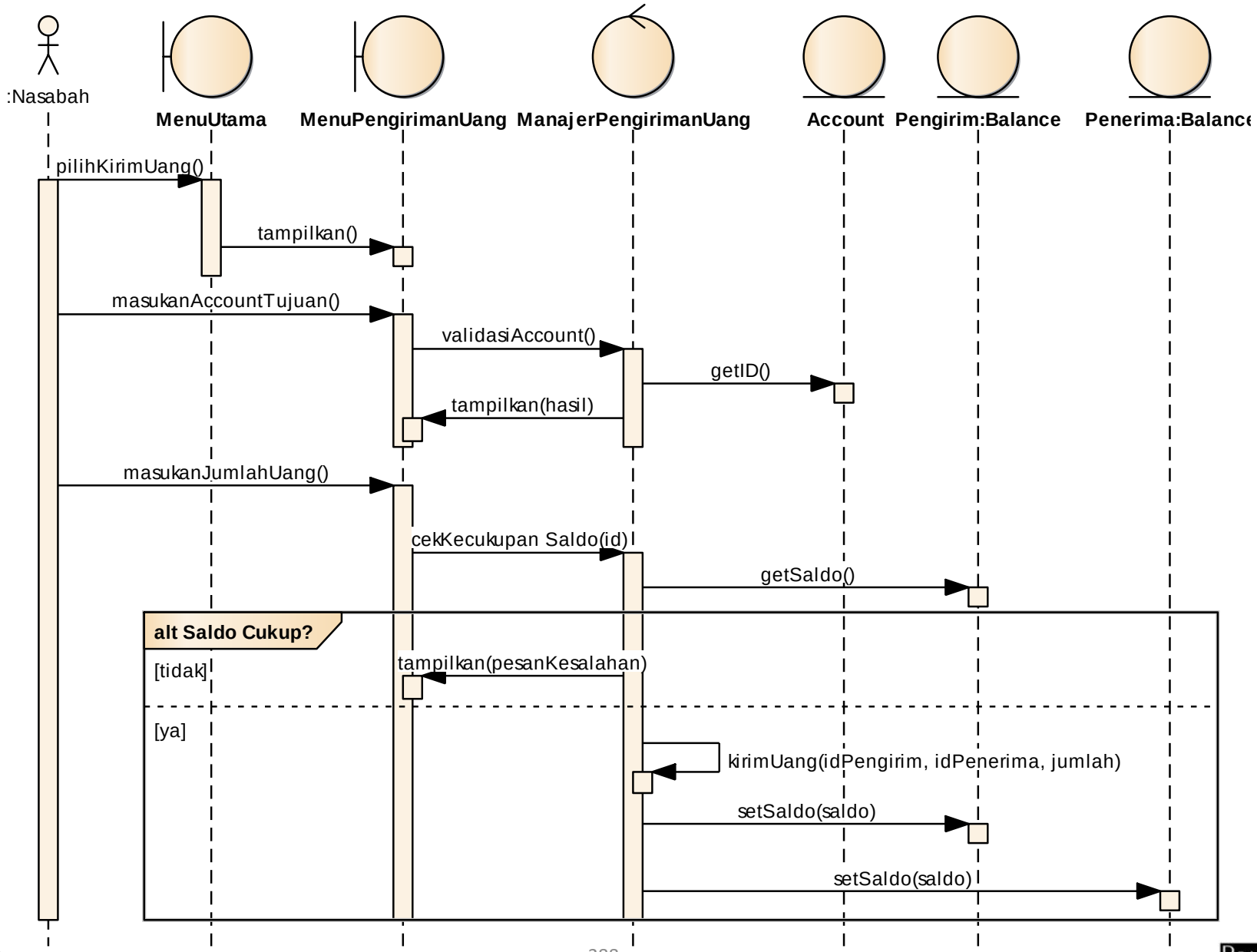
Sequence Diagram: Mengecek Saldo



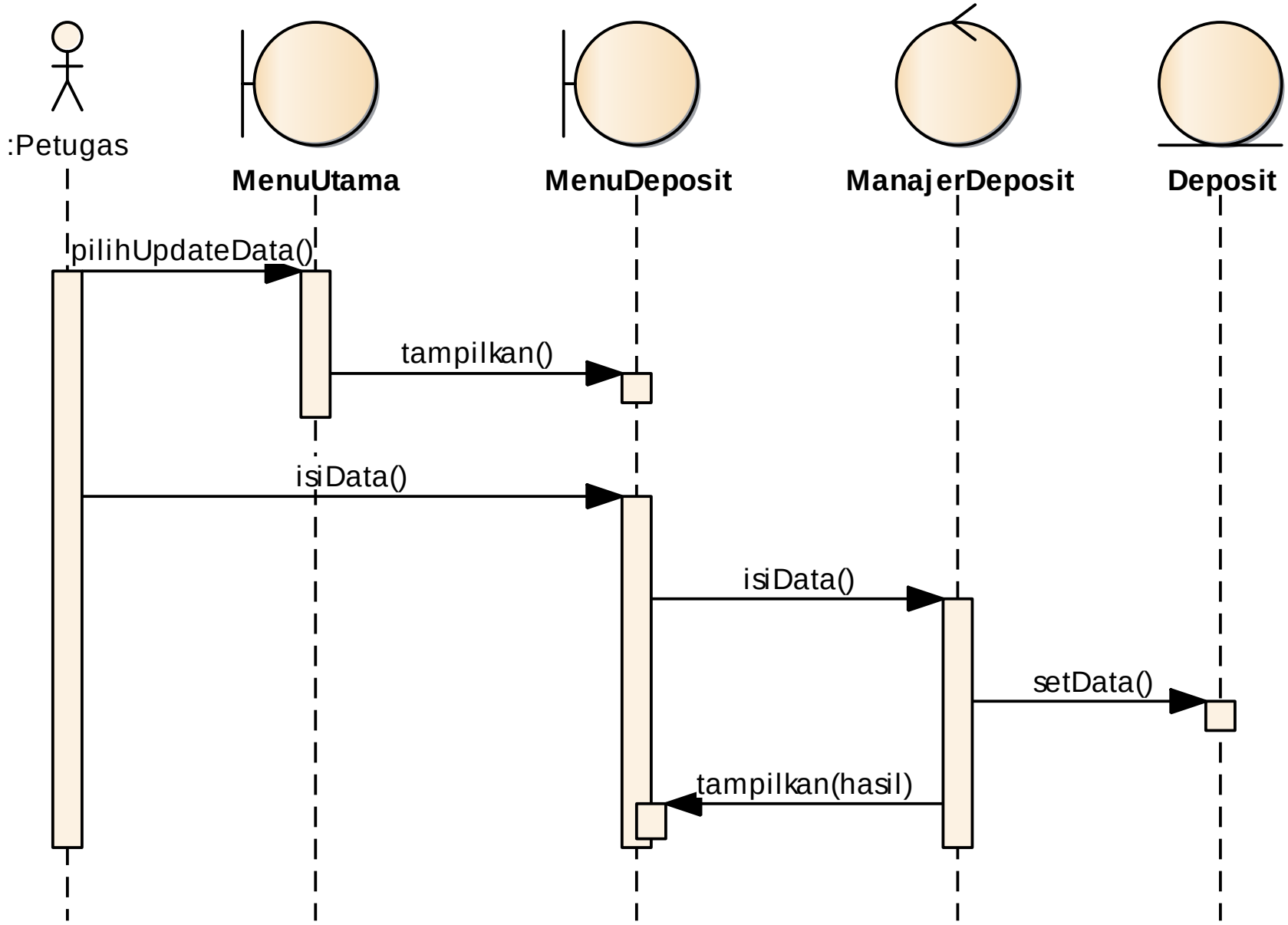
Sequence Diagram: Mengambil Uang



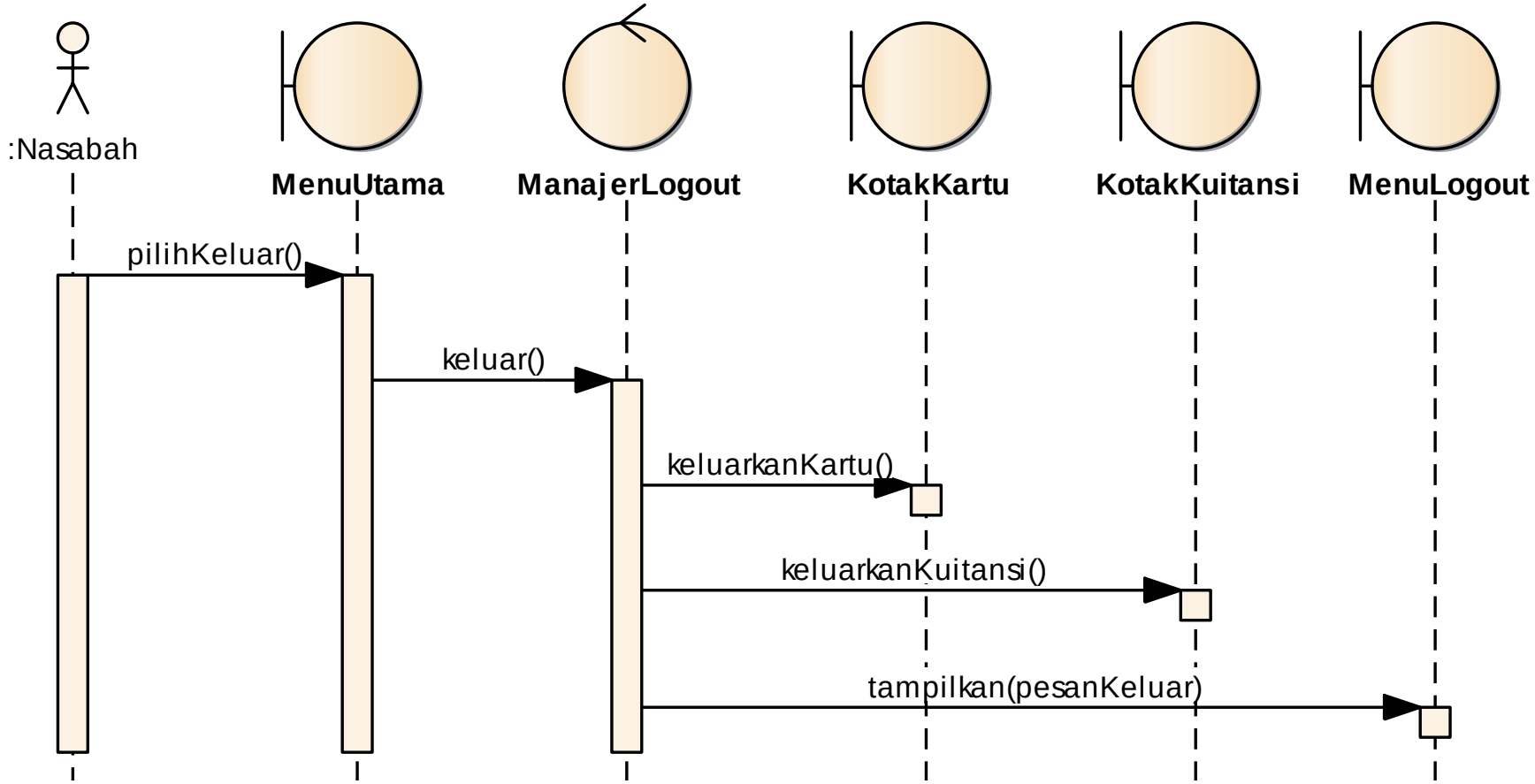
Sequence Diagram: Mengirim Uang



Sequence Diagram: Mengupdate Informasi Kotak Deposit

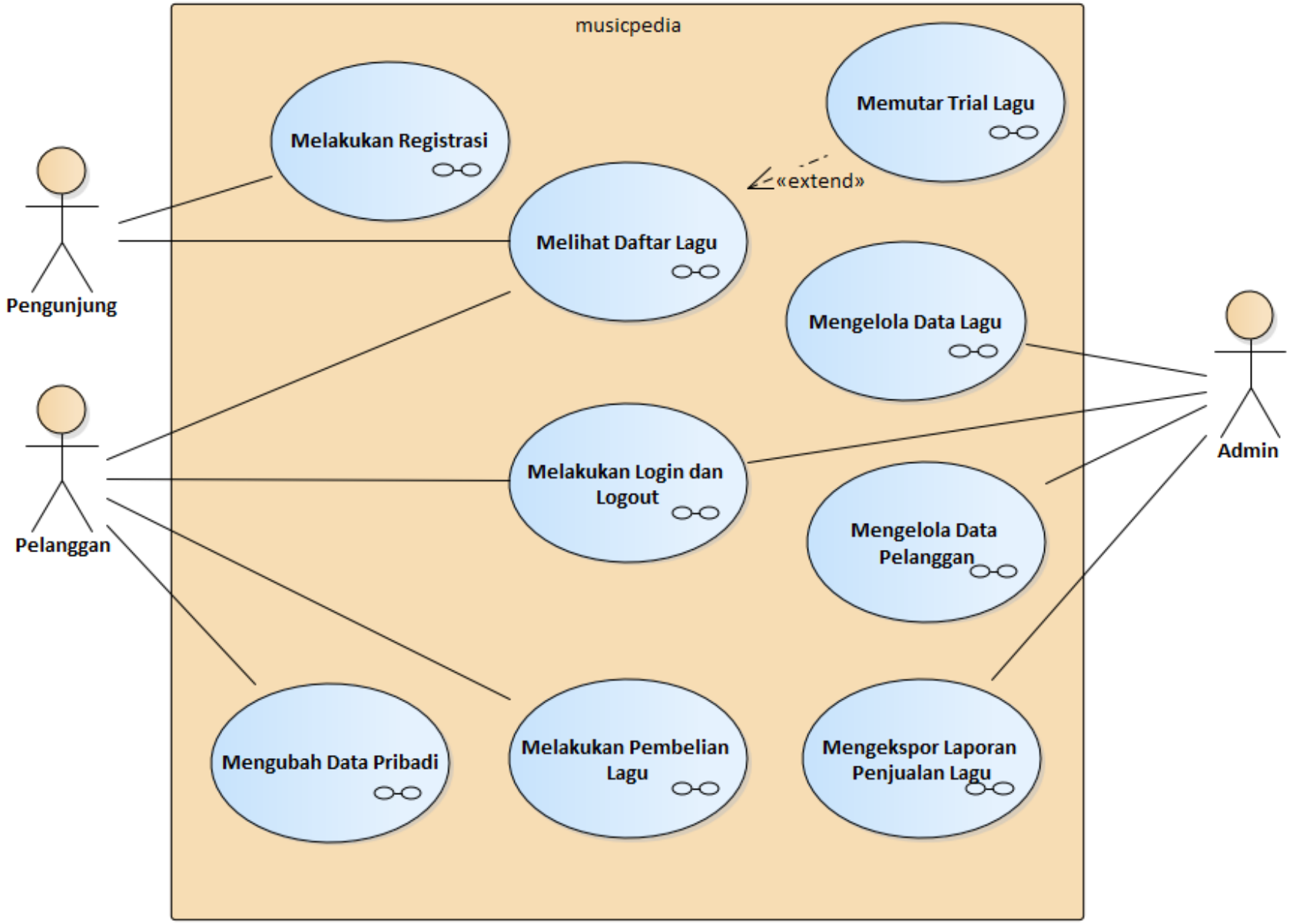


Sequence Diagram: Keluar Sistem

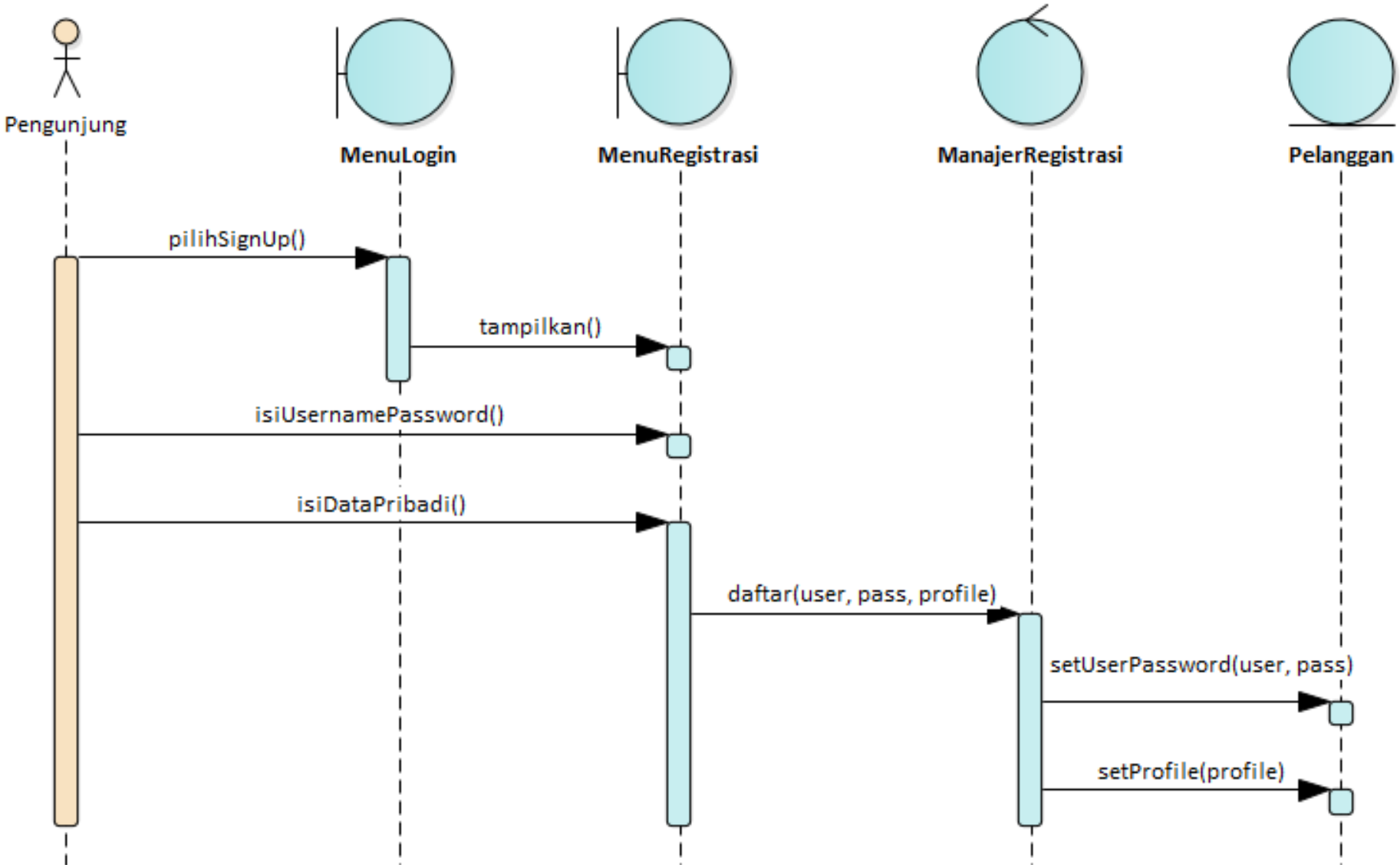


Studi Kasus: Sequence Diagram MusicPedia

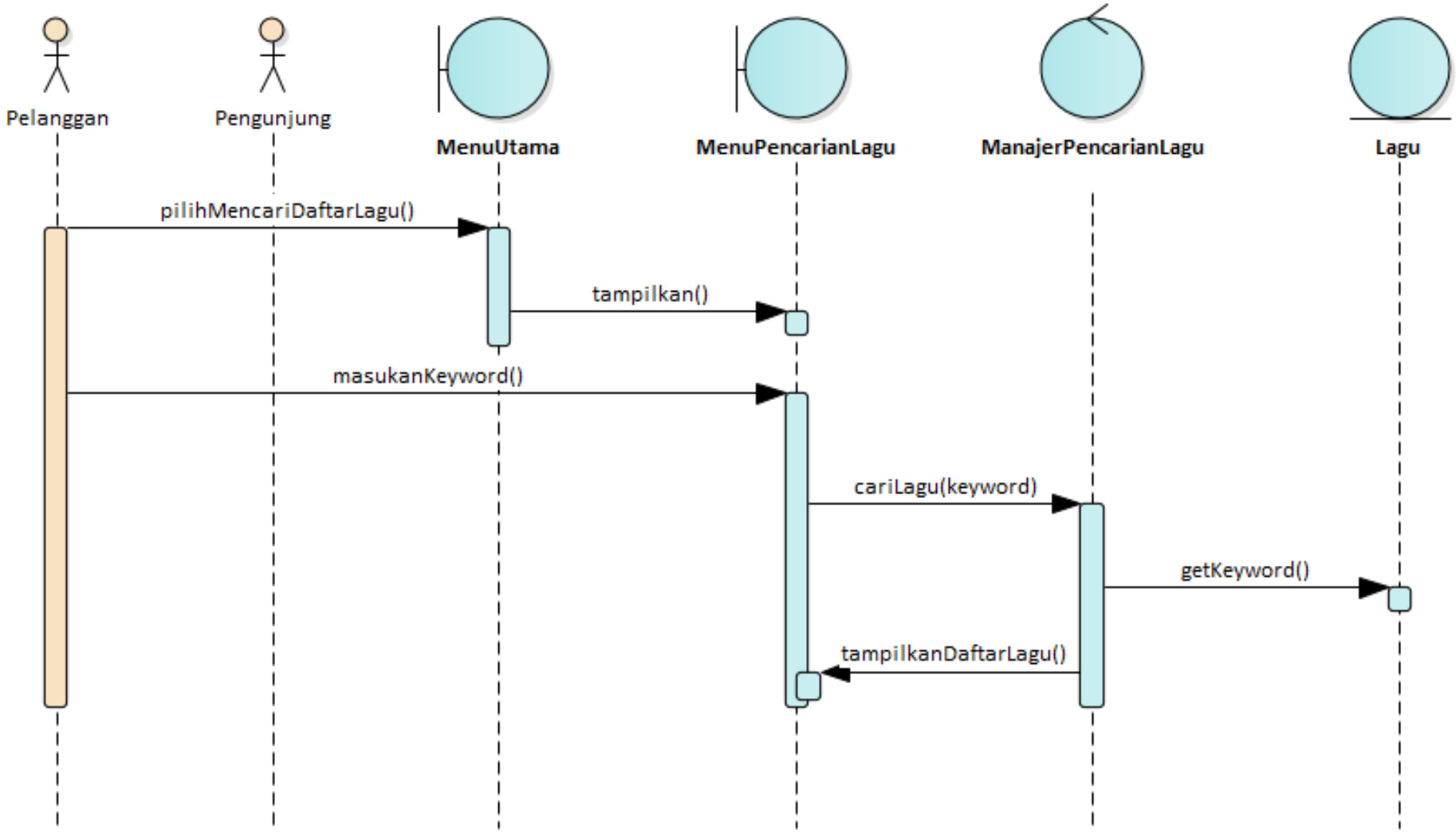
Use Case Diagram MusicPedia



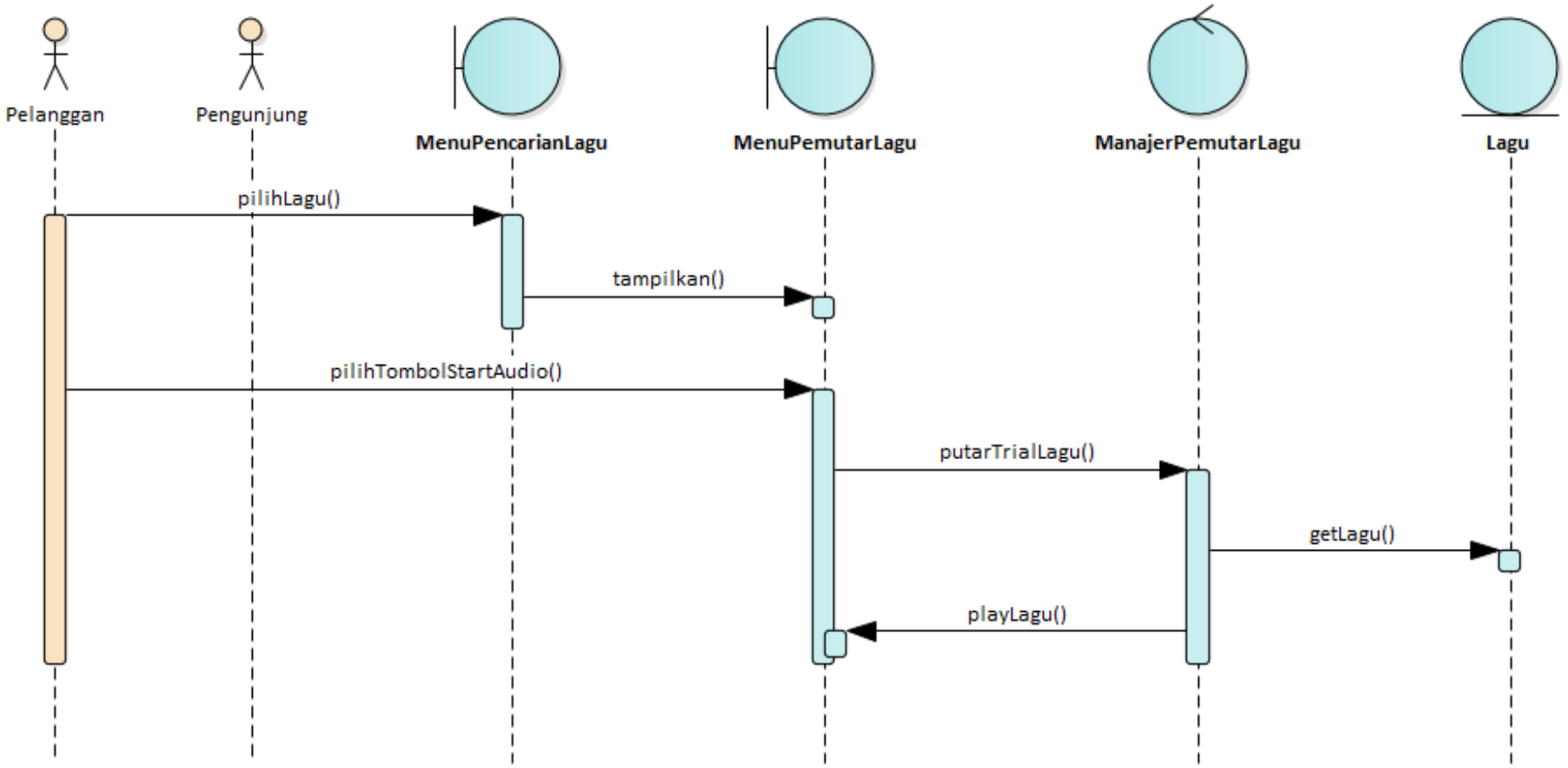
Sequence Diagram: Melakukan Registrasi



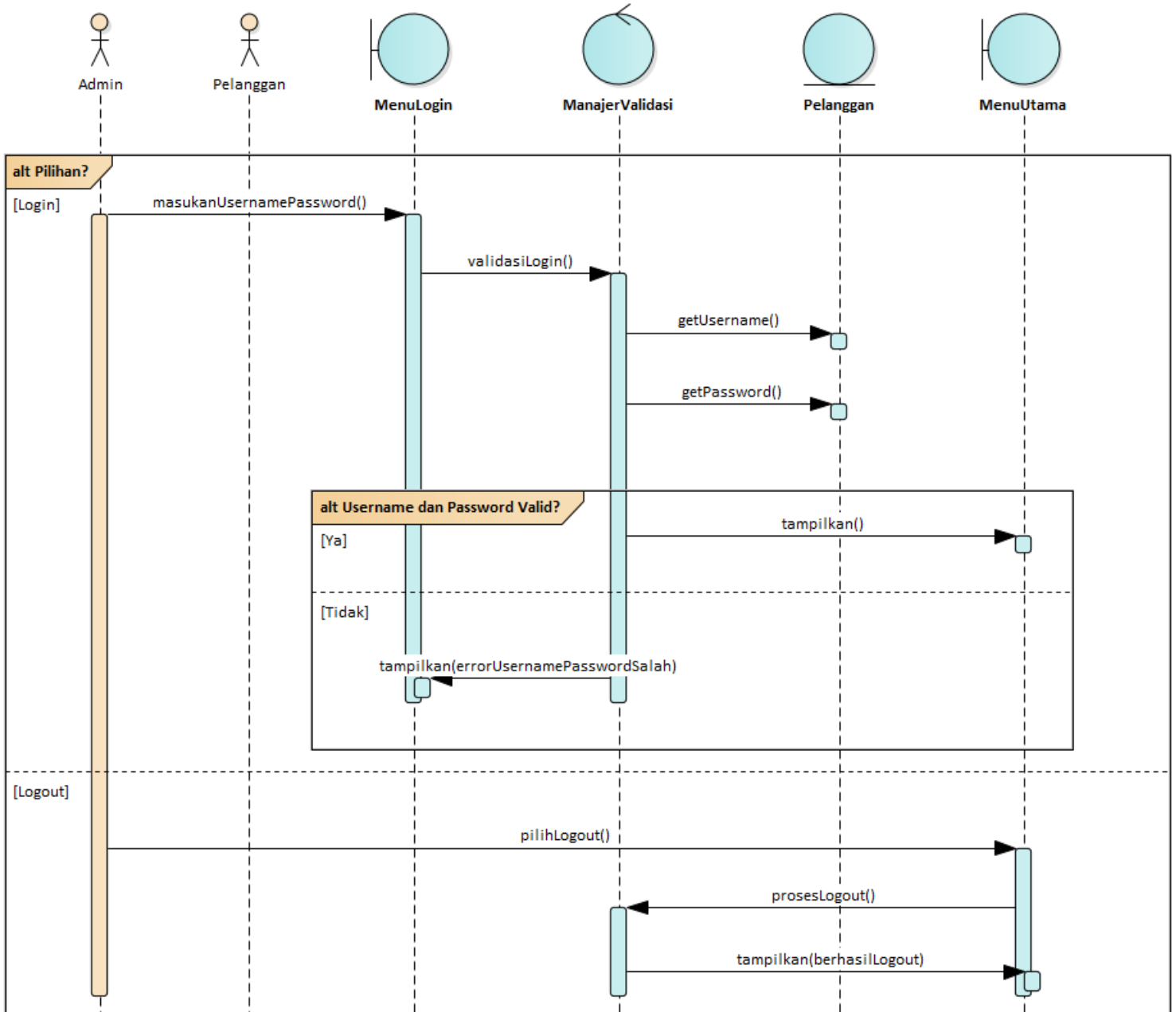
Sequence Diagram: Melihat Daftar Lagu



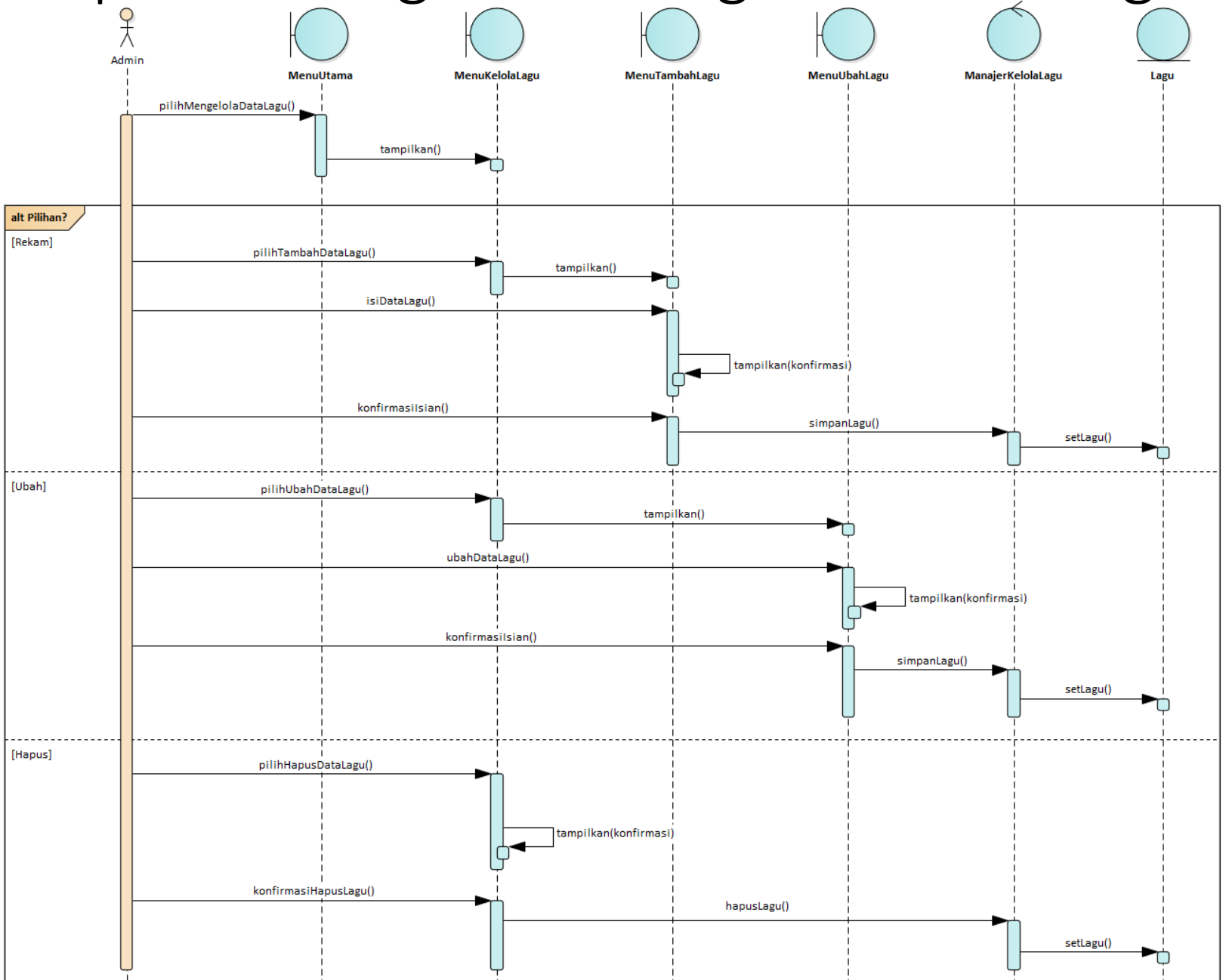
Sequence Diagram: Memutar Trial Lagu



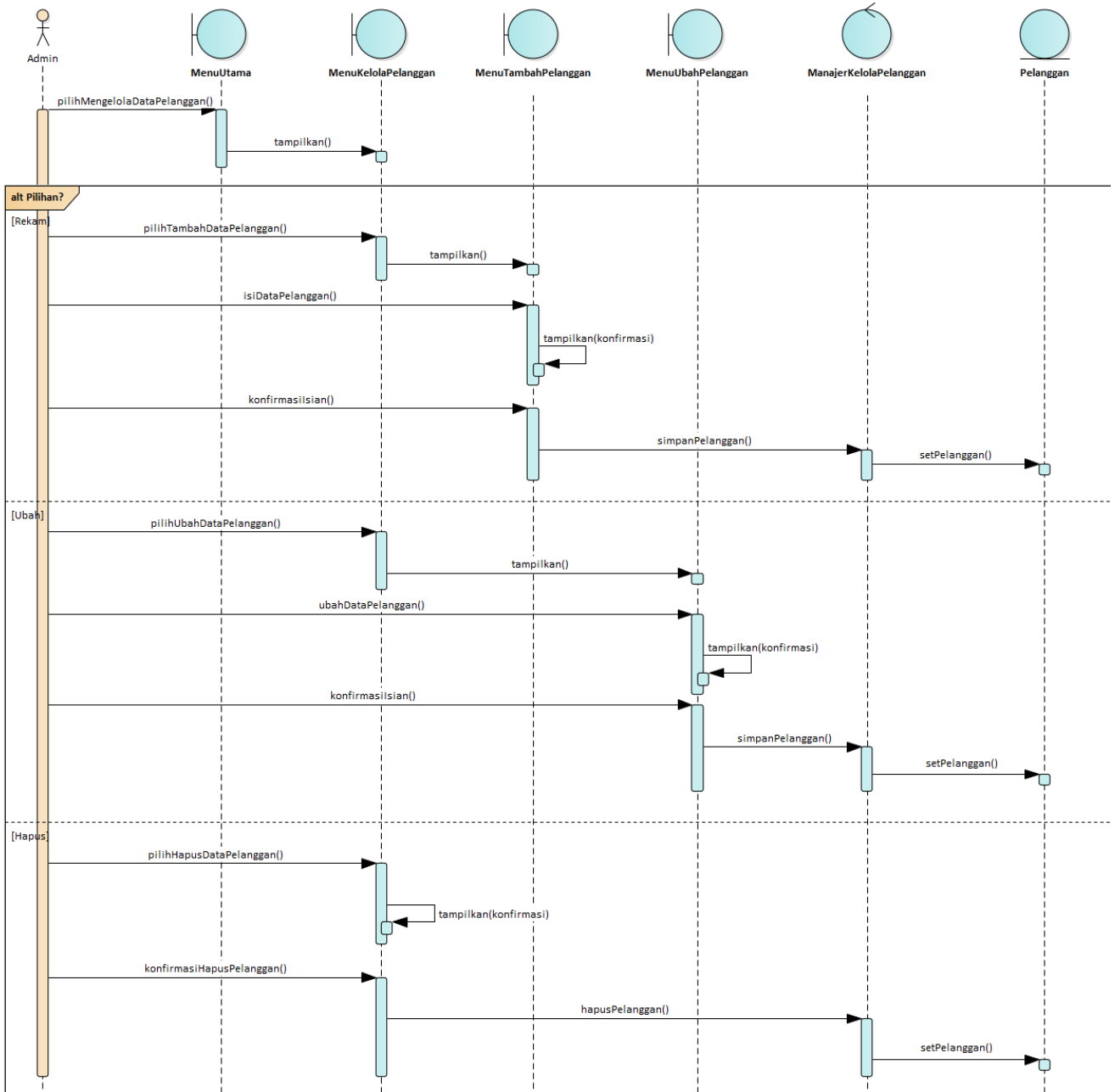
Sequence Diagram: Melakukan Login dan Logout



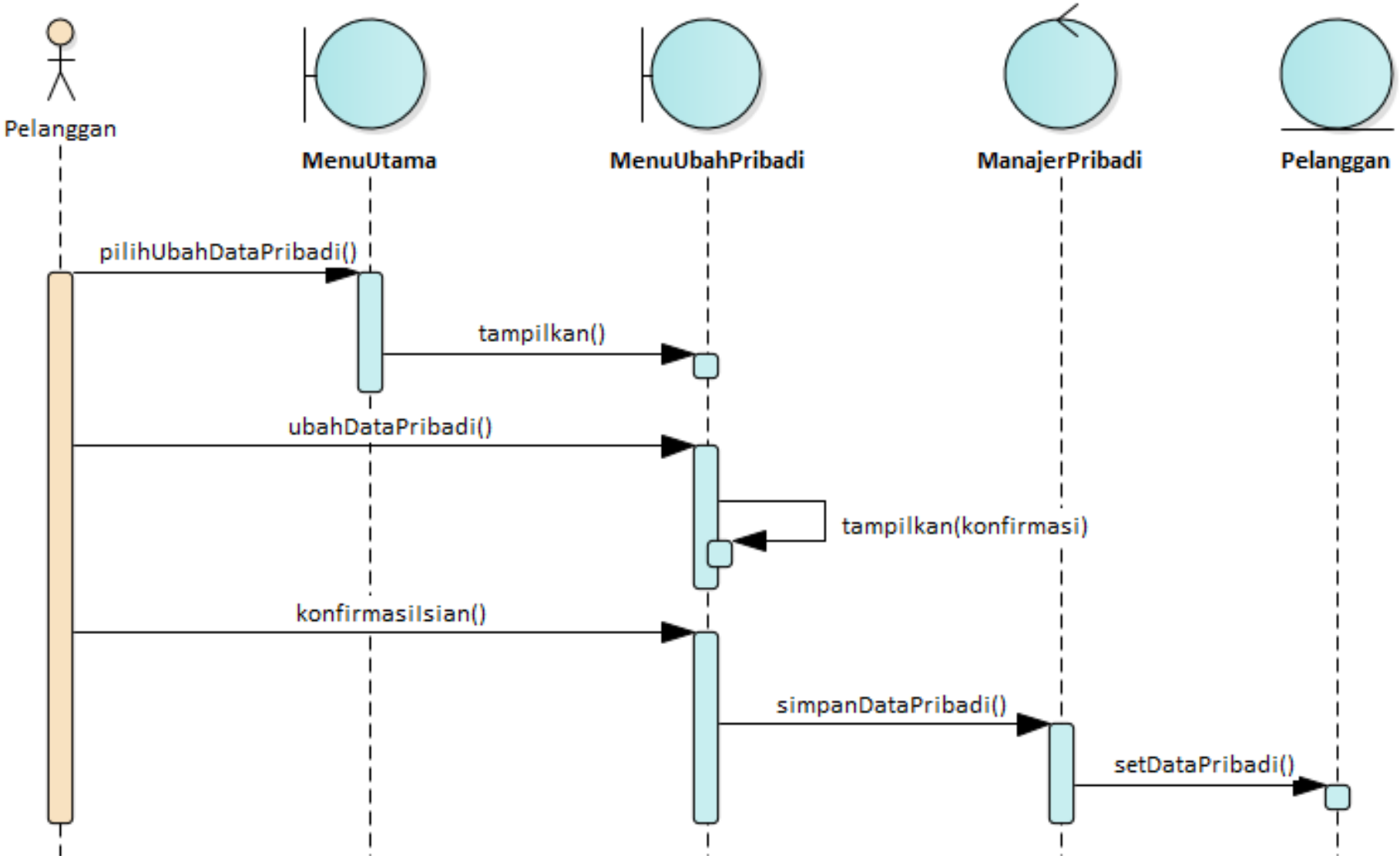
Sequence Diagram: Mengelola Data Lagu



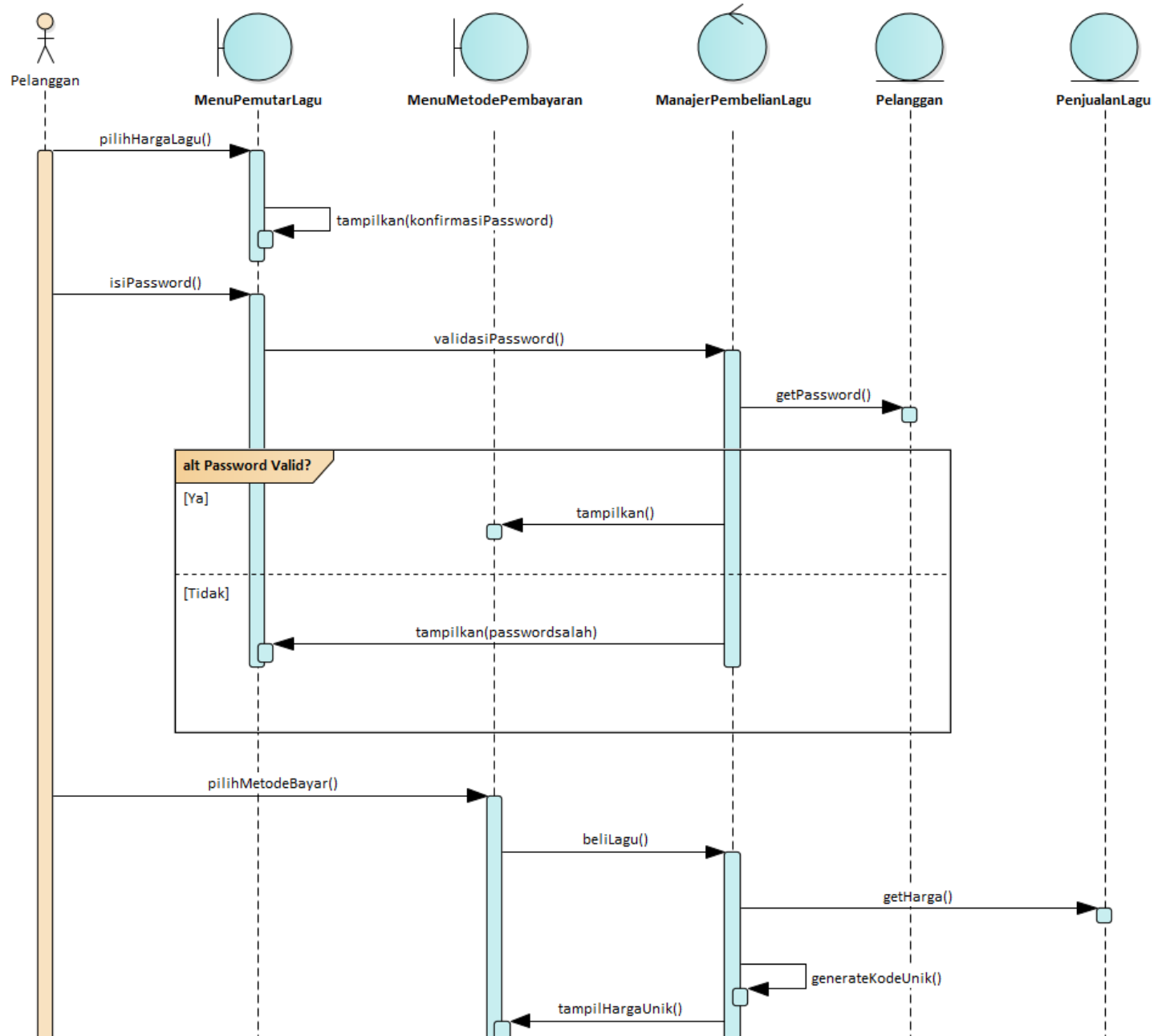
Sequence Diagram: Mengelola Data Pelanggan



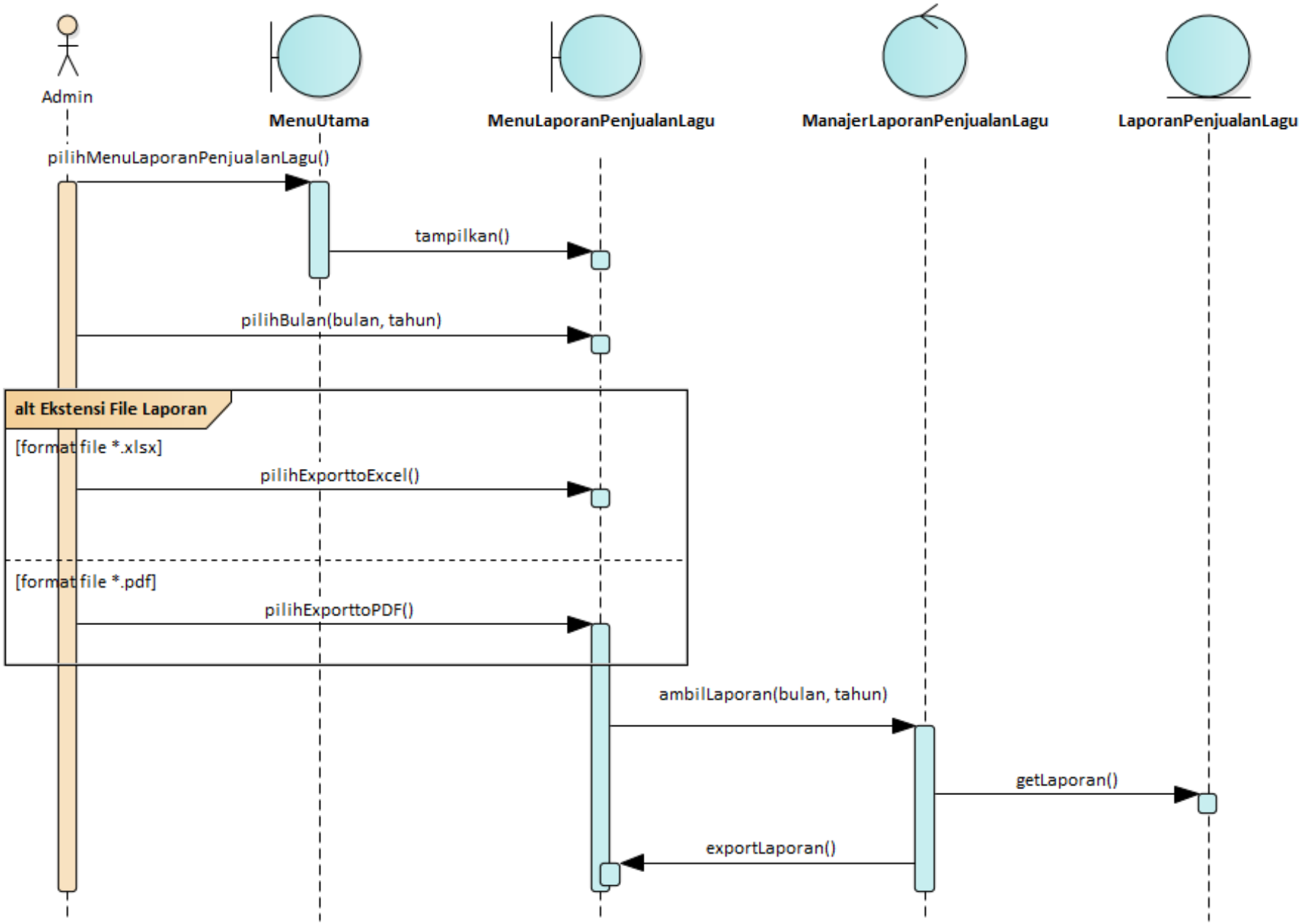
Sequence Diagram: Mengubah Data Pribadi



Sequence Diagram: Melakukan Pembelian Lagu



Sequence Diagram: Mengekspor Laporan Penjualan



Catatan Pola Kesalahan

- Gunakan pola Subject-Verb-Object (S-V-O) untuk use case diagram (Actor – Use Case) dan Activity Diagram (Partition – Action)
- Use Case Diagram adalah apa yang dilakukan Actor di sistem, bukan apa yang dilakukan oleh sistem
- Pada Sequence Diagram perhatikan transaksi yang harusnya datangnya dari actor, dan bukan dari object lain
- Naming untuk object dan class adalah kata benda (noun), untuk message di Sequence Diagram (method) adalah kata kerja
- Object lifeline dan message di Sequence Diagram dan nama class di Class Diagram, tidak boleh menggunakan spasi, karena akan jadi Class dan Method di kode program
- Actor (manusia) akan mengirim message hanya ke Boundary Class, tidak ke Control atau Entity Class. Sedangkan Actor (System) akan mengirim message ke Control
- Entity Class bukan consumer, jadi tidak pernah mengirim message ke Boundary atau Control Class
- Boundary class akan menjadi UI Design, entity class akan menjadi Data Model
- Class diagram tidak menunjukkan alur, tapi menunjukkan struktur dan komposisi dari sistem yg kita bangun

Exercise: System Analysis untuk System Request

1. Lihat kembali System Request yang sudah anda buat
2. Lakukan system analysis dengan membuat diagram di bawah:
    1. Use Case Diagram
    2. Activity Diagram or BPMN
    3. Sequence Diagram
3. Ujicoba menghitung size dari software dengan menggunakan Use Case Points
4. Ujicoba publish hasil ke HTML dan DOCX
5. Kirim file EAPX ke romi@brainmatics.com untuk dilakukan review bersama

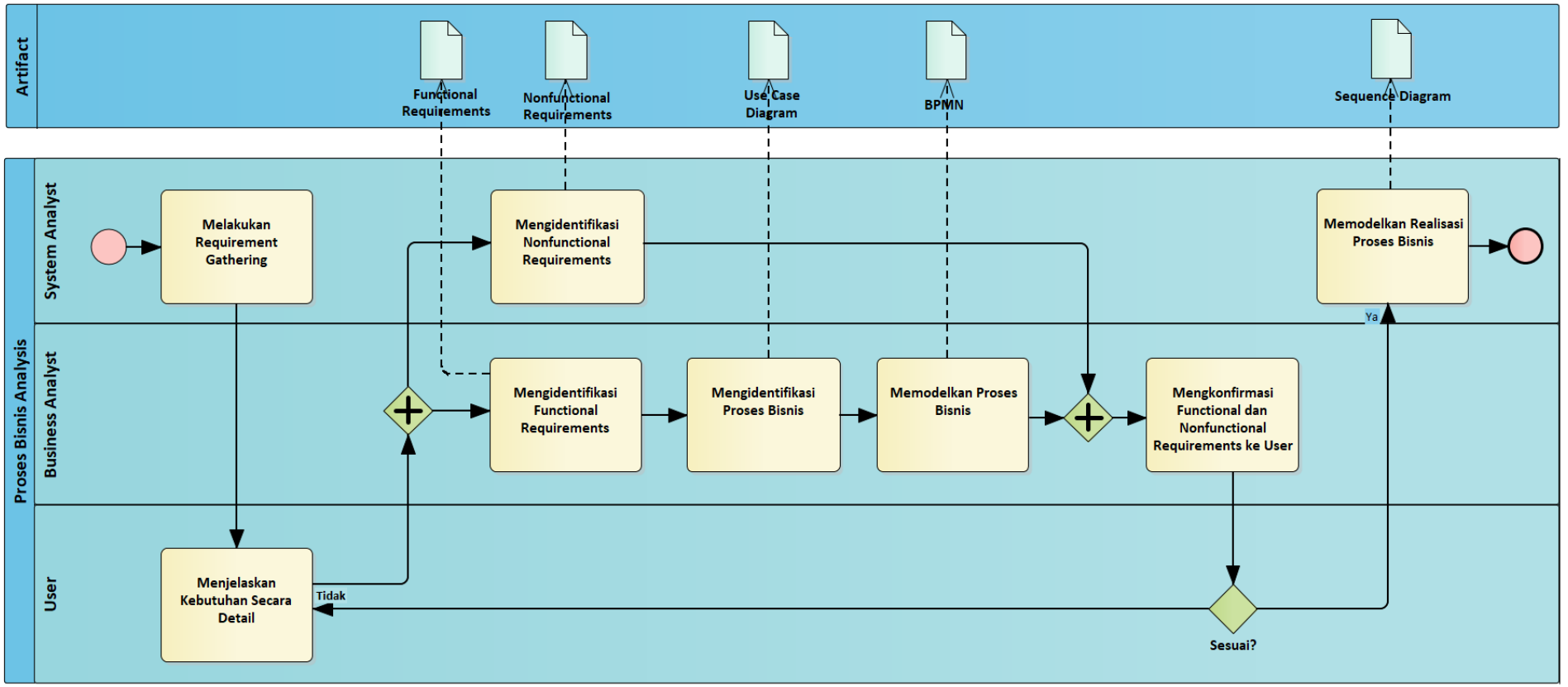
Exercise: Systems Analysis and Design

- Lakukan sistem analysis and design yang menghasilkan diagram:
  1. Use Case Diagram
  2. Activity Diagram or BPMN
  3. **Sequence Diagram**
  4. **Penghitungan Size dengan Use Case Points**

- Pilih salah satu aplikasi di bawah:

1. Aplikasi Rental Mobil
2. Aplikasi Pengelolaan Klinik
3. Aplikasi Pengelolaan Apotik
4. Aplikasi Pengelolaan Service Mobil
5. Aplikasi Penjualan Motor
6. Aplikasi Pengelolaan Perpustakaan
7. Aplikasi Penjualan Buku Online
8. Aplikasi Penjualan Tiket Kereta Online
9. Aplikasi Manajemen Universitas Online
10. Aplikasi Penjualan Laptop Online
11. Aplikasi Perpustakaan Digital
12. Aplikasi Pengelolaan Project Software

Systems Analysis



4. Systems Design

4.1 Paradigma Berorientasi Object

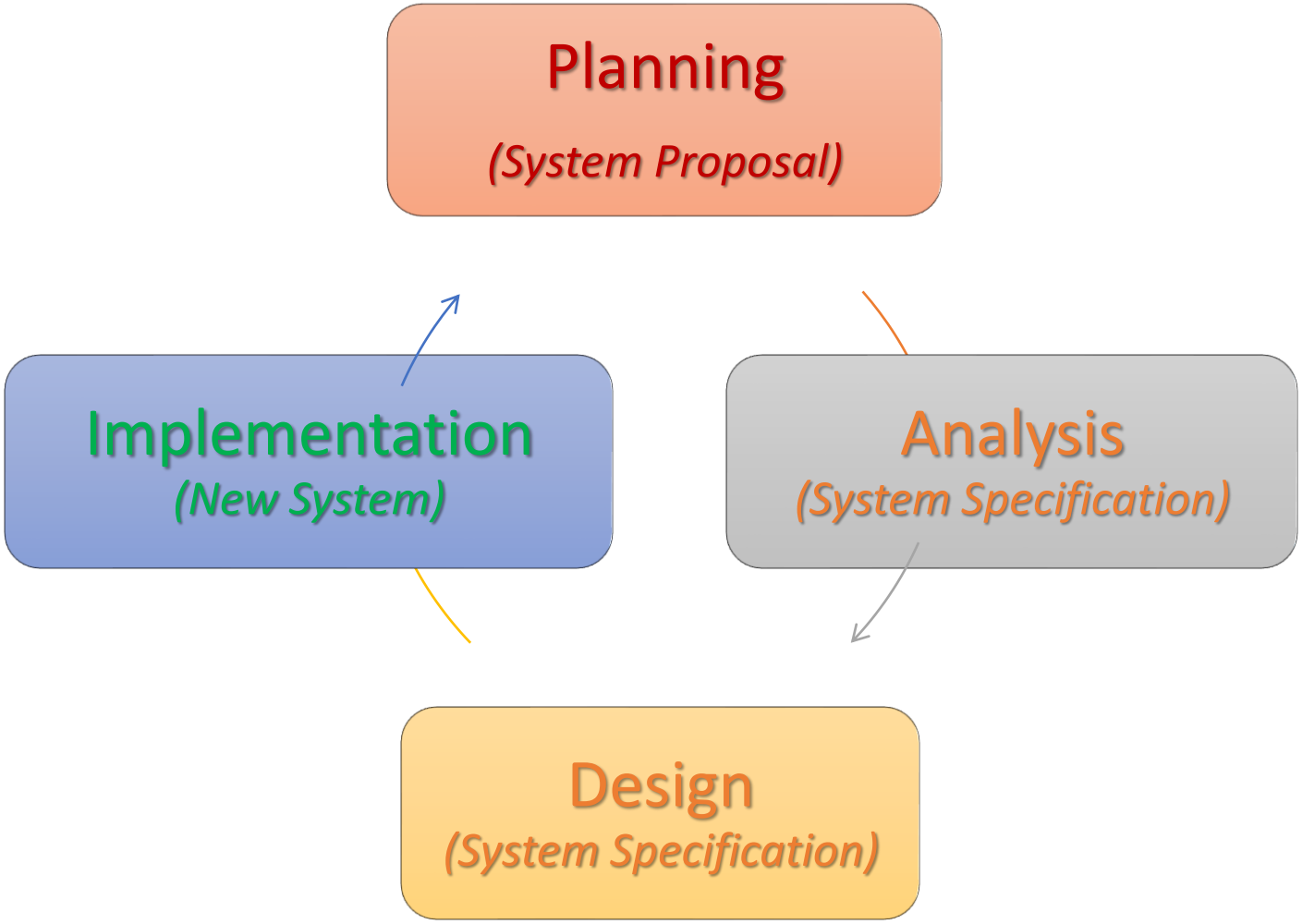
4.2 Pembuatan Class Diagram

4.3 Pembuatan User Interface Design

4.4 Pembuatan Data Model

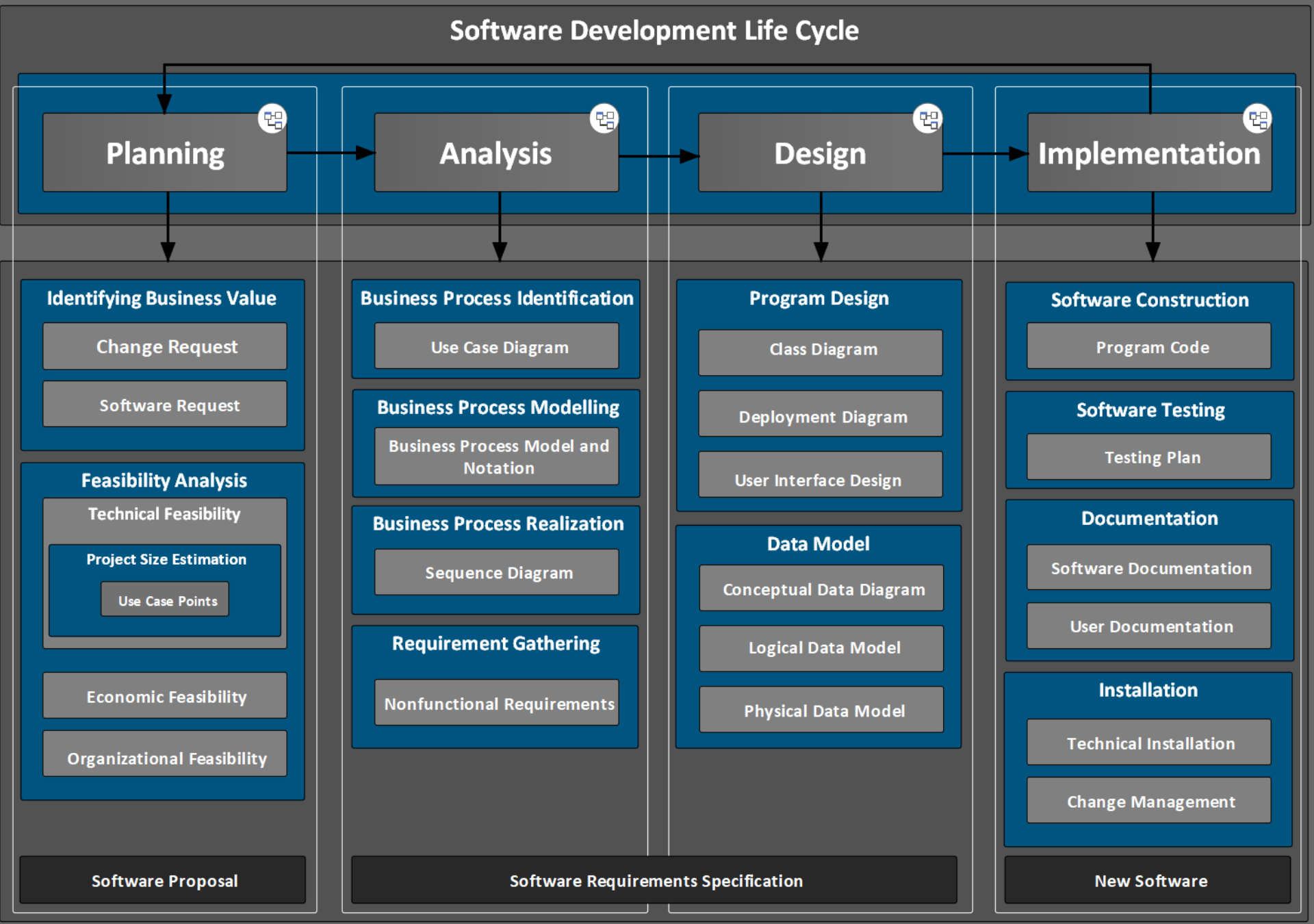
4.5 Pembuatan Deployment Diagram

Siklus Pengembangan Software



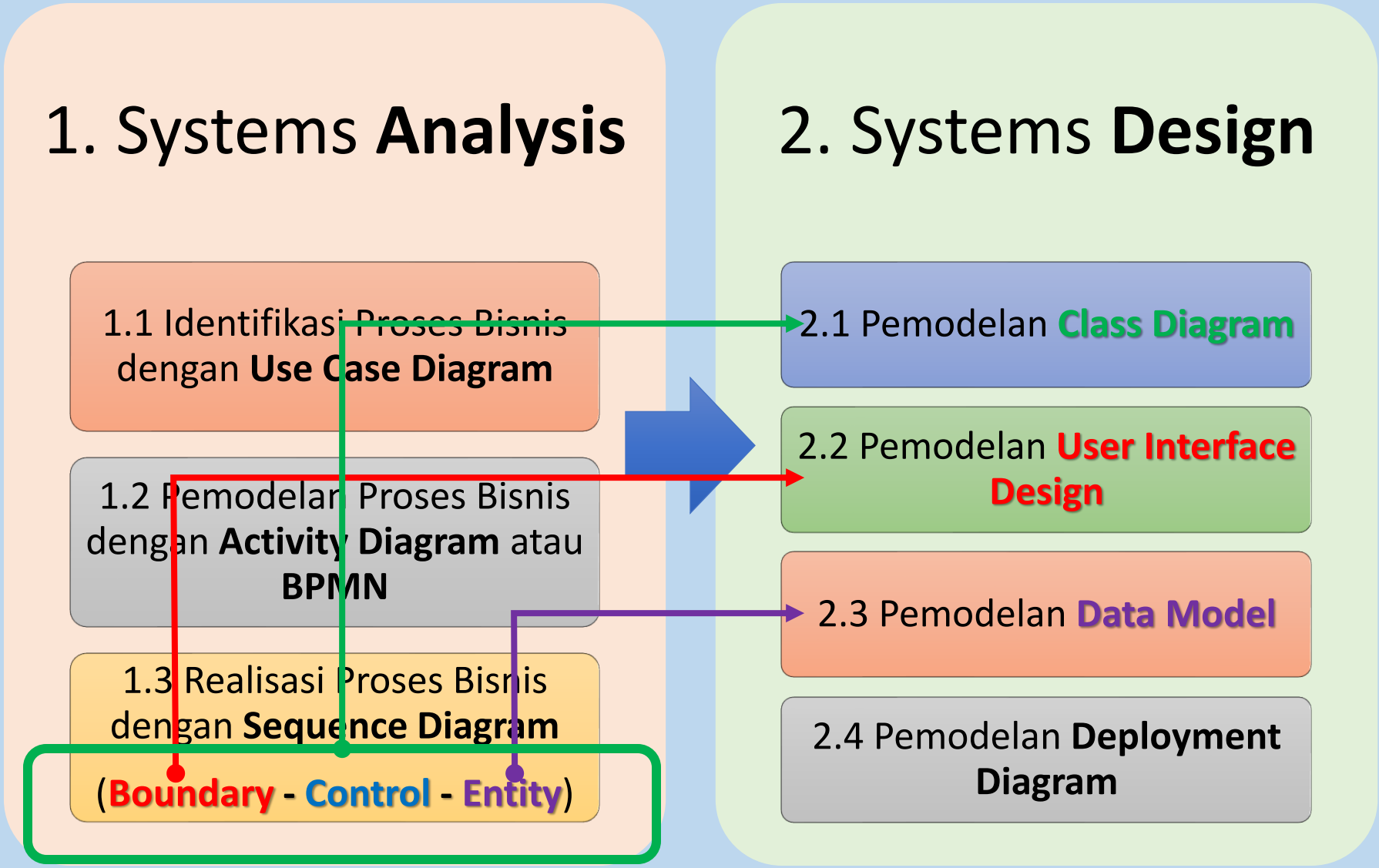
(Tilley, 2012) *(Dennis, 2016)* *(Valacich, 2017)*

Application Development Governance



UML based Software Analysis and Design

(Wahono, 2009)



4.1 Paradigma Berorientasi Objek

4.1.1 Konsep Dasar Paradigma Berorientasi Objek

Class , Object, Method, Attribute

Berorientasi Objek?

Attribute:

Baju, Jaket,
Tas Punggung,
Tangan, Kaki, Mata

Behavior:

Cara Jalan ke Depan
Cara Jalan Mundur
Cara Belok ke Kiri
Cara Memanjat

Berorientasi Objek?

Attribute (State):

Ban, Stir, Pedal Rem, Pedal Gas,
Warna, Tahun Produksi

Behavior:

Cara Menghidupkan Mesin
Cara Manjalankan Mobil
Cara Memundurkan Mobil

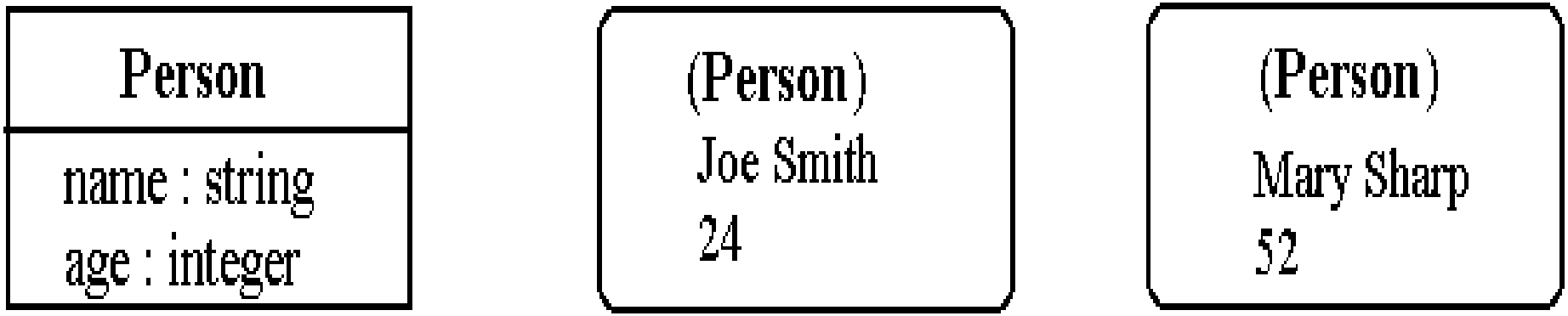
Attribute → Variable(Member)
Behavior → Method(Fungsi)

Perbedaan Class dan Object

- Class: konsep dan deskripsi dari sesuatu
  - Class mendeklarasikan method yang dapat digunakan (dipanggil) oleh object
- Object: instance dari class, bentuk (contoh) nyata dari class
  - Object memiliki sifat independen dan dapat digunakan untuk memanggil method
- Contoh Class dan Object:
  - Class: mobil
  - Object: mobilnya pak Joko, mobilku, mobil berwarna merah

Perbedaan Class dan Object

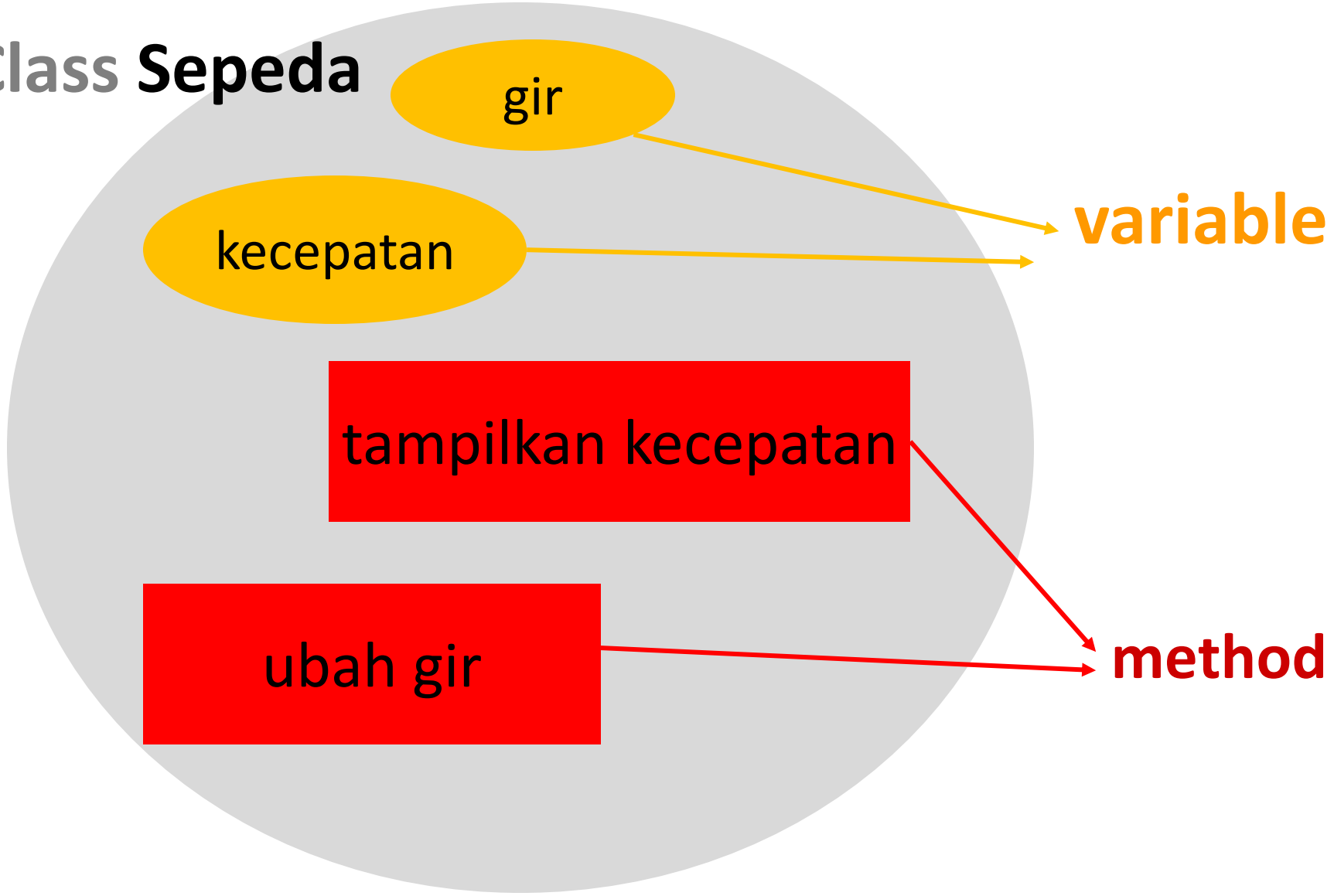
- Class seperti cetakan kue, dimana kue yg dihasilkan dari cetakan kue itu adalah object
- Warna kue bisa bermacam-macam meskipun berasal dari cetakan yang sama (object memiliki sifat independen)



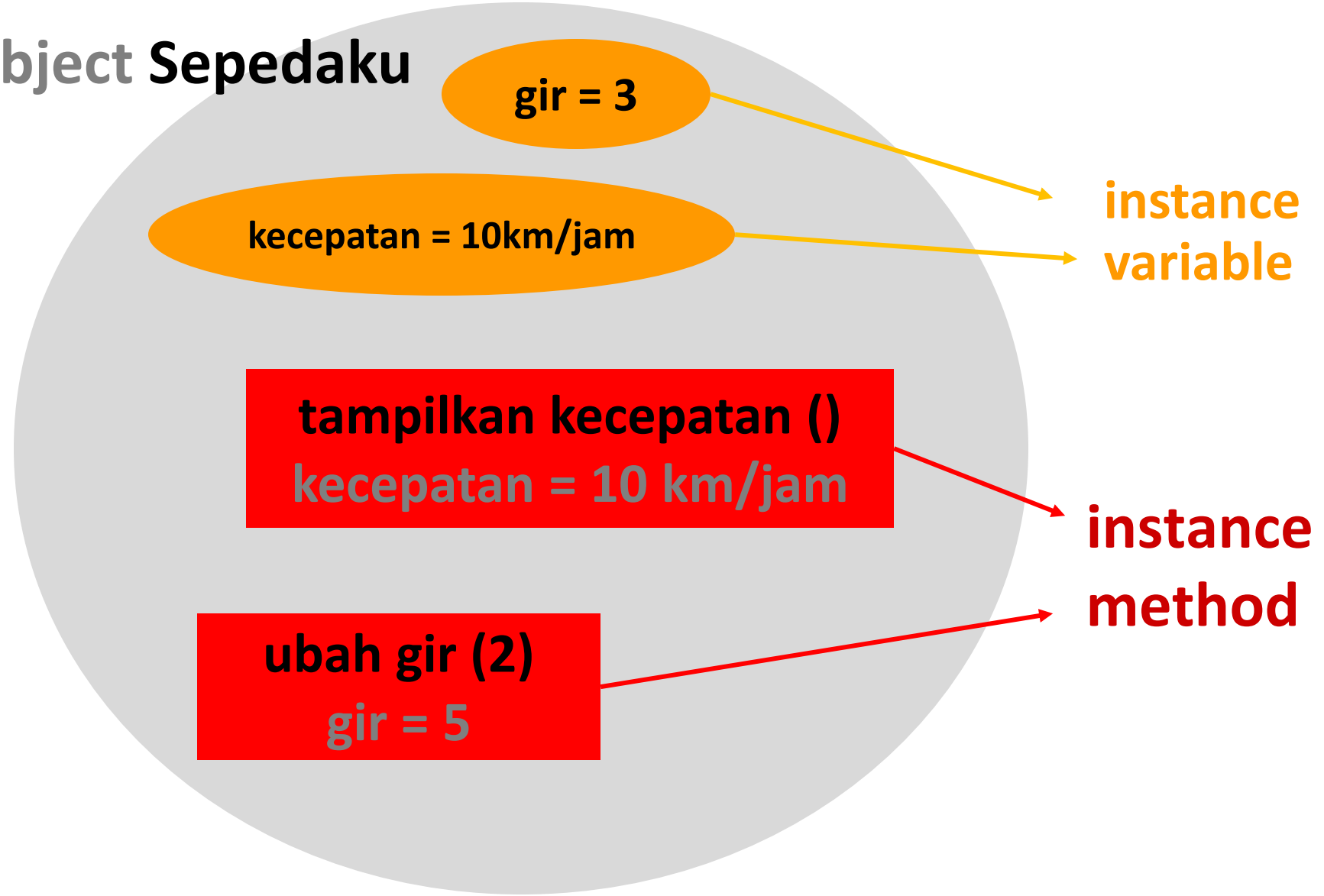
Class with Attributes

Objects with Values

Class = Method + Variable

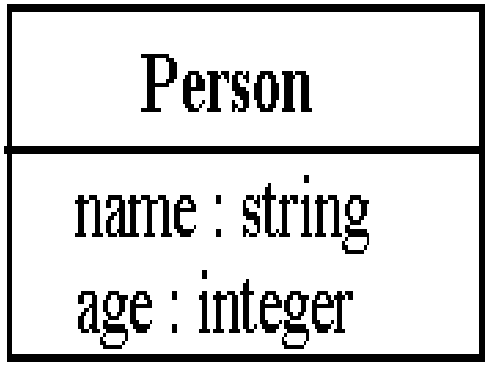


Object = Method + Variable Bernilai

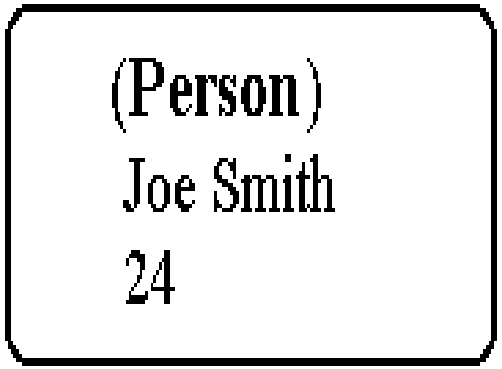


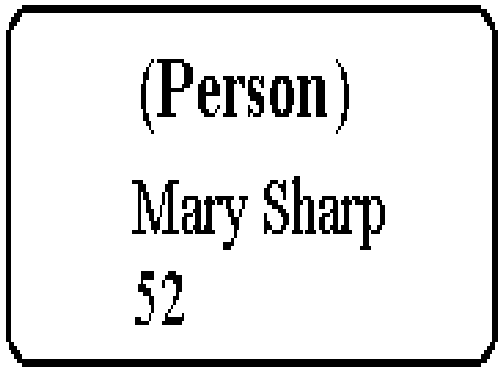
Attribute

- Variable yang mengitari class, dengan nilai datanya bisa ditentukan di object
- Variable digunakan untuk menyimpan nilai yang nantinya akan digunakan pada program
- Variable memiliki jenis (tipe), nama dan nilai
- Name, age, dan weight adalah atribute (variabel) dari class Person



Class with Attributes





Objects with Values

Membuat Class dan Object

Mobil.java

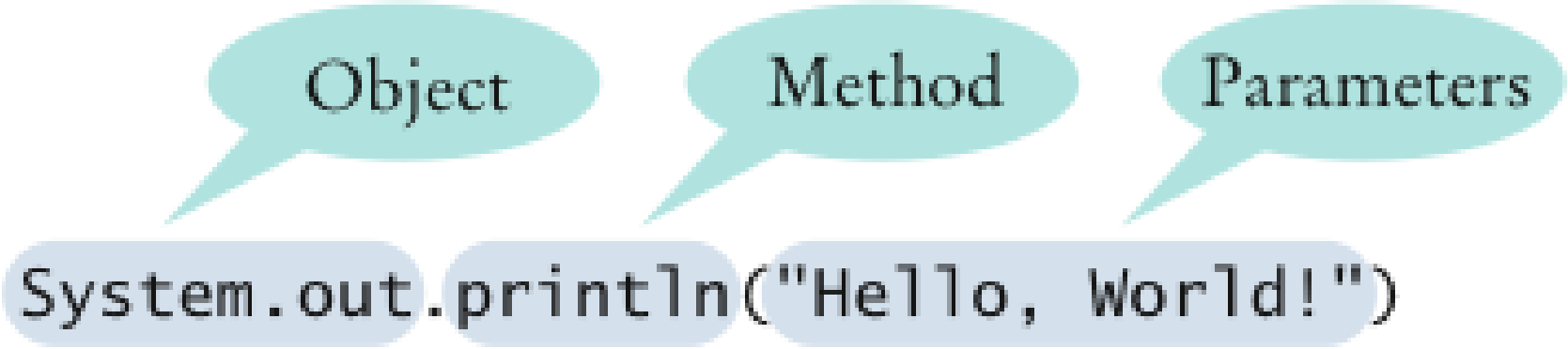
```
public  class  Mobil {
   String  warna;
   int  tahunProduksi;
}
```

MobilBeraksi.java

```
public  class  MobilBeraksi{
    public  static  void  main(String[] args){
          // Membuat object
         Mobil  mobilku = new Mobil();

          /* memanggil atribut  dan memberi nilai */
         mobilku.warna = "Hitam";
         mobilku.tahunProduksi = 2006;
         System.out.println("Warna: " + mobilku.warna);
         System.out.println("Tahun: " + mobilku.tahunProduksi);
    }
}
```

Method

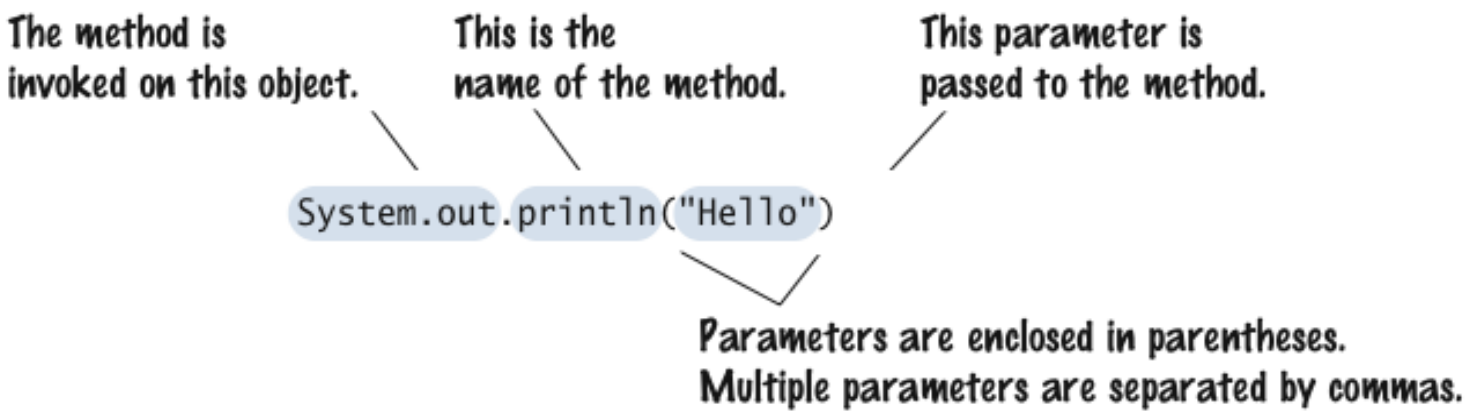
- Method adalah urutan instruksi yang mengakses data dari object
- Method melakukan:
  1. Manipulasi data
  2. Perhitungan matematika
  3. Memonitor kejadian dari suatu event



Method

Syntax *object.methodName(parameters)*

Example



Membuat dan Memanggil Method

Mobil2.java

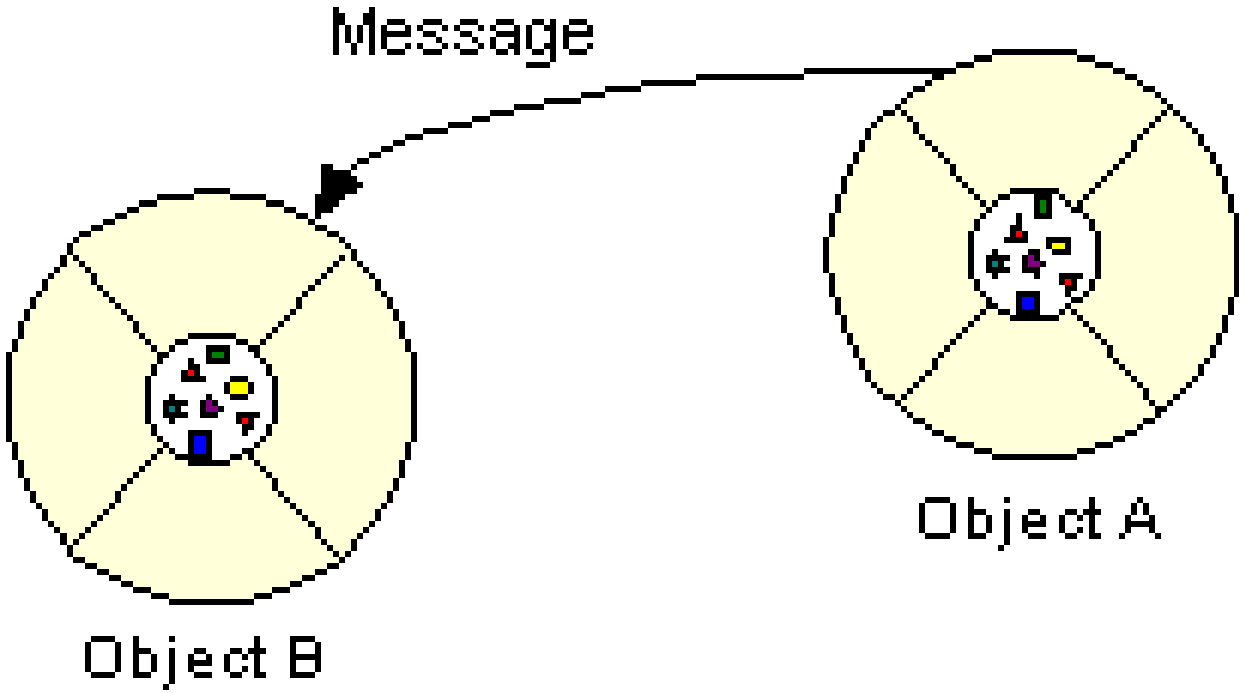
```
public class Mobil2{
    String warna;
    int tahunProduksi;

    void  printMobil(){
            System.out.println("Warna: " + warna);
            System.out.println("Tahun: " + tahunProduksi);
    }
}
```

Mobil2Beraksi.java

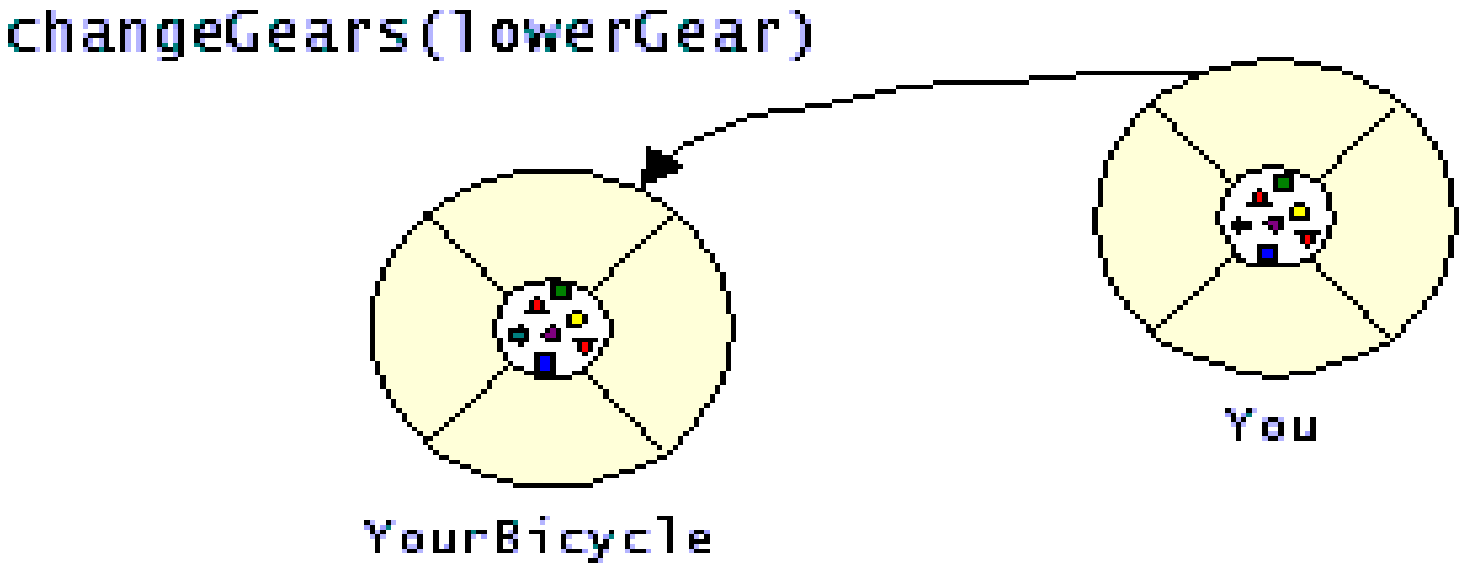
```
public class Mobil2Beraksi{
     public static void main(String[] args){
            Mobil2 mobilku = new Mobil2();

             mobilku.warna = "Hitam";
            mobilku.tahunProduksi = 2006;
            mobilku.printMobil();
      }
}
```

Parameter

- Sepeda akan berguna apabila ada object lain yang berinterasi dengan sepeda tersebut
- Object software berinteraksi dan berkomunikasi dengan object lain dengan cara mengirimkan message atau pesan
- Pesan adalah suatu method, dan informasi dalam pesan dikenal dengan nama parameter



Pengiriman Pesan dan Parameter



1. **You** → object pengirim
2. **YourBicycle** → object penerima
3. **changeGears** → pesan berupa method yang dijalankan
4. **lowerGear** → parameter yang dibutuhkan method (pesan) untuk dijalankan


ON/OFF

Sepeda.java

```
public class Sepeda{
	int gir;

		// method (mutator) dengan parameter
		void setGir(int pertambahanGir) {
			gir= gir+ pertambahanGir;
		}

		// method (accessor)
		int getGir() {
			return gir;
		}
}
```

SepedaBeraksi.java

```
public class SepedaBeraksi{
    public static void main(String[] args) {
        // Membuat object
        Sepeda sepedaku = new Sepeda();

        // Memanggil method dan menunjuk nilai parameter
        sepedaku.setGir(1); // menset nilai gir = 1
        System.out.println("Gir saat ini: " + sepedaku.getGir());

        sepedaku.setGir(3); // menambahkan 3 pada posisi gir saat ini (1)

        System.out.println("Gir saat ini: " + sepedaku.getGir());
    }
}
```

4.1.2 Karakteristik Pemrograman Berorientasi Objek

Abstraction, Encapsulation, Inheritance, Polymorphism

Abstraction

- Cara kita melihat suatu sistem dalam bentuk yang lebih sederhana, yaitu sebagai suatu kumpulan subsistem (object) yang saling berinteraksi.
  - Mobil adalah kumpulan sistem pengapian, sistem kemudi, sistem pengereman
- Alat meng-abstraksikan sesuatu adalah class
- Object bersifat modularity. Object dapat ditulis dan dimaintain terpisah (independen) dari object lain

C:\RSWLecture\romi-sad\02 case study\RentalDVD\RentalDVD.java - Notepad++

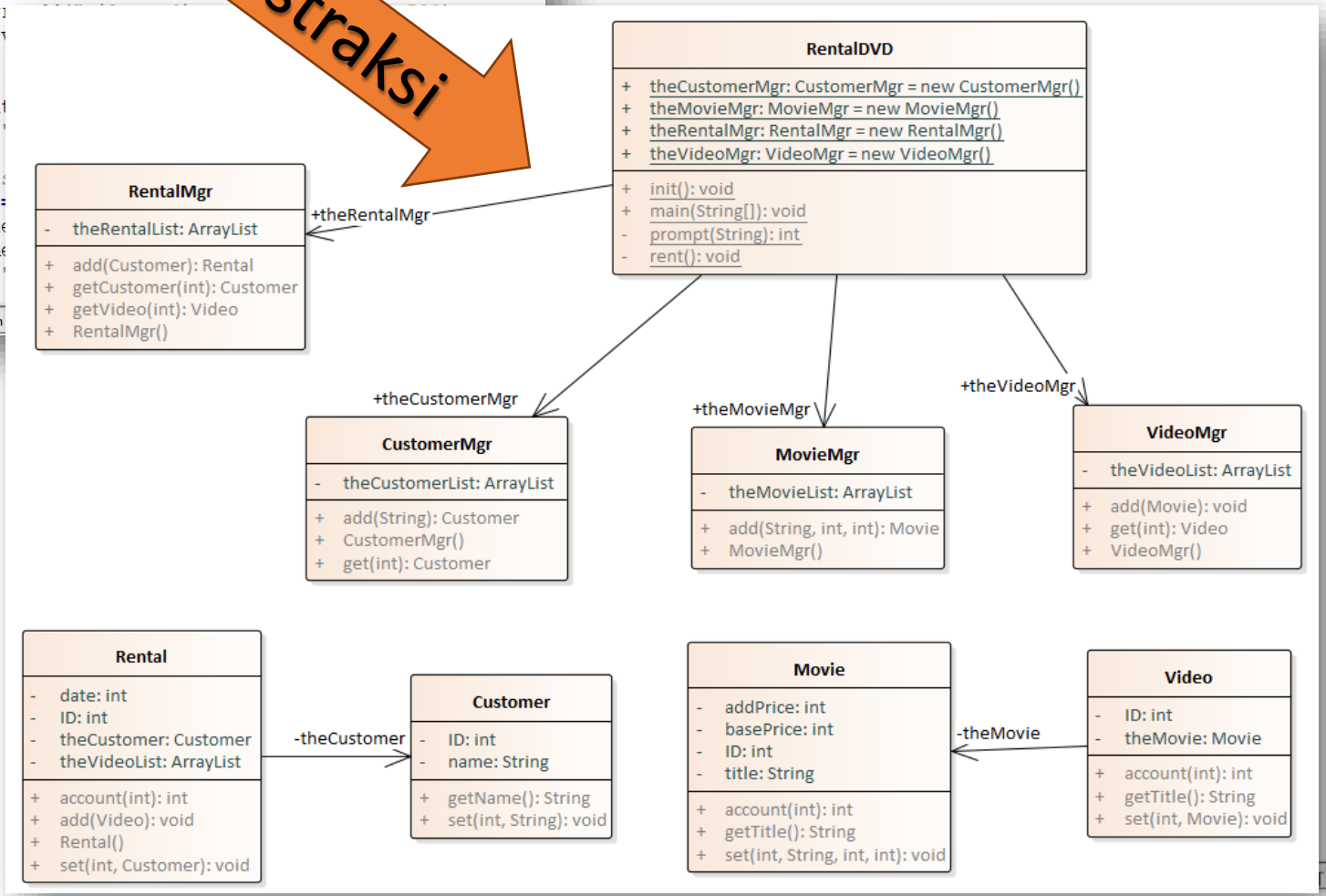
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

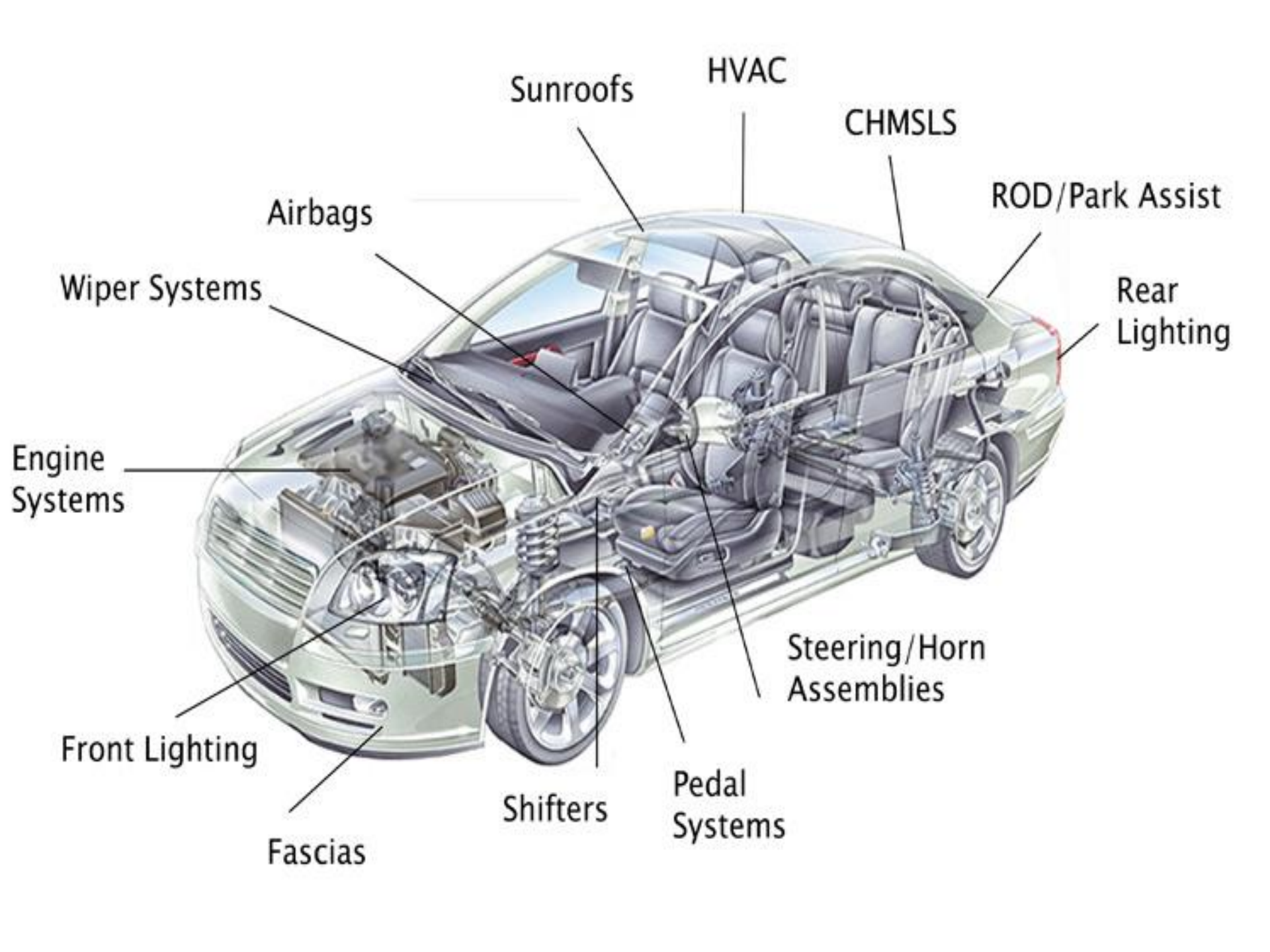
Account.java | MenuDeposit.java | ManajerValidasi.java | ManajerPengecekanSaldo.java | ManajerPengirimanUang.java | RentalDVD.java

```
21
22          Movie aMovie = new Movie();
23          aMovie = theMovieMgr.add("The Mirror",2000,1000);
24          theVideoMgr.add(aMovie);
25          aMovie = theMovieMgr.add("Meet Dave",2000,1000);
26          theVideoMgr.add(aMovie);
27          aMovie = theMovieMgr.add("Hallowen",2000,1000);
28          theVideoMgr.add(aMovie);
29
30          aMovie = theMovieMgr.add("Film                ru 1",1000,500);
31          theVideoMgr.add(aMovie);
32          aMovie = theMovieMgr.add("Film Ind             ",1000,500);
33          theVideoMgr.add(aMovie);
34          aMovie = theMovieMgr
35          theVideoMgr.add(aMov
36      }
37
38      private static void rent
39          System.out.println("
40
41          int mn = prompt("Ma
42          Customer aCustomer =
43          Rental aRental = the
44          String ms = aCustome
45          System.out.println("
            +ms+"\n");
```

va source file | length : 5,727 | lines : 253 | Ln

abstraksi




Sunroofs
HVAC
CHMSLS
ROD/Park Assist
Airbags
Wiper Systems
Rear Lighting
Engine Systems
Steering/Horn Assemblies
Front Lighting
Fascias
Shifters
Pedal Systems

BICYCLE

www.infovisual.info

Encapsulation

- Mekanisme menyembunyikan suatu proses dan data dalam sistem untuk menghindari interferensi, dan menyederhanakan penggunaan proses itu sendiri
  - Tongkat transmisi (gigi) pada mobil
  - Tombol on/off/pengaturan suhu pada AC
- Class access level (public, protected, privat) adalah implementasi dari konsep encapsulation
- Enkapsulasi data dapat dilakukan dengan cara:
  1. mendeklarasikan instance variable sebagai private
  2. mendeklarasikan method yang sifatnya public untuk mengakses variable tersebut

```
public  class  Titik {
    private void pixel(){
   .....
   }
  private void pen(){
  .....
  }
  private void koordinat(){
 ...
  }
  public void gambarTitik(){
      pixel();
       pen();
  }
}
```

```
public  class  Garis {
      public  void gambarGaris(){
      koordinat1 = Titik.gambarTitik();


}}
```


ON/OFF

Encapsulation dan Access Modifier

Modifier	Dalam Class yang Sama	Dalam Package yang Sama	Dalam SubClass	Dalam Package Lain
private	✓			
tanpa tanda	✓	✓		
protected	✓	✓	✓	
public	✓	✓	✓	✓

Encapsulation

Enkapsulasi data juga dapat dilakukan dengan cara:

1. mendeklarasikan instance variable sebagai private
2. mendeklarasikan method yang sifatnya public untuk mengakses variable tersebut

Syntax

```
accessSpecifier class ClassName
{
   instance variables
   constructors
   methods
}
```

Example

```
public class Counter
{
   private int value;

   public Counter(double initialValue) { value = initialValue; }

   public void count() { value = value + 1; }
   public int getValue() { return value; }
}
```

Public interface

Private implementation

Sepeda.java

```
public class Sepeda{
    int gir;

      void setGir(int pertambahanGir) {
            gir= gir+ pertambahanGir;
      }

      int getGir() {
            return gir;
      }
}
```

SepedaBeraksi.java

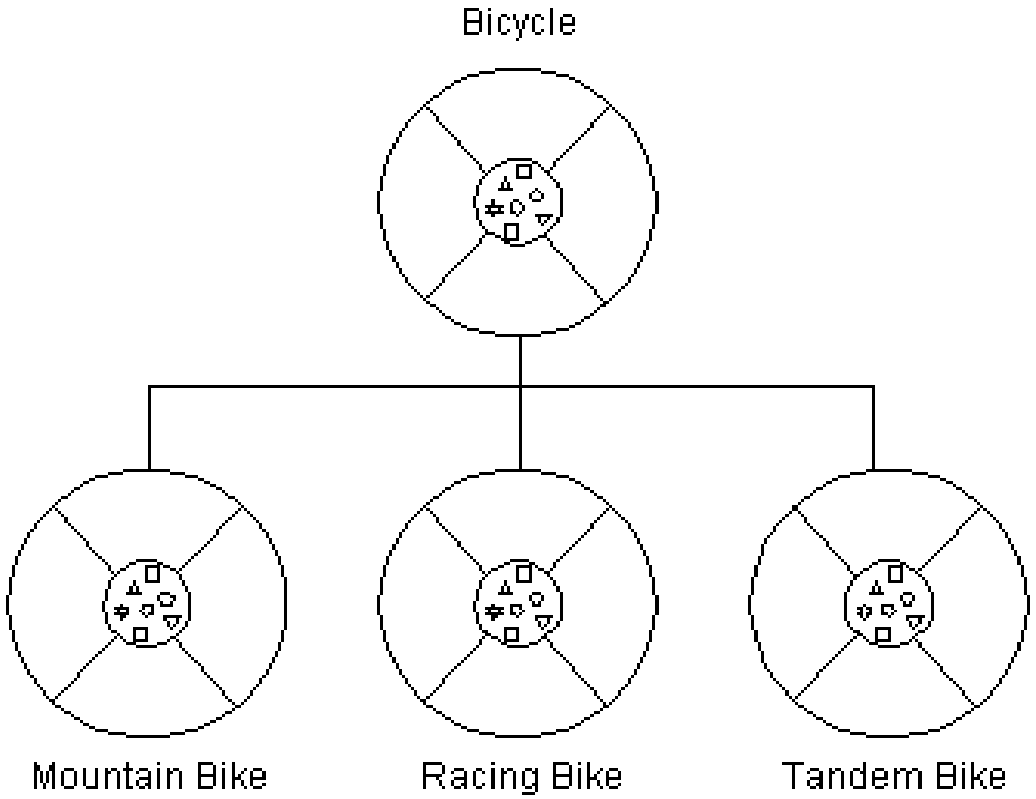
```
public class SepedaBeraksi{
    public static void main(String[] args) {
        Sepeda sepedaku = new Sepeda();

        sepedaku.setGir(1);
        /* Variabel bisa diubah atau tidak sengaja diubah.
           Hal ini berbahaya dan sering menimbulkan bug.
           Berikan access modifier private pada instance variable */
        sepedaku.gir = 3;
        System.out.println("Gir saat ini: " + sepedaku.getGir());
    }
}
```

Sepeda.java

```
public class Sepeda{
    private int gir; // access modifier private pada instance variable

      void setGir(int pertambahanGir) {
            gir= gir+ pertambahanGir;
      }

      int getGir() {
            return gir;
      }
}
```

Inheritance (Pewarisan)

- Suatu class dapat mewariskan atribut method kepada cla lain (subclass), serta membentuk class hierarchy
- Penting untuk Reusability
- Java Keyword: extends



Sepeda.java

```
public class Sepeda{
    private int gir;

      void setGir(int pertambahanGir) {
            gir= gir+ pertambahanGir;
      }

      int getGir() {
            return gir;
      }
}
```

Class SepedaGunung Mewarisi Class Sepeda

```
public class SepedaGunung extends Sepeda{

    private int sadel;

    void setSadel (int jumlah) {
            sadel = getGir() - jumlah;
    }

    int getSadel(){
            return sadel;
    }
}
```

SepedaGunung.java

```
public class SepedaGunungBeraksi {
   public static void main(String[] args) {

        SepedaGunung sg=new SepedaGunung();

         sg.setGir(3);
         System.out.println(sg.getGir());

         sg.setSadel(1);
         System.out.println(sg.getSadel());
     }
}
```

SepedaGunungBeraksi.java

Polymorphism

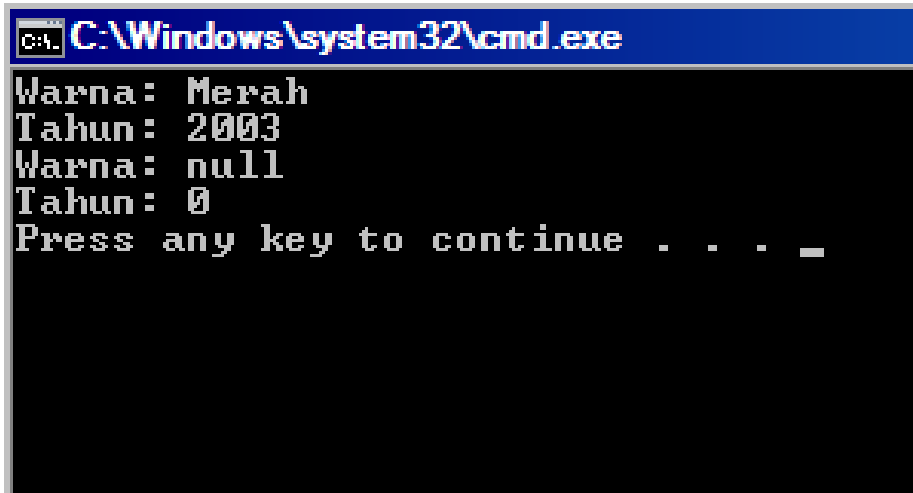
- Kemampuan untuk memperlakukan object yang memiliki perilaku (bentuk) yang berbeda

- Implementasi konsep polymorphism:
  1. Overloading: Kemampuan untuk menggunakan nama yang sama untuk beberapa method yang berbeda parameter (tipe dan atau jumlah)
  2. Overriding: Kemampuan subclass untuk menimpa method dari superclass, yaitu dengan cara menggunakan nama dan parameter yang sama pada method

Polymorphism – Overloading

```
class Mobil {
    String warna;
    int tahunProduksi;

    public Mobil(String warna, int tahunProduksi){
        this.warna = warna;
        this.tahunProduksi = tahunProduksi;
    }

    public Mobil(){
    }

    void info(){
        System.out.println("Warna: " + warna);
        System.out.println("Tahun: " + tahunProduksi);
    }
}
```

```
public class MobilKonstruktor{
  public static void main(String[] args){
      Mobil mobilku = new Mobil("Merah", 2003);
      mobilku.info();

      Mobil mobilmu = new Mobil();
      mobilmu.info();
      }
}
```



Polymorphism – Overloading

```
class Lingkaran{
   void gambarLingkaran(){
   }
    void gambarLingkaran(int diameter){
   ...
   }
    void gambarLingkaran(int diameter, int x, int y){
   ...
   }
    void gambarLingkaran(int diameter, int x, int y, int warna, String
   namaLingkaran){
   ...
   }
}
```

Polymorphism - Overriding

```
public class Sepeda{
      protected int gir;

         void setGir(int pertambahanGir) {
                  gir= gir+ pertambahanGir;
         }

         int getGir() {
                  return gir;
         }
}
```

Polymorphism - Overriding

```
public class SepedaGunung extends Sepeda{

   void setGir(int pertambahanGir) {
        super.setGir(pertambahanGir);
         gir = 2*getGir();
   }
}
```

SepedaGunung.java

```
public class SepedaGunungBeraksi {
   public static void main(String[] args) {

         SepedaGunung sg=new SepedaGunung();

         sg.setGir(2);
          System.out.println(sg.getGir());

          sg.setGir(3);
          System.out.println(sg.getGir());
      }
}
```

SepedaGunungBeraksi.java

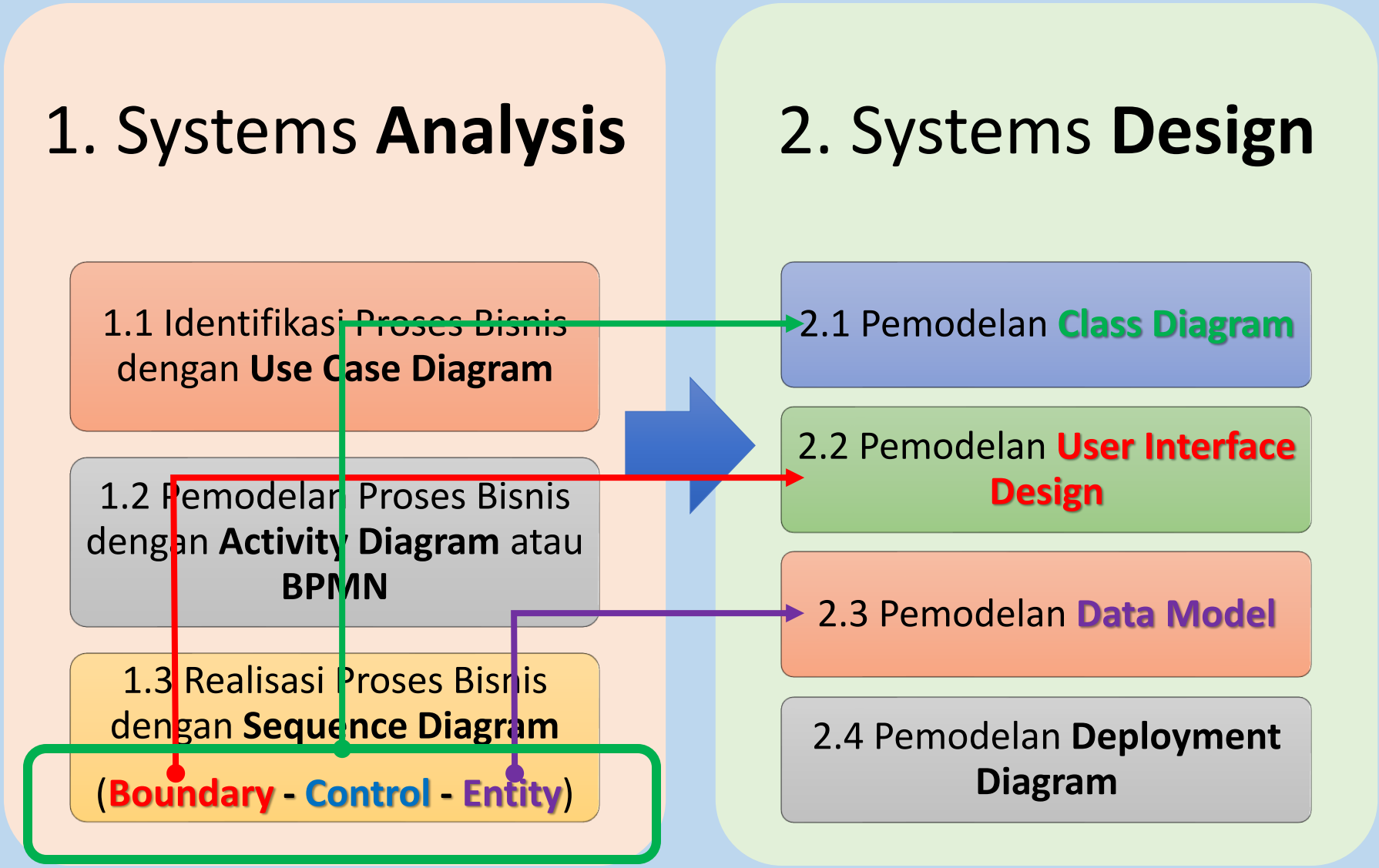
Matematika.java

```
public class Matematika{
   void pertambahan (int a, int b){
            int hasil= a + b;
            System.out.println(“hasil:” + hasil);
   }

   void pertambahan (double a, double b, double c){
            int hasil= a + b + c;
            System.out.println(“hasil:” + hasil);
   }
   ...
}
```

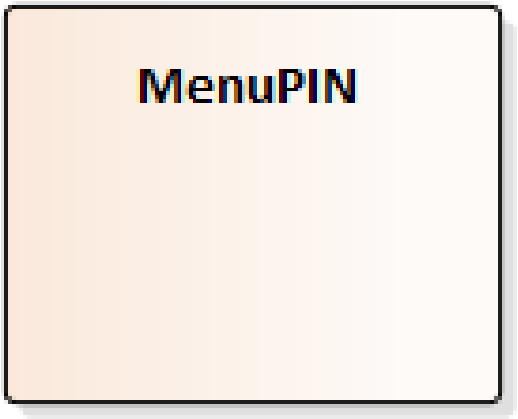
4.2 Pemodelan Class Diagram

UML based Software Analysis and Design

(Wahono, 2009)



Class

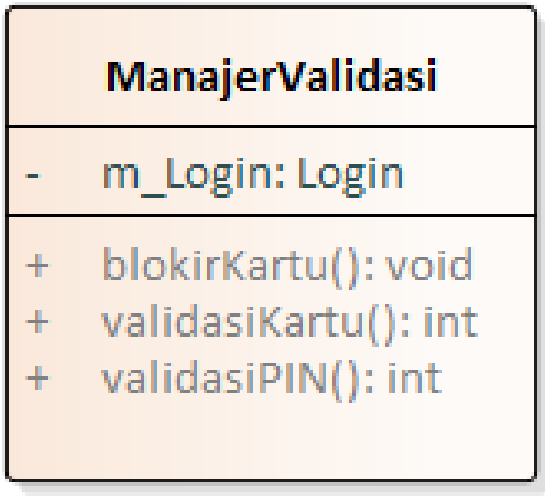



```
public class MenuPIN{

    ........

}
```

Attributes

- Visibility of attributes
  - **+ Public**: not hidden from any object
  - **# Protected**: hidden from all but immediate subclasses
  - **– Private**: hidden from all other classes
- Default is private

Class with Attribute and Method

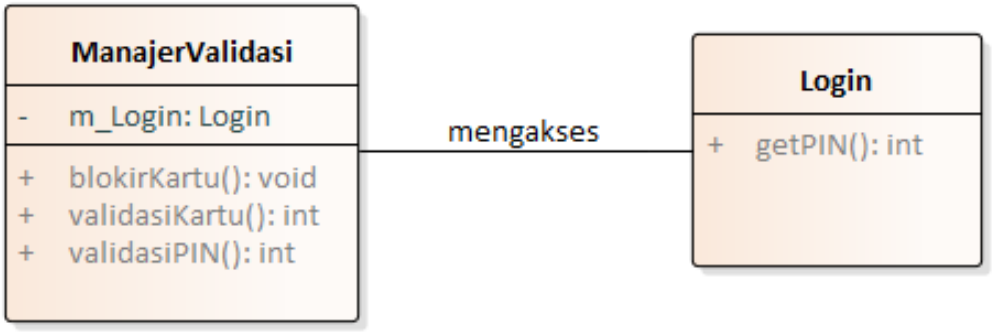


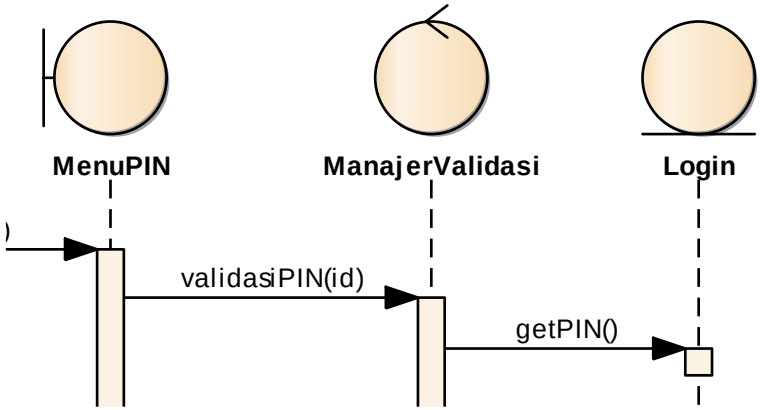

```
public class ManajerValidasi {

        private Login m_Login;

        public int validasiKartu(){
            return 0;
         }

         public int validasiPIN(){
            return 0;
         }

          public void blokirKartu(){

          }
   }
```

Relationships

1. Generalization
   - "Is-A" relationship
   - Enables inheritance of attributes & operations
   - Subclasses and superclasses
2. Aggregation
   - "Has-A" relationship
   - Relates parts to wholes
   - Uses decomposition
3. Composition
   - "Has-A" relationship
   - Similar but stronger than Aggregation
4. Association
   - Relationships that don't fit "Is-A" or "Has-A"
   - Often a weaker form of "Has-A"
   - Miscellaneous relationships between classes

Class Association

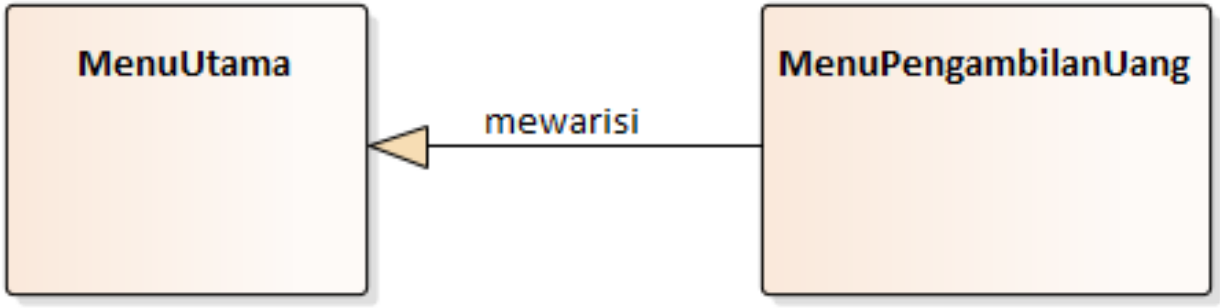





```
public class ManajerValidasi {

        private Login m_Login;

        public int validasiKartu(){
              return 0;
         }

         public int validasiPIN(){
              int pin = m_login.getPIN();
              return 0;
         }

          public void blokirKartu(){

          }
}
```

```
public class Login {


        public int getPIN(){
              return 0;
         }

}
```

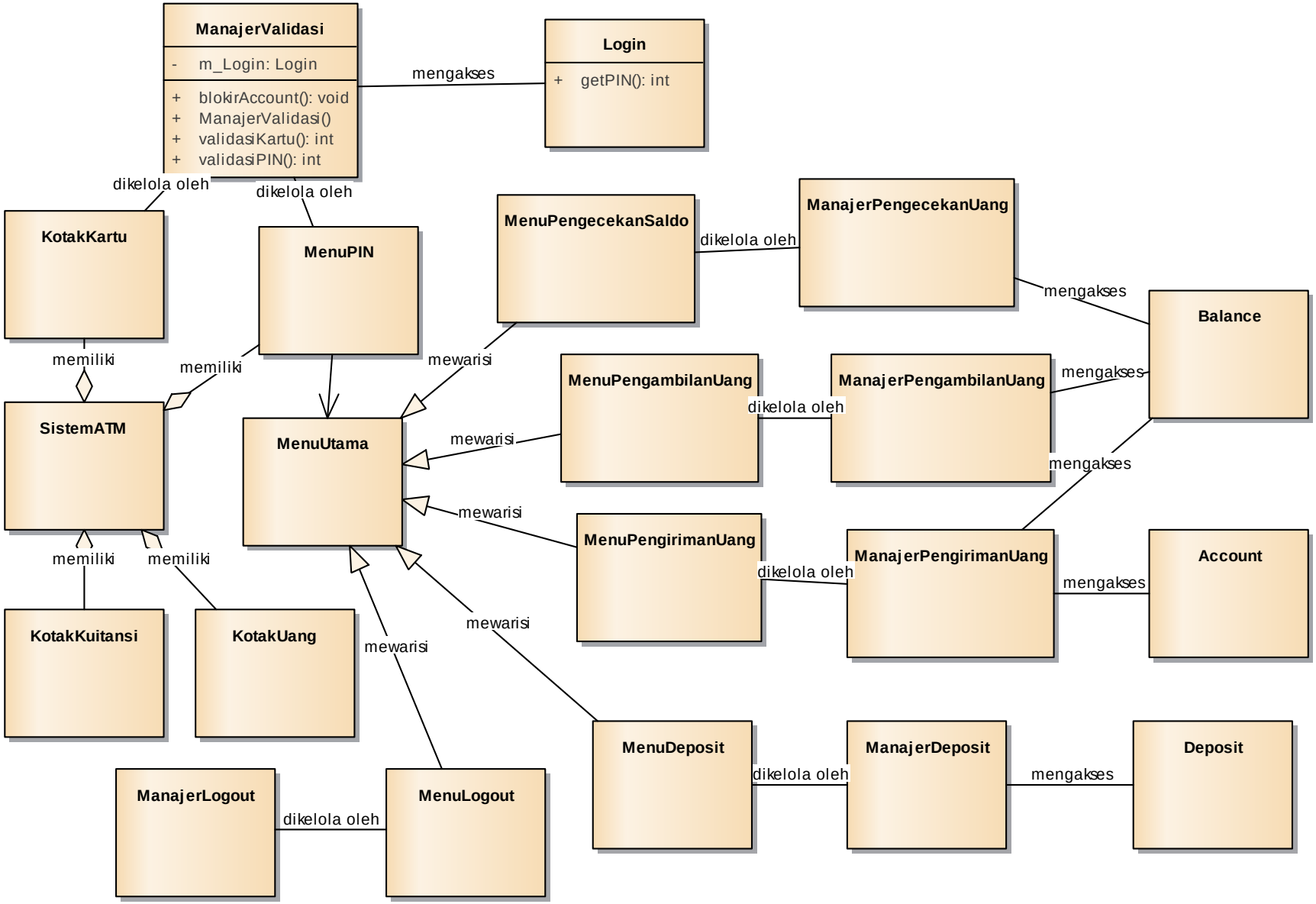
Class Inheritance



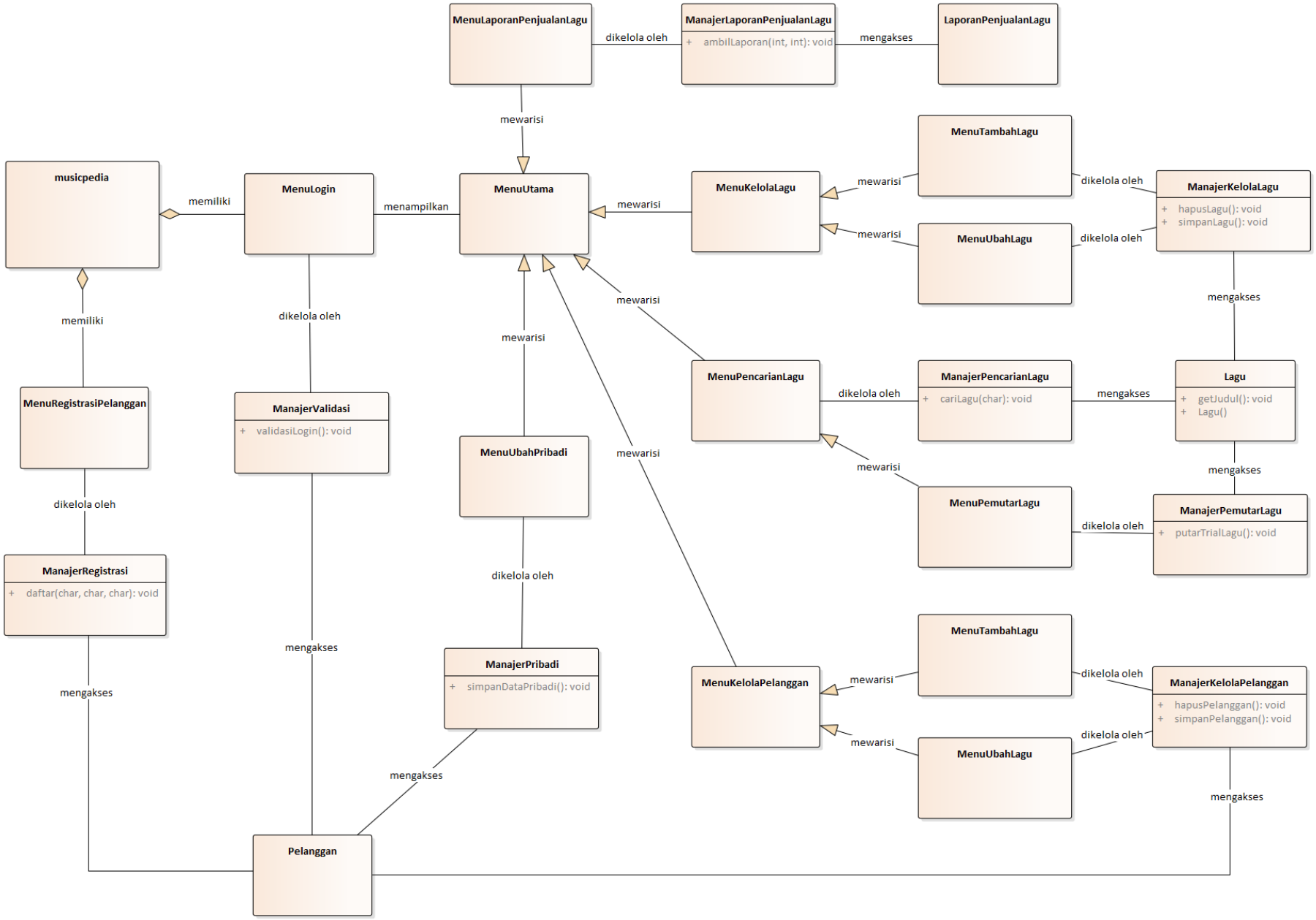

```
public class MenuUtama {

    ......

}
```

```
public class MenuPengambilanUang
        extends MenuUtama {

    .....

}
```

Class Diagram: Sistem ATM



Class Diagram: MusicPedia



Exercise: Class Diagram

1. Lihat kembali System Request yang sudah anda buat
2. Pahami kembali Activity Diagram, Use Case Diagram, dan Sequence Diagram yang telah anda buat
3. Lanjutkan dengan membuat Class Diagram
4. Masukkan method (message di Sequence Diagram) ke dalam setiap class
5. Generate code secara otomatis dari Class Diagram tersebut

Exercise: Systems Analysis and Design

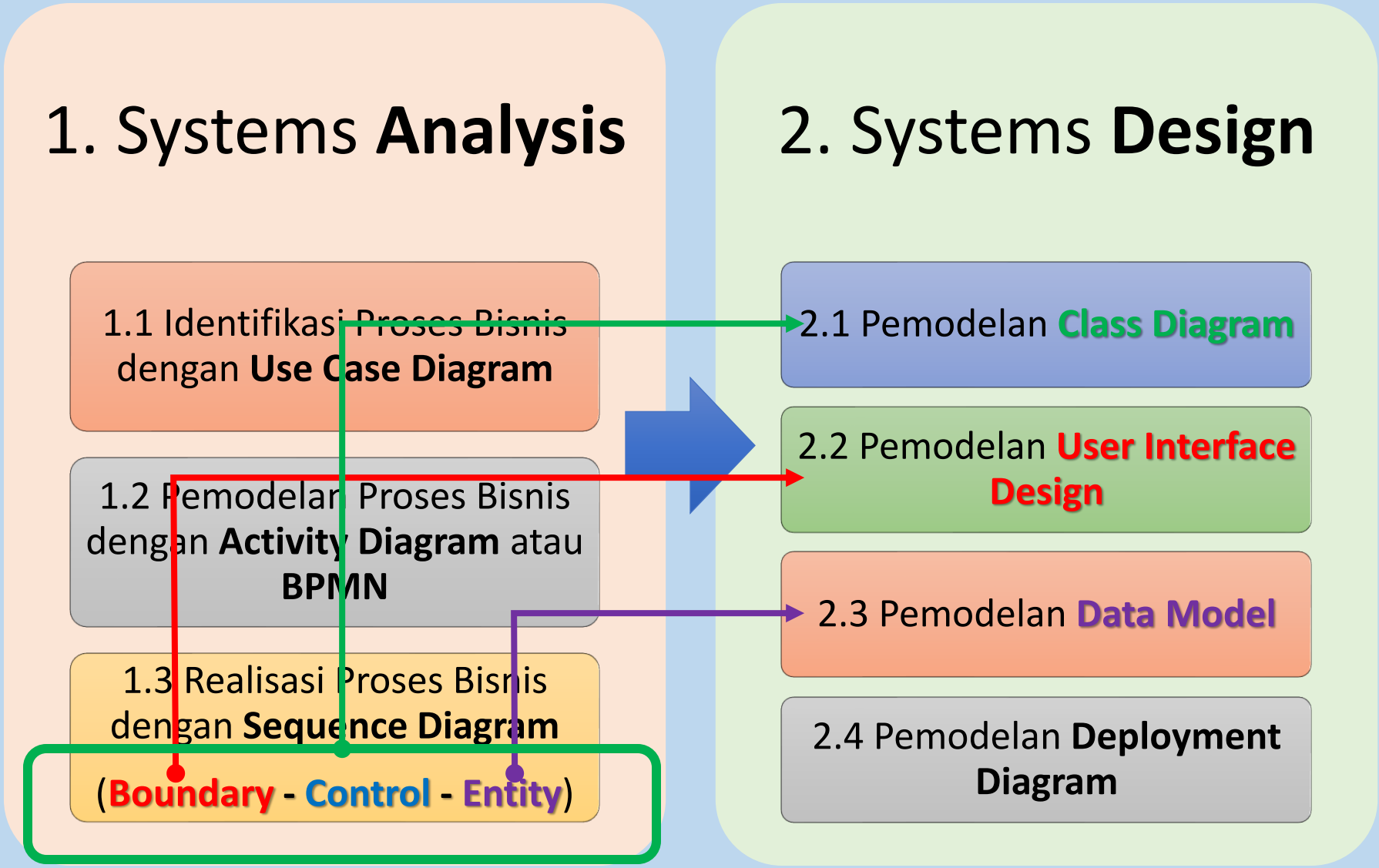
- Lakukan sistem analysis and design yang menghasilkan diagram:
  1. Use Case Diagram
  2. Activity Diagram
  3. Sequence Diagram
  4. **Class Diagram**

- Pilih salah satu aplikasi di bawah:
  1. Aplikasi Rental Mobil
  2. Aplikasi Pengelolaan Klinik
  3. Aplikasi Pengelolaan Apotik
  4. Aplikasi Pengelolaan Service Mobil
  5. Aplikasi Penjualan Motor
  6. Aplikasi Pengelolaan Perpustakaan
  7. Aplikasi Penjualan Buku Online
  8. Aplikasi Penjualan Tiket Kereta Online
  9. Aplikasi Manajemen Universitas Online
  10. Aplikasi Penjualan Laptop Online
  11. Aplikasi Perpustakaan Digital
  12. Aplikasi Pengelolaan Project Software

4.3 Pemodelan User Interface Design

UML based Software Analysis and Design

(Wahono, 2009)



4.3.1 User Interface Concepts

Philosophy and Principles

- Interface design is an art
- Balance between
  - Making the interface useful and
  - Presenting too much information

- Principles for User Interface Design:
  1. Layout: Consistent use of screen area
  2. Content awareness: Users know where they are
  3. Aesthetics: White space vs. functionality
  4. User experience: Ease of use vs. learning curve
  5. Consistency: User can predict what will happen for each action
  6. Minimal user effort: Simple to use, three click rule

1. Layout Concepts

- The screen is often divided into three boxes
  1. Navigation area (top)
  2. Status area (bottom)
  3. Work area (middle)
- Information can be presented in multiple areas, like areas should be grouped together
- Areas and information should minimize user movement from one to another
- Ideally, areas will remain consistent in:
  - Size, Shape, Placement for entering data, Reports presenting retrieved data
- Biggest Problem:
  - Too much information to process
- Provide a logical structure
  - Chronological
  - Most used to least used
  - General to specific

Layout Example

System Navigation

Report and form area

2. Content Awareness

- Make the user aware of the content without even knowing it
- All interfaces should have titles
- Menus should show
  - where you are
  - where you came from to get there

3. Aesthetics

- Interfaces need to be functional and inviting to use
- Pleasing to the eye
- Avoid squeezing in too much
  - Particularly for novice users
  - Experienced users can handle more info
- Design text carefully
  - Be aware of font and size
  - Avoid using all capital letters
- Colors and patterns should be used carefully
  - Test quality of colors
    - Try the interface on a monochrome monitor
  - Use colors to separate or categorize items
  - Don't rely on color alone to convey meaning

4. User Experience

- How easy is the program to learn?
- How easy is the program to use for the expert?
  - Consider shortcuts for the expert
- Some applications are used by many people
  - But not often by any of them → Focus on easy learning
- Some applications used frequently
  - By the same people → Focus on ease of use

5. Consistency

- Lets users predict what will happen
- Reduces learning curve
- Considers items within an application and across applications (Windows API)
- Pertains to many different levels
  - Navigation controls
  - Terminology
    - Same word $\Rightarrow$ same meaning
  - Report and form design

6. Minimize Effort

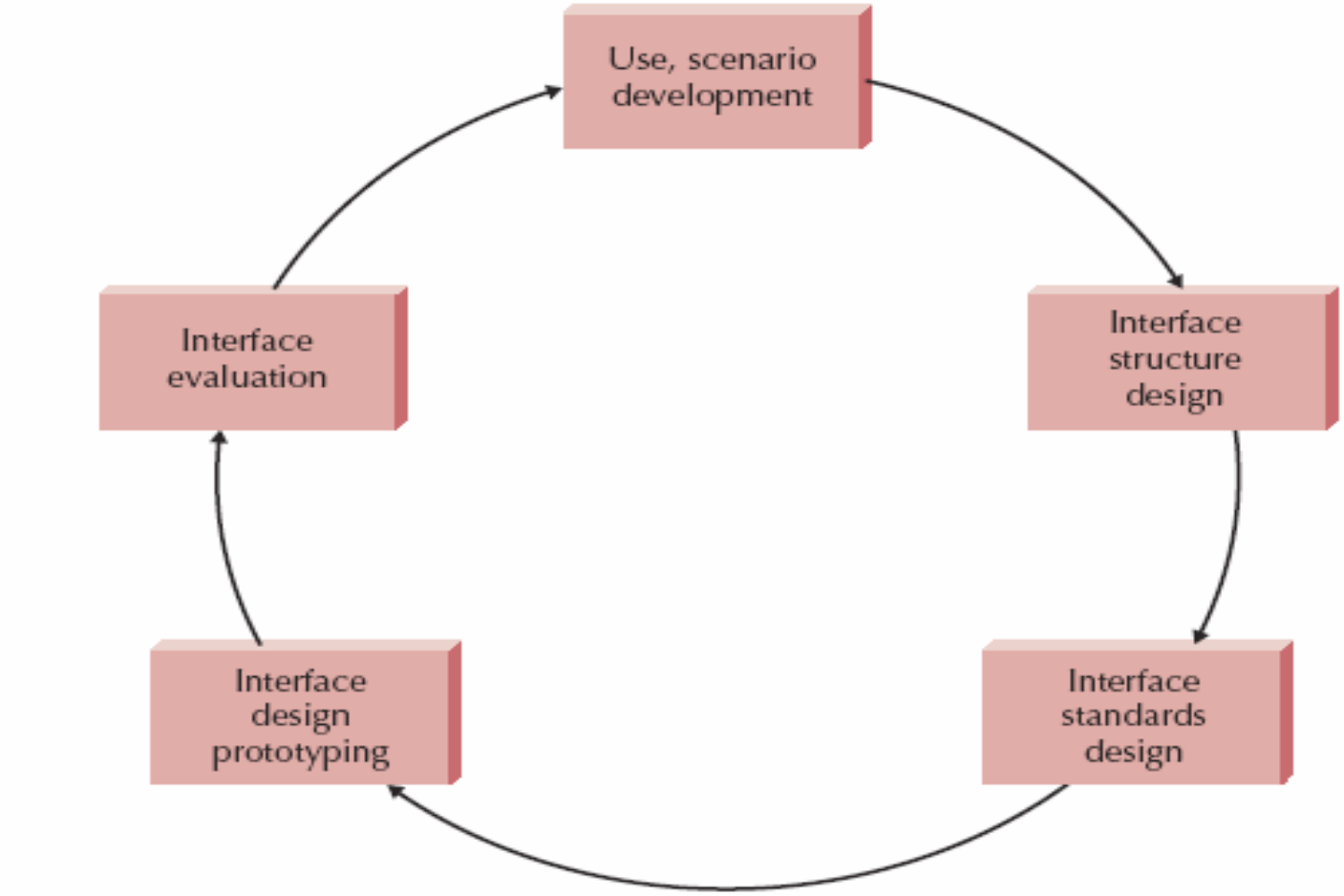
- Three click rule
  - Users should be able to go from the main menu to the information they want
    - in no more than three clicks or keystrokes

4.3.2 Process of User Interface Design

Five Step Process

1. Look at use cases and sequence diagrams
   - Develop use scenarios:
     - How will the users use the interface
   - ID common user actions
2. Develop Window Navigation Diagram (WND)
3. Design Interface Standards
   - Basic design elements upon which interfaces are designed
4. Create Interface Design Prototype
   - One for each interface in the system
   - Use real use cases
5. Perform interface evaluations
   - Are the interfaces ok?

Interface Evaluation



1. Use Scenario Development

- Outline of steps users use to perform their work
- Presented in a simple narrative
- Document the most common cases so interface designs will be easy to use for those situations

2. Interface Structure Design

- Window navigation diagram (WND)
  - Shows how all screens, forms, and reports are related
  - Shows how user moves from one to another
  - Like a state diagram for the user interface
    - Boxes represent components
    - Arrows represent transitions
    - Stereotypes show interface type

Window Navigation Diagram

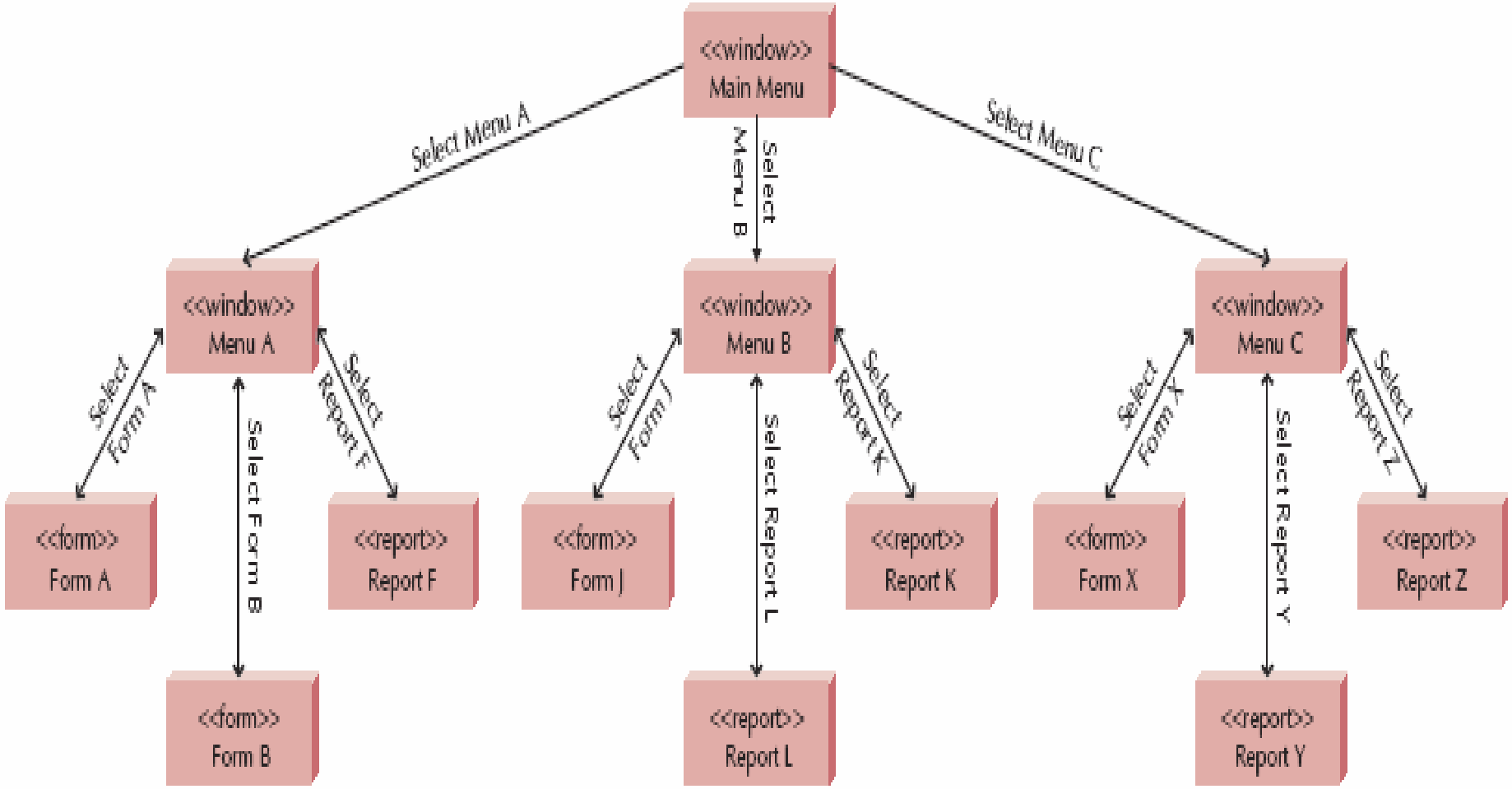


FIGURE 12-7 An Example Window Navigation Diagram

3. Interface Standards Design

- The basic elements that are common across
  - Screens
  - Forms
  - Reports
- Interface metaphor
  - A concept from the real world
  - Used as a model for the computer system
  - Desktop, checkbook, shopping cart

4. Interface Elements

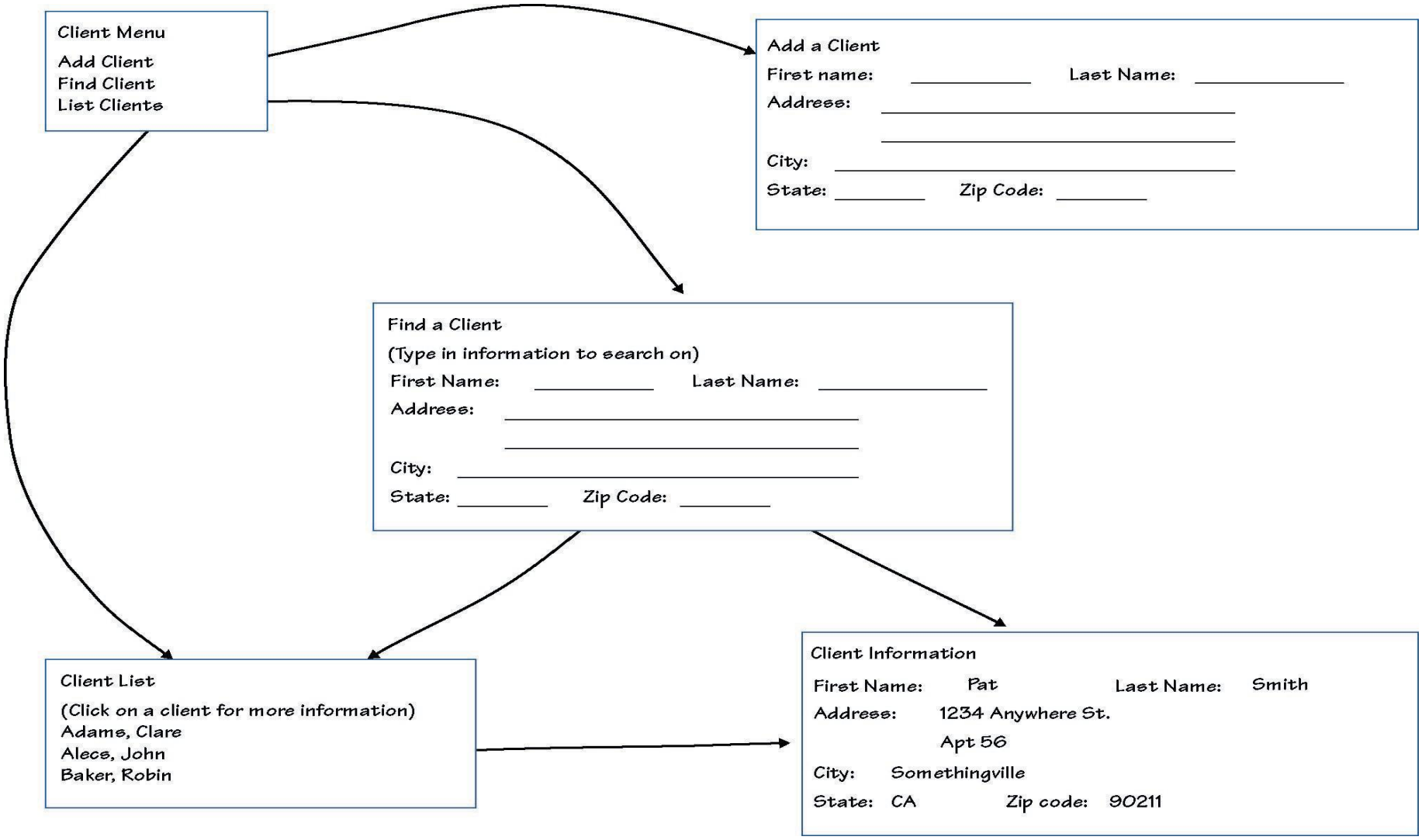
- Interface objects
  - Building blocks of the interface
  - Give each object an understandable name
  - Understandable names are better than precise names
- Interface actions
  - Navigation and command
    - Menu, buttons, drop-down lists
  - Grammar
    - Buy vs. purchase
    - Modify vs. change
- Interface icons
  - Represent interface objects & actions
- Interface templates
  - Templates for screens, forms, reports

4. Interface Design Prototyping

- A mock-up or simulation of screen, form, or report

- Common methods include
  - Paper Forms
  - Storyboarding
  - HTML
  - Language

Storyboard Example



5. Interface Evaluation Methods

- Understand how to improve the interface
- Should be done before system is finished
- Heuristic evaluation
  - Check the prototype
  - Design by checklist
  - At least three people should participate
- Walkthrough evaluation
  - Walk users through the system
  - Team simulates movement through components
- Interactive evaluation
  - Use prototype
  - User tries out the system with a team member present
- Formal usability testing
  - Expensive
  - Detailed use of special lab testing

4.3.3 Navigation Design

Basic Principles

- Assume users:
  - Have not read the manual
  - Have not attended training
  - Do not have external help readily at hand

- Controls should be:
  - Clear and understandable
  - Placed in an intuitive location on the screen
  - Anticipate what the user will do

Basic Principles

- Prevent mistakes
  - Limit choices
  - Never display commands that can't be used
  - Confirm actions that are difficult or impossible to undo
- Simplify recover from mistakes
  - No matter how hard you try, users will make mistakes
  - If possible:
    - Have an "undo"
    - Transaction rollbacks
    - Automatic backups
- Use consistent grammar order
  - Normally specify an Action and an Object
  - Can use Action-Object, or Object-Action
  - Windows uses Object-Action (cut and paste)
  - Whatever you choose, it should be used consistently

Types of Navigation Control

- Languages
  - Command language
    - User enters commands using special language
    - UNIX, DOS, SQL, etc.
    - Hard to learn, fast and easy to use
  - Natural language
    - Easy to learn
    - Slow and imprecise
- Menus
  - Generally aim at broad shallow menus
  - Consider using "hot keys"
- Direct Manipulation
  - Used with icons to start programs
  - Used to shape and size objects
  - May not be intuitive for all commands

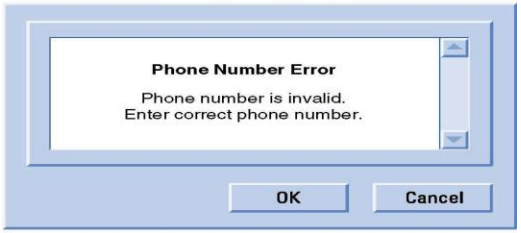
Message Tips

- Should be clear, concise, and complete
- Should be grammatically correct and free of jargon and abbreviations (unless they are the users)
- Avoid negatives and humor

Stating corrective action before the problem explanation can be confusing.

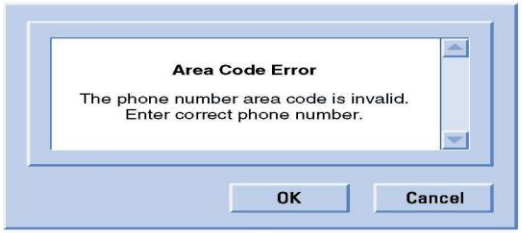
(a)

Stating the problem before the corrective action is more intuitive.



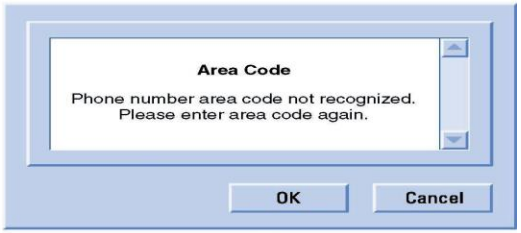
(b)

The problem should be stated in as much detail as possible.



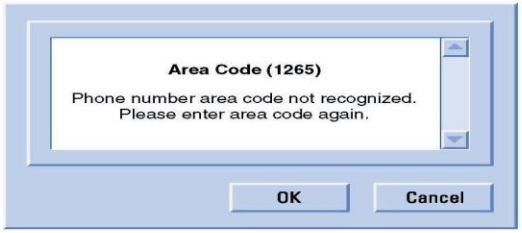
(c)

The error message should be polite.



(d)

Error messages should include a message number.



(e)

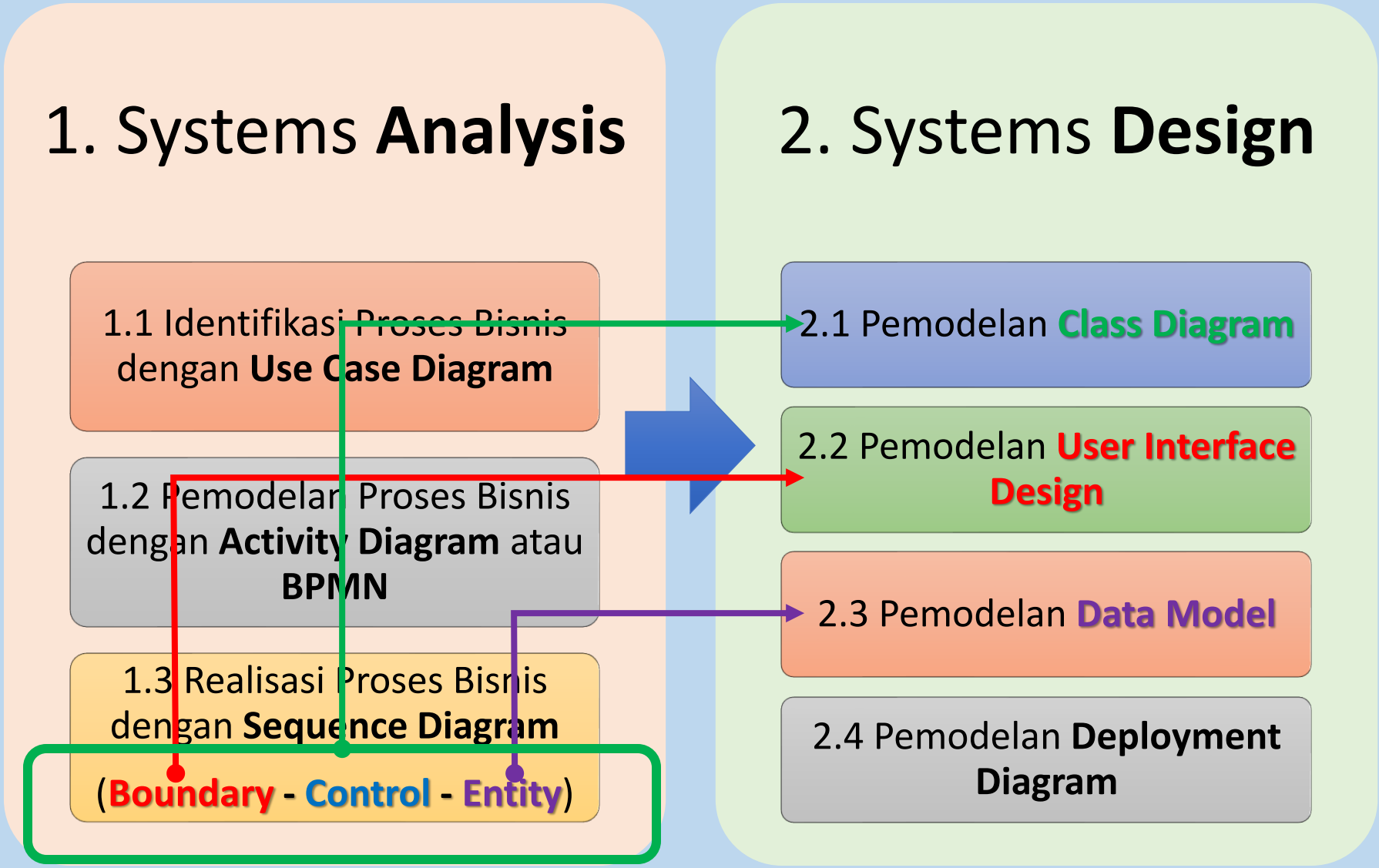
Exercise: User Interface Design

1. Lihat kembali Activity Diagram, Use Case Diagram, Sequence Diagram dan Class Diagram yang telah anda buat
2. Lanjutkan dengan membuat User Interface Design dengan menggunakan Java Frame dengan Netbeans
3. User Interface Design diekstraksi dari Entity Class

4.4 Pemodelan Data Model

UML based Software Analysis and Design

(Wahono, 2009)



What is Database

- Database system is a computer based record keeping system
- It is a system whose overall purpose is to record and maintain information that is deemed important to the organization
- Database is collection of stored operational data which can be used and shared by different applications and users of any organization

Why Database

- Database system provides the organization with centralized control of its operational data, which is one of its most valuable assets

- This is totally opposite of the situation that is happening in many organizations, where typically each application has its own private files (flat file)
  - This makes the operational data widely dispersed and difficult to control

Advantage of Centralized Database

- Redundancy can be reduced
- Inconsistency can be avoided
- Data can be shared
- Standards can be enforced
- Security restrictions can be applied
- Integrity can be maintained
- Conflicting requirements can be balanced

Disadvantage of Database Systems

- Database is more vulnerable to destruction thru:
  - machine malfunction
  - personal error
  - Deliberate human tampering

- Cost: the cost of required hardware, DB development, and DB maintenance is high

- Complexity: Due to its complexity, the user should understand it well enough to use it efficiently and effectively

Database Models - Product - Vendor

MODEL	PRODUCT	VENDOR
1. Relational	DB2	IBMSQL/DS
	Ingress	Relational Tech.
	Oracle	Oracle corp
	Access	Microsoft
	PostgreSQL	
	MySQL	
2. Network	DMS100	Unysis
	IDMS	Cullinet
3. Heirarchical	IMS	IBM
	System 2000	Intel
4. Object oriented	Starburst	IBM
	Gemstone	
	Orion	

Relational Database

- Relational database is a collection of tables
- Formally a table is called a relation
- Database is a structure that can hold information about tables, rows, and columns

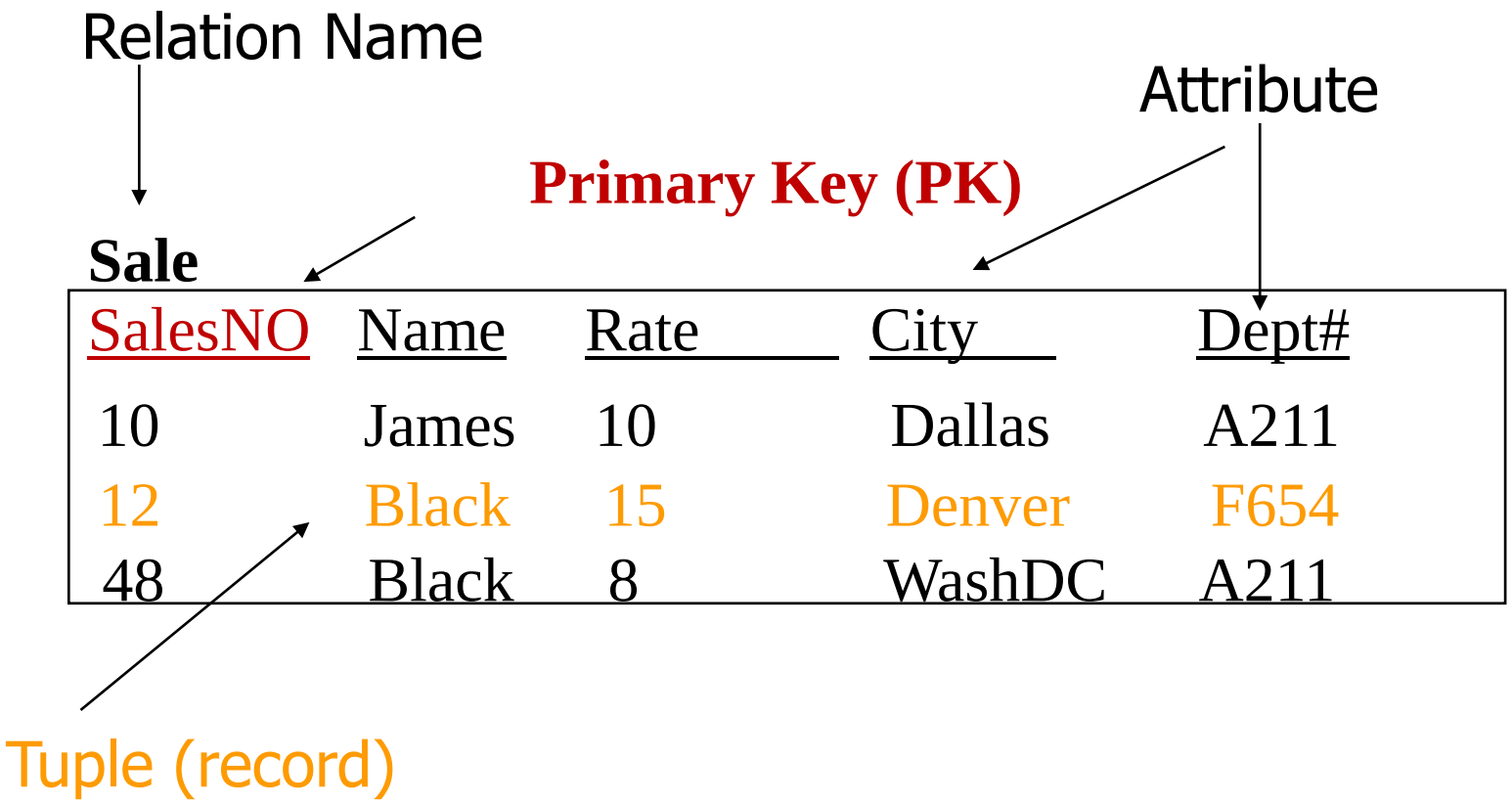
Relational Model	Relational DBMS	Traditional File System
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field
Primary Key (PK)	Primary Key (PK)	Search Key
Relationship (FK)	Relationship (FK)	Not Used

Relational Database

1. Primary Key (PK): An attribute which can uniquely identify each record (tuple) of a relation (table)

2. Foreign Key (FK): An attribute which is a regular attribute in one table but a primary key in another table

Example of a Relational Database



Example of a Relational Database

Customer

CustID	Name	Balance	City	SaleNo
132	Black	2000.00	Dallas	10
135	Tom	129.89	Denver	12
198	Tom	(132.90)	Dallas	10

SalesNO is PK in Sales table

Sales

SalesNO	Name	Rate	City	Dept#
10	James	10	Dallas	A211
12	Black	15	Denver	F654
48	Black	8	WashDC	A211

Example: Order Entry Database

Order

ONO	DATE	CustID	SalesNO
102	11/2/94	132	10
199	2/15/95	135	12
92	10/4/94	102	53

OrderLine

ONO	Oline#	Part#	Qty	Part#
102	1	12.00	10	EX454
102	2	129.89	1	DE012
199	1	32.90	3	DC810

Customer

CustID	Name	Balance	City	SaleNo
132	Black	2000.00	Dallas	10
135	Tom	129.89	Denver	12
198	Tom	(132.90)	Dallas	10

Sales

SalesNO	Name	Rate	City	Dept#
10	James	10	Dallas	A211
12	Black	15	Denver	F654
48	Black	8	WashDC	A211

Functionality of a DBMS

- The programmer sees SQL, which has two components:
  1. Data Definition Language (DDL)
  2. Data Manipulation Language (DML)

- Behind the scenes the DBMS has:
  1. Query engine
  2. Query optimizer
  3. Storage management
  4. Transaction Management (concurrency, recovery)

How the Programmer Sees the DBMS

1. Start with DDL to *create tables*:

```
CREATE TABLE Students (
        Name CHAR(30)
        SSN CHAR(9) PRIMARY KEY NOT NULL,
        Category CHAR(20)
)  . . .
```

2. Continue with DML to *populate tables:*

```
INSERT INTO Students
VALUES('Charles', '123456789', 'undergraduate')
. . . .
```

Transactions

- Enroll “Mary Johnson” in “CSE444”:

```
BEGIN TRANSACTION;

INSERT INTO Takes
    SELECT Students.SSN, Courses.CID
    FROM Students, Courses
    WHERE Students.name = 'Mary Johnson' and
                Courses.name = 'CSE444'

-- More updates here....

IF everything-went-OK
     THEN COMMIT;
ELSE ROLLBACK
```

If system crashes, the transaction is still either committed or aborted

Transactions

- A transaction = sequence of statements that either all succeed, or all fail
- Transactions have the ACID properties:
  1. A = atomicity (a transaction should be done or undone completely)
  2. C = consistency (a transaction should transform a system from one consistent state to another consistent state)
  3. I = isolation (each transaction should happen independently of other transactions)
  4. D = durability (completed transactions should remain permanent)

Queries

- Find all courses that “Mary” takes

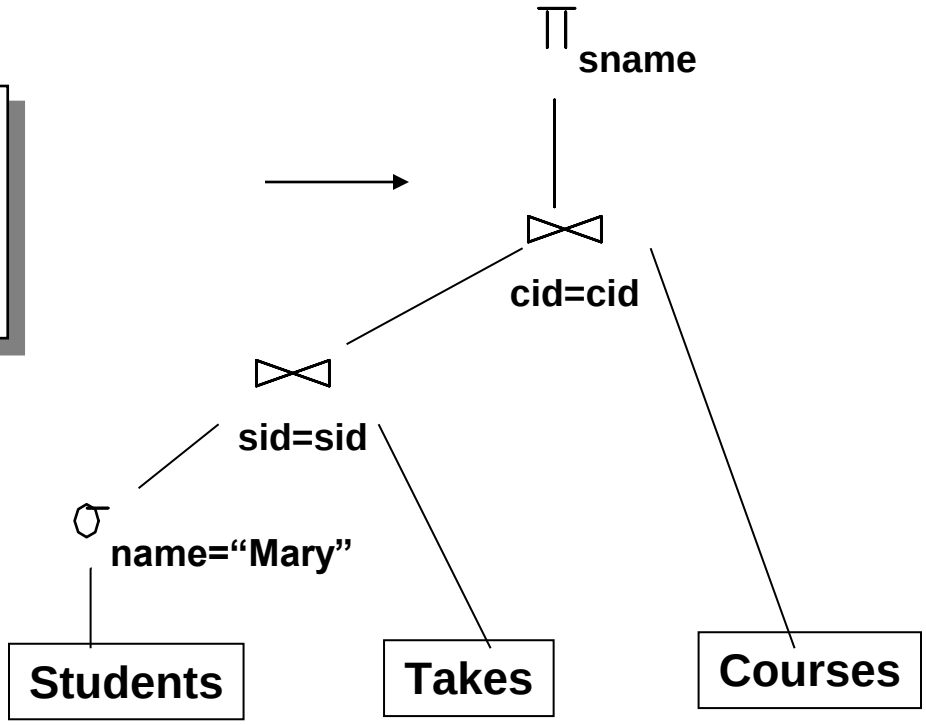
```
SELECT  C.name
FROM    Students S, Takes T, Courses C
WHERE   S.name=“Mary” and
                S.ssn = T.ssn and T.cid = C.cid
```

- What happens behind the scene ?
  - Query processor figures out how to answer the query efficiently.

Queries, Behind the Scene

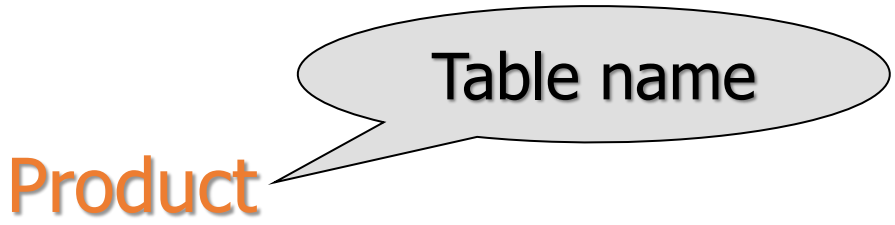
Declarative SQL query → Imperative query execution plan:

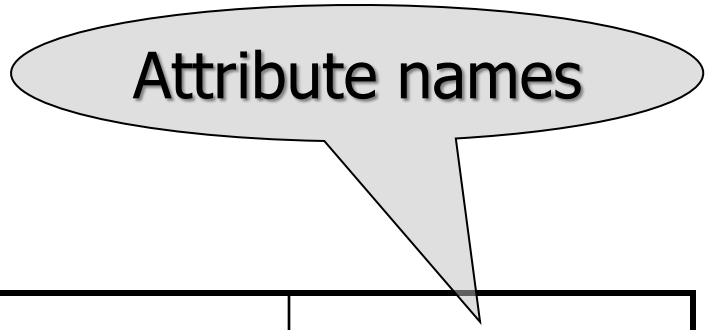
```
SELECT  C.name
FROM Students S, Takes T, Courses C
WHERE S.name="Mary" and
        S.ssn = T.ssn and T.cid = C.cid
```



The **optimizer** chooses the best execution plan for a query

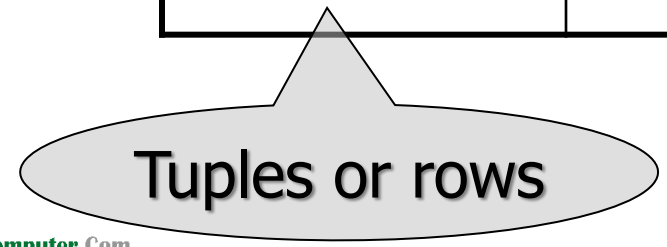
Tables in SQL





Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi



Tables Explained

- A tuple = a record
  - Restriction: all attributes are of atomic type
- A table = a set of tuples
  - Like a list…
  - …but it is unorderd: no **first()**, no **next()**, no **last()**.
- No nested tables, only flat tables are allowed!

Tables Explained

- The schema of a table is the table name and its attributes:

  Product(PName, Price, Category, Manfacturer)

- A key is an attribute whose values are unique; we underline a key

  Product(PName, Price, Category, Manfacturer)

SQL Query

Basic form: (plus many many more bells and whistles)

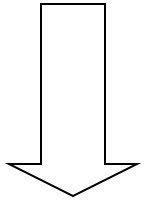
```
SELECT  attributes
FROM    relations (possibly multiple, joined)
WHERE   conditions (selections)
```

Simple SQL Query

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  *
FROM    Product
WHERE   category='Gadgets'
```



PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

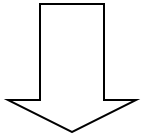
"selection"

Simple SQL Query

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

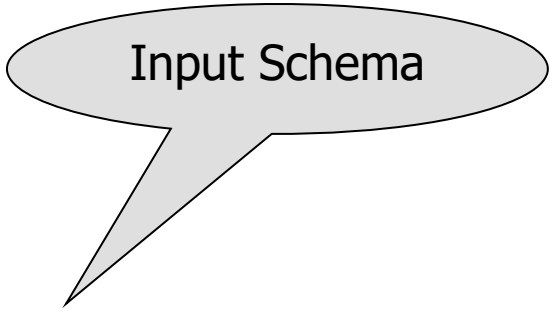
```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100
```



"selection" and "projection"

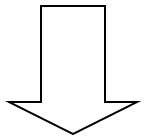
PName	Price	Manufacturer
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A Notation for SQL Queries

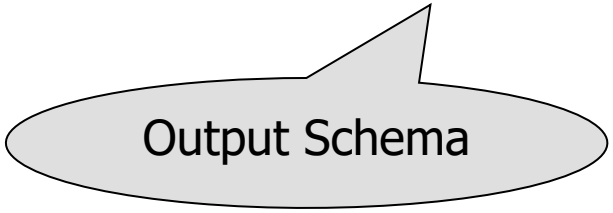


Product(PName, Price, Category, Manfacturer)

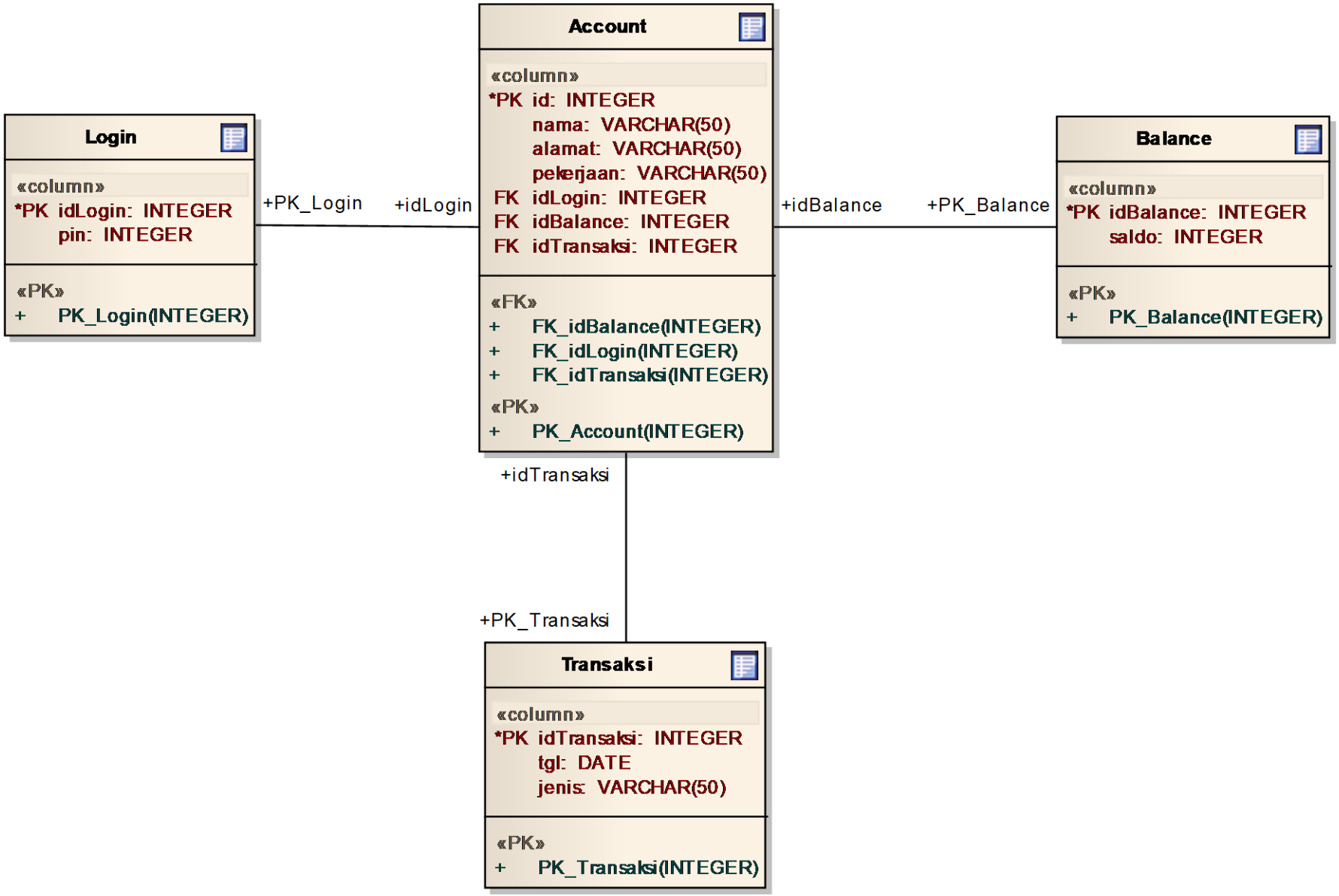
```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100
```



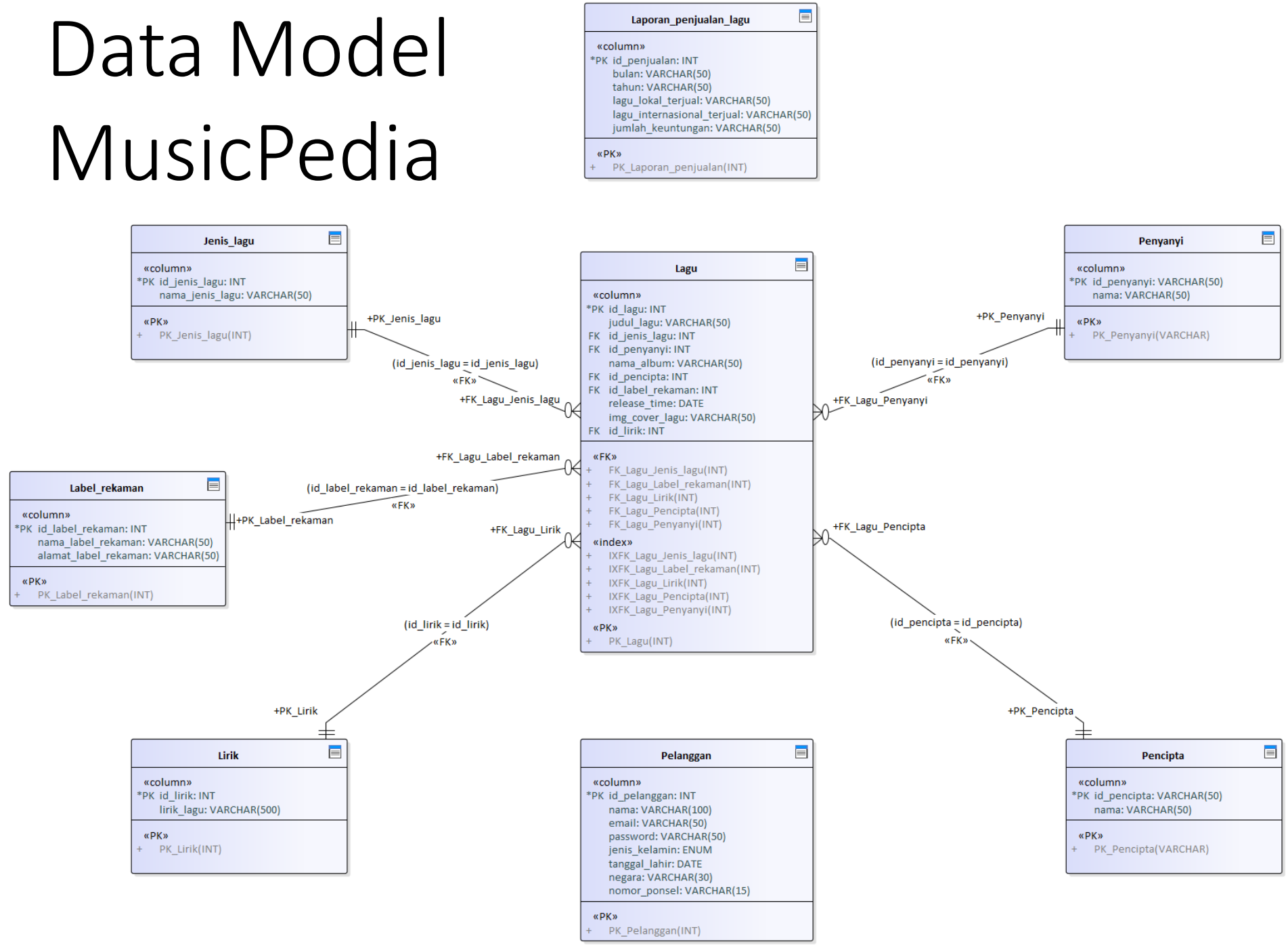
Answer(PName, Price, Manfacturer)



Data Model Sistem ATM



Data Model MusicPedia



Exercise: Data Model

1. Lihat kembali Activity Diagram, Use Case Diagram, Sequence Diagram dan Class Diagram yang telah anda buat
2. Lanjutkan dengan membuat Data Model
3. Data Model diekstraksi dari Boundary Class
4. Generate SQL code dari Data Model yang dibuat

4.5 Pemodelan Deployment Diagram

Deployment Diagram

- Servers
  - Mainframes, Minis, Micros
- Clients
  - Input/Output HW used by users
  - Terminals, PCs, special purpose HW
- Network
  - HW and SW to connect clients to servers
- Nodes
  - Any piece of hardware in the model
  - A computational resource
  - Labeled by its name
  - Stereotype to label the type of node

Node

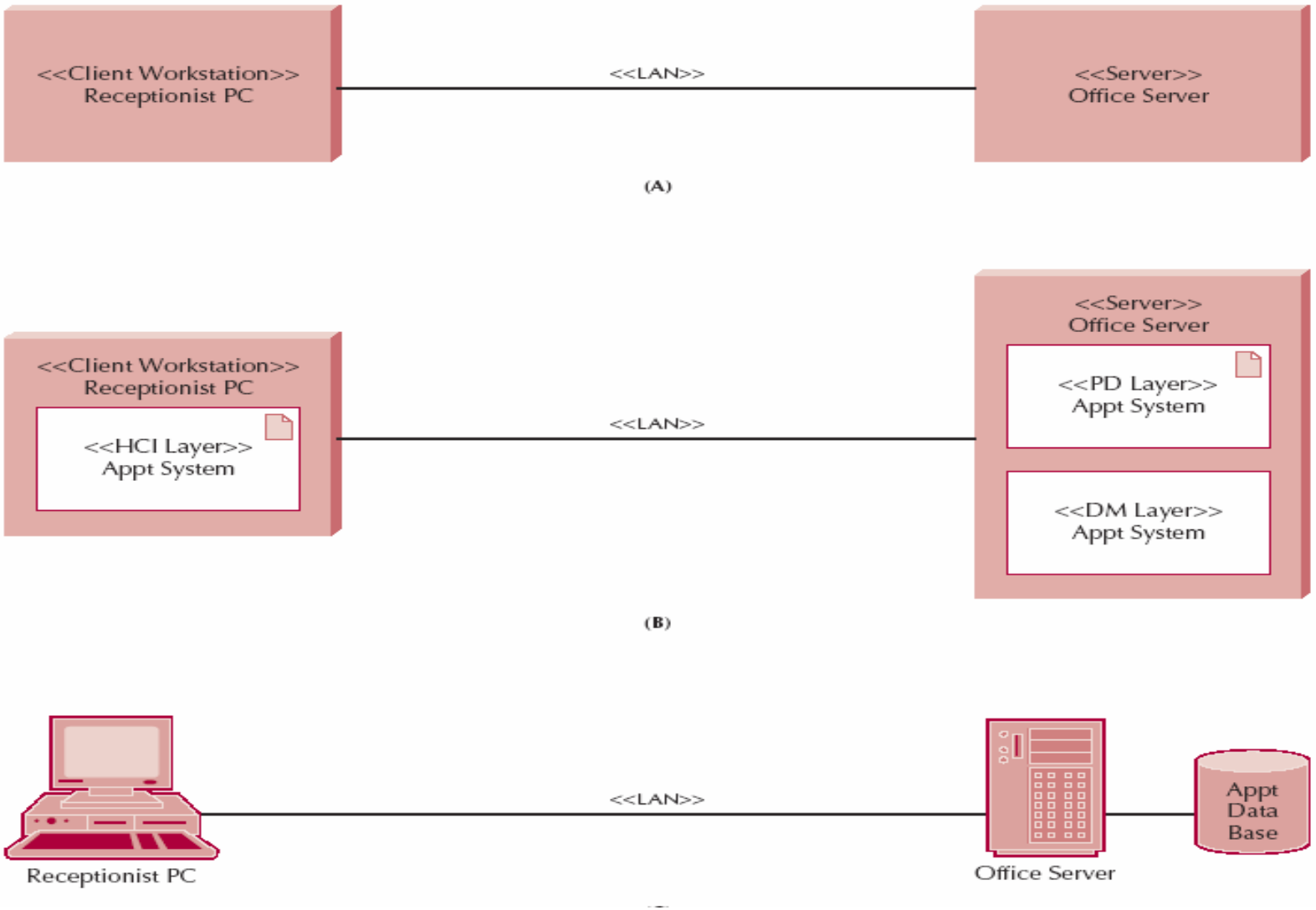
Artifact

Node with Deployment Artifact

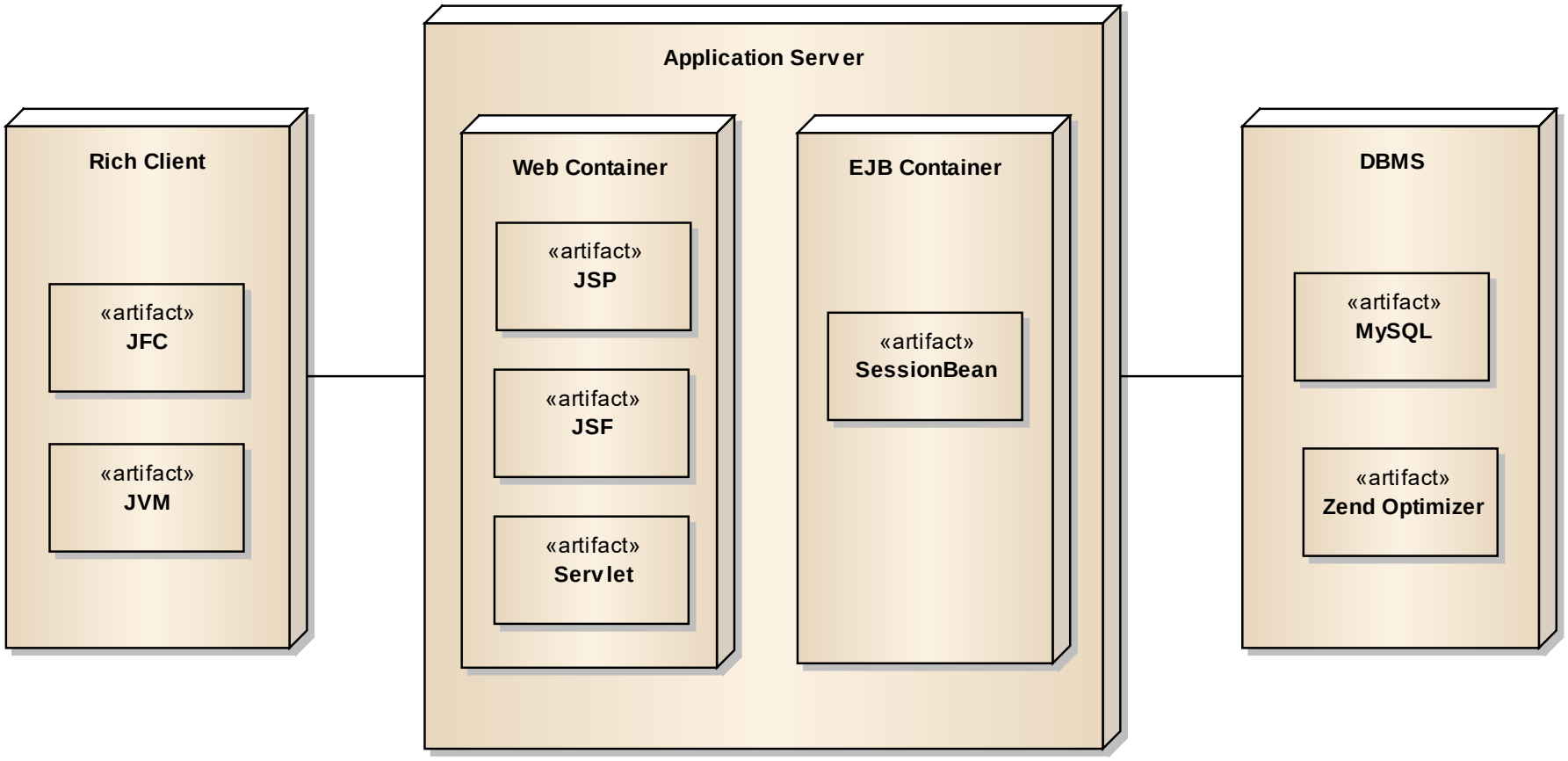
Communication Path

- Artifacts
  - Piece of the information system, such as software or a database table
- Node with Deployed Artifact
  - Shows artifact placed on a physical node
  - Good for showing distribution data or software
- Communication paths
  - Links between nodes of the network

Diagram Examples

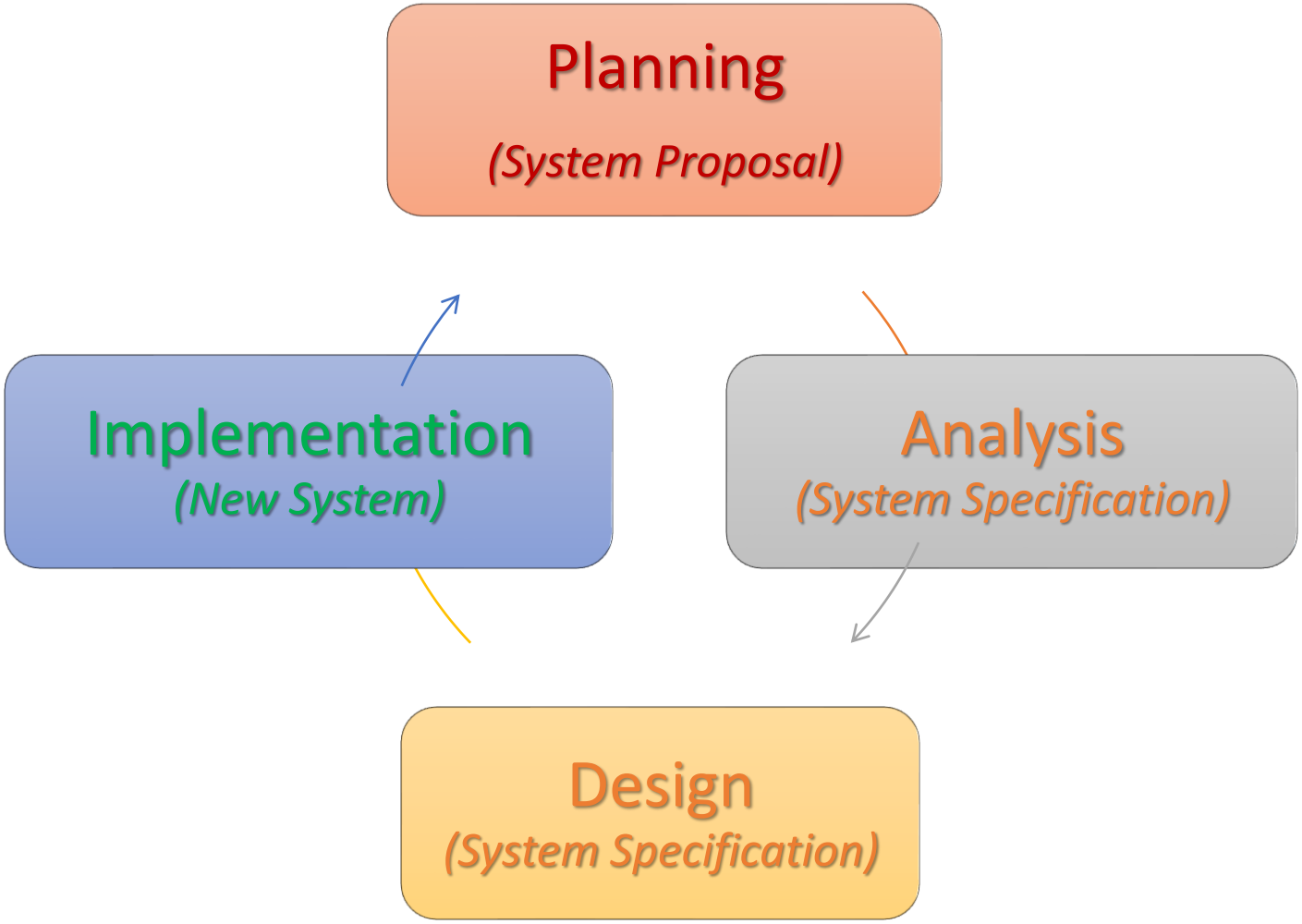


Deployment Diagram (3 Tier)



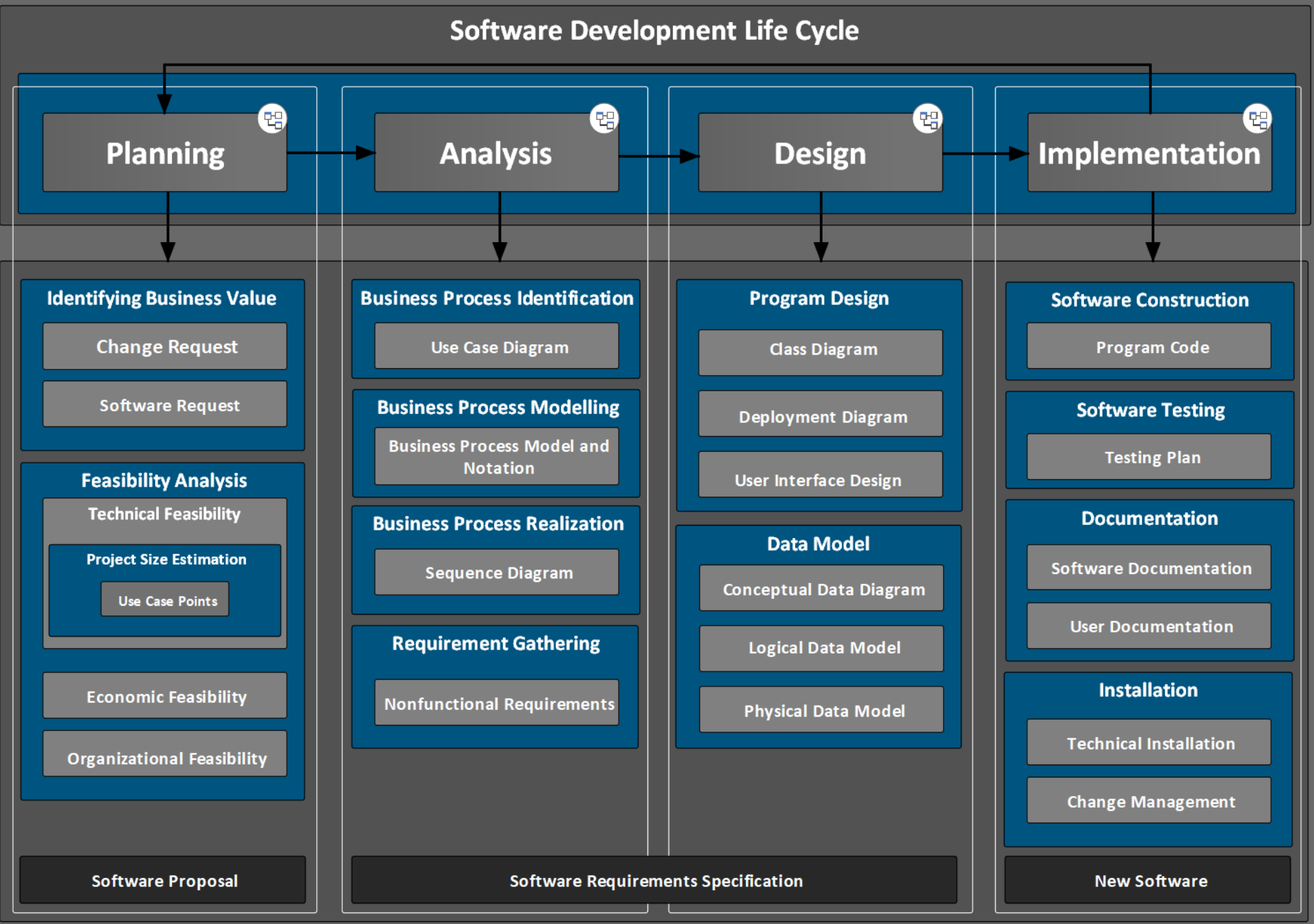
Rangkuman Studi Kasus Sistem ATM

Siklus Pengembangan Software

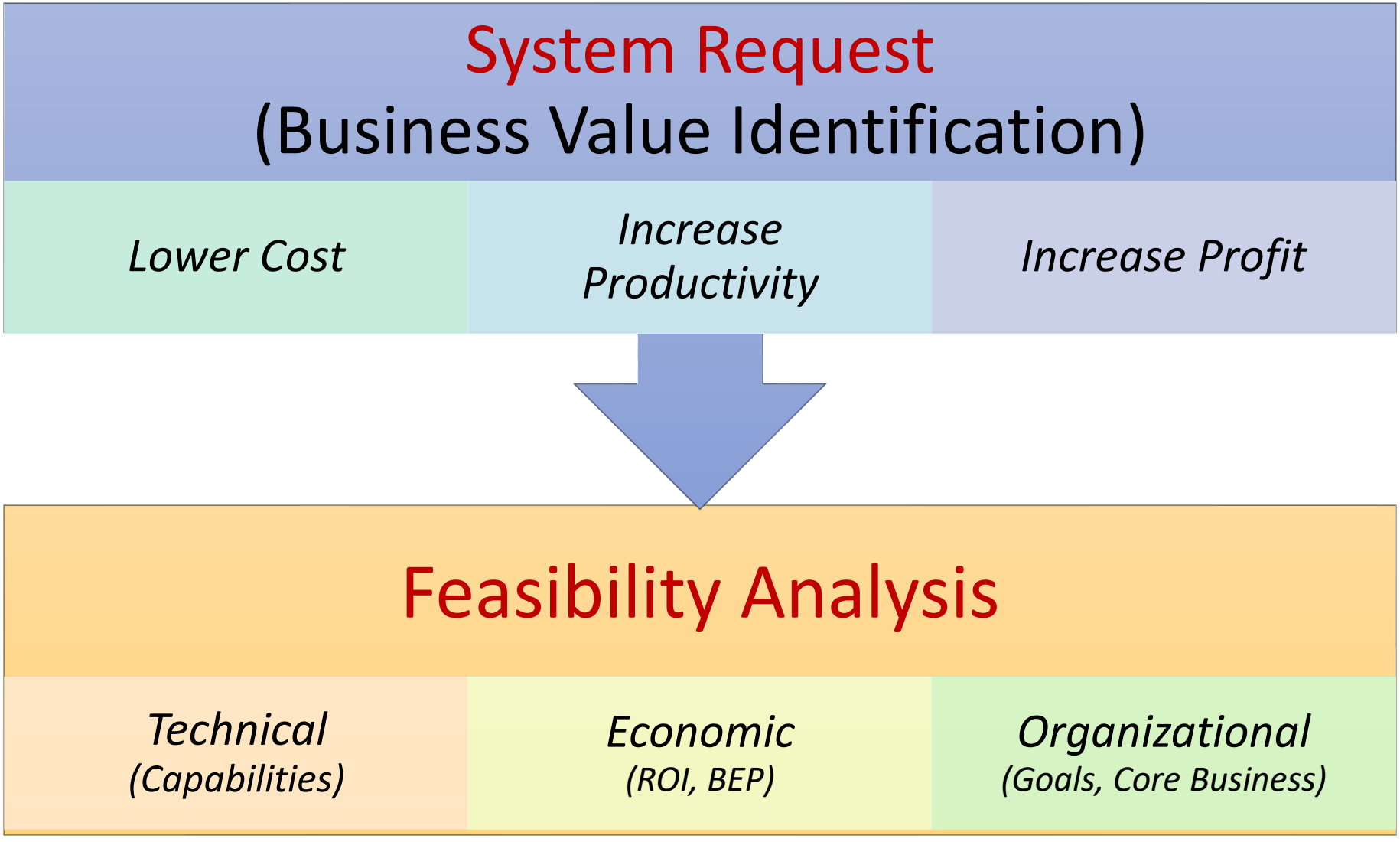


(Tilley, 2012) *(Dennis, 2016)* *(Valacich, 2017)*

Application Development Governance



Planning


System Request
(Business Value Identification)
Lower Cost
Increase Productivity
Increase Profit
Feasibility Analysis
Technical
(Capabilities)
Economic
(ROI, BEP)
Organizational
(Goals, Core Business)

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
2. Memberikan layanan pendukung dengan menggunakan internet

Business Requirements:

Sistem yang mendukung penjualan musik secara online. Fitur-fitur yang harus ada:

1. Fitur Pencarian Produk
2. Fitur Pencarian Toko yang Menyediakan Stok Produk
3. Fitur Pemesanan Produk Melalui Toko yang Menyediakan
4. Fitur Pembayaran dengan Berbagai Pilihan Pembayaran

Business Value:

Intangible Value:

- Meningkatkan kenyamanan dan kepuasan pelanggan
- Meningkatkan brand recognition tentang perusahaan di dunia Internet

Tangible Value:

1. Meningkatkan penjualan dari pelanggan baru lewat Internet:
   - Rp 400 juta peningkatan penjualan dari pelanggan baru dan Rp 600 juta dari pelanggan lama
2. Mengurangi biaya operasional untuk menangani komplain dari pelanggan
   - Rp 100 juta pengurangan tahunan biaya telepon untuk menangani pelanggan

Studi Kelayakan Sistem Penjualan Musik Online

Margaret Mooney dan Alec Adams membuat studi kelayakan untuk pengembangan Sistem Penjualan Musik Online

Kelayakan Teknis

Sistem penjualan musik online layak secara teknis, meskipun memiliki beberapa risiko.

Risiko Berhubungan dengan **Kefamilieran dengan Aplikasi**: Resiko Tinggi

- Divisi Marketing tidak memiliki pengalaman menggunakan sistem penjualan online
- Divisi IT memiliki pemahaman yang baik tentang sistem penjualan offline, akan tetapi tidak berpengalaman mengembangkan sistem penjualan musik online

Risiko Berhubungan dengan **Kefamilieran dengan Teknologi**: Resiko Sedang

- Divisi IT tidak menguasai masalah infrastruktur dan ISP, tetapi akan menyewa konsultan
- Divisi IT cukup familier dengan framework dan IDE yang akan digunakan
- Divisi Marketing tidak memiliki pengalaman menggunakan teknologi Web

Risiko berhubungan dengan **Ukuran Project**: Risiko Rendah

- Perusahaan memiliki total 30 orang pengembang
- Project dikerjakan oleh 5 orang pengembang dengan estimasi waktu 6 bulan

Kompatibilitas dengan sistem dan infrastruktur yang ada: Risiko Rendah

- Sistem pemesanan yang ada sekarang menggunakan *open standard*, jadi sangat kompatibel dengan sistem penjualan berbasis web yang akan dibangun

Kelayakan Ekonomi

Cost benefit analysis telah dilakukan. Sistem Penjualan musik online memiliki peluang yang baik untuk bisa meningkatkan pendapatan perusahaan.

- Return on Investment (ROI) setelah 3 tahun: **31%**
- Break-even point (BEP): **2.25 tahun**
- Total keuntungan setelah 3 tahun: **Rp. 503.559.986,-**

Keuntungan Intangible

- Meningkatkan kepuasaan pelanggan
- Meningkatkan branding perusahaan

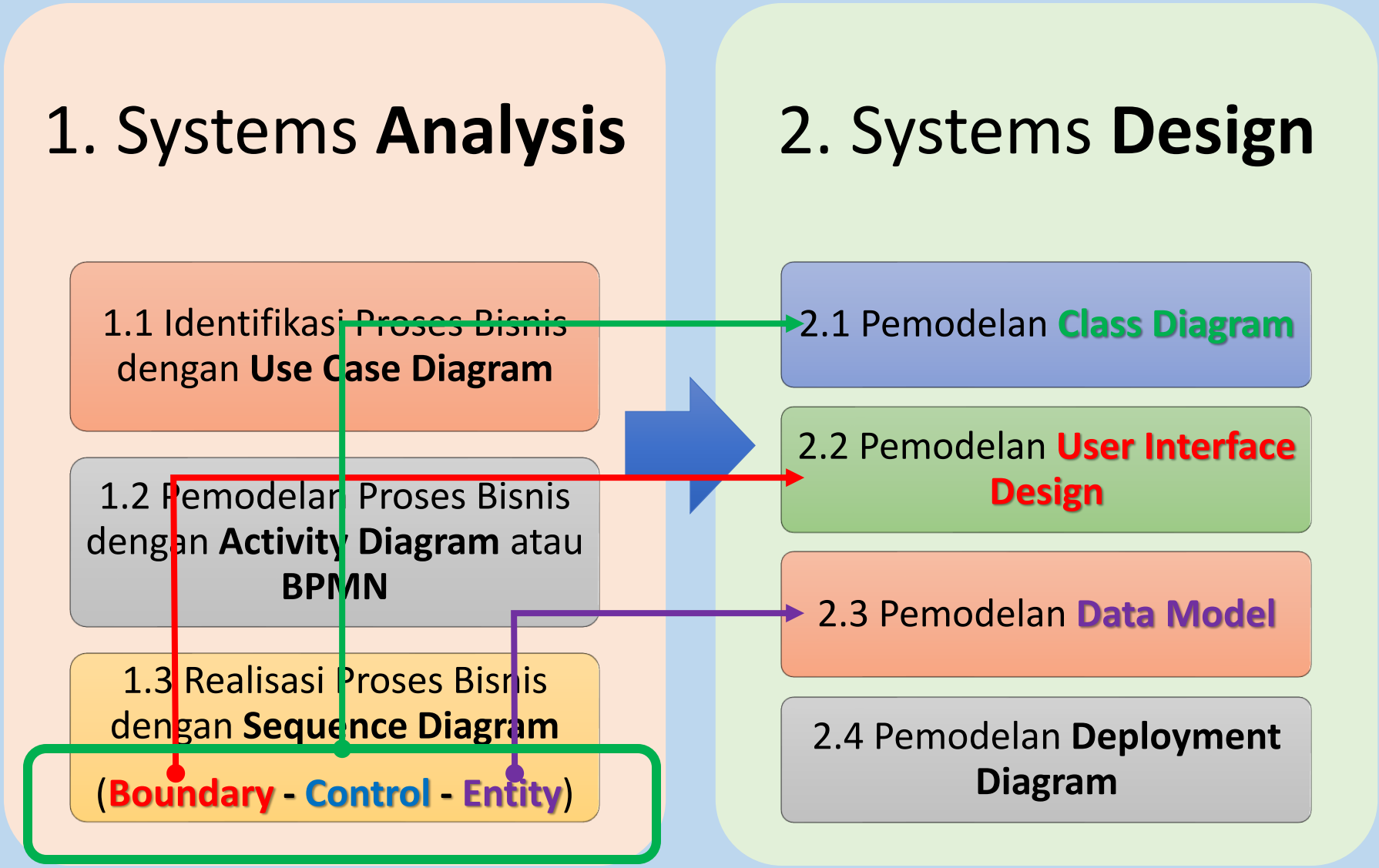
Kelayakan Organisasi

- Secara organisasi, resikonya rendah. Tujuan dari pengembangan sistem penjualan musik online adalah meningkatkan penjualan perusahaan. Dan ini selaras dengan KPI marketing yang ke arah peningkatan kuantitas penjualan
- Project champion dari pengembangan sistem penjualan musik online ini adalah Margaret Mooney, Vice President of Marketing

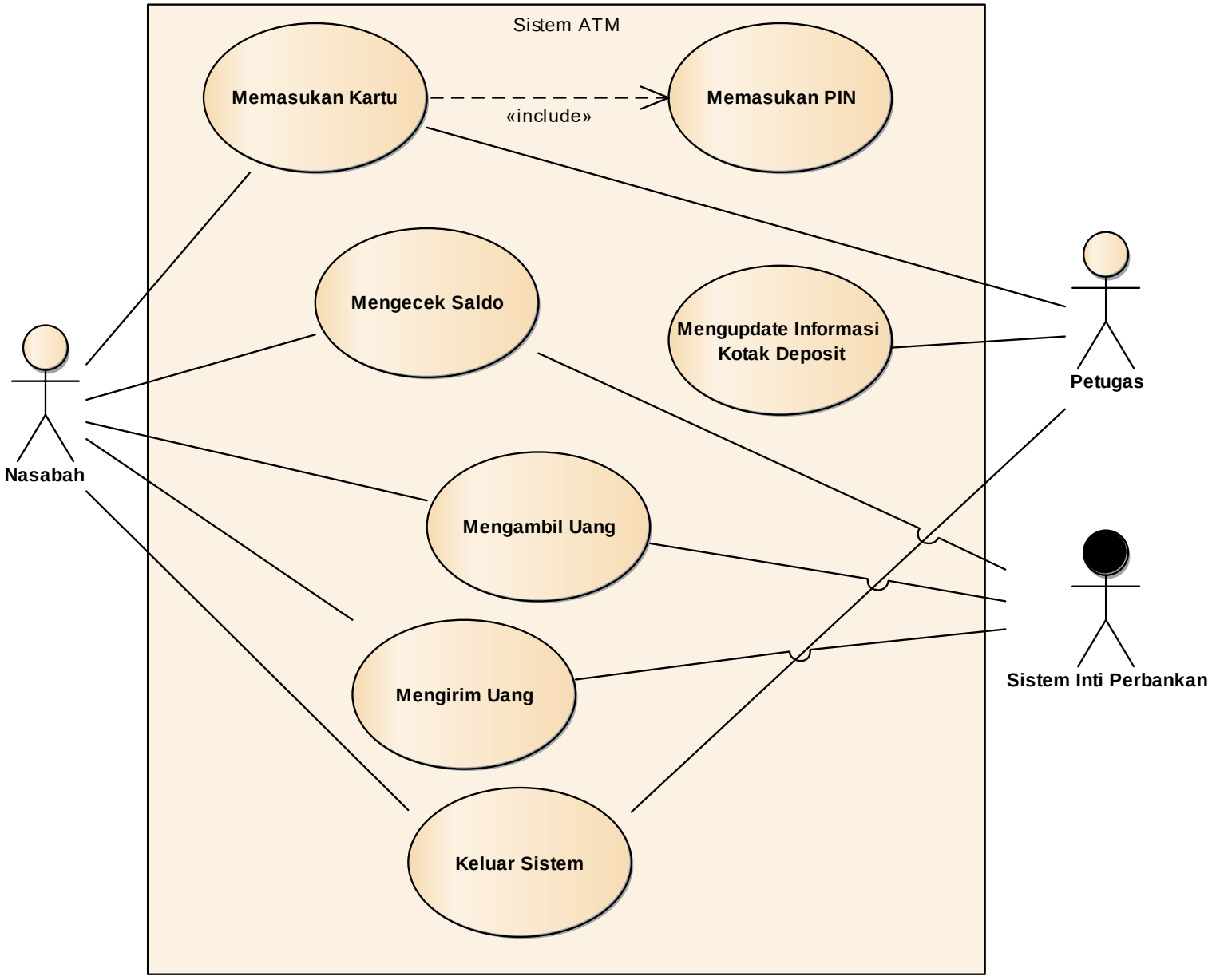
	2019	2020	2021
Peningkatan penjualan dari pelanggan baru	0	400,000,000	500,000,000
Peningkatan penjualan dari pelanggan lama	0	600,000,000	700,000,000
Pengurangan biaya operasional dan telepon	0	100,000,000	100,000,000
Total Benefits:	**0**	**1,100,000,000**	**1,300,000,000**
PV of Benefits:	**0**	**978,996,084**	**1,091,505,068**
PV of All Benefits:	**0**	**978,996,084**	**2,070,501,152**
Honor Tim (Analysis, Design and Implementation)	360,000,000	0	0
Honor Konsultan	90,000,000	0	0
Total Development Costs:	**450,000,000**	**0**	**0**
Honor Pengelola Web	60,000,000	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**	**669,811,321**
Cumulative NPV:	**-735,849,057**	**-166,251,335**	**503,559,986**
Return on Investment (ROI) di Tahun 3: 30.70%	**-100.00%**	**-0.145166304**	**0.306932213**
Break-even Point (BEP): 2.25 tahun			**2.248206218**

UML based Software Analysis and Design

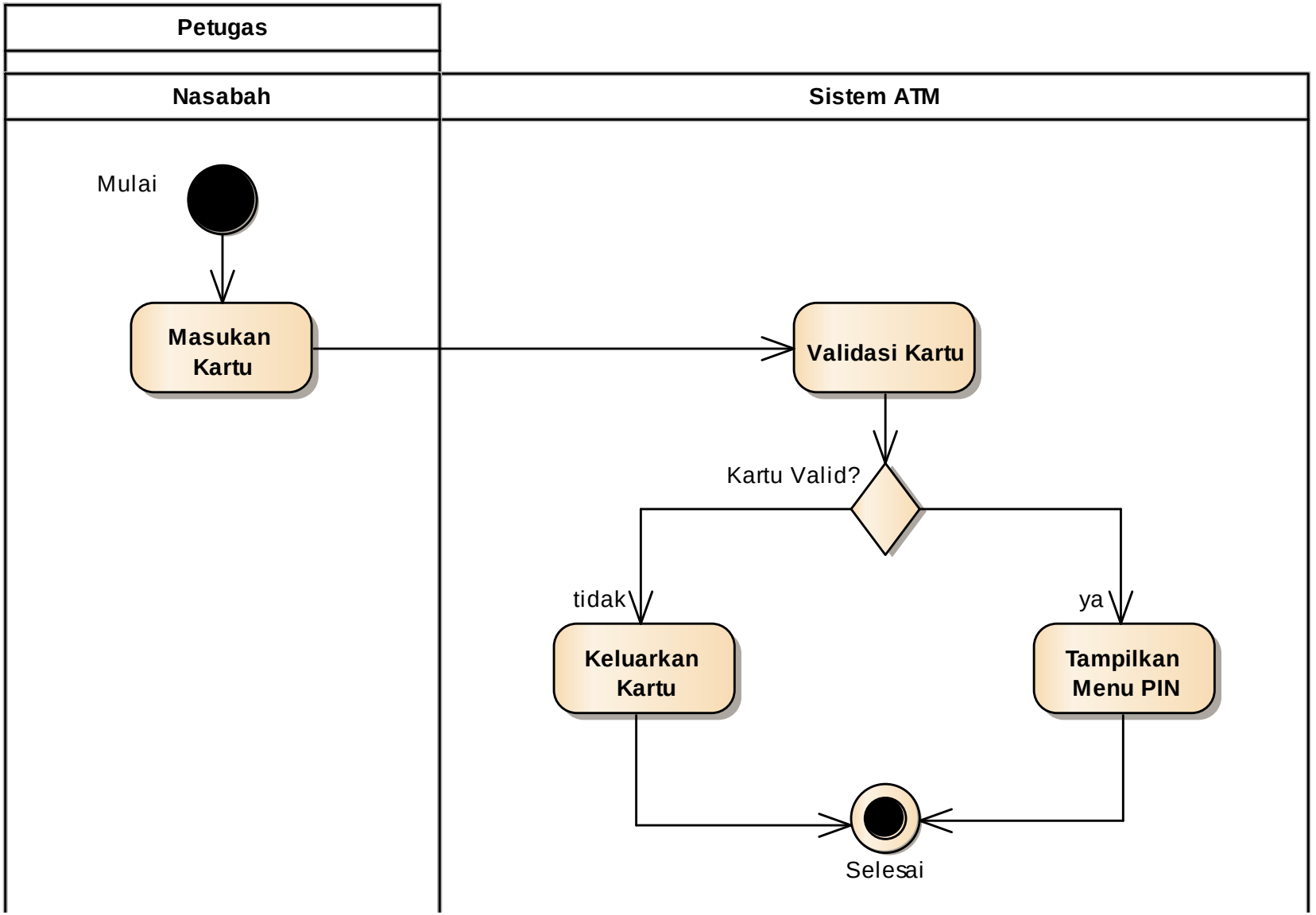
(Wahono, 2009)



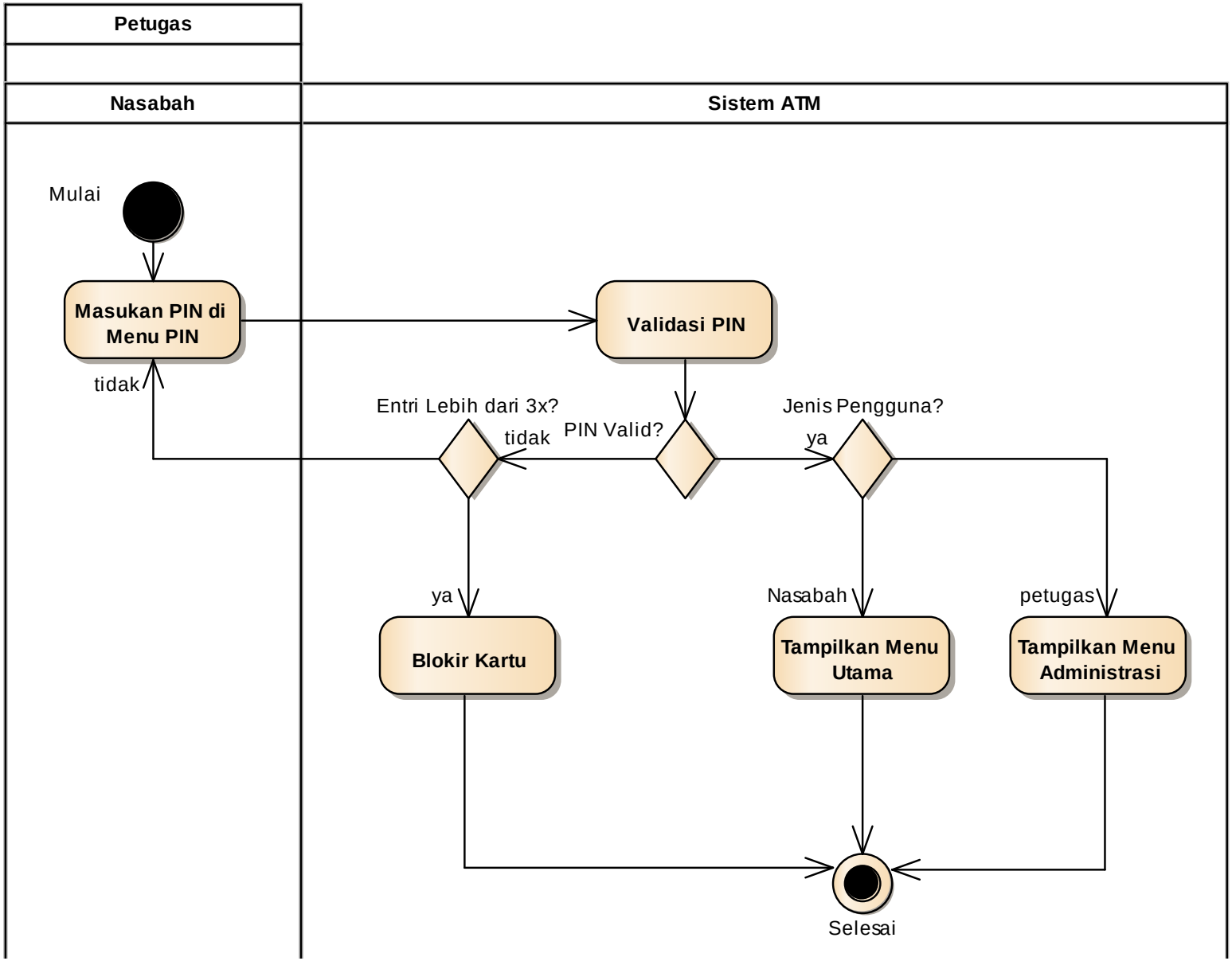
Use Case Diagram



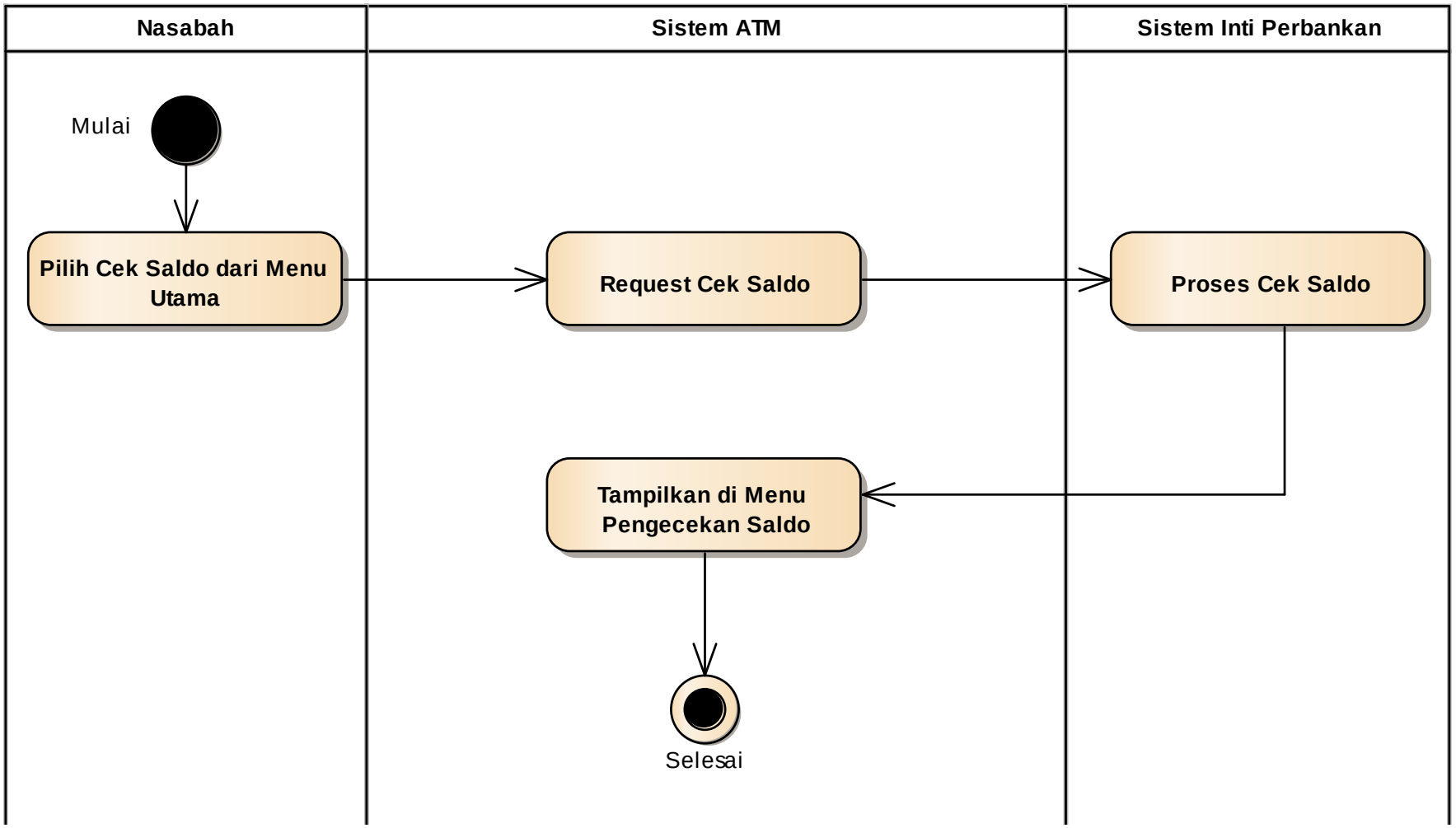
Activity Diagram: Memasukkan Kartu



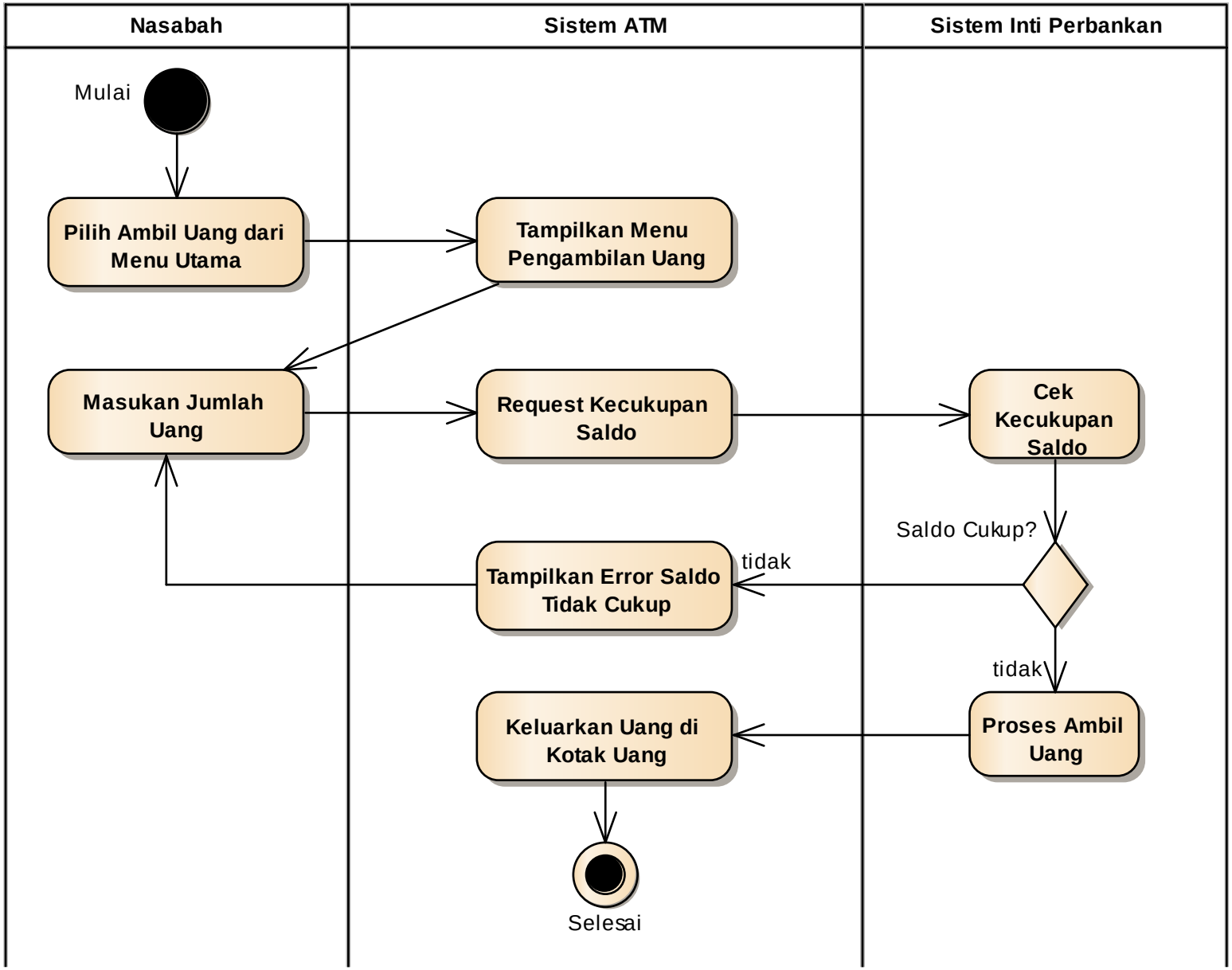
Activity Diagram: Memasukkan PIN



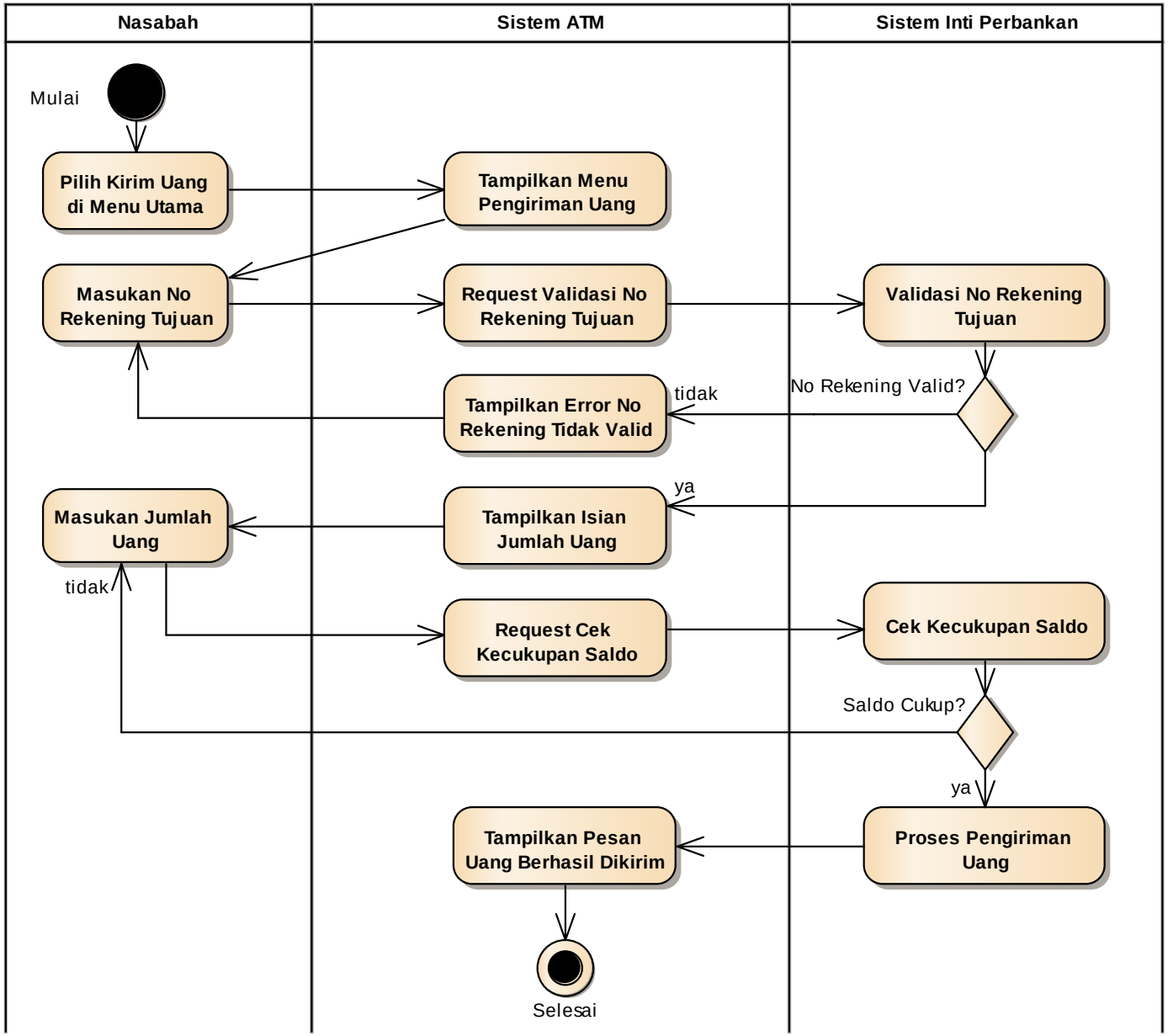
Activity Diagram: Mengecek Saldo



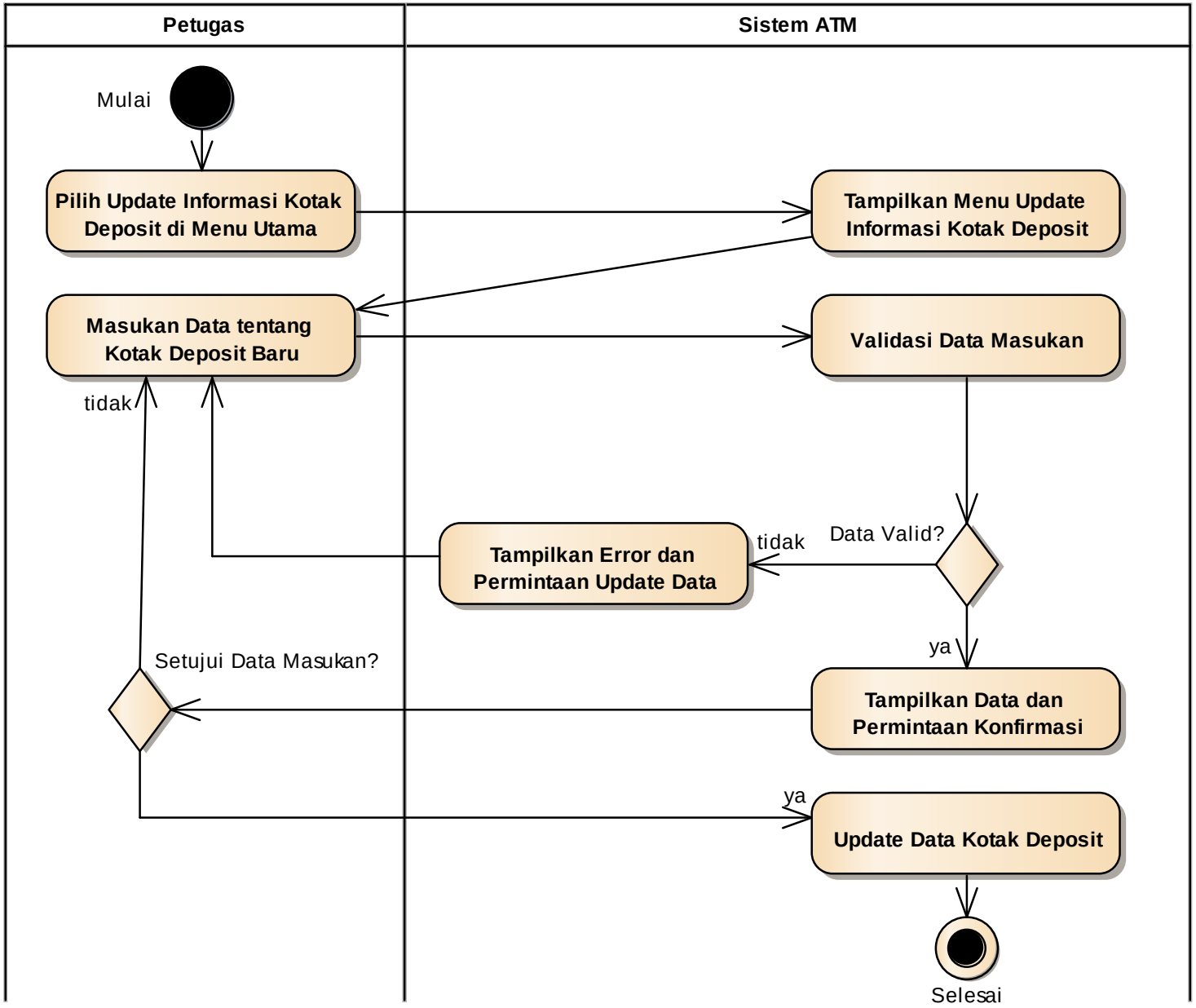
Activity Diagram: Mengambil Uang



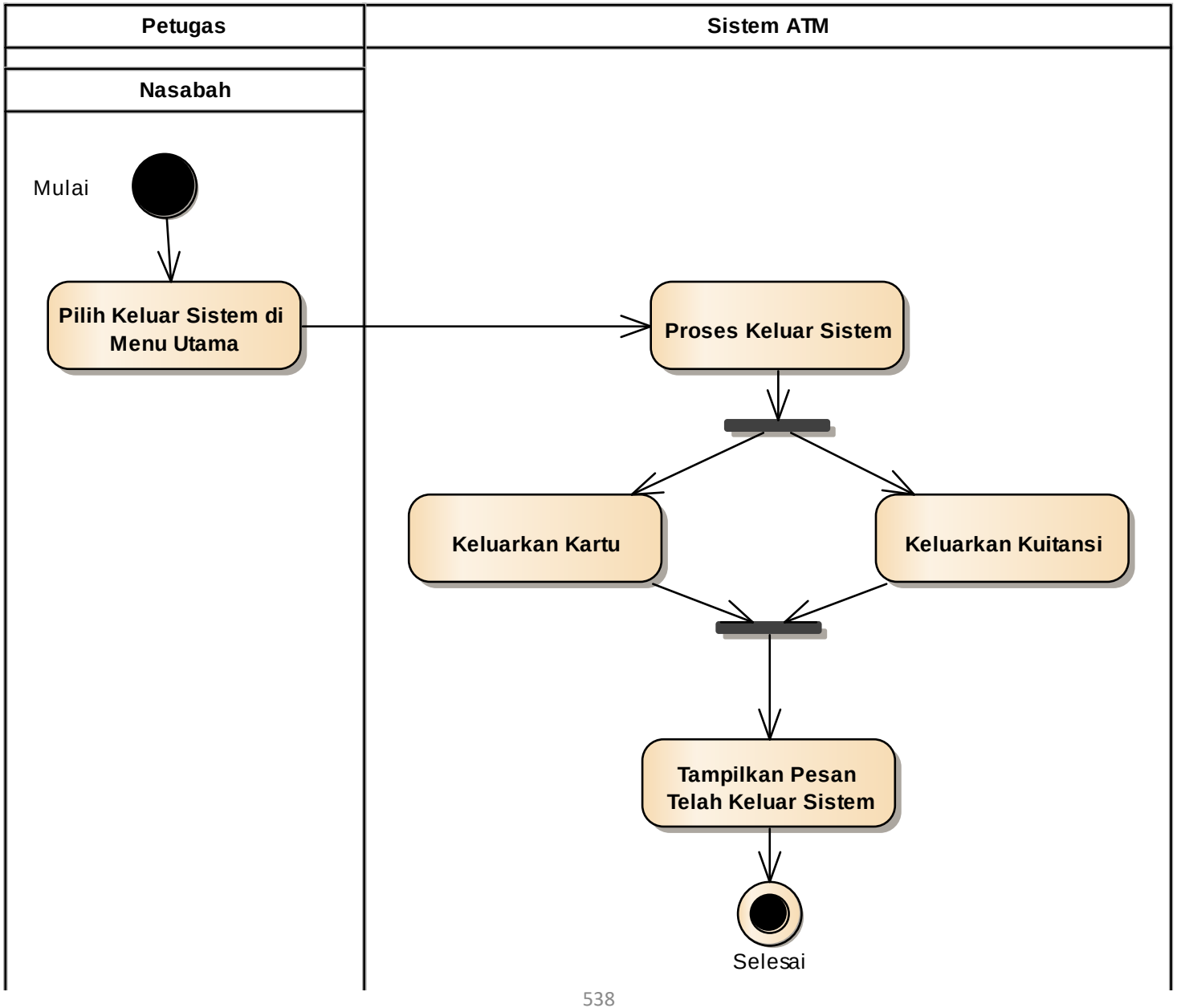
Activity Diagram: Mengirim Uang



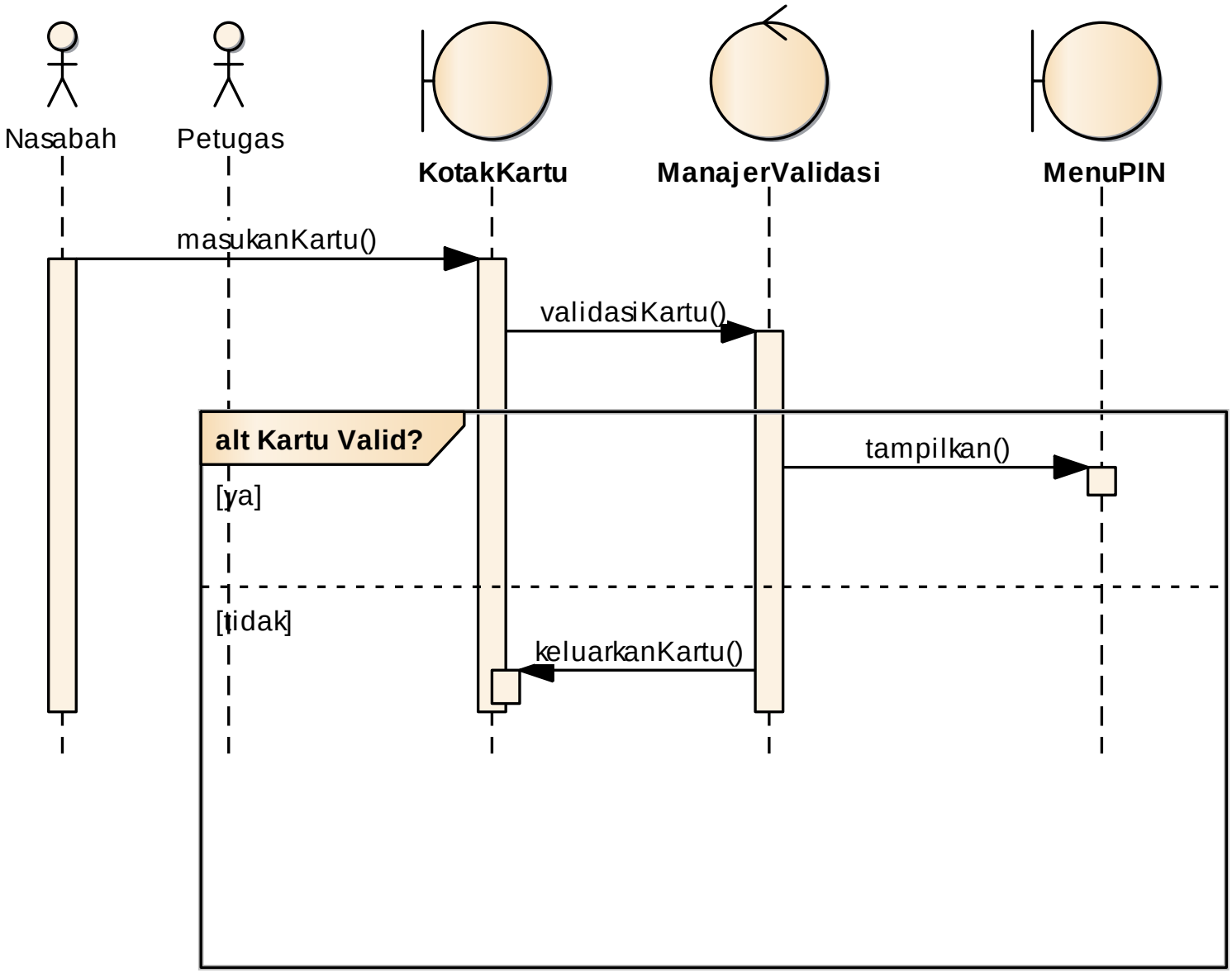
Activity Diagram: Mengupdate Informasi Kotak Deposit



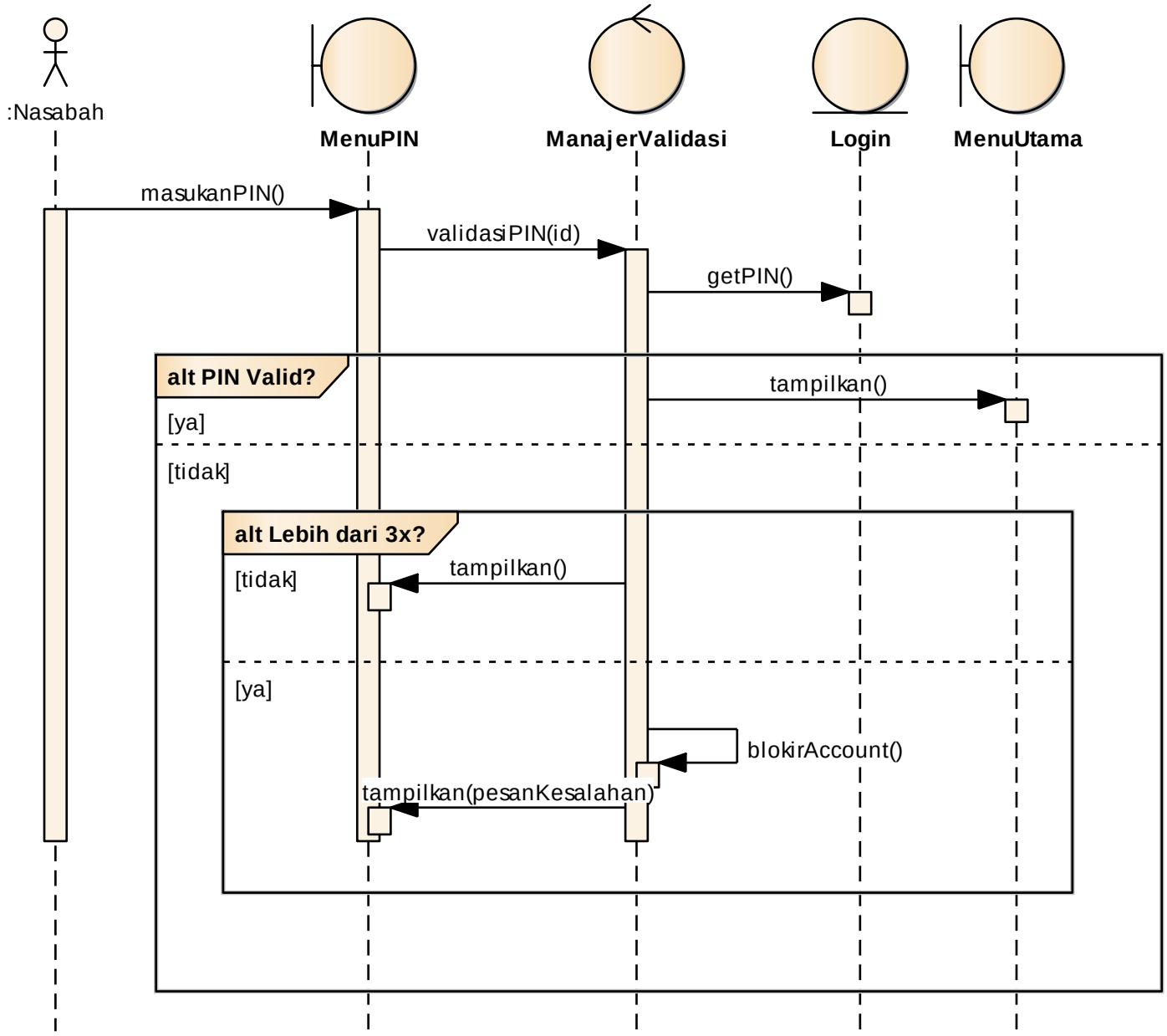
Activity Diagram: Keluar Sistem



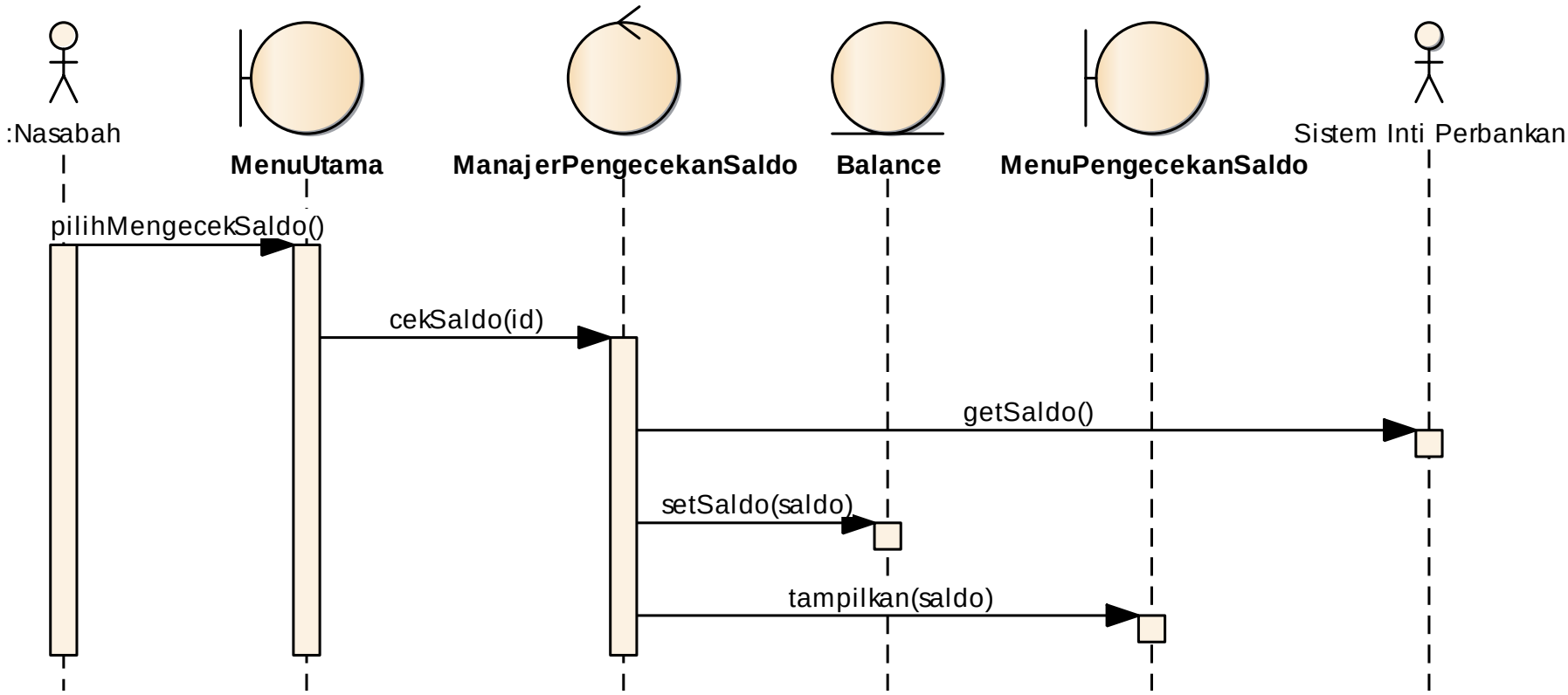
Sequence Diagram: Memasukkan Kartu



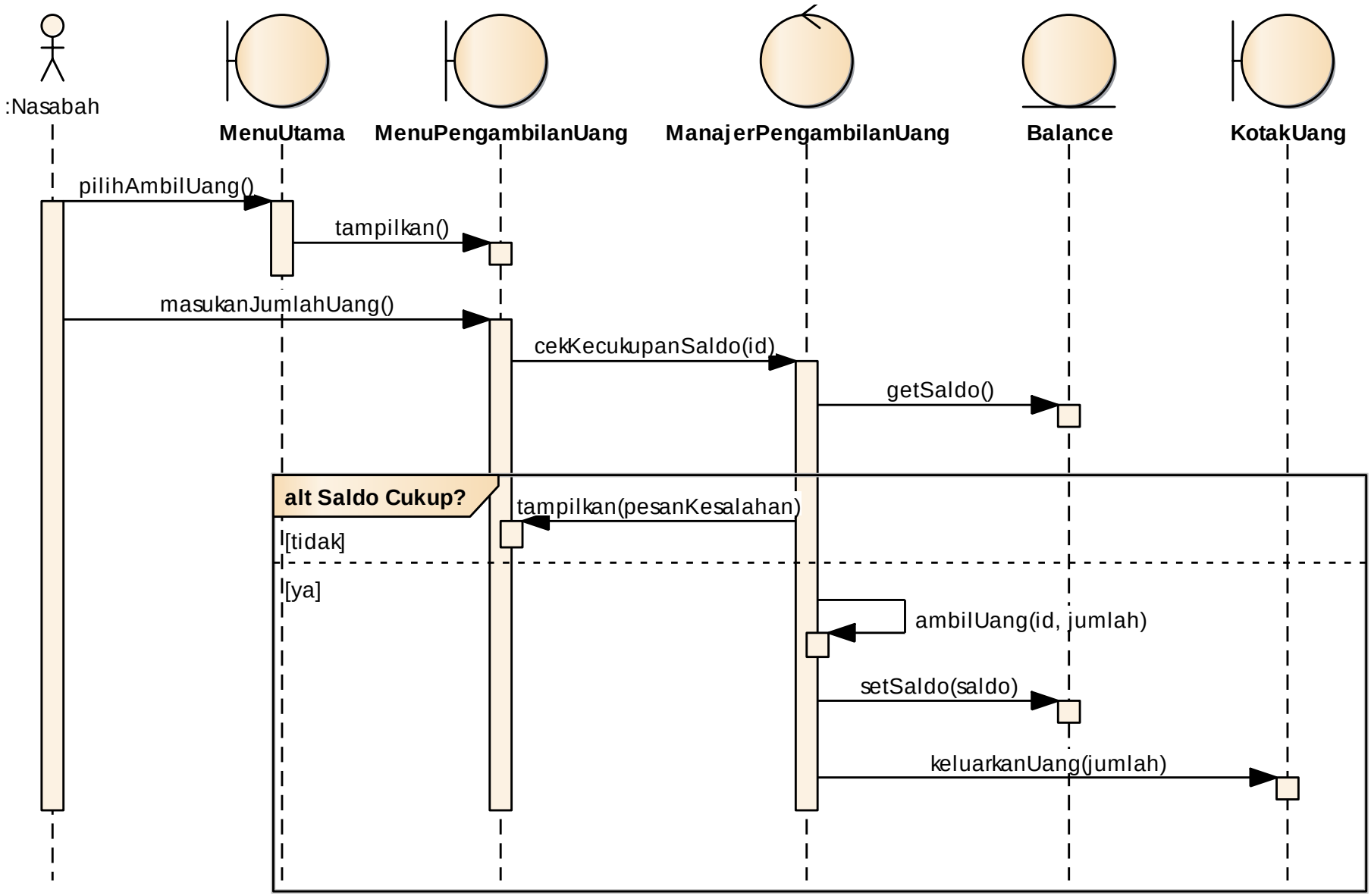
Sequence Diagram: Memasukkan PIN



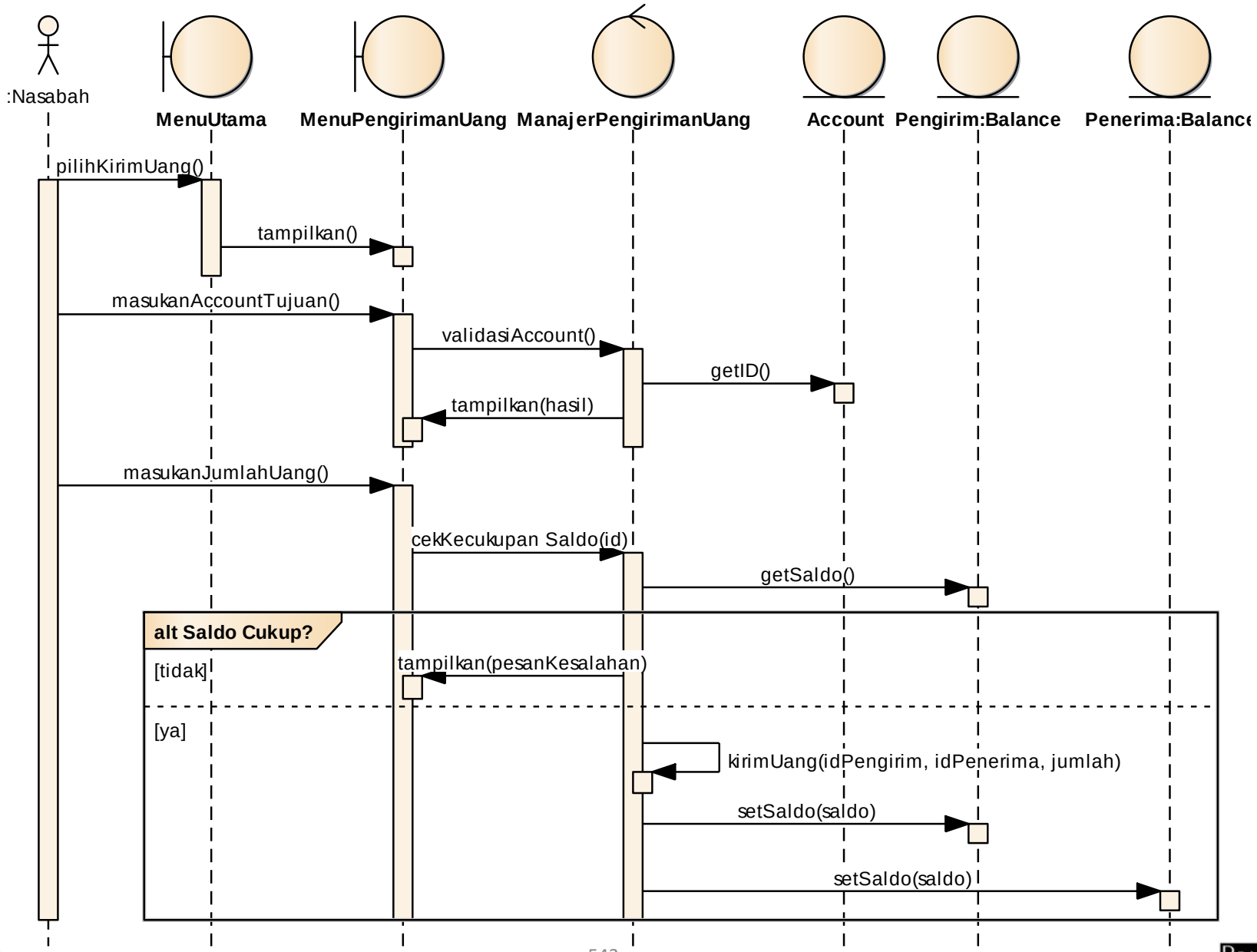
Sequence Diagram: Mengecek Saldo



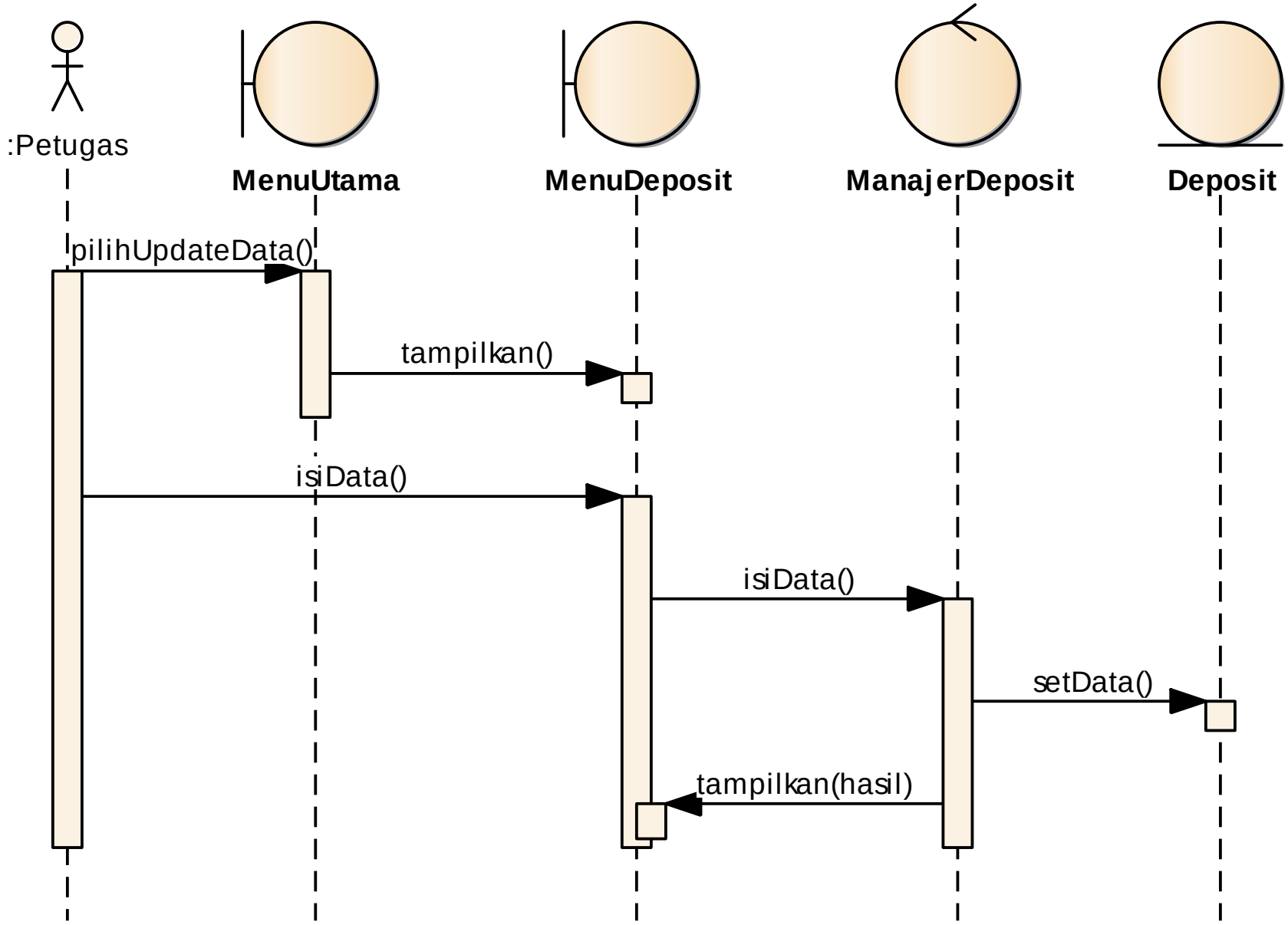
Sequence Diagram: Mengambil Uang



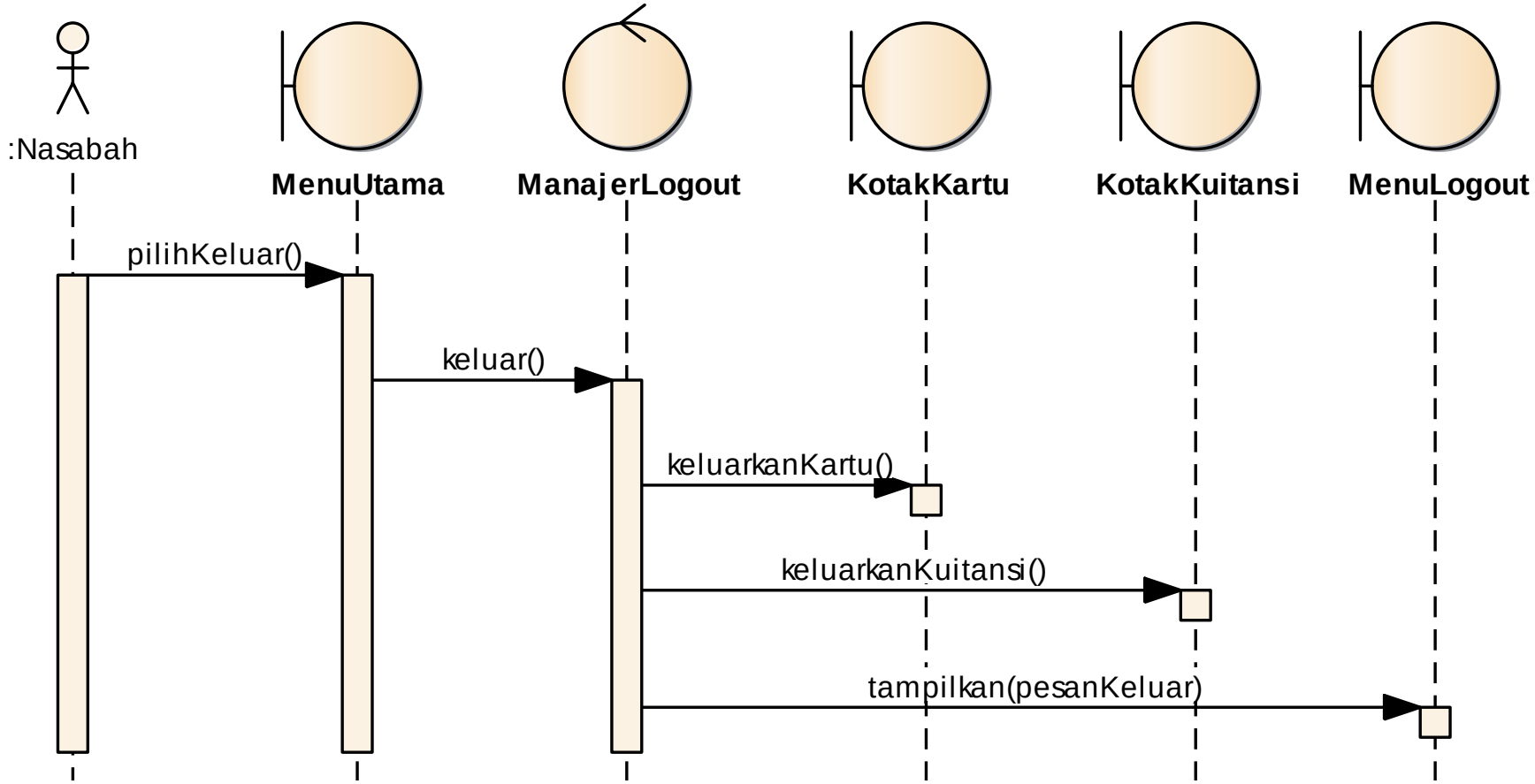
Sequence Diagram: Mengirim Uang



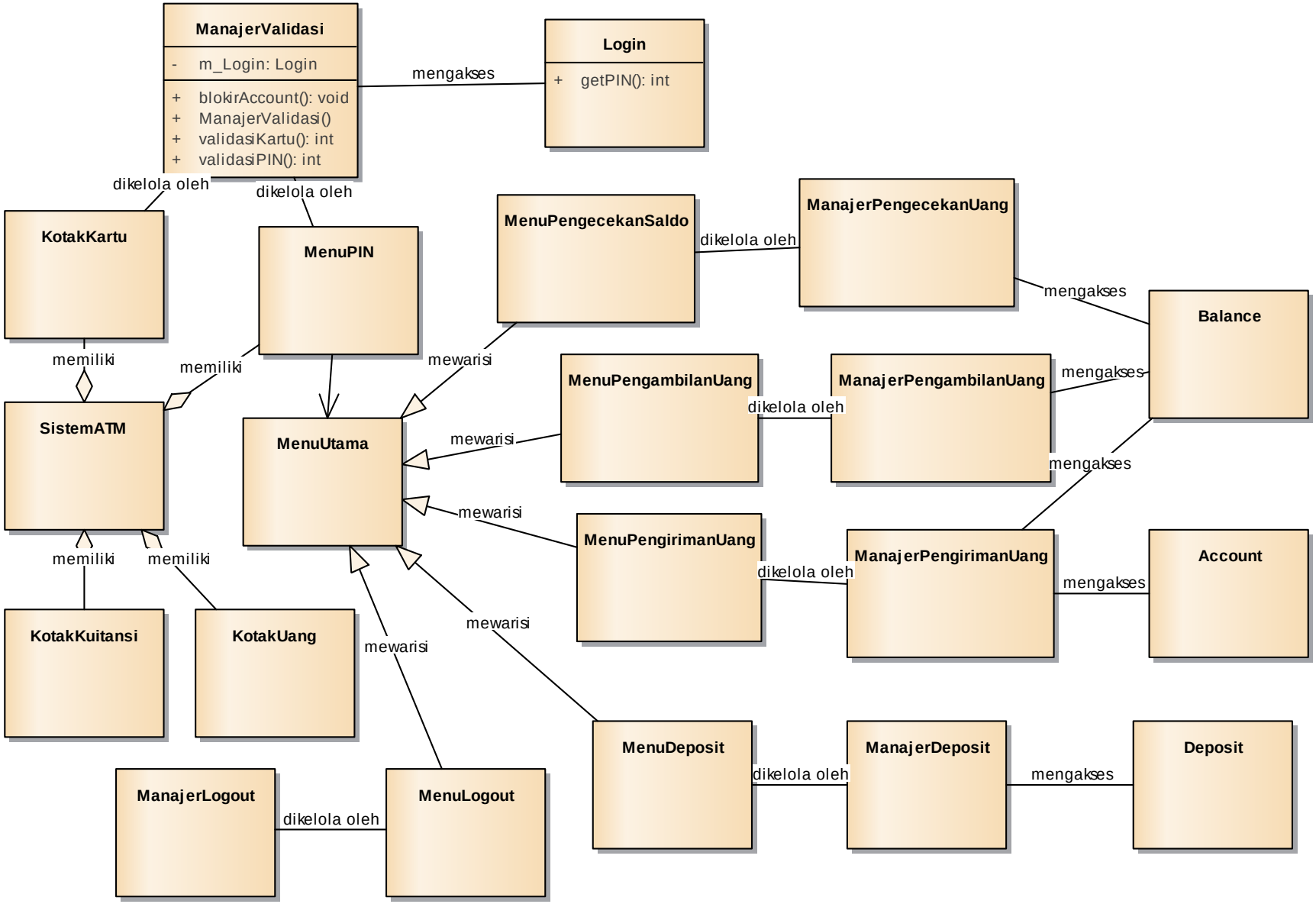
Sequence Diagram: Mengupdate Informasi Kotak Deposit



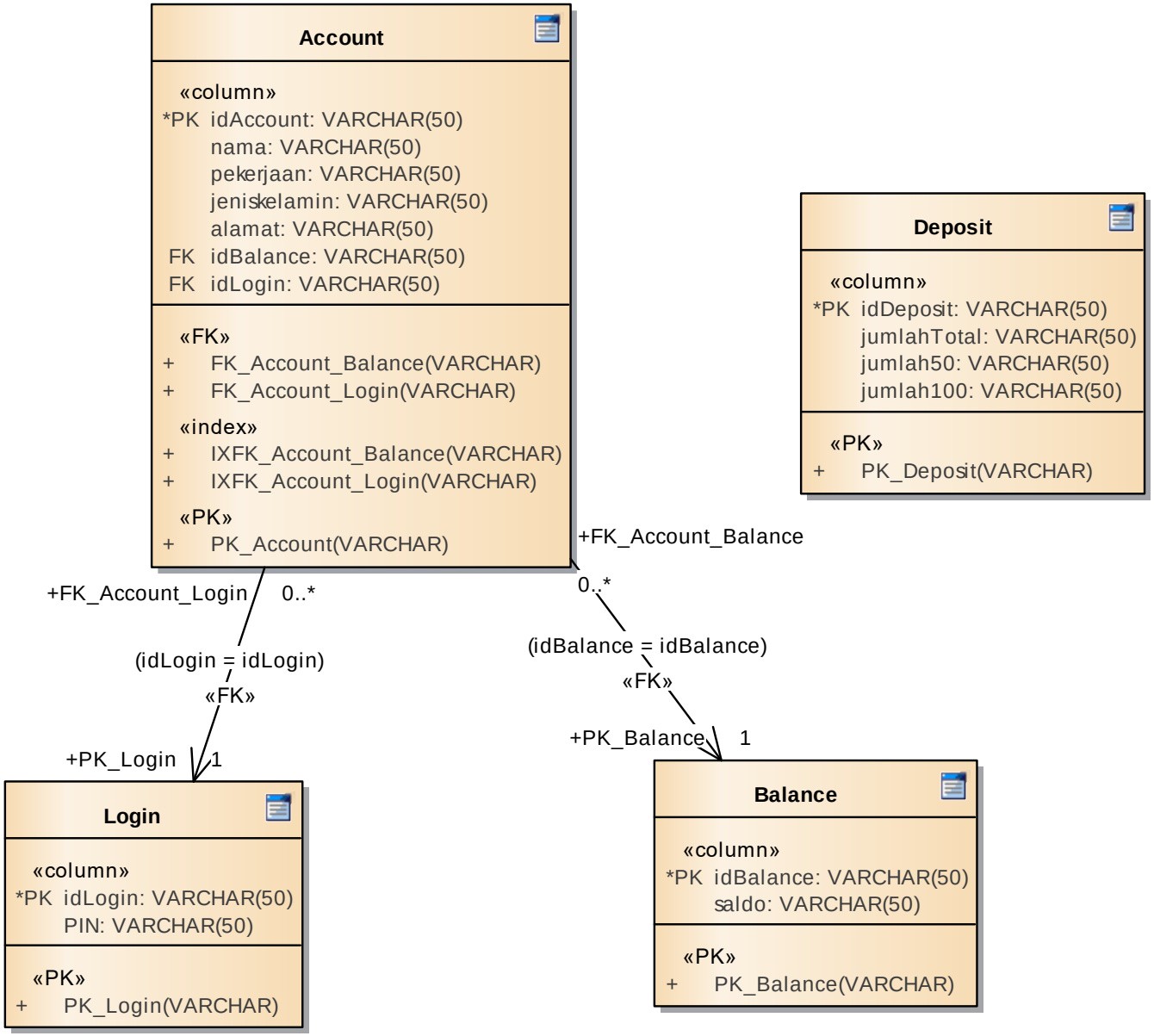
Sequence Diagram: Keluar Sistem



Class Diagram

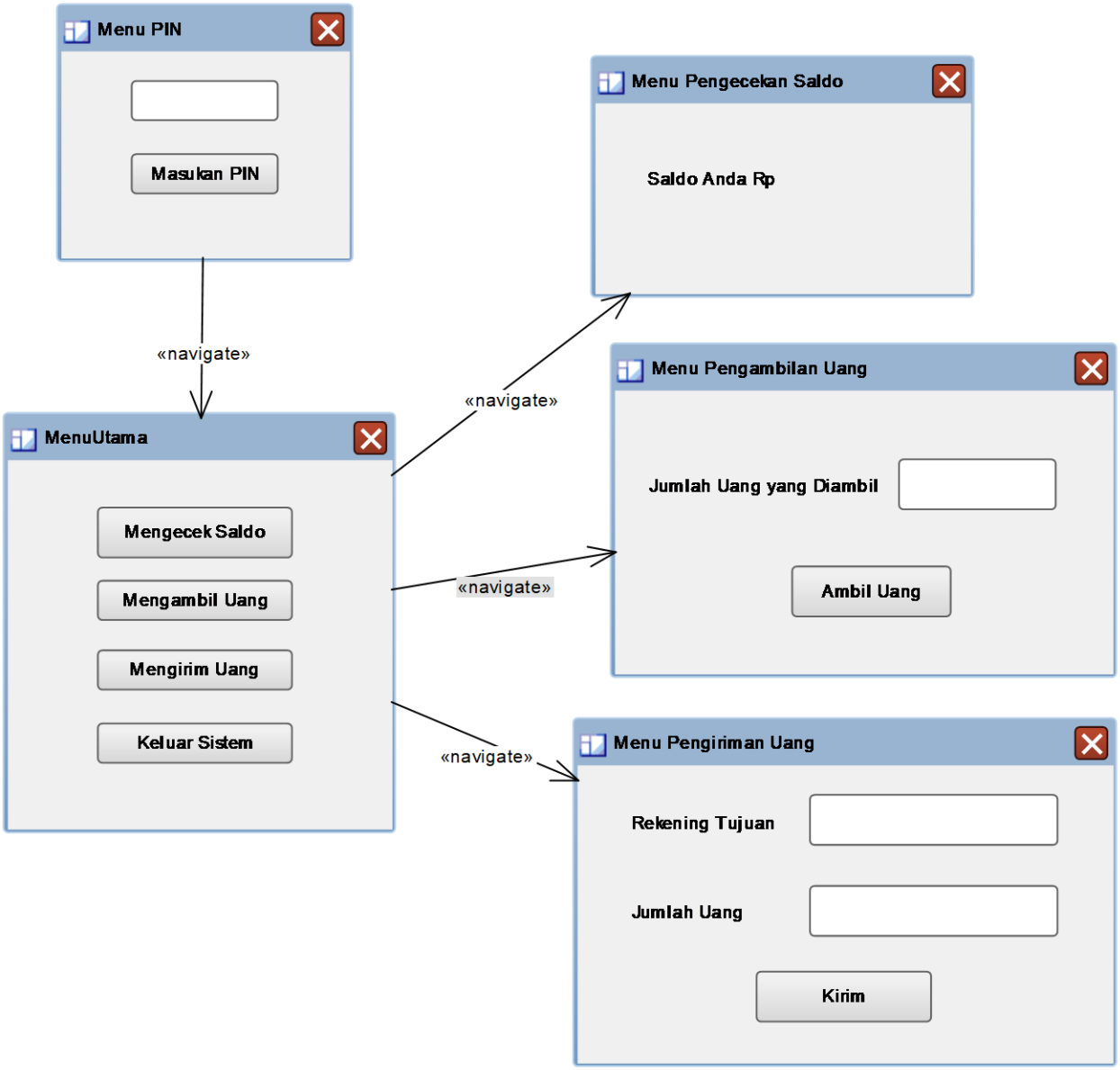


Data Model

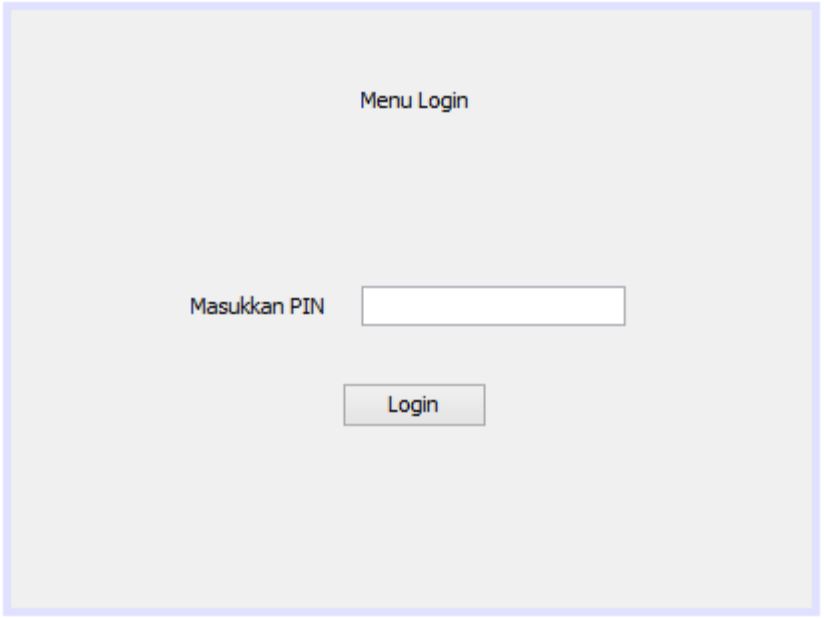

Account
«column»
*PK idAccount: VARCHAR(50)
nama: VARCHAR(50)
pekerjaan: VARCHAR(50)
jeniskelamin: VARCHAR(50)
alamat: VARCHAR(50)
FK idBalance: VARCHAR(50)
FK idLogin: VARCHAR(50)
«FK»
+ FK_Account_Balance(VARCHAR)
+ FK_Account_Login(VARCHAR)
«index»
+ IXFK_Account_Balance(VARCHAR)
+ IXFK_Account_Login(VARCHAR)
«PK»
+ PK_Account(VARCHAR)
Deposit
«column»
*PK idDeposit: VARCHAR(50)
jumlahTotal: VARCHAR(50)
jumlah50: VARCHAR(50)
jumlah100: VARCHAR(50)
«PK»
+ PK_Deposit(VARCHAR)
+FK_Account_Login
0..*
(idLogin = idLogin)
«FK»
+PK_Login
1
+FK_Account_Balance
0..*
(idBalance = idBalance)
«FK»
+PK_Balance
1
Login
«column»
*PK idLogin: VARCHAR(50)
PIN: VARCHAR(50)
«PK»
+ PK_Login(VARCHAR)
Balance
«column»
*PK idBalance: VARCHAR(50)
saldo: VARCHAR(50)
«PK»
+ PK_Balance(VARCHAR)

User Interface Design

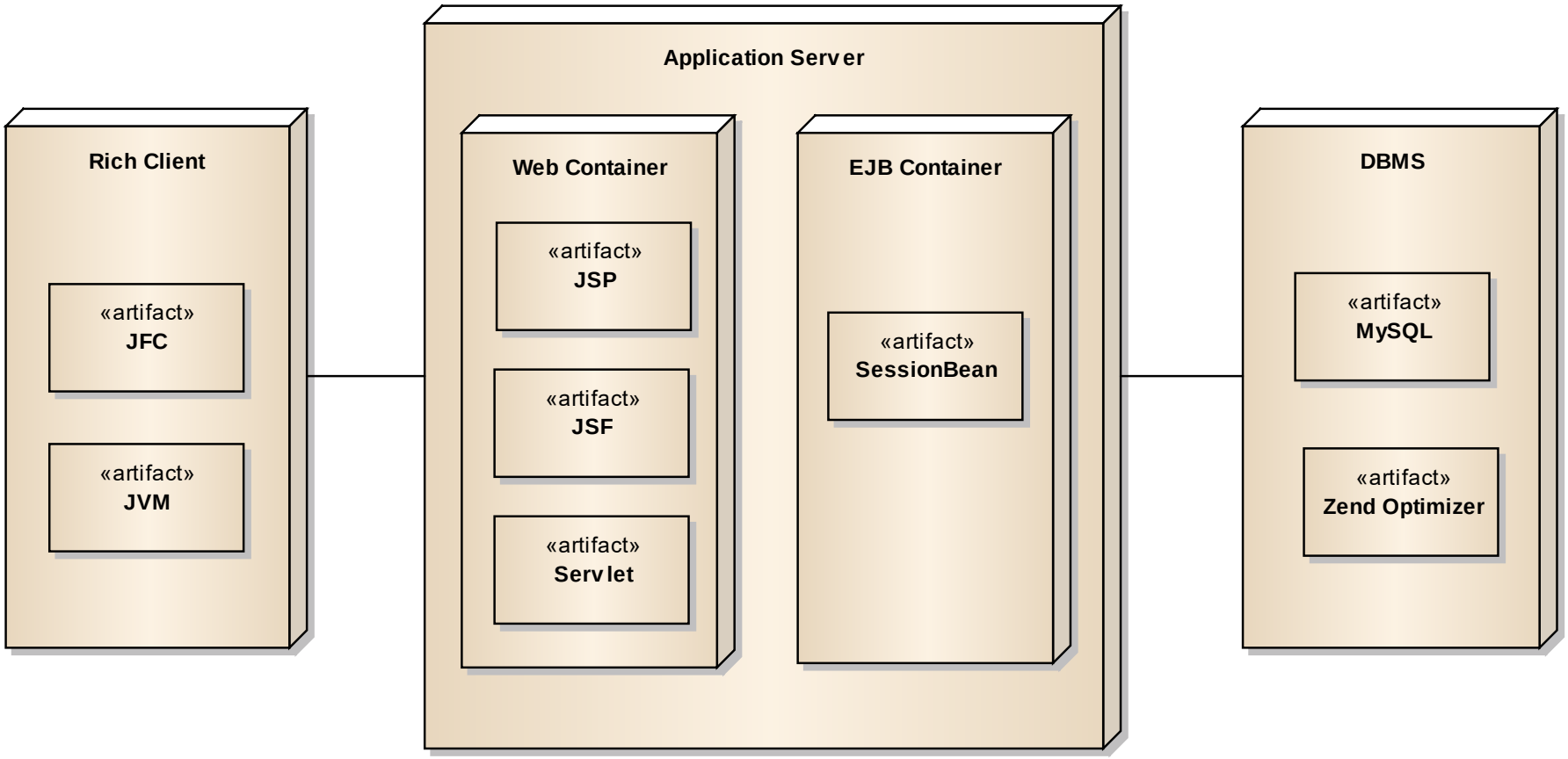
User Interface Design (Sparx EA)



User Interface Design (Netbeans)



Deployment Diagram (3 Tier)



Contoh Template System Specification (SRS, BRD, FSD)

1. System Planning
   - 1.1 Project Scope
   - 1.2 Project Schedule
   - 1.3 Project Team
2. System Design
   - 2.1 Functional Requirements
     - 2.1.1 Actor
     - 2.1.2 Use Case Diagram
     - 2.1.3 Activity Diagram (BPMN)
     - 2.1.4 Sequence Diagram
     - 2.1.5 Class Diagram
     - 2.1.6 Data Model
     - 2.1.7 User Interface Design
     - 2.1.8 Deployment Diagram
     - 2.1.9 Relational Matrices
       - 2.1.9.1 Actor – Activity Diagram
       - 2.1.9.2 Actor – Sequence Diagram
   - 2.2 Nonfunctional Requirements
     - 2.2.1 Operational
     - 2.2.2 Performance
     - 2.2.3 Security
     - 2.2.4 Hardware
     - 2.2.5 Development Platform
     - 2.2.6 Deadline
3. System Implementation
   - 3.1 Testing Strategy
   - 3.2 Installation Strategy
   - 3.3 Change Management Strategy

Exercise: Deployment Diagram

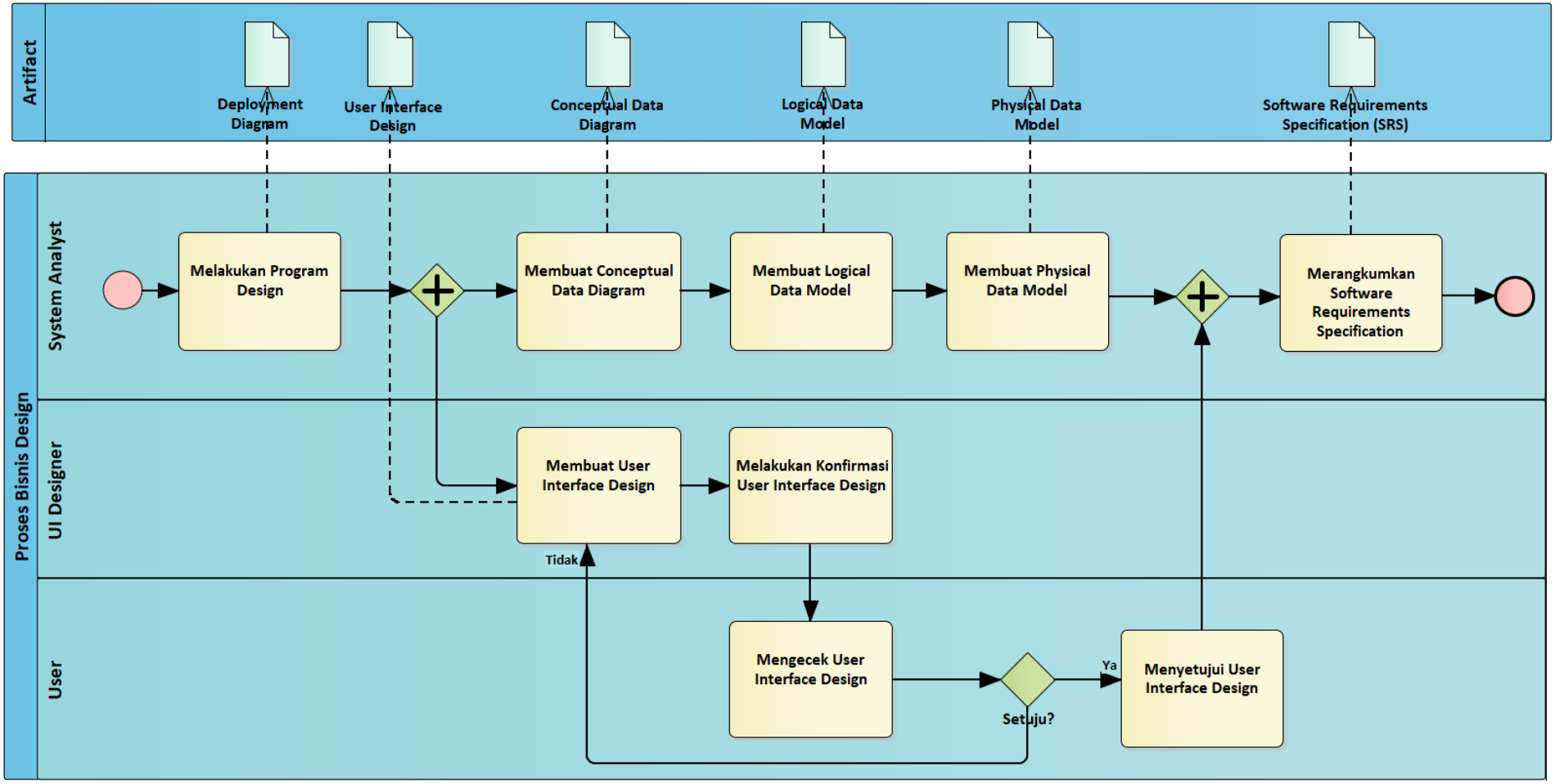
1. Lihat kembali Activity Diagram, Use Case Diagram, Sequence Diagram dan Class Diagram yang telah anda buat

2. Lanjutkan dengan membuat Deployment Diagram

Exercise: System Analysis and Design untuk System Request

1. Lihat kembali System Request yang sudah anda buat
2. Lakukan system analysis dengan membuat diagram di bawah:
    1. Use Case Diagram (+ Use Case Points)
    2. Activity Diagram atau BPMN
    3. Sequence Diagram
    4. Class Diagram (+ Forward/Reverse Engineering)
    5. Data Model (+ Forward/Reverse Engineering)
    6. User Interface Design
    7. Deployment Diagram
3. Ujicoba publish hasil ke HTML dan DOCX
4. Kirim file EAPX ke romi@brainmatics.com untuk review bersama

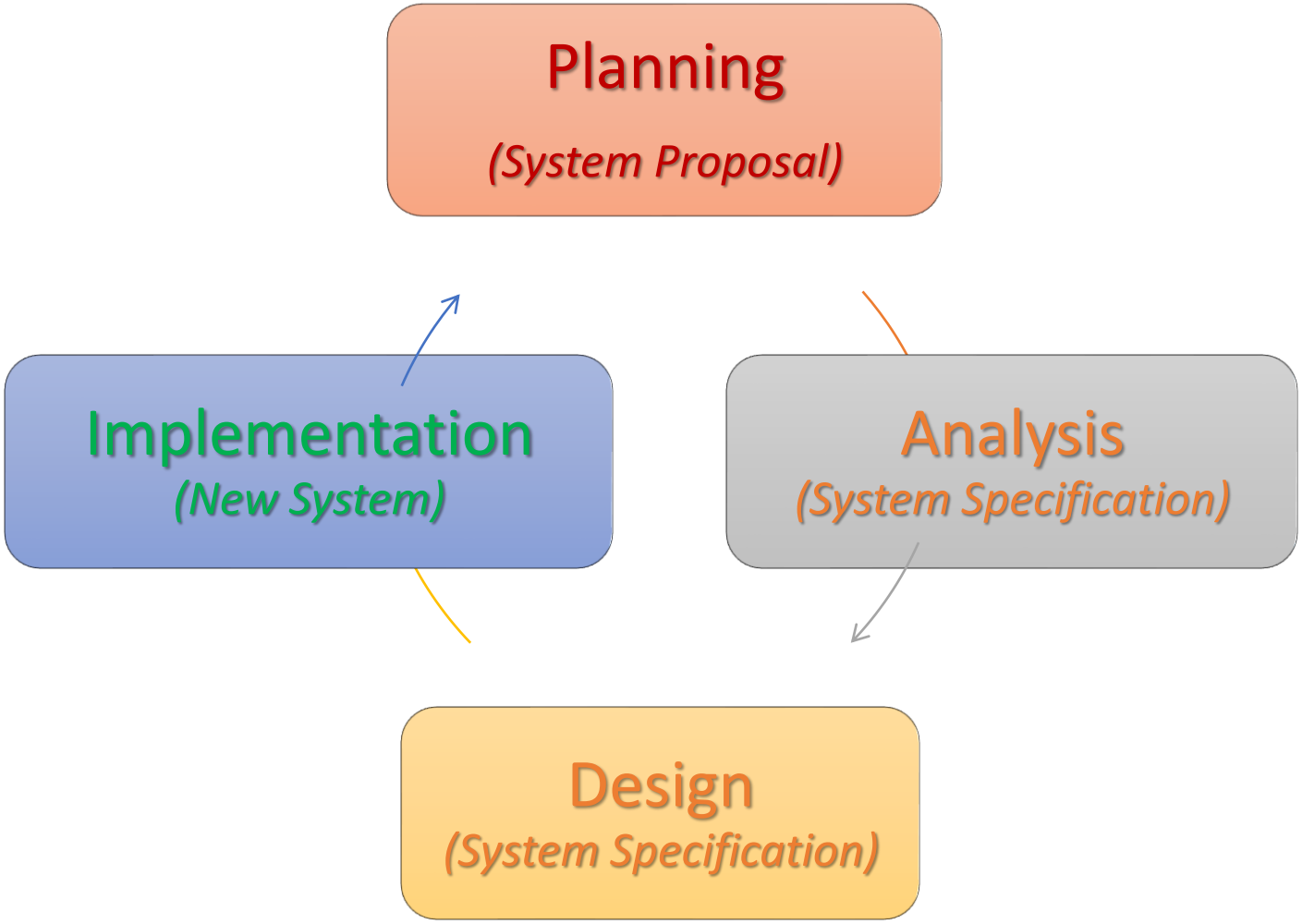
Systems Design



5. Systems Implementation

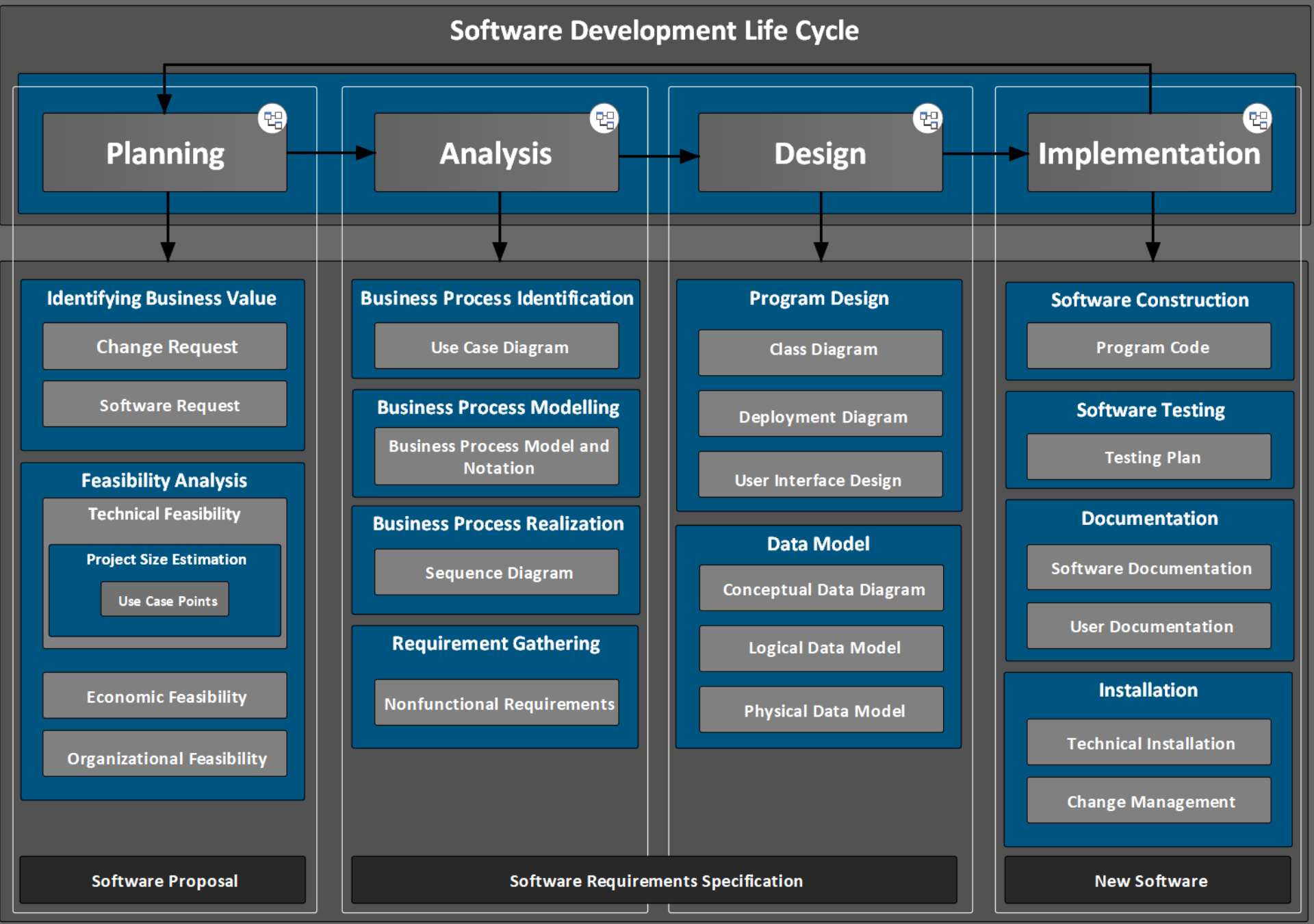
5.1 Konstruksi Software
5.2 Pengujian Software
5.3 Pembuatan Dokumentasi
5.4 Instalasi dan Manajemen Perubahan

Siklus Pengembangan Software



(Tilley, 2012) *(Dennis, 2016)* *(Valacich, 2017)*

Application Development Governance



System Implementation

1. Construction: The development of all parts of the software itself, documentation, and new operating procedures

2. Testing: A form of insurance. It's cheaper to fix bugs earlier, rather than later

3. Documentation: Provides information to make the system easier to use and maintain

4. Installation: Technical aspect (conversion) and organizational aspect (change management)

5.1 Konstruksi Software

Kegiatan Utama Konstruksi Software

1. Assigning the programmers
2. Coordinating the activities
3. Managing the schedule

1. Assigning Programmers

- Start by looking at the package diagrams
- Assign similar modules to the same programmer

- Remember the "programmer paradox"
  - Can't just add more people
- Fewer programmers is normally better
- Adding manpower to a late project makes it later (Brook, 1975)
  - “Just because a woman can make a baby in nine months, it does not follow that nine women can make a baby in one month”

Adagium di Pengembangan Software

- Project Manager is a person who thinks nine women can deliver a baby in one month
- Marketing Manager is a person who thinks he can deliver a baby even if no man and woman are available
- Human Resource is a person who thinks that a donkey can deliver a human baby if given 9 months
- Client is the one who doesn't know why he wants a baby
- Developer is a person who thinks it will take 18 months to deliver a baby
- Tester is a person who always tells his wife that this is not the right baby

2. Coordinating Activities

- Hold weekly project meetings
  - discuss changes to the system
  - discuss other issues of the past week
- Create and follow standards
- Set up separate workspace for
  - development, testing, production
  - as a minimum, separate files
- Use change control
  - program log, sign-in/-out
- Use CASE tools

3. Managing the Schedule

- Use initial time estimates as a baseline
- Revise time estimates as construction proceeds
- Fight against scope creep!
- Monitor “minor” slippage
- Create risk assessment and track changing risks
  - Risks change as deadline approaches
- Fight the temptation to lower quality to meet unreasonable schedule demands

Avoid Classic Mistakes

- Research-oriented development
  - If you use state-of-the art technology, lengthen planned time
- Using "low-cost" personnel
  - You get what you pay for. If using a significant number of entry level personnel, lengthen planned time
- Lack of code control
  - Use source code library to keep programmers from changing the same code at the same time. Why?
- Inadequate testing
  - Always allocate sufficient time for formal testing

5.2 Pengujian Software

Testing Philosophy

- Testing can never prove there are no errors
  - The purpose is not to demonstrate that the system is free of errors
  - The purpose is to detect as many errors as possible
- It is dangerous to test early modules without an overall testing plan
  - It may be difficult to reproduce sequence of events causing an error
  - Testing must be done systematically and results documented carefully

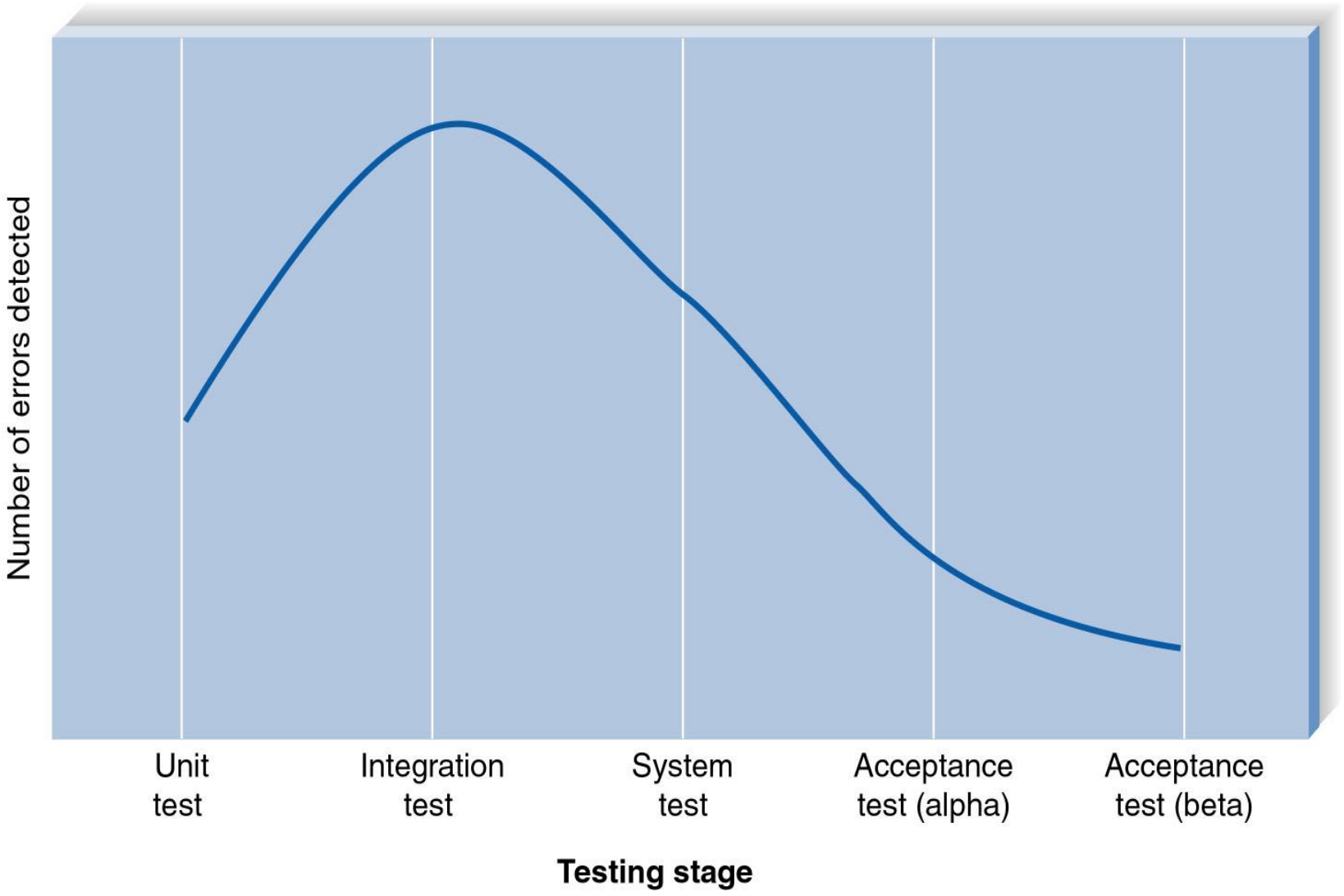
Test Planning

- Address all products created during development
  - So develop test plan early
  - Example: test completeness of CRC cards
- Each test has:
  - Specific objective
  - Specific test cases to examine
  - Test specifications
- If the tested class requires methods that aren't ready
  - Use stubs (hard coded fake methods)

Stages of Testing

1. Unit testing
   - Tests each module to assure that it performs its function
2. Integration testing
   - Tests the interaction of modules to assure that they work together
3. System testing
   - Tests to assure that the software works well as part of the overall system
4. Acceptance testing
   - Tests to assure that the system serves organizational needs

Error Discover Rates



Unit Testing

- Tests a single unit (a class)

- Type of unit testing:
  1. Black Box Testing
     - Most common
     - Looks just at inputs and outputs
     - Tests whether the unit meets requirements stated in specification
  2. White-Box Testing
     - Looks inside the module to test its major elements
     - Limited usefulness in OO design
       - because units are so small

Integration Testing

- After the classes pass unit tests
- Test classes that must work together
- Four types of Integrating testing:
  1. User interface testing
     - Tests each interface function
     - Move through each menu/screen
  2. Use-case testing
     - Ensures that each use case works correctly
     - Step through each use case
  3. Interaction testing
     - Start with a package
     - Each method is a stub
     - Add methods one at a time, testing as you go
     - Once all packages are done, repeat on the package level
  4. System interface testing
     - Ensures data transfer between systems

System Testing

- See that all classes work together
- Similar to integration testing but broader

- Five types of system testing:
  1. Requirements Testing
     - Are business requirements met?
     - Ensures that integration did not cause new errors
  2. Usability Testing
     - Tests how easy and error-free the system is in use
     - Informal or formal
  3. Security Testing
     - Assures that security functions are handled properly
     - e.g. Disaster recovery
  4. Performance Testing
     - Assures that the system works under high volumes of activity
  5. Documentation Testing
     - Analysts check that documentation and examples work properly

User Acceptance Testing

- Done by users with support from project team
- Ensure the system meets the originally stated requirements

- Two types of acceptance testing:
  1. Alpha Testing
     - Repeat tests by users to assure they accept the system, uses known data
  2. Beta Testing
     - Uses real data, not test data

5.3 Pembuatan Dokumentasi

Documentation

System Documentation

- Revised and final version of System Specification
- Helps programmers and analysts understand the application
- Used for development and maintenance
- Largely a by product of the system analysis and design phases
- Can be automated (JavaDoc)

User Documentation

- Type: Reference documents (help system), Procedures manuals, Tutorials
- Help users operate the system
- High quality documentation takes about 3 hours per page to produce
- Should not be left to the end of the project

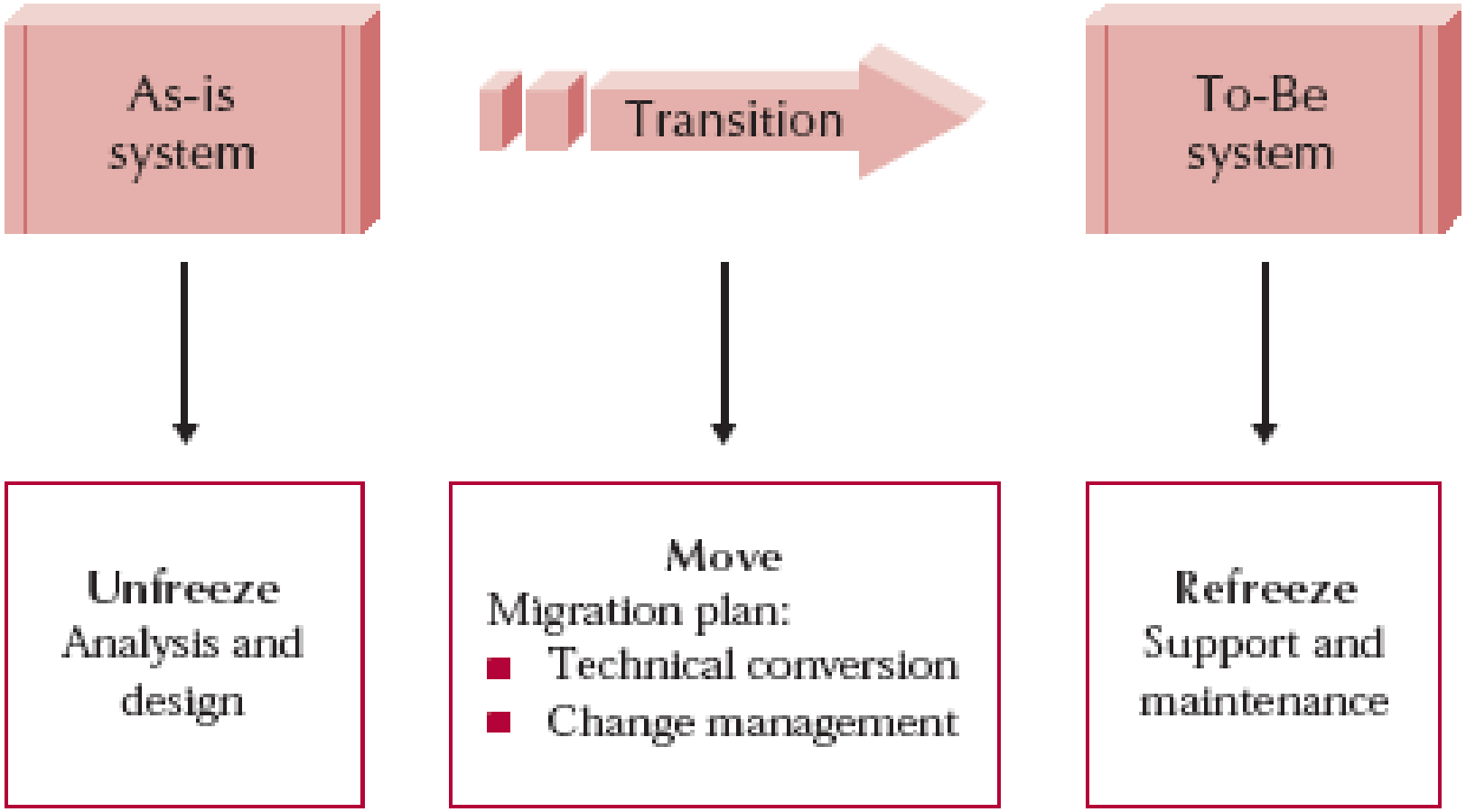
5.4 Instalasi dan Manajemen Perubahan

Installation and Change Management

- Transitioning to new systems involves managing change from pre-existing norms and habits

- Change management involves:
  1. Unfreezing: loosening up peoples' habits and norms
  2. Moving: transition from old to new systems
  3. Refreezing: institutionalize and make efficient the new way of doing things

Big Picture of Change Management



1. Unfreezing

- Activities to date facilitate unfreezing

- Users:
  - Already know of the new system
  - Helped in the analysis phase
  - Helped in the design

- This probably has already unfreezed current habits and norms

2. Moving

- Helps move people from As-Is system to To-Be system

- What activities will be performed ?
  1. Technical aspects (Conversion)
     - Installing hardware and software
     - Converting data
     - Conversion strategy depends on Style, Location and Modules
  2. Organizational aspects (Change Management)
     - Training users on the system
     - Motivating employees to use the new system to aid in their work

Conversion Styles

1. Direct conversion

- The new system instantly replaces the old
- Upgrading to new version of word processor
- Simplest and most straightforward
- Most risky

2. Parallel conversion

- Old and new systems used side by side
- Old is turned off when new is shown to work
- Provides a safety net
- Added expense and complexity of running both

Conversion Location

1. Pilot Conversion

- A few locations are converted first
- Once bugs are worked out, other locations are converted
- Provides additional testing before going live
- Requires more time
- Different parts of the organization are using different versions

2. Phased Conversion

- Partition the organization
- Convert each partition one at a time
- Allows smaller installation team
- Same pros and cons as pilot conversion

3. Simultaneous Conversion

- All locations are converted at the same time
- Can be used with direct or parallel conversion
- Everyone uses same version
- Requires large staff to perform conversion

Conversion Modules

1. Whole system conversion
   - All modules converted in one step
   - Most common
   - May be a steep learning curve for users

2. Modular conversion
   - Separate modules are converted one at a time
   - Application must be written for this

Characteristics of Conversion Strategies

	Conversion Style		Conversion Location			Conversion Modules	
Characteristic	Direct Conversion	Parallel Conversion	Pilot Conversion	Phased Conversion	Simultaneous Conversion	Whole-System Conversion	Modular Conversion
Risk	High	Low	Low	Medium	High	High	Medium
Cost	Low	High	Medium	Medium	High	Medium	High
Time	Short	Long	Medium	Long	Short	Short	Long

Change Management (Key Players)

- The sponsor
  - The business person who initiated the request for the new system
  - Usually senior management
  - Must be visible leader of change
  - Development team doesn't have the clout to change minds and hearts
- The change agent
  - The person(s) leading the change effort
  - Plan and implement the change
  - Usually outside the business unit
  - Has no management authority
- Potential adopter(s)
  - The people who must change
  - This is who the system is designed for

Change Management (Tools)

- Standard operating procedures (SOPs)
  - Formal and informal
  - Become the norms for an organization
  - Formal SOPs must match To-Be system
  - Informal SOPs will then follow
- Measurements and rewards
  - What does it mean to “do a good job”
  - Measurement and rewards help workers know
  - Measurements indicate what is important to the organization
  - Rewards enforces the measurements
- Resource allocation
  - Put your money where your mouth is
  - Has tangible and symbolic impact
    - Tangible – the benefits of the resources
    - Symbolic – Shows that the new project is important to management

Change Management (Training)

- Users must be capable of adopting the system
- They may need new skills
  - Learning these skills may involve use of the technology itself
  - New skills may be needed to handle the changed business processes
- Helping users accomplish their tasks
  - Don't show off the new system
- Show users how the system connects to the big picture
  - Use case diagrams are a good place to look to see what to train

Types of Adopters

1. Ready Adopters (20% - 30%)
   - Recognize the benefits
   - Quickly adopt the system
   - Become proponents of the system

2. Resistant adopters (20% - 30%)
   - Refuse to accept the change
   - Fight against the system

3. Reluctant Adopters (40% - 60%)
   - Apathetic & blow with the wind

3. Refreezing

- To institutionalize the use of the new system
  - Make it the normal, routine, accepted way of doing business

- Three key activities
  a. System support
  b. System maintenance
  c. Project assessment

a. System Support

- On-demand training
  - At the time of users' need
- Online support
  - Frequently asked questions (FAQ)
- Help desk
  - Phone service for known issues
    - 80% success with first call
  - Level 2 Support
    - For the other 20%

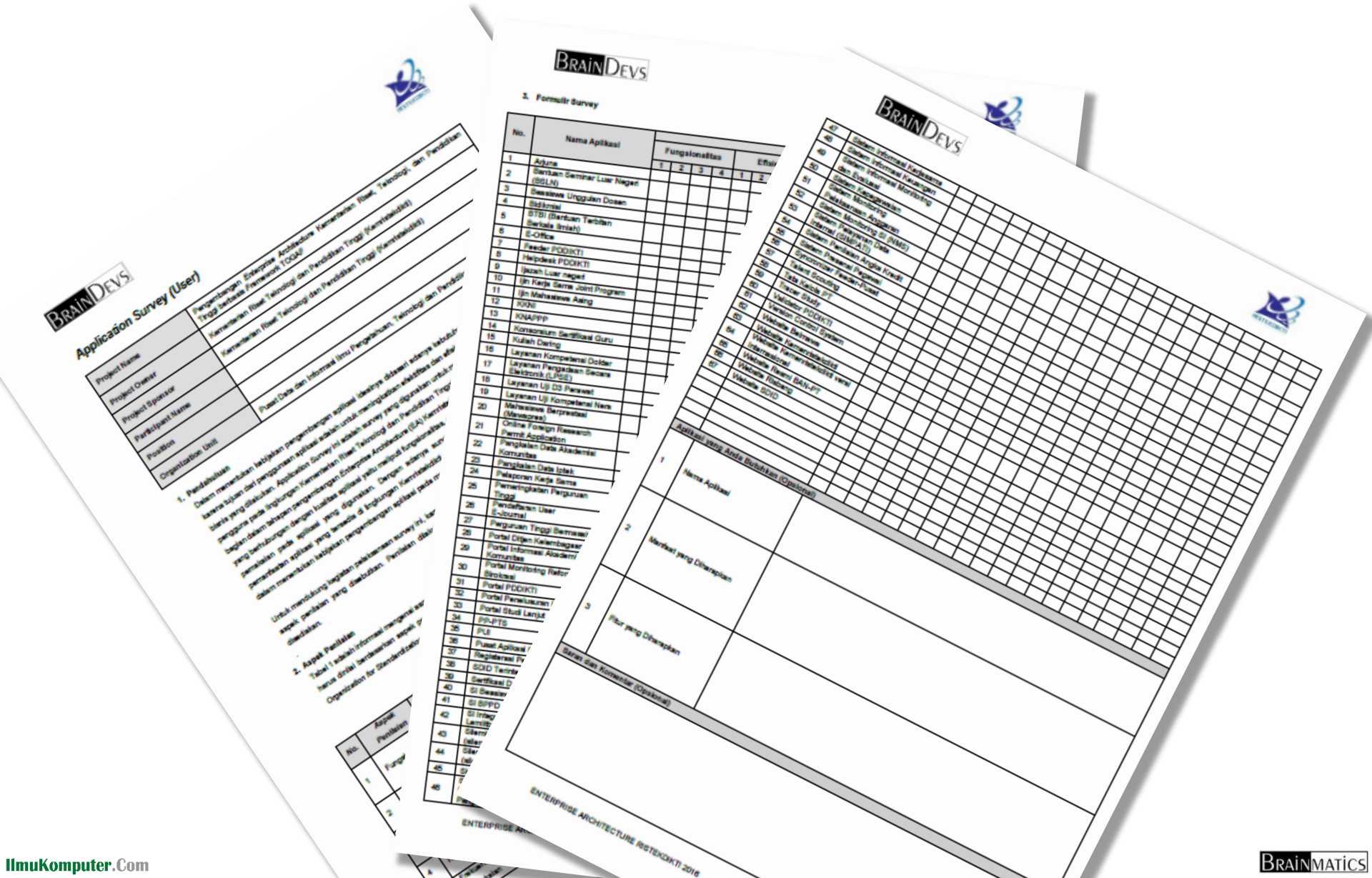
b. System Maintenance

- Refining the system so it continues to meet business needs
- More money spent on maintenance than initial development

c. Project Assessment

- Determine what worked, and what didn't
  - Important for future projects
  - Especially important for junior personnel
- Start with system request and feasibility analysis
  - Was the cost/benefit estimate valid?
- Helps improve future cost/benefit estimates
- Helps initiators to be honest about the need for a new system

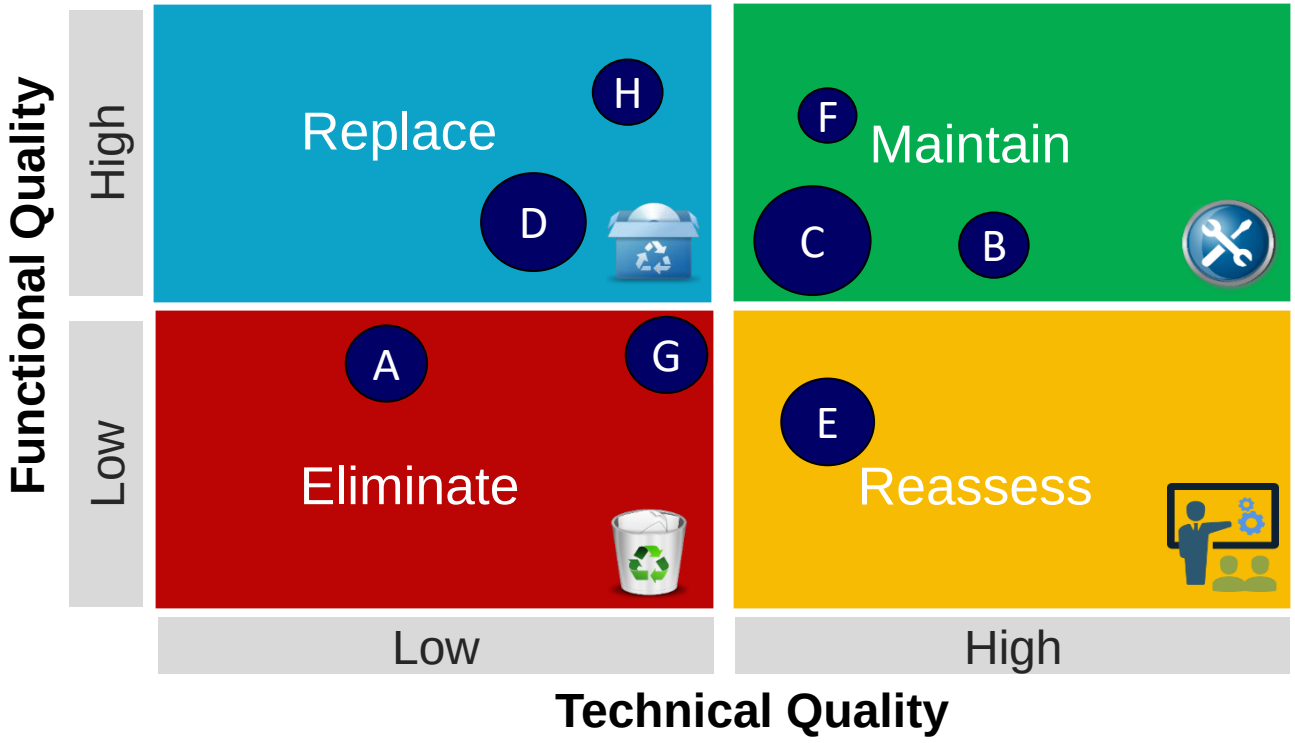
Application Portfolio Assessment



Aspek Penilaian (ISO 25010 Model)

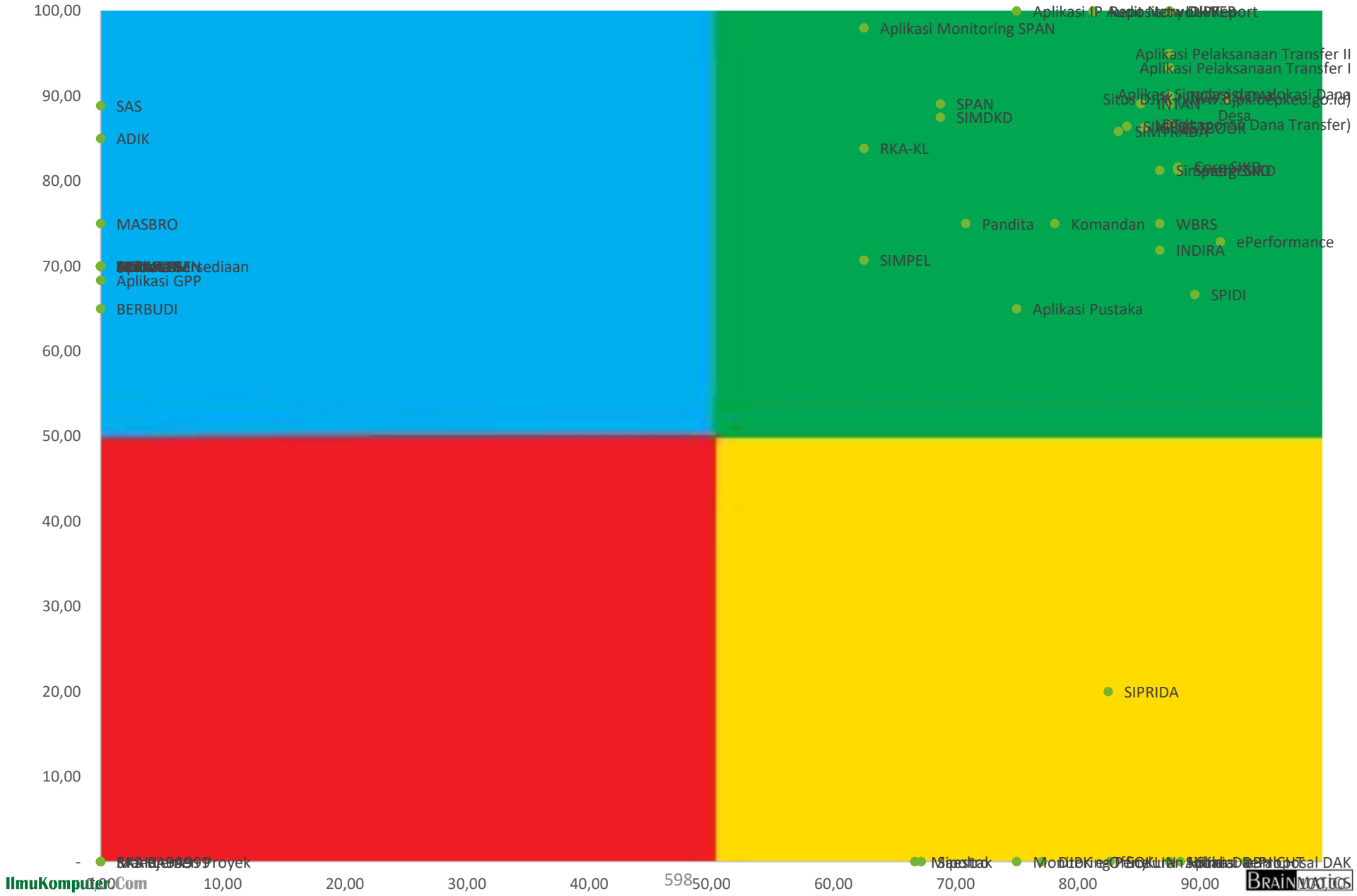
Kategori	Aspek Penilaian	Target Partisipan
Functional	Fungsionalitas	Semua Pengguna Aplikasi
	Efisiensi	
	Kemudahan	
	Keandalan	
	Frekuensi Pemakaian	
Technical	Kompatibilitas	Divisi TI atau Pengembang Aplikasi
	Keamanan	
	Pemeliharaan	
	Portabilitas	

Application Portfolio Assessment

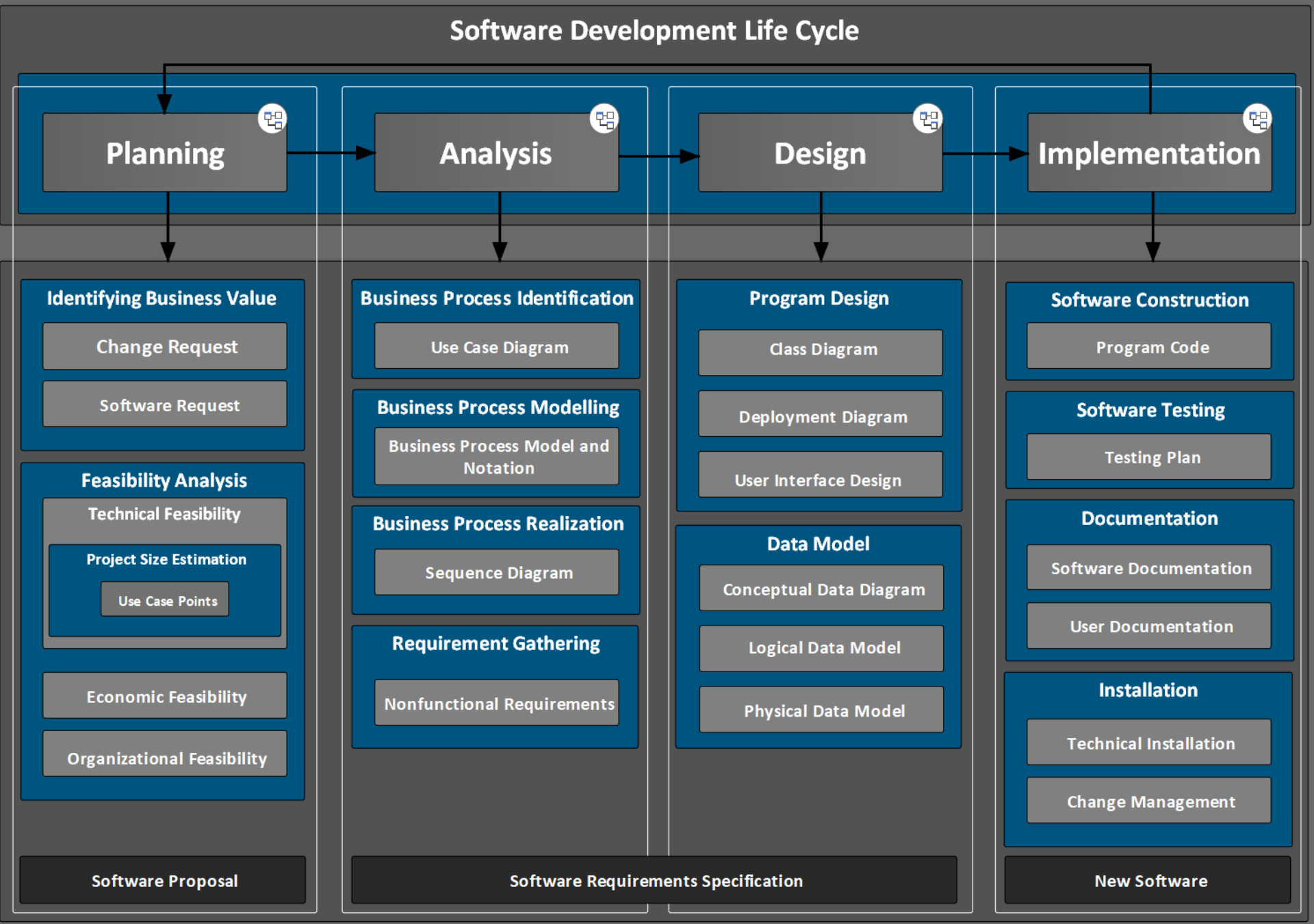


Verdict	Functional Quality	Technical Quality	Action
Eliminate	Low	Low	**Jangan gunakan** lagi sistem ini
Replace	High	Low	**Ganti dengan sistem baru** yang menjalankan fungsi yang sama
Reassess	Low	High	**Tambahkan fitur baru** yang lebih bermanfaat
Maintain	High	High	**Maintain sistem** supaya bisa terus berjalan

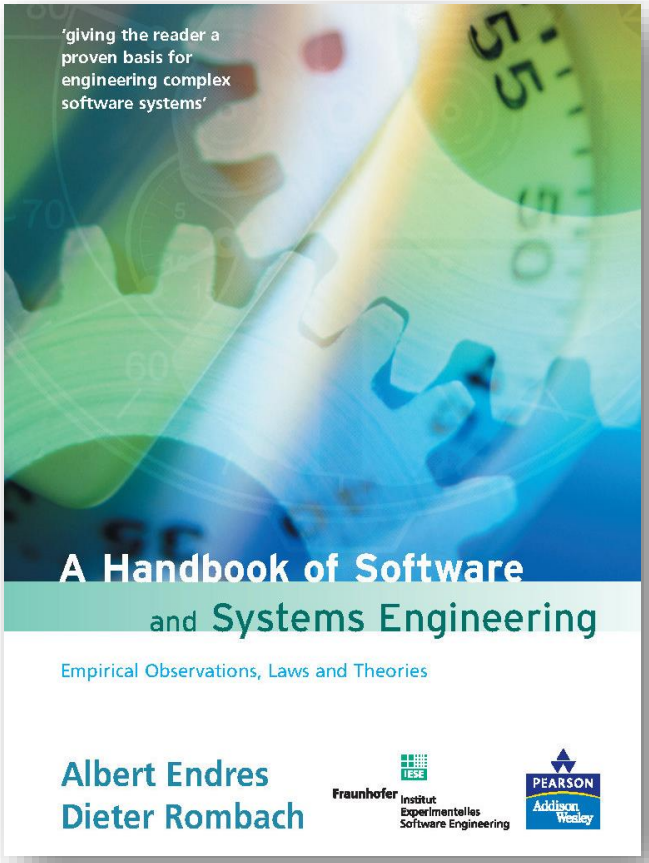
Rekapitulasi Hasil *Application Portfolio Assessment Survey*



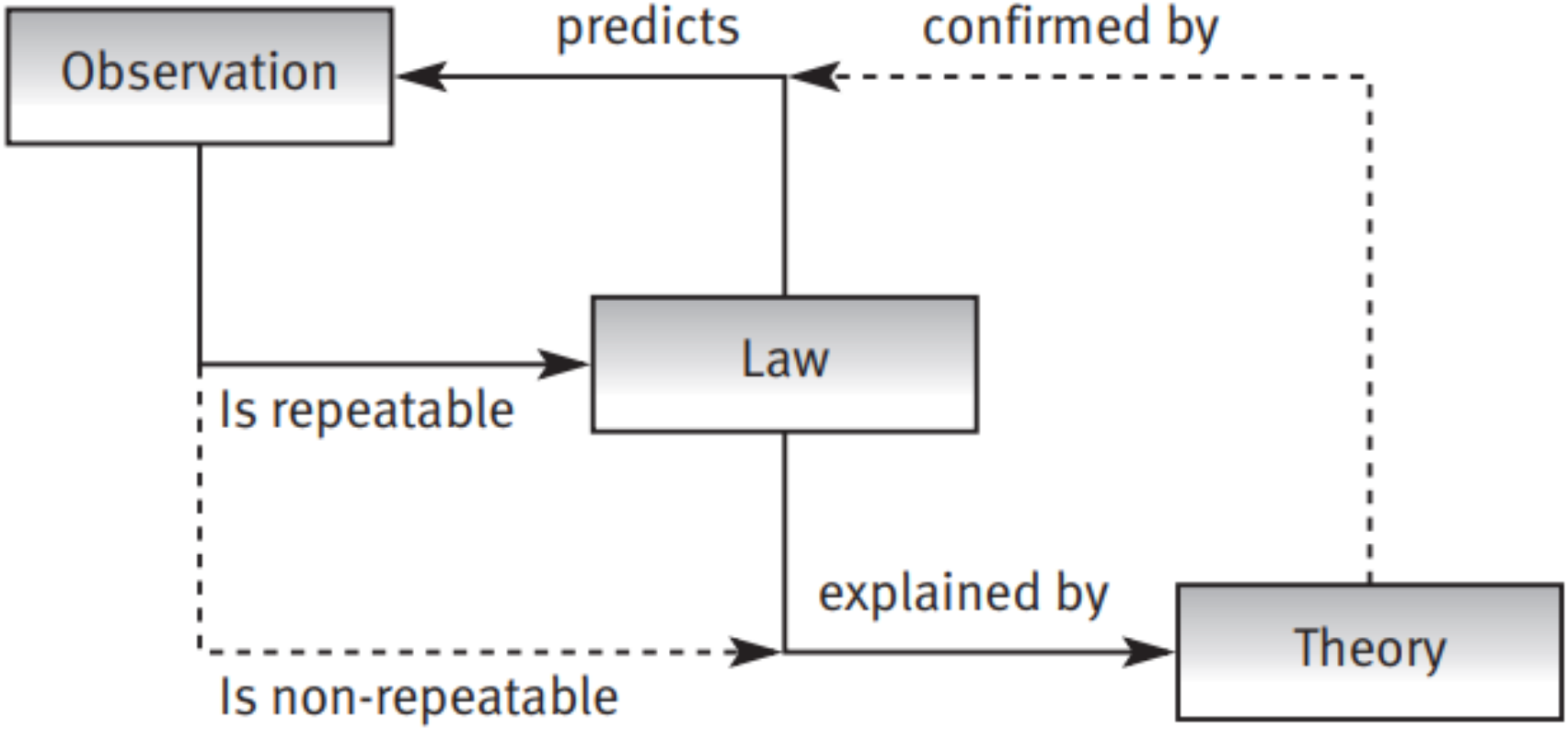
Application Development Governance



5.5 Software Engineering Laws



Observation vs Law vs Theory



Software Engineering Laws

ANALYSIS AND DESIGN

1. Requirement deficiencies are the prime source of project failures (Glass)
2. Errors are most frequent during the requirements and design activities and are the more expensive the later they are removed (Boehm 1)
3. Prototyping significantly reduces requirement and design errors, especially for user interfaces (Boehm 2)
4. The value of models depends on the view taken, but none is best for all purposes (Davis)
5. Good designs require deep application domain knowledge (Curtis)
6. Hierarchical structures reduce complexity (Simon)
7. A structure is stable if cohesion is strong and coupling low (Constantine)
8. Only what is hidden can be changed without risk (Parnas)
9. Separation of concerns leads to standard architectures (Denert)
10. Screen pointing-time is a function of distance and width (Fitts-Shneiderman)

Software Engineering Laws

CONSTRUCTION

11. What applies to small systems does not apply to large ones (DeRemer)
12. Productivity and reliability depend on the length of a program's text, independent of language level used (Corbató)
13. Well-structured programs have fewer errors and are easier to maintain (Dijkstra-Mills-Wirth)
14. The larger and more decentralized an organization, the more likely it is that it has reuse potential (Lanergan)
15. Software reuse reduces cycle time and increases productivity and quality (McIlroy)
16. A system reflects the organizational structure that built (Conway)

Software Engineering Laws

TESTING

17. Inspections significantly increase productivity, quality, and project stability (Fagan)
18. Effectiveness of inspections is fairly independent of its organizational form (Porter–Votta)
19. Perspective-based inspections are (highly) effective and efficient (Basili)
20. A combination of different V&V methods outperforms any single method alone (Hetzel–Myers)
21. Online debugging is more efficient than offline debugging (Sackman 1)
22. Testing can show the presence but not the absence of errors (Dijkstra)
23. A developer is unsuited to test his or her code (Weinberg)
24. Approximately 80 percent of defects come from 20 percent of modules (Pareto–Zipf)
25. Performance testing benefits from system-level benchmarks (Gray–Serlin)
26. Usability is quantifiable (Nielsen-Norman)

Software Engineering Laws

EVOLUTION

27. A system that is used will be changed (Lehman 1)
28. An evolving system increases its complexity, unless work is done to reduce it (Lehman 2)
29. System evolution is determined by a feedback process (Lehman 3)
30. Smaller changes have a higher error density than large ones (Basili–Möller)

Software Engineering Laws

PLANNING

31. Individual developer performance varies considerably (Sackman 2)
32. A multitude of factors influence developer productivity (Nelson–Jones)
33. Development effort is a (non-linear) function of product size (Boehm 3)
34. Most cost estimates tend to be too low (DeMarco–Glass)
35. Mature processes and personal discipline enhance planning, increase productivity, and reduce errors (Humphrey)
36. Adding manpower to a late project makes it later (Brooks)
37. Products replace services through productivity gains (Baumol)

Software Engineering Laws

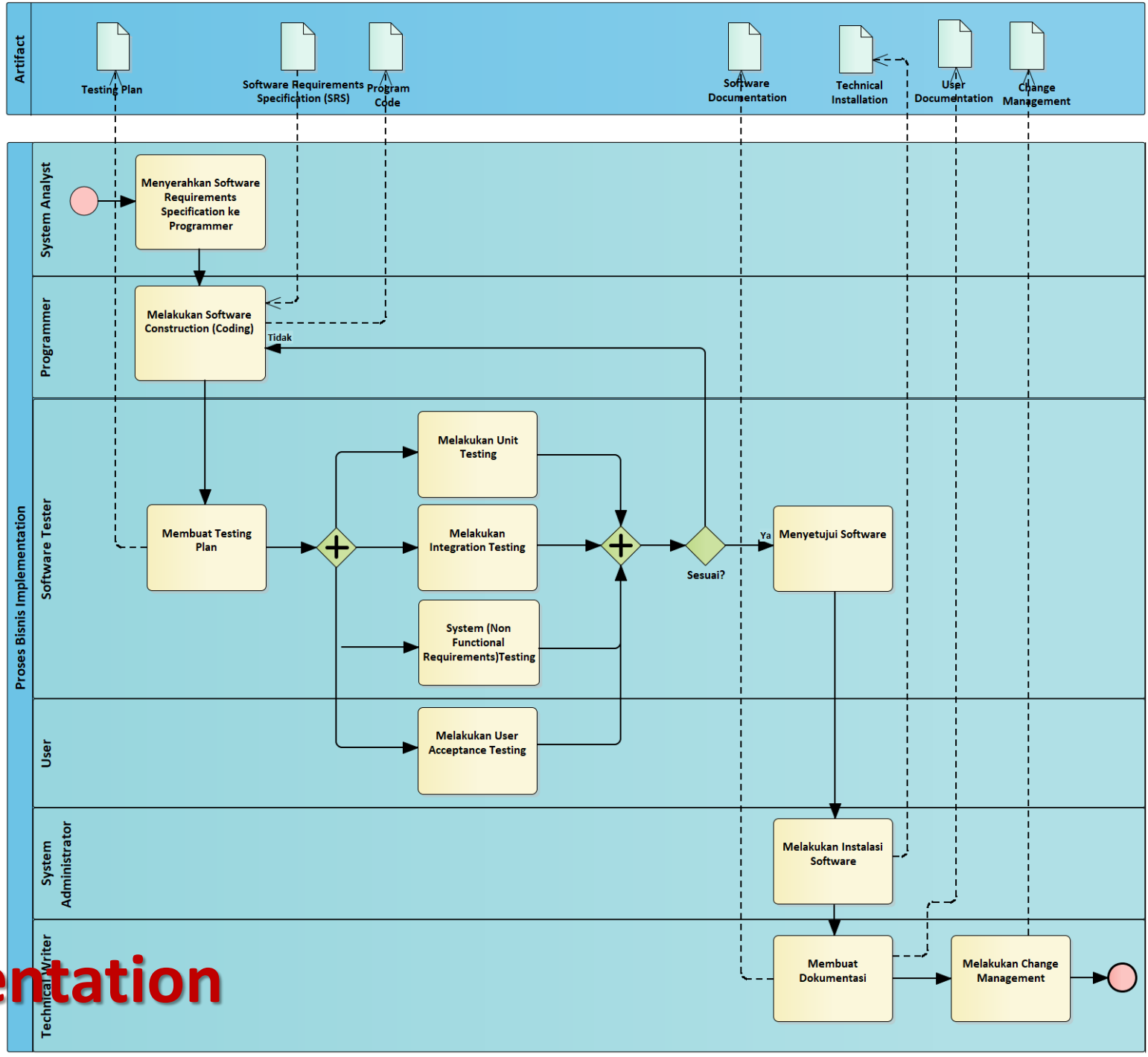
HUMAN RESOURCE

38. Humans receive most information through the visual system and store it in a spatially organized memory (Kupfmüller)
39. Humans tend to structure what they see to form cohesive patterns (Gestalt)
40. Short-term memory is limited to 7 ± 2 chunks of information (Miller)
41. Multimodal information is easier to remember than single mode (Krause)
42. The more knowledge that is available, the more effort has to be spent on the processes to use it (Librarian)
43. It takes 5000 hours to turn a novice into an expert (Apprentice)
44. Human needs and desires are strictly prioritized (Maslow-Herzberg)

Software Engineering Laws

TECHNOLOGY

45. The price and performance of processors is halved every 18 months (Moore)
46. The capacity of magnetic devices increases by a factor of ten every decade (Hoagland)
47. Wireless bandwidth doubles every 2.5 years (Cooper)
48. Architecture wins over technology (Morris-Ferguson)
49. The value of a network increases with the square of its users (Metcalfe)
50. The probability that a hypothesis is true increases the more unlikely the new event is that confirms this hypothesis (Bayes)



Systems Implementation

Post-Test

1. Sebutkan tahapan pengembangan sistem atau system development life cycle (SDLC)!
2. Sebutkan beberapa metodologi pengembangan sistem yang anda ketahui!
3. Gambarkan *requirement* di bawah dengan use case diagram!

SISTEM ELIBRARY
- Sistem elibrary memungkinkan pengguna untuk melakukan registrasi dan login
- Setelah menjadi member, pengguna dapat memodifikasi profile, serta mencari dan mendownload koleksi buku di sistem elibrary
- Admin sistem elibrary melakukan approval terhadap registrasi dan menampilkan laporan aktifitas pengguna secara individual maupun total

4. Gambarkan activity diagram, sequence diagram dan class diagram dari requirement di atas!
5. Hitung dengan metode yang anda ketahui, berapa orang dan waktu (bulan) yang dibutuhkan untuk mengembangkan sistem di atas!

References

1. Alan Dennis et al, Systems Analysis and Design with UML 5th Edition, *John Wiley and Sons*, 2016
2. Joseph S. Valacich and Joey F. George, Modern Systems Analysis and Design 8th Edition, *Pearson Education*, 2017
3. Scott Tilley and Harry J. Rosenblatt, Systems Analysis and Design 11th Edition, *Cengage Learning*, 2017
4. Kenneth E. Kendall and Julie E Kendall, Systems Analysis and Design 8th Edition, *Prentice Hall*, 2010
5. John W. Satzinger, Robert B. Jackson, Stephen D. Burd, Systems Analysis and Design in a Changing World 6th Edition, *Course Technology*, 2012
6. Hassan Gomaa, Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures, *Cambridge University Press*, 2011
7. Howard Podeswa, UML for the IT Business Analyst 2nd Edition, *Course Technology*, 2009
8. Jeffrey A. Hoffer et al, Modern Systems Analysis and Design 6th Edition, *Prentice Hall*, 2010
9. Albert Endres and Dieter Rombach, A Handbook of Software and Systems Engineering, *Pearson Education*, 2003

@brainmatics.id facebook.com/Brainmatics.id @brainmatics_id

PT Brainmatics Cipta Informatika
Menara Bidakara, Suite 0205 Jl. Gatot Subroto Kav. 71-73 Jakarta 12870
Phone: +62283793383, 83793384, 83793260, 83793261
https://brainmatics.com info@brainmatics.com