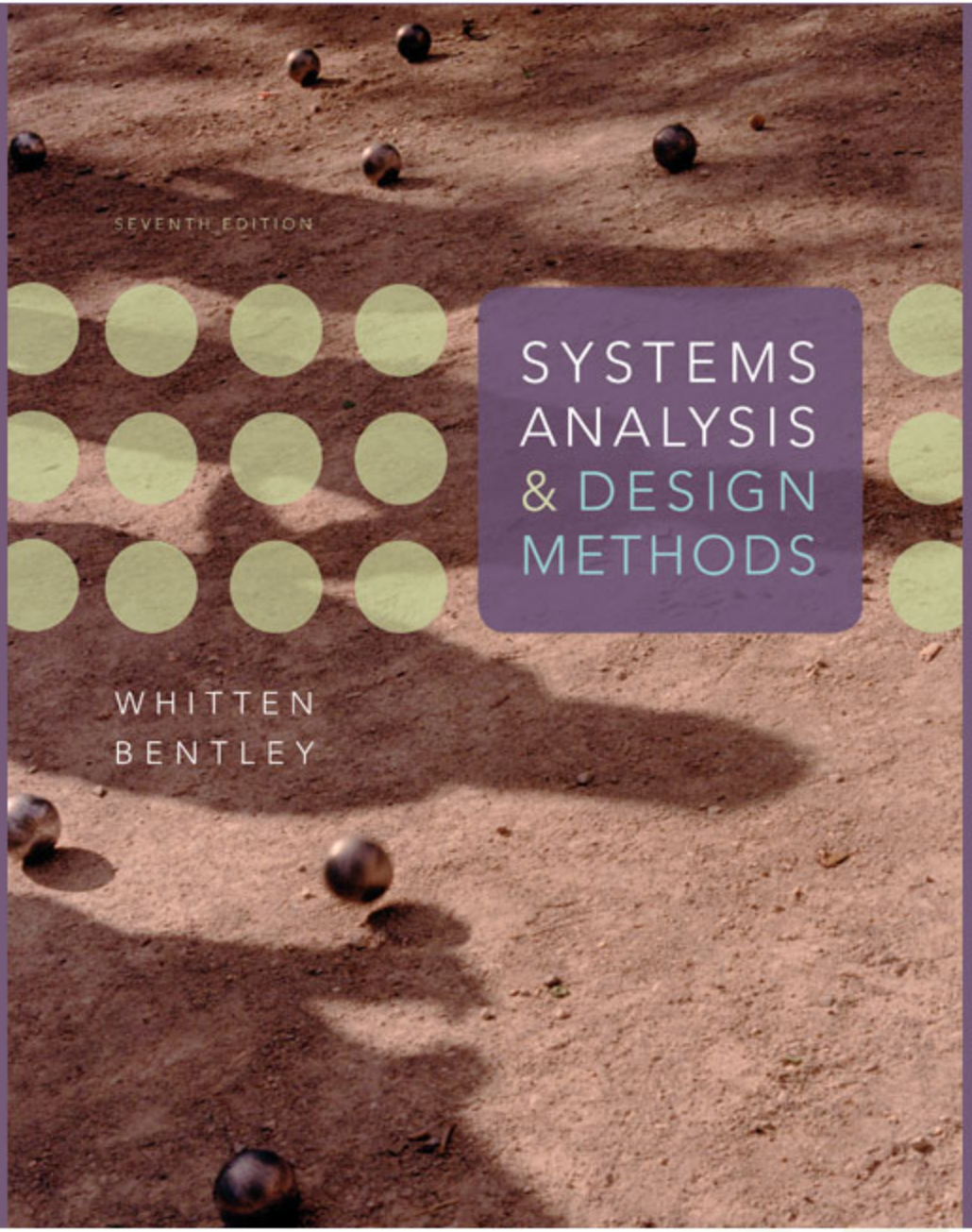
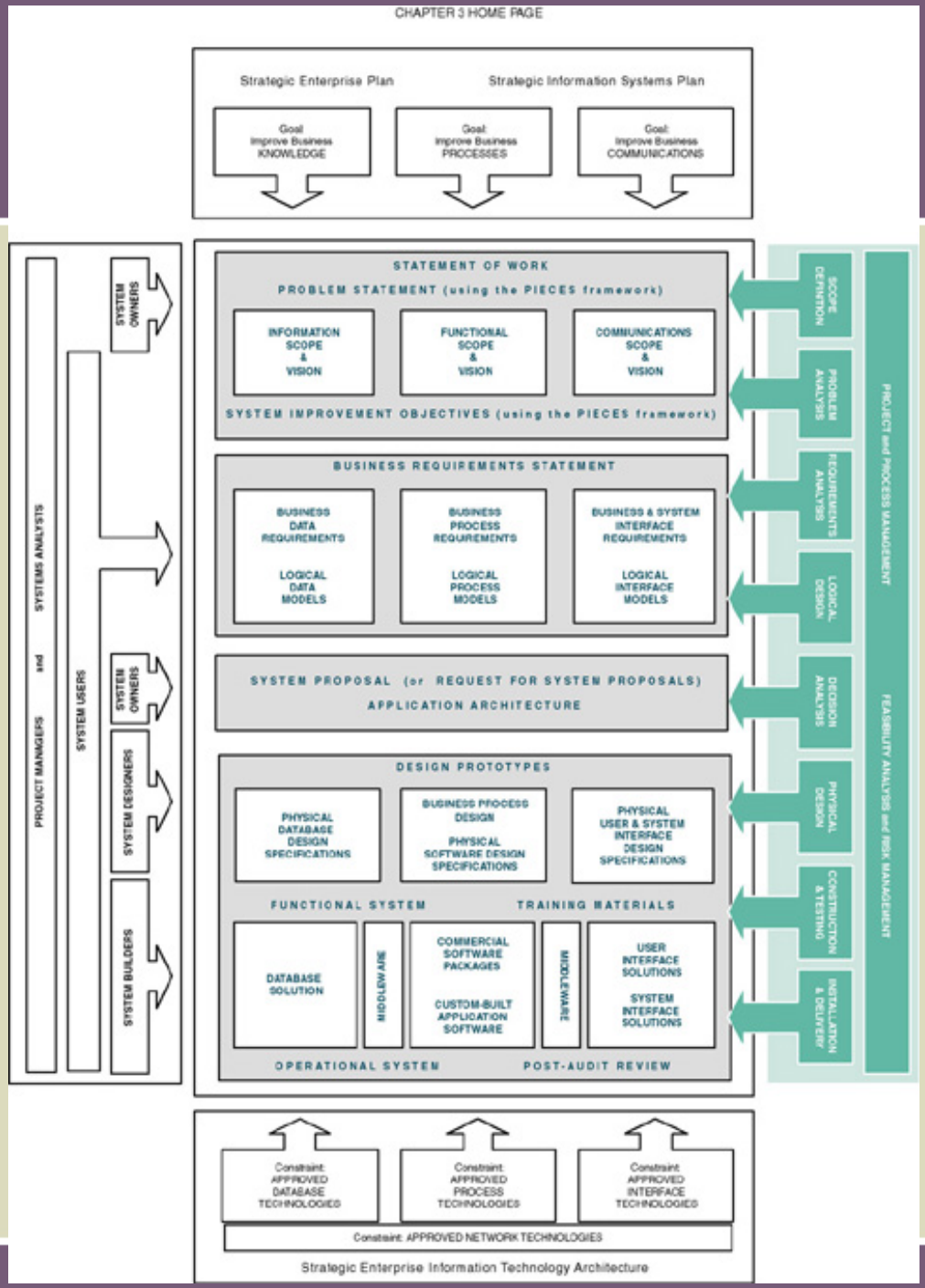
Rumuskan dengan baik hasil jawaban saudara di topik diskusi forum E-L 3


SEVENTH EDITION
SYSTEMS ANALYSIS & DESIGN METHODS
WHITTEN
BENTLEY

Objectives

- Describe the motivation for a system development process in terms of the Capability Maturity Model (CMM) for quality management.
- Differentiate between the system life cycle and a system development methodology.
- Describe 10 basic principles of system development.
- Define problems, opportunities, and directives—the triggers for systems development projects.
- Describe the PIECES framework for categorizing problems, opportunities, and directives.
- Describe the essential phases of system development. For each phase, describe its purpose, inputs, and outputs.
- Describe cross life cycle activities that overlap multiple system development phases.
- Describe typical alternative "routes" through the basic phases of system development. Describe how routes may be combined or customized for different projects.
- Describe various automated tools for system development.


CHAPTER 3 HOME PAGE
Strategic Enterprise Plan
Strategic Information Systems Plan
Goal: Improve Business KNOWLEDGE
Goal: Improve Business PROCESSES
Goal: Improve Business COMMUNICATIONS
PROJECT MANAGERS and SYSTEMS ANALYSTS
SYSTEM USERS
SYSTEM OWNERS
SYSTEM OWNERS
SYSTEM DESIGNERS
SYSTEM BUILDERS
STATEMENT OF WORK
PROBLEM STATEMENT (using the PIECES framework)
INFORMATION SCOPE & VISION
FUNCTIONAL SCOPE & VISION
COMMUNICATIONS SCOPE & VISION
SYSTEM IMPROVEMENT OBJECTIVES (using the PIECES framework)
BUSINESS REQUIREMENTS STATEMENT
BUSINESS DATA REQUIREMENTS
LOGICAL DATA MODELS
BUSINESS PROCESS REQUIREMENTS
LOGICAL PROCESS MODELS
BUSINESS & SYSTEM INTERFACE REQUIREMENTS
LOGICAL INTERFACE MODELS
SYSTEM PROPOSAL (or REQUEST FOR SYSTEM PROPOSALS)
APPLICATION ARCHITECTURE
DESIGN PROTOTYPES
PHYSICAL DATABASE DESIGN SPECIFICATIONS
BUSINESS PROCESS DESIGN
PHYSICAL SOFTWARE DESIGN SPECIFICATIONS
PHYSICAL USER & SYSTEM INTERFACE DESIGN SPECIFICATIONS
FUNCTIONAL SYSTEM
TRAINING MATERIALS
DATABASE SOLUTION
MIDDLEWARE
COMMERCIAL SOFTWARE PACKAGES
CUSTOM-BUILT APPLICATION SOFTWARE
MIDDLEWARE
USER INTERFACE SOLUTIONS
SYSTEM INTERFACE SOLUTIONS
OPERATIONAL SYSTEM
POST-AUDIT REVIEW
SCOPE DEFINITION
PROBLEM ANALYSIS
REQUIREMENTS ANALYSIS
LOGICAL DESIGN
DECISION ANALYSIS
PHYSICAL DESIGN
CONSTRUCTION & TESTING
INSTALLATION & DELIVERY
PROJECT and PROCESS MANAGEMENT
FEASIBILITY ANALYSIS and RISK MANAGEMENT
Constraint: APPROVED DATABASE TECHNOLOGIES
Constraint: APPROVED PROCESS TECHNOLOGIES
Constraint: APPROVED INTERFACE TECHNOLOGIES
Constraint: APPROVED NETWORK TECHNOLOGIES
Strategic Enterprise Information Technology Architecture

Process of System Development

System development process – a set of activities, methods, best practices, deliverables, and automated tools that stakeholders (Chapter 1) use to develop and continuously improve information systems and software (Chapters 1 and 2).

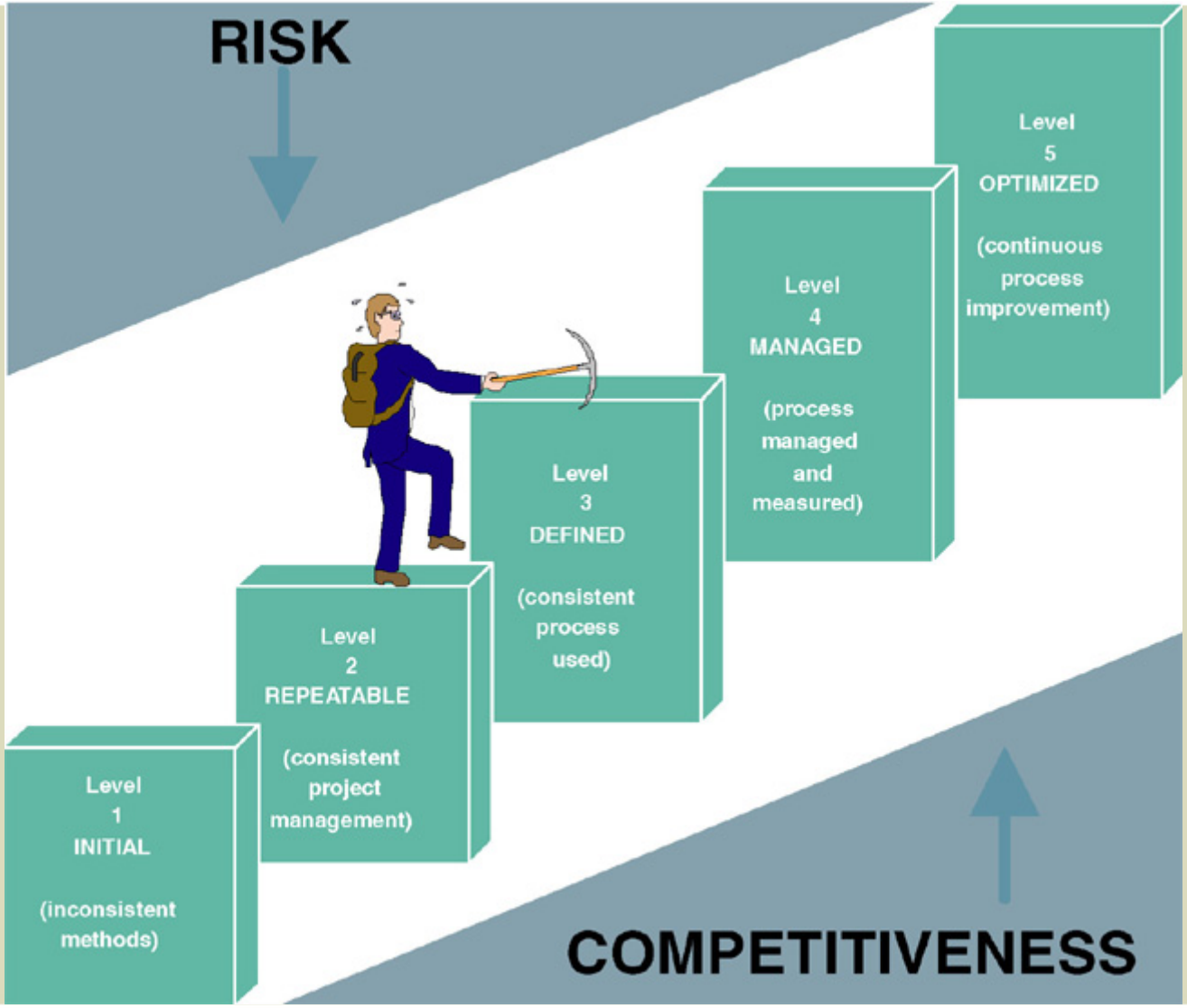
- Many variations
- Using a consistent process for system development:
  - Create efficiencies that allow management to shift resources between projects
  - Produces consistent documentation that reduces lifetime costs to maintain the systems
  - Promotes quality

CMM Process Management Model

Capability Maturity Model (CMM) – a standardized framework for assessing the maturity level of an organization's information system development and management processes and products. It consists of five levels of maturity:

- **Level 1—Initial**: System development projects follow no prescribed process.
- **Level 2—Repeatable**: Project management processes and practices established to track project costs, schedules, and functionality.
- **Level 3—Defined**: Standard system development process (methodology) is purchased or developed. All projects use a version of this process.
- **Level 4—Managed**: Measurable goals for quality and productivity are established.
- **Level 5—Optimizing**: The standardized system development process is continuously monitored and improved based on measures and data analysis established in Level 4.

Capability Maturity Model (CMM)



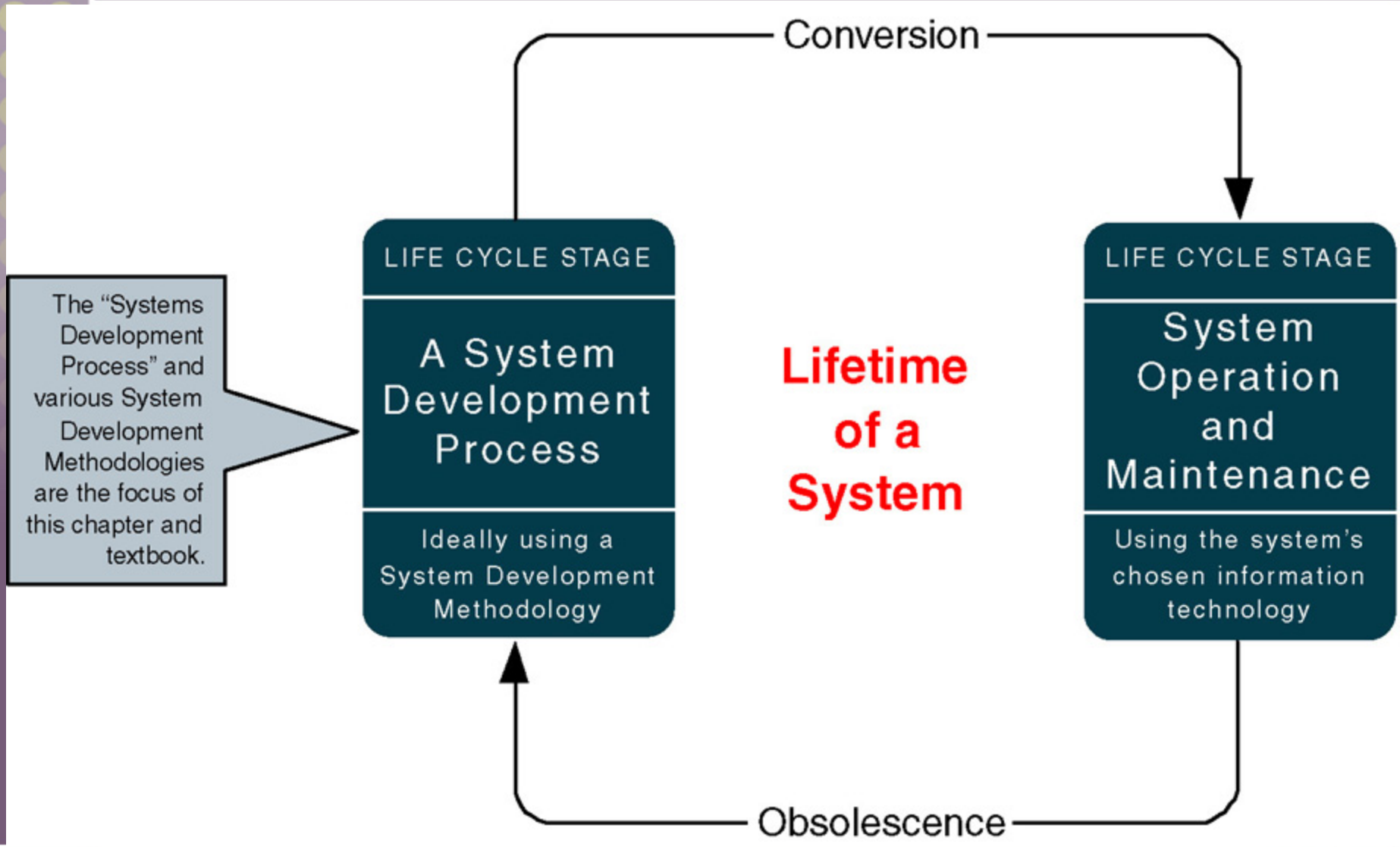
Impact of System Development "Process" on Quality

CMM Project Statistics for a Project Resulting in 200,000 Lines of Code						
Organization's CMM Level	Project Duration (months)	Project Person-Months	Number of Defects Shipped	Median Cost ($ millions)	Lowest Cost ($ millions)	Highest Cost ($ millions)
1	30	600	61	5.5	1.8	100+
2	18.5	143	12	1.3	.96	1.7
3	15	80	7	.728	.518	.933

Life Cycle versus Methodology

- **System life cycle** – the factoring of the lifetime of an information system into two stages, (1) systems development and (2) systems operation and maintenance.
- **System development methodology** – a formalized approach to the systems development process; a standardized development process that defines (as in CMM Level 3) a set of activities, methods, best practices, deliverables, and automated tools that system developers and project managers are to use to develop and continuously improve information systems and software.

A System Life Cycle



Representative System Development Methodologies

- Architected Rapid Application Development (Architected RAD)
- Dynamic Systems Development Methodology (DSDM)
- Joint Application Development (JAD)
- Information Engineering (IE)
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Structured Analysis and Design
- eXtreme Programming (XP)

Principles of System Development

- Get the system users involved.
- Use a problem-solving approach.
- Establish phases and activities.
- Document through development.
- Establish standards.
- Manage the process and projects
- Justify systems as capital investments.
- Don't be afraid to cancel or revise scope.
- Divide and conquer.
- Design systems for growth and change.

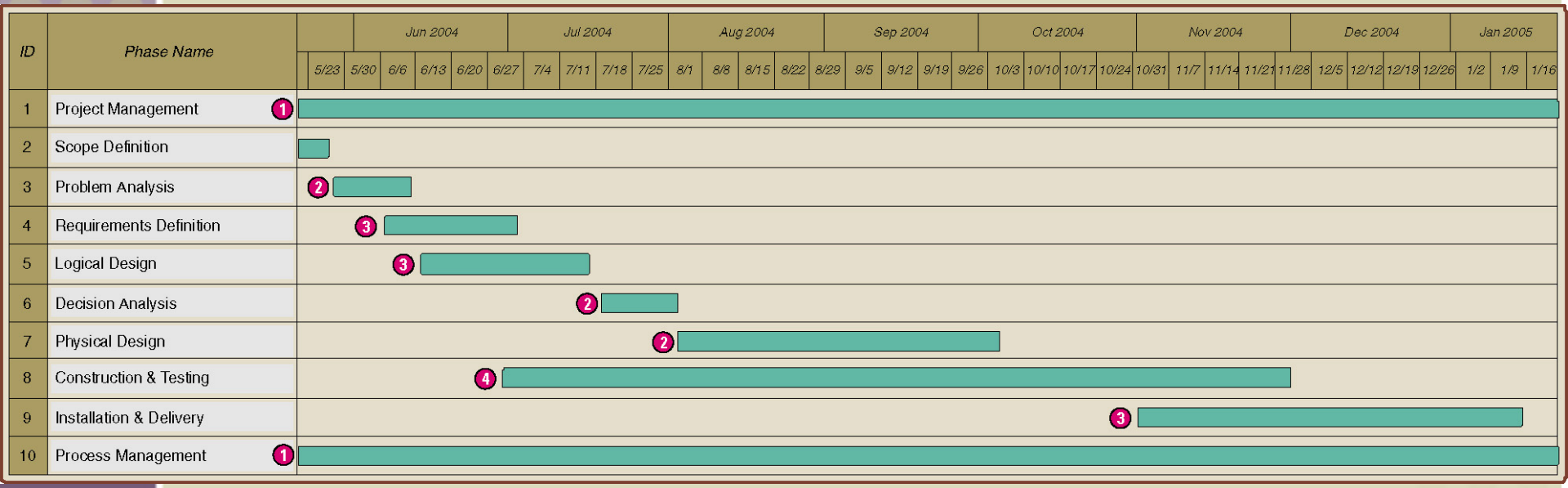
Use a Problem-Solving Approach

Classical Problem-solving approach

1. Study and understand the problem, its context, and its impact.
2. Define the requirements that must be meet by any solution.
3. Identify candidate solutions that fulfill the requirements, and select the "best" solution.
4. Design and/or implement the chosen solution.
5. Observe and evaluate the solution's impact, and refine the solution accordingly.

Establish Phases and Activities

Overlap of System Development Phases



Manage the Process and Projects

Process management – an ongoing activity that documents, manages, oversees the use of, and improves an organization's chosen methodology (the "process") for system development. Process management is concerned with phases, activities, deliverables, and quality standards should be consistently applied to all projects.

Project management is the process of scoping, planning, staffing, organizing, directing, and controlling a project to develop an information system at a minimum cost, within a specified time frame, and with acceptable quality.

Justify Information Systems as Capital Investments

Cost-effectiveness – The result obtained by striking a balance between the lifetime costs of developing, maintaining, and operating an information system and the benefits derived from that system. Cost-effectiveness is measured by a cost-benefit analysis.

Strategic information systems plan – a formal strategic plan (3-5 years) for building and improving an information technology infrastructure and the information system applications that use that infrastructure.

Strategic enterprise plan – a formal strategic plan (3-5 years) for an entire business that defines its mission, vision, goals, strategies, benchmarks, and measures of progress and achievement. Usually, the strategic enterprise plan is complemented by strategic business unit plans that define how each business unit will contribute to the enterprise plan. The information systems plan is one of those unit-level plans.

Don’t Be Afraid to Cancel or Revise Scope

Creeping commitment – a strategy in which feasibility and risks are continuously reevaluated throughout a project. Project budgets and deadlines are adjusted accordingly.

Risk management – the process of identifying, evaluating, and controlling what might go wrong in a project before it becomes a threat to the successful completion of the project or implementation of the information system. Risk management is drive by risk analysis or assessment.

Where Do Systems Development Projects Come From?

- **Problem** – an undesirable situation that prevents the organization from fully achieving its purpose, goals, and/or objectives.
- **Opportunity** – a chance to improve the organization even in the absence of an identified problem.
- **Directive** - a new requirement that is imposed by management, government, or some external influence.

Where Do Systems Development Projects Come From?

- Planned Projects
  - An **information systems strategy plan** has examined the business as a whole to identify those system development projects that will return the greatest strategic (long-term) value to the business
  - A **business process redesign** has thoroughly analyzed a series of business processes to eliminate redundancy and bureaucracy and to improve efficiency and value added. Not it is time to redesign the supporting information system for those redesigned business processes.

Where Do Systems Development Projects Come From?

- Unplanned projects
  - Triggered by a specific problem, opportunity, or directive that occurs in the course of doing business.
  - **Steering committee** – an administrative body of system owners and information technology executives that prioritizes and approves candidate system development projects.
  - **Backlog** – a repository of project proposals that cannot be funded or staffed because they are a lower priority than those that have been approved for system development.

The PIECES Problem-Solving Framework

James Watherbee: framework for classifying problem

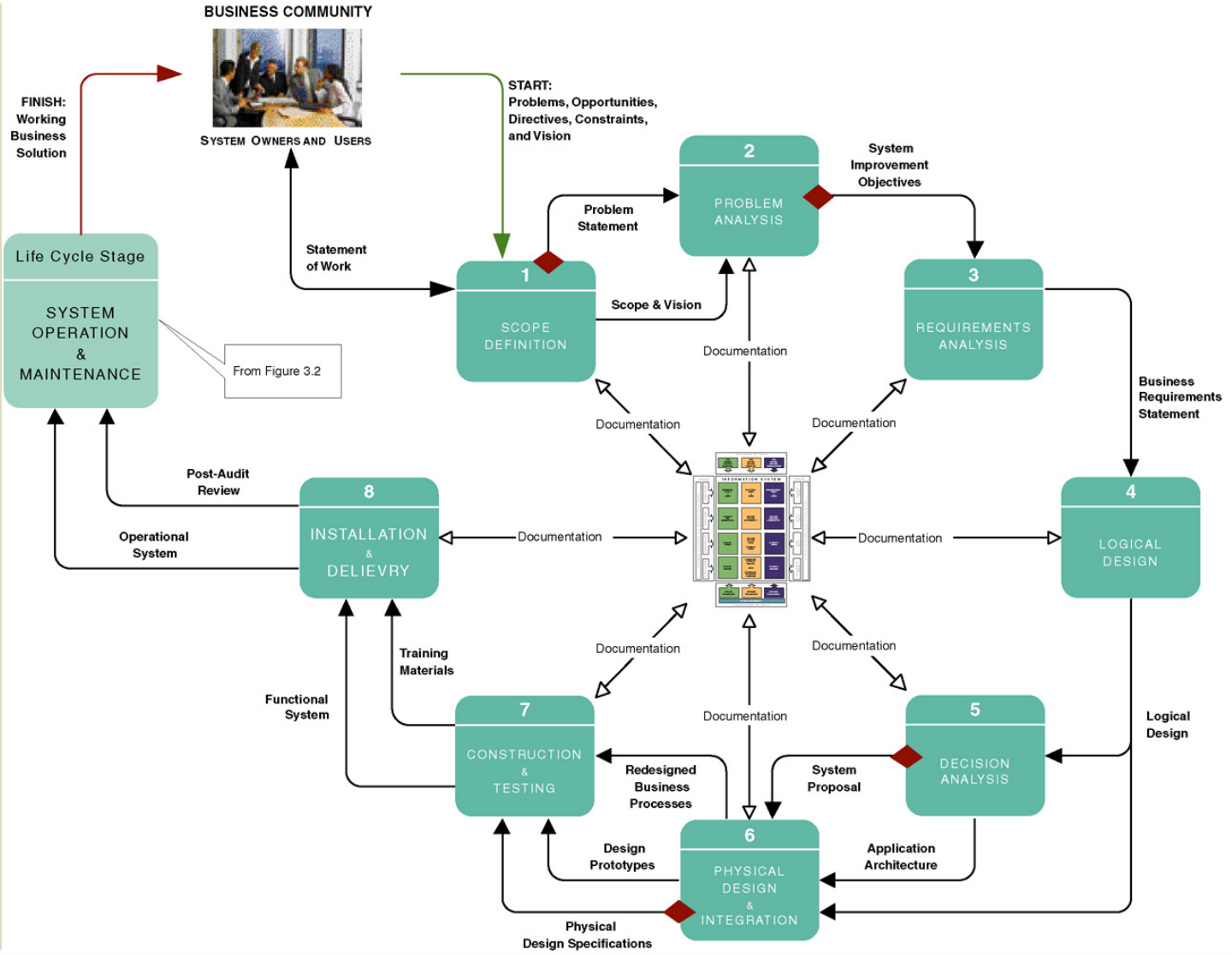
P	the need to improve performance
I	the need to improve information (and data)
E	the need to improve economics, control costs, or increase profits
C	the need to improve control or security
E	the need to improve efficiency of people and processes
S	the need to improve service to customers, suppliers, partners, employees, etc.

Project Phases

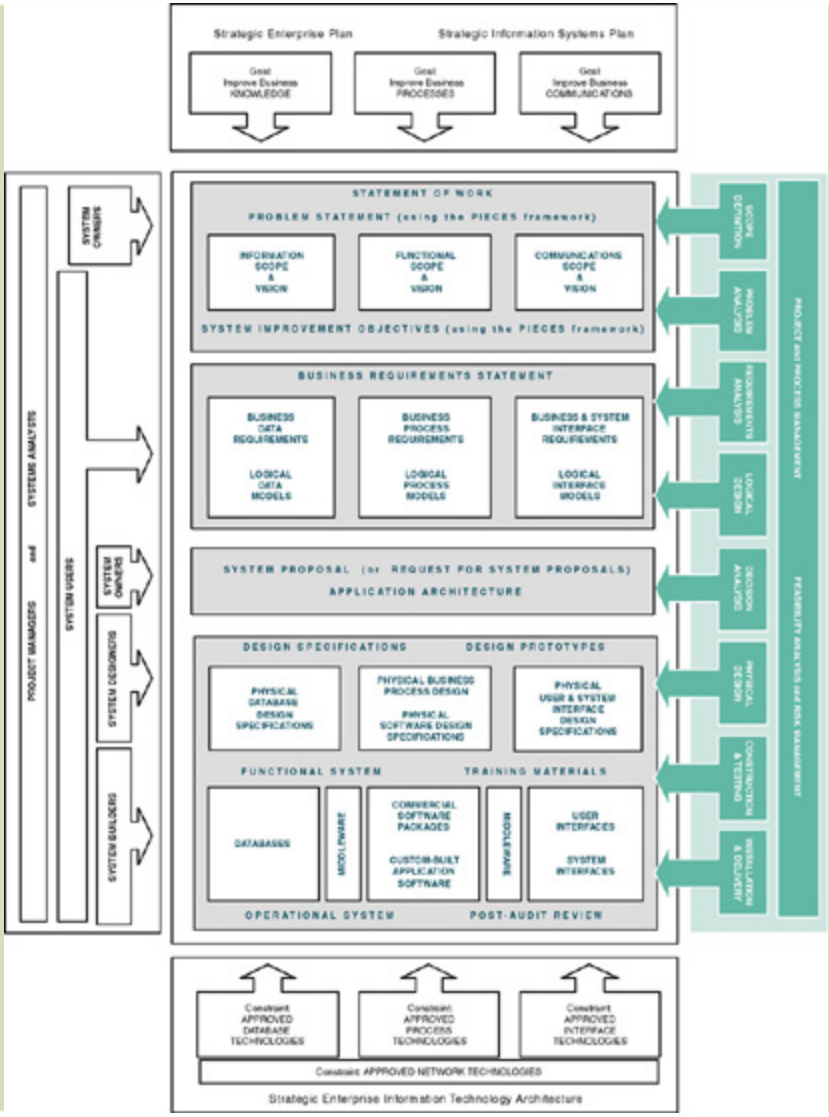
- **FAST** - (Framework for the Application of Systems Thinking) a hypothetical methodology used throughout this book to demonstrate a representative systems development process.
- Each methodology will use different project phases.

FAST Phases	**Classic Phases (from Chapter 1)**			
	Project Initiation	System Analysis	System Design	System Implementation
Scope Definition	X			
Problem Analysis	X	X		
Requirements Analysis		X		
Logical Design		X		
Decision Analysis	(a system analysis transition phase)			
Physical Design and Integration			X	
Construction and Testing			X	X
Installation and Delivery				X

FAST Project Phases



Building Blocks View of System Development



Scope Definition Phase

Problem statement – a statement and categorization of problems, opportunities, and directives; may also include constraints and an initial vision for the solution. Synonyms include *preliminary study* and *feasibility assessment*.

Constraint – any factor, limitation, or restraint that may limit a solution or the problem-solving process.

Scope creep – a common phenomenon wherein the requirements and expectations of a project increase, often without regard to the impact on budget and schedule.

Statement of work – a contract with management and the user community to develop or enhance an information system; defines vision, scope, constraints, high-level user requirements, schedule, and budget. Synonyms include *project charter*, *project plan*, and *service-level agreement*.

Requirements Analysis Phase

- What capabilities should the new system provide for its users?
- What data must be captured and stored?
- What performance level is expected?
- What are the priorities of the various requirements?

Logical Design Phase

Logical design – the translation of business user requirements into a system model that depicts only the business requirements and not any possible technical design or implementation of those requirements. Common synonyms include *conceptual design* and *essential design*.

System model – a picture of a system that represents reality or a desired reality. System models facilitate improved communication between system users, system analysts, system designers, and system builders.

Analysis paralysis – a satirical term coined to describe a common project condition in which excessive system modeling dramatically slows progress toward implementation of the intended system solution.

Decision Analysis Phase

- Candidate solutions evaluated in terms of:
  - **Technical feasibility** – Is the solution technically practical? Does our staff have the technical expertise to design and build this solution?
  - **Operational feasibility** – Will the solution fulfill the users' requirements? To what degree? How will the solution change the users' work environment? How do users feel about such a solution?
  - **Economic feasibility** – Is the solution cost-effective?
  - **Schedule feasibility** – Can the solution be designed and implemented within an acceptable time?
  - **Risk feasibility** – What is the probability of a successful implementation using the technology and approach?

Physical Design & Integration Phase

Physical design – the translation of business user requirements into a system model that depicts a technical implementation of the users' business requirements. Common synonyms include *technical design* or *implementation model*.

Two extreme philosophies of physical design

- *Design by specification* – physical system models and detailed specification are produced as a series of written (or computer-generated) blueprints for construction.
- *Design by prototyping* – Incomplete but functioning applications or subsystems (called *prototypes*) are constructed and refined based on feedback from users and other designers.

Construction and Testing Phase

- Construct and test system components
  - Software
    - Purchased
    - Custom-built
  - Databases
  - User and System Interfaces
  - Hardware
  - Networks

Installation and Delivery Phase

- Deliver the system into operation (production)
- Deliver User training
- Deliver completed documentation
- Convert existing data

System Operation & Maintenance

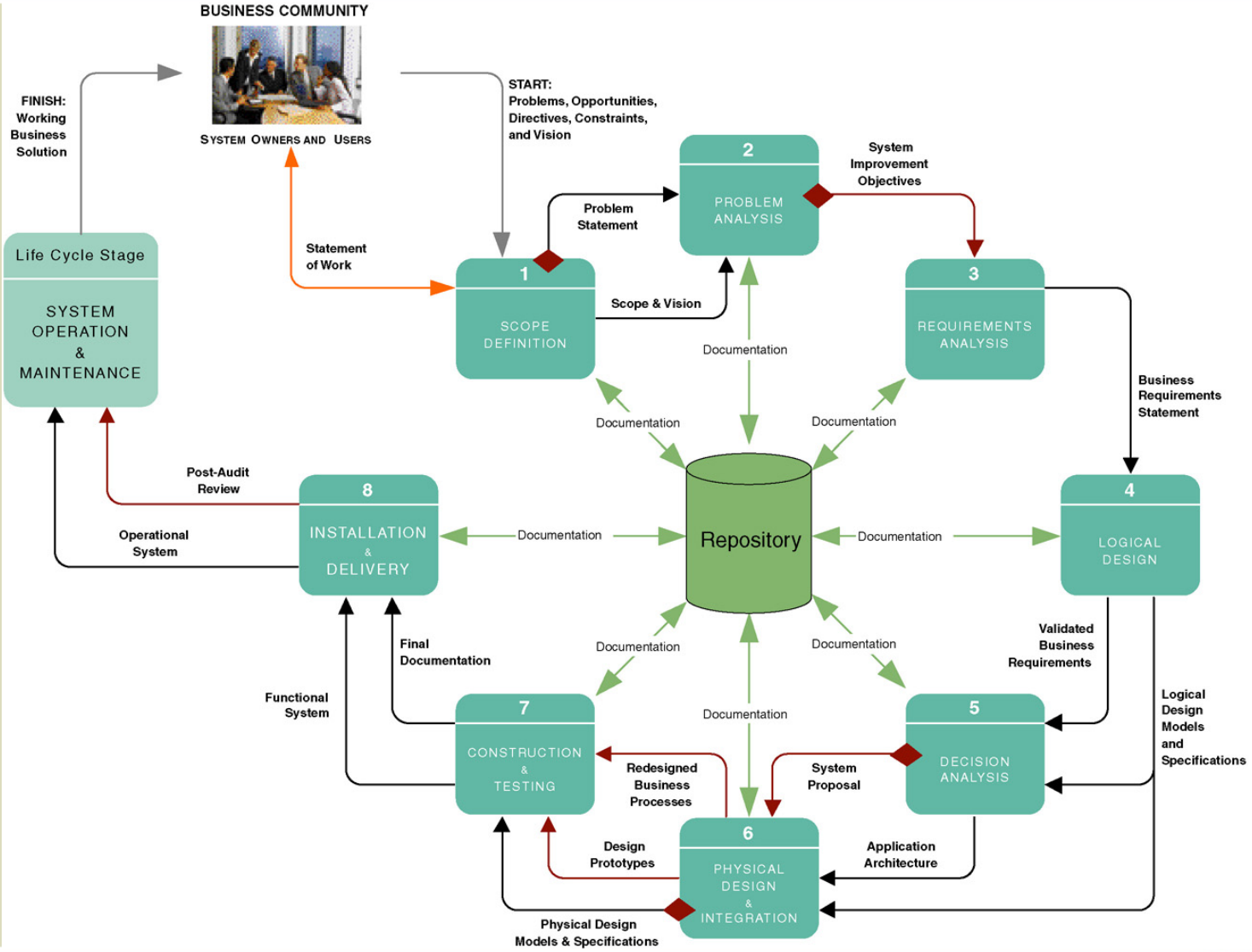
System support – the ongoing technical support for users of a system, as well as the maintenance required to deal with any errors, omissions, or new requirements that may arise.

Cross Life-Cycle Activities

Cross life-cycle activity – activities that overlap multiple phases

- **Fact-finding** - formal process of using research, interviews, meetings, questionnaires, sampling, and other techniques to collect information about system problems, requirements,and preferences.
- **Documentation and presentation**
  - **Documentation** – recording facts and specifications for a systems for current and future reference.
  - **Presentation** – communicating findings, recommendations, and documentation for review by interested users and mangers.
  - **Repository** – database and/or file directory where system developers store all documentation, knowledge, and artifacts for information systems or project(s).
- **Feasibility analysis**
- **Process and project management**

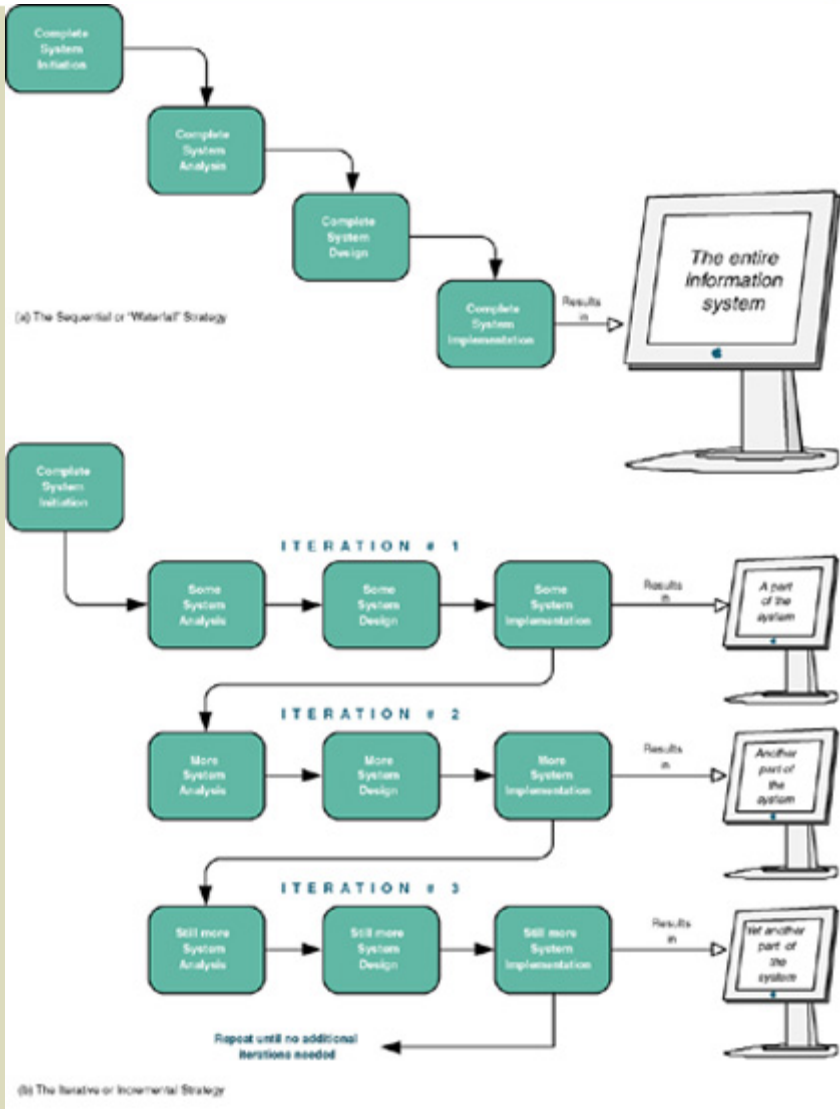
System Development Documentation, Repository, and Presentations



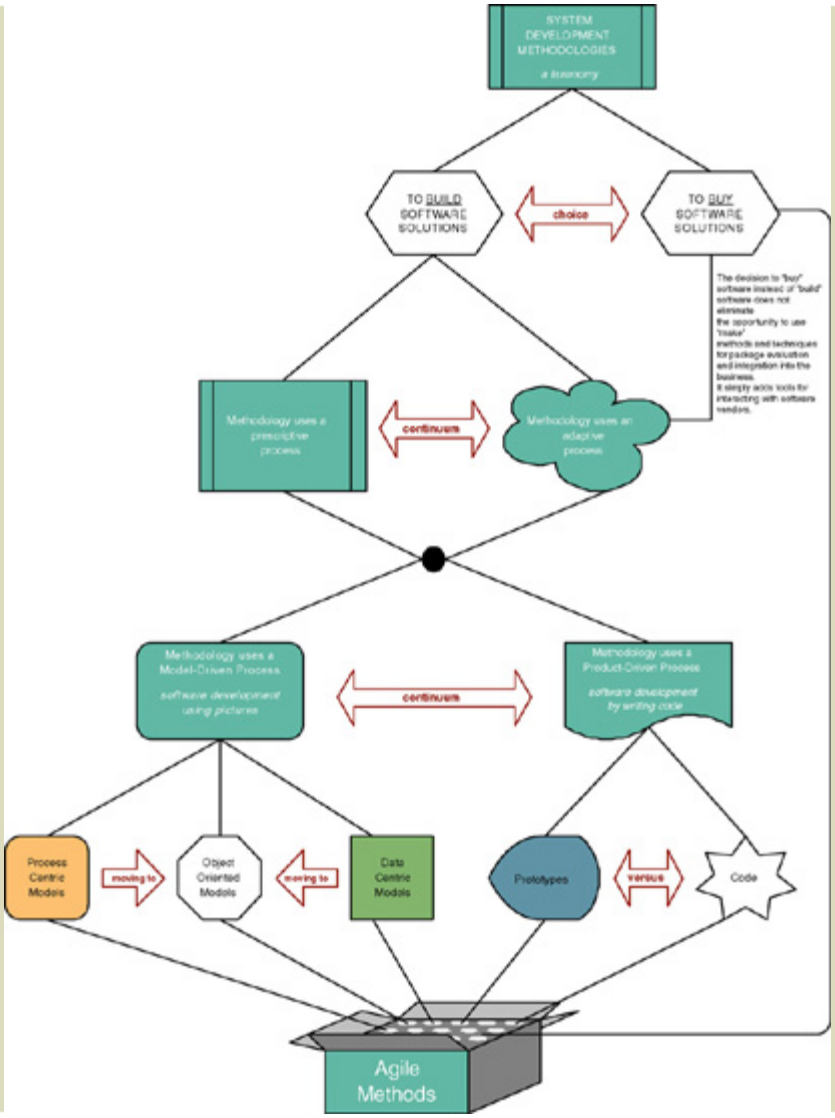
Sequential versus Iterative Development

Waterfall development approach an approach to systems analysis and design that completes each phase one after another and only once .

Iterative development approach an approach to systems analysis and design that completes the entire information system in successive iterations. Each iterations does some analysis, some design, and some construction. Synonyms include incremental and spiral.



A Taxonomy for System Development Methodologies & Strategies



Model-Driven Development Strategy

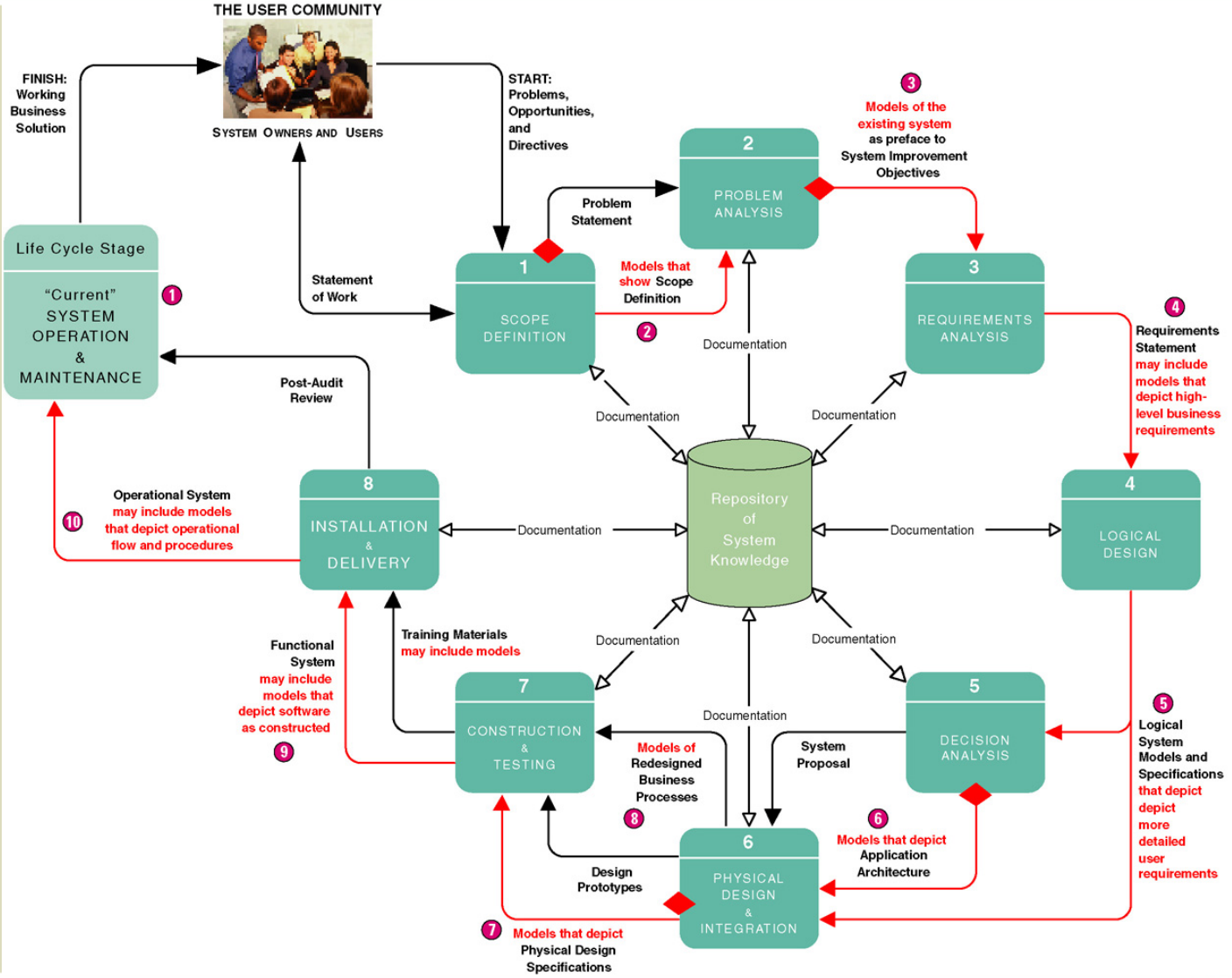
- **Model-driven development** – a system development strategy that emphasizes the drawing of system models to help visualize and analyze problems, define business requirements, and design information systems.
  - **Process modeling** – a process-centered technique popularized by the structured analysis and design methodology that used models of business process requirements to derive effective software designs for a system.
  - **Data modeling** – a data-centered technique used to model business data requirements and design database systems that fulfill those requirements.
  - **Object modeling** – a technique that attempts to merge the data and process concerns into singular constructs called objects. Object models are diagrams that document a system in terms of its objects and their interactions.

Logical vs. Physical Models

Logical model - a pictorial representation that depicts what a system is or does.

Physical model - a technical pictorial representation that depicts what a system is or does and how the system is implemented.

Model-Driven Development Strategy



Model-Driven Development Strategy

Advantages

- Requirements often more thorough
- Easier to analyze alternatives
- Design specifications often more stable and flexible
- Systems can be constructed more correctly the first time

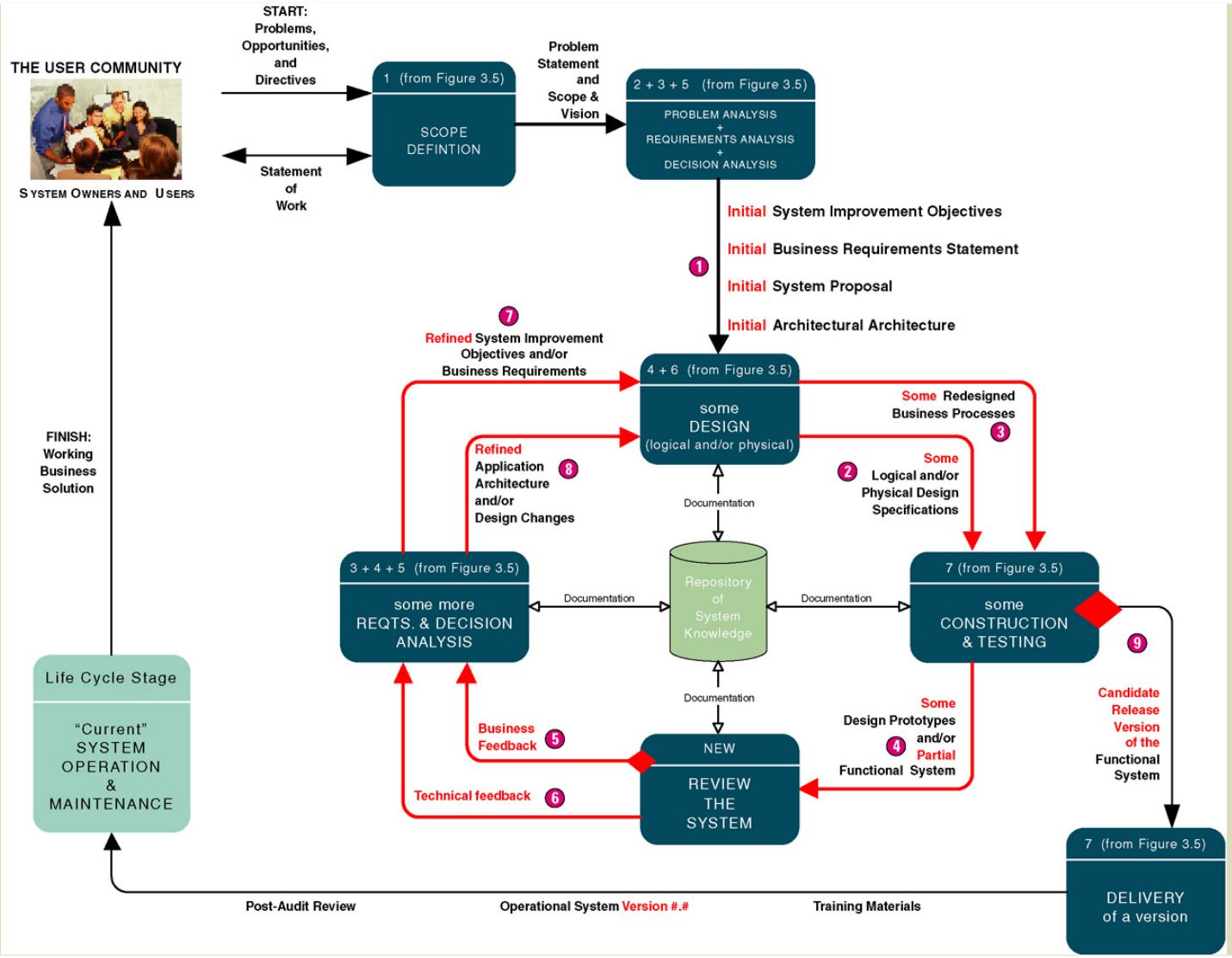
Disadvantages

- Time consuming
- Models only as good as users' understanding of requirements
- Reduces users' role because pictures are not software
- Can be Inflexible

Rapid Application Development Strategy

- **Rapid application development** (RAD) – a system development strategy that emphasizes speed of development through extensive user involvement in the rapid, iterative, and incremental construction of series of functioning prototypes of a system that eventually evolves into the final system.
  - **Prototype** – a small-scale, representative, or working model of the users' requirements or a proposed design for an information system.
  - **Time box** – the imposition of a non-extendable period of time, usually 60-90 days, by which the first (or next) version of a system must be delivered into operation.

Rapid Application Development Strategy



Rapid Application Development Strategy

Advantages

- User requirements often uncertain or imprecise
- Encourages active user and management participation
- Projects get higher visibility and support
- Stakeholders see working solutions more rapidly
- Errors detected earlier
- Testing and training are natural by-products
- More natural process because change is expected

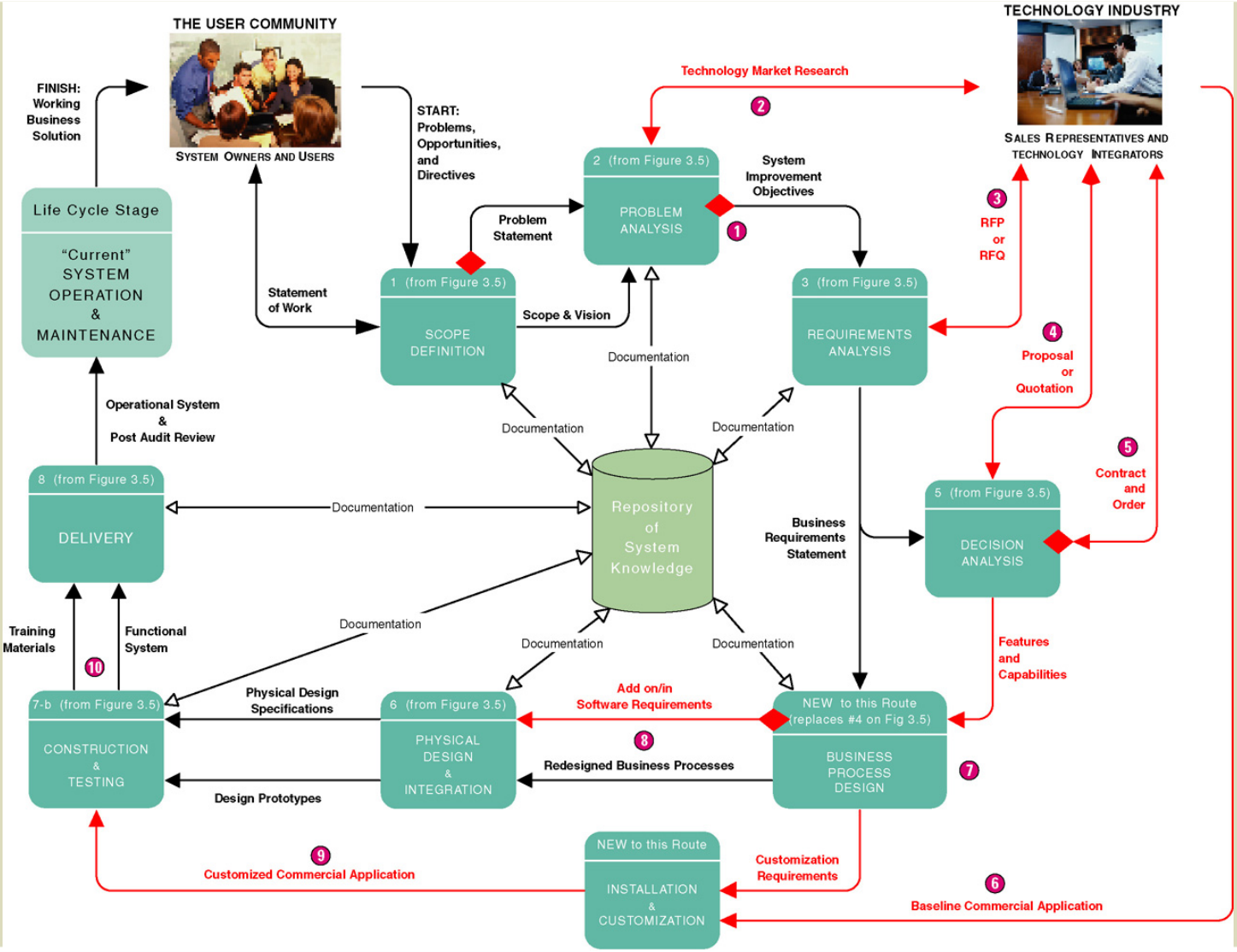
Disadvantages

- May encourage "code, implement, repair" mentality
- Can solve wrong problem since problem analysis is abbreviated
- May discourage analysts from considering alternatives
- Stakeholders reluctant to throw away prototype
- Emphasis on speed can adversely impact quality

Commercial Application Package Implementation Strategy

- **Commercial application package** – software application that can be purchased and customized to meet business requirements of a large number of organizations or specific industry. A synonym is *commercial off-the-shelf* (COTS) system.
  - **Request for proposal (RFP)** – formal document that communicates business, technical, and support requirements for application software package to vendors that may wish to compete for the sale of application package and services.
  - **Request for quotation (RFQ)** – formal document that communicates business, technical, and support requirements for an application software package to a single vendor that has been determined as being able to supply that application package and services.
  - **Gap analysis** – comparison of business and technical requirements for a commercial application package against capabilities and features of a specific commercial application package to define requirements that cannot be met.

Commercial Application Package Implementation Strategy



Commercial Application Package Implementation Strategy

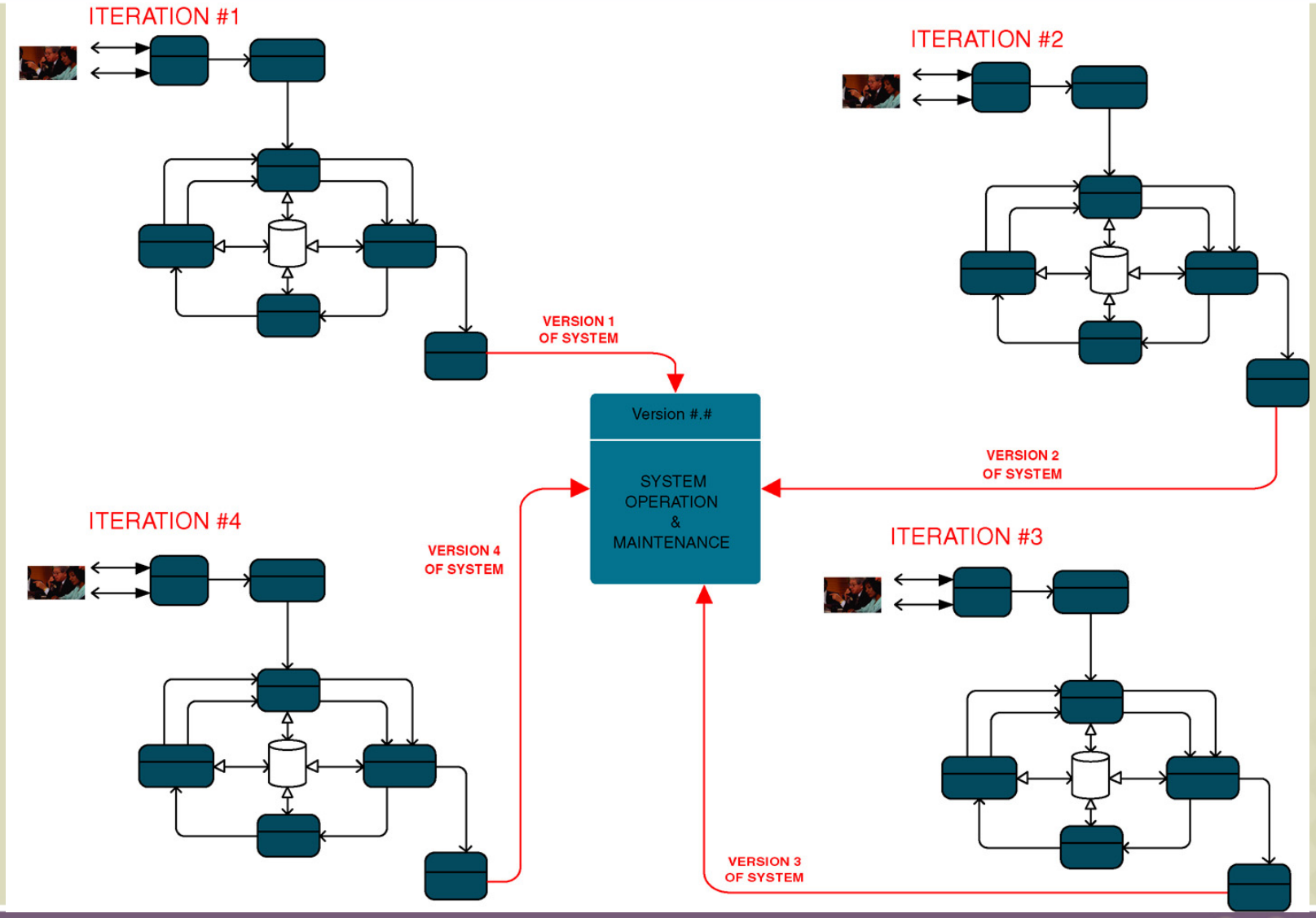
Advantages

- Systems usually implemented more quickly
- Avoids staffing required to develop in-house solutions
- Generally less expensive
- Vendor assumes responsibility for improvements and corrections
- Many business functions more similar than dissimilar for all businesses in a given industry

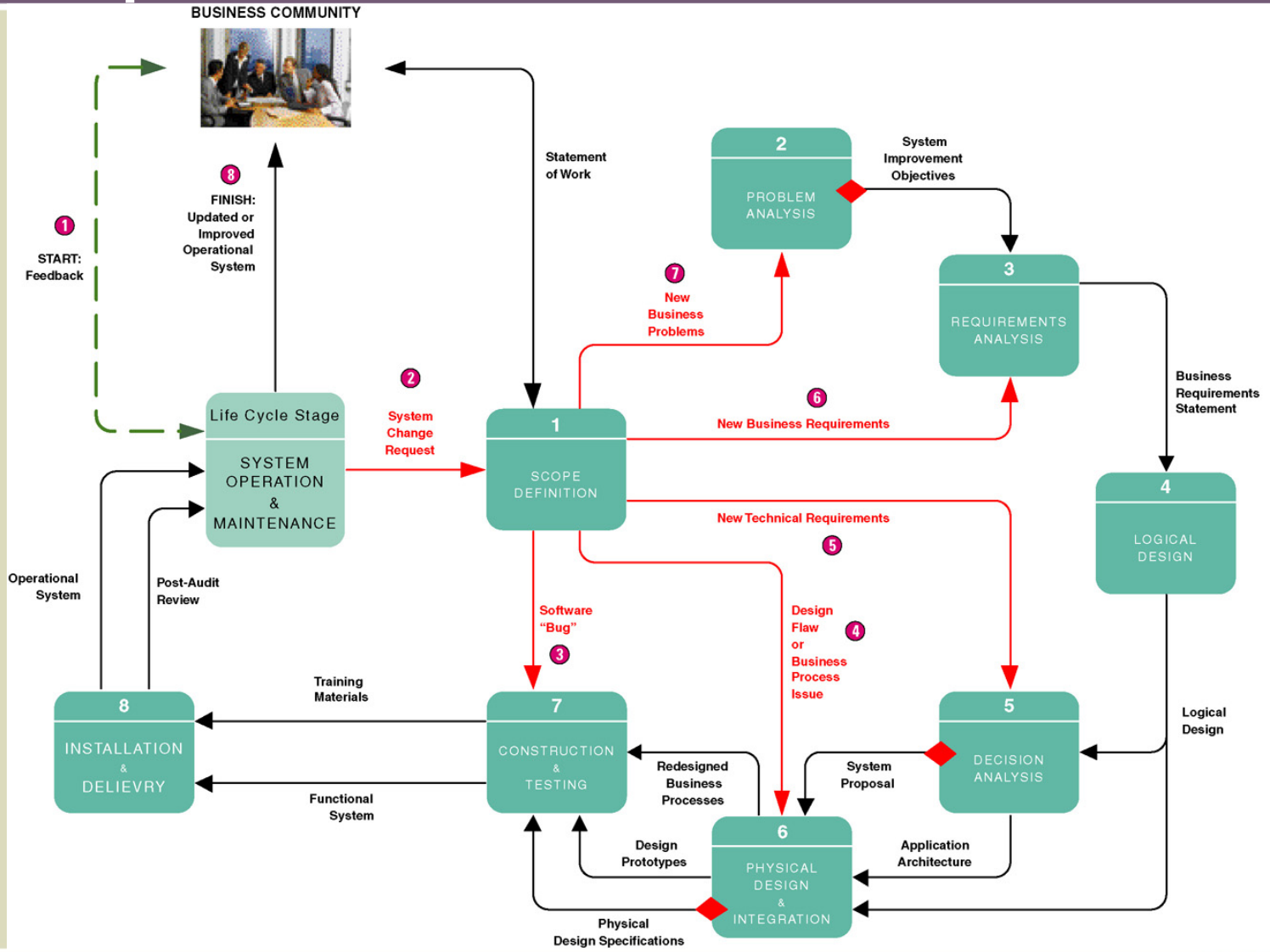
Disadvantages

- Dependent on long-term viability of vendor
- Rarely reflects ideal solution
- Often resistance to changes business processes to adapt to software

Hybrid Strategies



A System Maintenance Perspective



Automated Tools and Technology

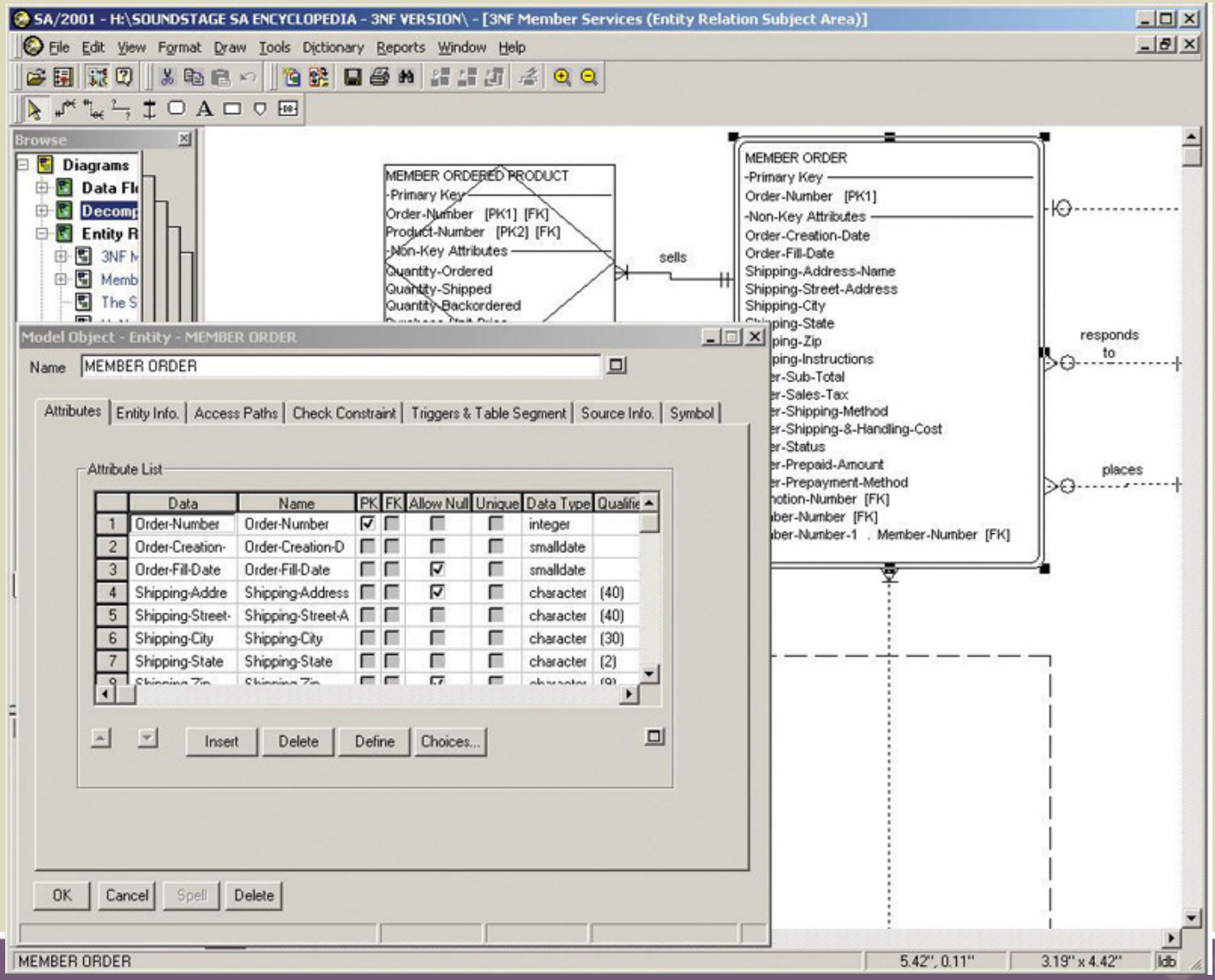
- Computer-aided systems engineering (CASE)
- Application development environments (ADEs)
- Process and project managers

Computer-Assisted Software Engineering (CASE)

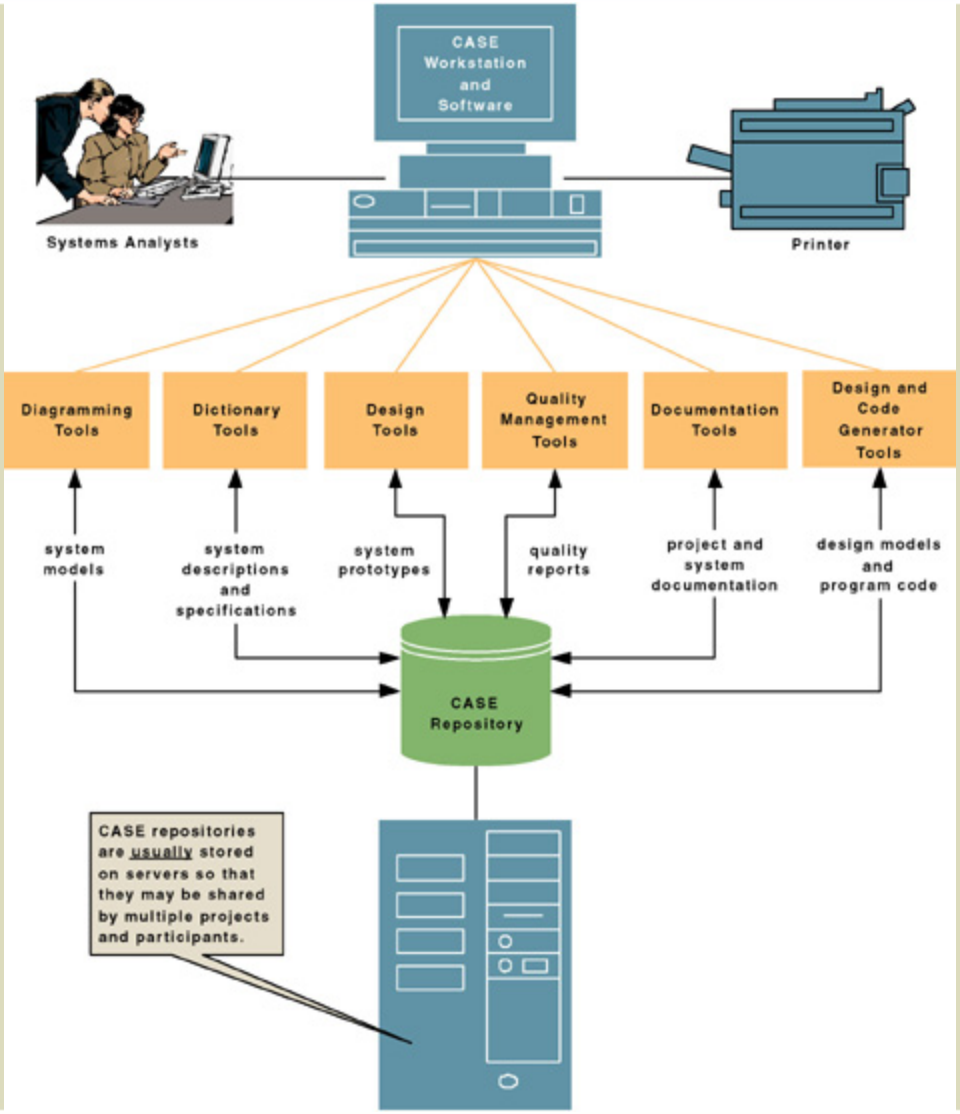
Computer-aided systems engineering (CASE) – automated software tools that support the drawing and analysis of system models and associated specifications. Some CASE tools also provide prototyping and code generation capabilities.

- **CASE repository** – system developers' database where developers can store system models, detailed descriptions and specifications, and other products of system development. Synonyms: *dictionary* and *encyclopedia*.
- **Forward engineering** – CASE tool capability that can generate initial software or database code directly from system.
- **Reverse engineering** – CASE tool capability that can generate initial system models from software or database code.

Using a CASE Tool for System Development



CASE Tool Architecture



Application Development Environments

Application development environments (ADEs) – an integrated software development tool that provides all the facilities necessary to develop new application software with maximum speed and quality. A common synonym is *integrated development environment* (IDE)

- ADE facilities may include:
  - Programming languages or interpreters
  - Interface construction tools
  - Middleware
  - Testing tools
  - Version control tools
  - Help authoring tools
  - Repository links

Process and Project Managers

- **Process manager application** – an automated tool that helps document and manage a methodology and routes, its deliverables, and quality management standards. An emerging synonym is *methodware*.
- **Project manager application** – an automated tool to help plan system development activities (preferably using the approved methodology), estimate and assign resources (including people and costs), schedule activities and resources, monitor progress against schedule and budget, control and modify schedule and resources, and report project progress.

Advanced IS Analys and Design

Nama : Rahmi

NIM : 192420046

CMM dapat diartikan sebagai penyederhanaan yang representatif yang digunakan untuk mengukur tingkat kematangan sebuah software development house dalam menyajikan/membuat/mengembangkan perangkat lunak sebagaimana telah dijanjikan secara tertulis dalam perjanjian kerjasama. Sehingga bisa dikatakan bahwa CMM adalah mengukur.

CMM awalnya ditujukan sebagai suatu alat untuk secara objektif menilai kemampuan kontraktor pemerintah untuk menangani proyek perangkat lunak yang diberikan. Walaupun berasal dari bidang pengembangan perangkat lunak, model ini dapat juga diterapkan sebagai suatu model umum yang membantu pemahaman kematangan kapabilitas proses organisasi di berbagai bidang. Misalnya rekayasa perangkat lunak, rekayasa sistem, manajemen proyek, manajemen risiko, teknologi informasi, serta manajemen sumber daya manusia.

1. ***Initial Level***

Level ini biasa disebut *anarchy* atau *chaos*. Pada pengembangan sistem ini masing – masing *developer* menggunakan peralatan dan metode sendiri. Berhasil atau tidaknya tergantung dari *project* teamnya. *Project* ini seringkali menemukan saat – saat krisis, kadang kelebihan budget dan di belakang rencana. Dokumen sering tersebar dan tidak konsisten dari satu *project* ke *project* lainnya. Level initial bercirikan sebagai berikut :

- Tidak adanya manajemen proyek
- Tidak adanya quality assurance
- Tidak adanya mekanisme manajemen perubahan (change management)
- Tidak ada dokumentasi
- Adanya seorang guru/dewa yang tahu segalanya tentang perangkat lunak yang dikembangkan.
- Sangat bergantung pada kemampuan individual

2. Repeatable level

Proses *project management* dan prakteknya telah membuat aturan tentang biaya projectnya, *schedule*, dan funsionalitasnya. Fokusnya adalah pada *project management* bukan pada pengembangan sistem. Proses pengembangan sistem selalu diikuti, tetapi akan berubah dari *project* ke *project*. Sebuah konsep upaya dibuat untuk mengulang kesuksesan *project* dengan lebih cepat. *Level Repeatable* bercirikan sebagai berikut :

- Kualitas perangkat lunak mulai bergantung pada proses bukan pada orang
- Ada manajemen proyek sederhana
- Ada quality assurance sederhana
- Ada dokumentasi sederhana
- Ada software configuration managemen sederhana
- Tidak adanya knowledge managemen
- Tidak ada komitment untuk selalu mengikuti SDLC dalam kondisi apapun
- Tidak ada statiskal control untuk estimasi proyek
- Rentan terhadap perubahan struktur organisasi.

3. Defined level

Standard proses pengembangan sistem telah dibeli dan dikembangkan dan ini telah digabungkan seluruhnya dengan unit sistem informasi dari organisasi. Dari hasil penggunaan proses standard, masing – masing project akan mendapatkan hasil yang konsisten dan dokumentasi dengan kualitas yang baik dan dapat dikirim. Proses akan bersifat stabil, terprediksi, dan dapat diulang. Level Defined bercirikan :

•SDLC sudah dibuat dan dibakukan
•Ada komitmen untuk mengikuti SDLC dalam keadaan apapun
• Kualitas proses dan produk masih bersifat kwalitatif bukan kualitatif (tidak terukur hanya kira-kira saja)

•Tidak menerapkan *Activity Based Costing*
• *Tidak ada mekanisme umpan balik yang baku*

4. Managed level

Tujuan yang terukur untuk kualitas dan produktivitas telah dibentuk. Perhitungan yang rinci dari standard proses pengembangan sistem dan kualitas produk secara rutin akan dikumpulkan dan disimpan dalam database. Terdapat suatu usaha untuk mengembangkan individual project management yang didasari dari data yang telah terkumpul. Level Managed bercirikan :

• Sudah adanya Activity Based Costing dan dan digunakan untuk estimasi untuk proyek berikutnya
• Proses penilaian kualitas perangkat lunak dan proyek bersifat kuantitatif.
• Terjadi pemborosan biaya untuk pengumpulan data karena proses pengumpulan data masih dilakukan secara manual
• Cenderung bias. Ingat efect thorne, manusia ketika diperhatikan maka prilakunya cenderung berubah.
• Tidak adanya mekanisme pencegahan defect
• Ada mekanisme umpan balik

5. Optimized level

Proses pengembangan sistem yang distandardisasi akan terus dimonitor dan dikembangkan yang didasari dari perhitungan dan analisis data yang dibentuk pada level 4. Ini dapat termasuk perubahan teknologi dan praktek – praktek terbaik yang digunakan untuk menunjukkan aktivitas yang diperlukan pada standard proses pengembangan sistem . Level Optimized bercirikan :

•Pengumpulan data secara automatis
•Adanya mekanisme pencegahan defect
•Adanya mekanisme umpan balik yang sangat baik
• Adanya peningkatan kualitas dari SDM dan peningkatan kualitas proses.

Nama : Rani Okta Felani

Nim : 192420048

Mata Kuliah : Advanced is Analisis and Design

Dosen Pengampu : M. Izman Herdiansyah, M.M, Ph.D

EL-3

Beri penjelasan tentang CMM level Apa pentingnya memahami konsep CMM level dalam Menganalisis dan Merancang Sistem !

Jawab :

Pengertian CCM Dalam jurnal Lusiana Dewi, (2016) CMM level (Capabillity Maturity Model) merupakan model acuan untuk menaksir tingkat kematangan proses pengembangan perangkat lunak dan juga merupakan model normatif bagi organisasi pengembang. Perangkat lunak untuk secara evolusioner menyempurnakan proses pengembangan perangkat lunak dari kondisi kacau dan ad-hoc menuju ke proses matang dan terdisiplin (hebsleb et al, 1997) untuk beragam jenis terapan CMM berisi kumpulan elemen penting dari proses yang telah terbukti efektif . CMM telah menjadi semacam standar de facto untuk menaksir dan mengembangkan proses pengembangan perangkat lunak .

Pentingnya memahami konsep CMM dalam menganalisis dan merancang sistem adalah kerangka kerja untuk mengukur tingkat kematangan pengembangan sistem informasi dalam menganalisis dan merancang sistem CMM dibagi menjadi 5 tingkatan kematangan yang mengatur kualitas dari proses pengembangan sistem dan perangkat lunak (setiap level menjadi pra-persyaratan untuk level selanjutnya)

Capability Maturity Model terdapat 5 Level / Skala Kematangan yaitu :

1. Initial

2. Repeatable

3. Defined

4. Managed

5. Optimizaed

Penjelasan CMM (Capability Maturity Model)

CMM merupakan mekanisme kualifikasi sebuah Software Development House yang dapat memberikan gambaran tentang kemampuan perusahaan tersebut dalam melakukan development software. Capability Maturity Model adalah sebuah model yang dikembangkan oleh Software Engineering Institute atas permintaan Departement of Defense(DOD) Amerika Serikat dengan tujuan membuat ujian saringan masuk bagi kontraktor yang mendaftarkan diri untuk menjadi konsultan

Pengertian secara harfiah:

Capability, diartikan menjadi kapabilitas yang berarti kemampuan yang bersifat laten. Capability lebih mengarah kepada integritas daripada kapabilitas yang berarti itu sendiri.

Maturity, berarti matang atau dewasa. Matang merupakan hasil proses. Dewasa merupakan hasil pertumbuhan

Model, didefinisikan sebagai suatu penyederhanaan yang representatif terhadap keadaan di dunia nyata

Secara keseluruhan pengertian CMM, yaitu:

CMM dapat didefinisikan sebagai sebuah penyederhanaan yang representatif yang digunakan untuk mengukur tingkat kematangan sebuah software development house dalam menyajikan/membuat/mengembangkan perangkat lunak sebagaimana telah dijanjikan secara tertulis dalam perjanjian kerjasama. Sehingga bisa dikatakan bahwa CMM adalah mengukur.

Nilai-nilai yang dilihat dalam pengukuran tersebut:

a) Apa yang diukur (Parameter)
b) Bagaimana cara mengukurnya (Metode)
c) Bagaimana standar penilaiannya (Skala Penilaian)
d) Bagaimana Interpretasinya (Bagi Manusia)

Capability Maturity Model terdapat 5 level/skala kematangan yaitu :

1. Initial

   Ciri-ciri dari fungsi initial adalah tidak ada manajemen proyek, tidak adanya quality assurance, tidak adanya mekanisme manajemen perubahan (change management), tidak ada dokumentasi, adanya seorang ahli yang tau segalanya tentang perangkat lunak yang dikembangkan, dan sangat bergantung pada kemampuan individual.

2. Repeatable

   Ciri-ciri dari fungsi repeatable adalah kualitas perangkat lunak mulai bergantung pada proses bukan pada oprang, ada manajemen proyek sederhana, ada quality assurance sederhana, ada dokumen sederhana, ada software configuration management sederhana, tidak adanya knowledge management, tidak adanya komitmen untuk selalu mengikuti SDLC dalam kondisi apapun, tidak adanya stastikal control untuk estimasi proyek dan rentan perubahan struktur organisasi.

3. Defined

   Ciri-ciri dari level Defined adalah SDLC sudah ditentukan, ada komitmrn untuk mengikuti SDLC dalam keadaan apapun, kualitas proses dan produk masih bersifat kualitatif

atau hanya perkiraan saja, idak menerapkan Activity Based Costing, dan tidak adanya mekanisme umpan balik yang baku.

4. Managed

   Ciri-cirinya adalah sebagai berikut, sudah ada Activity Based Costing dan digunakan untuk estimasi proyek berikutnya, proses penilaian kualitas perangkat lunak dan proyek bersifat kuantitatif, terjadi pemborosan biaya untuk pengumpulan data karena proses pengumpulan data masih dilakukan secara manual, cenderung belum jelas disebabkan karena masnusia ketika diperhatikan perilakuknya cenderung berubah, tidak ada mekanisme pencegahan defect dan adanya mekanisme umpan balik

5. Optimized

   Pengumpulan data secara automatis, adanya mekanisme pencegahan defect, adanya mekanisme umpan balik yang sangat baik, dan adanya peningkatan kualitas dari SDM dan juga peningkatan kualitas proses.

Hubungan CMM dengan Programming

Programming atau pembuatan program dapat dibuat kesamaannya dengan CMM. Programming in small (coding red) ekivalen dengan CMM level 1. Programming in large (proyek managemen, documentasi, dll) ekivalen dengan CMM level 2. Keduanya dapat dikelompokan menjadi programming as art proccess karena tidak memiliki unsur engineering. Unsur engineering yang perlu ditambahkan adalah standarisasi (pembakuan) dan pengukuran. Jika sudah dilakukan standarisasi maka ekivalen dengan CMM level 3. Jika sudah ada pengukuran maka ekivalen dengan CMM level 4. Jika sudah sampai di level 4 maka programming dapat dianggap sebagai engineering process. Keseluruhan level dari 1-4 dapat dipandang sebagai programming as discreet process dimana tidak ada pengembangan berkelanjutan (life time quality improvment) . Baru pada level 5 programming dapat dianggap sebagai continues process dimana peningkatan kualitas sumber daya manusia dan proses dilakukan secara terus menerus.

Kegunaan CMM meliputi:

• Menilai tingkat kematangan sebuah organisasi pengembang perangkat lunak

• Memfilter kontraktor yang akan menjadi pengembang perangkat lunak

• Memberikan arah untuk peningkatan organisasi bagi top managemen di dalam sebuah organisasi pengembang perangkat lunak

• Sebagai alat bantu untuk menilai keunggulan kompetitif yang dimiliki sebuah perusahaan dibandingkan perusahaan pesaingnya.

Program Pascasarjana Magister Teknik informatika

Universitas Bina Darma Palembang

Nama : Suwani

Nim : 192420049

Mata Kuliah :Advanced IS Analysis And Design

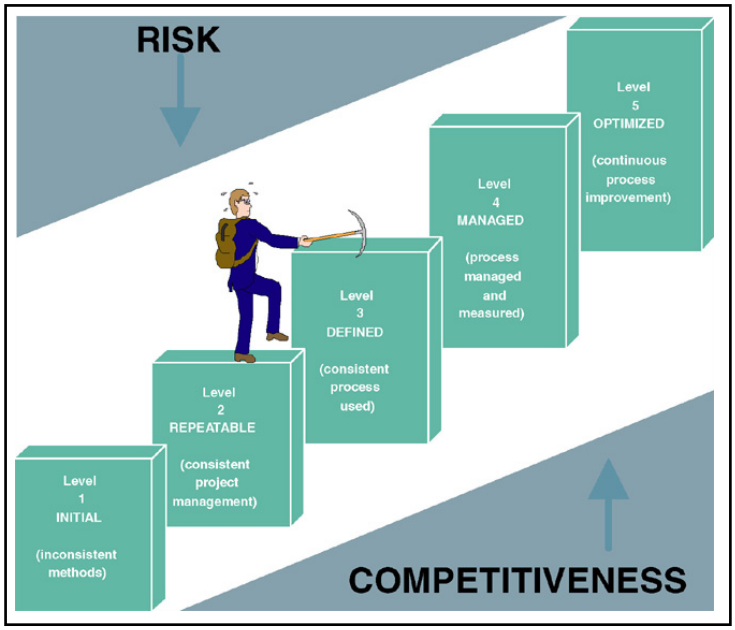
Tugas Pertemuan Elearning ke 3

Soal : Rumuskan dengan baik hasil jawaban saudara di topik diskusi forum E-L 3

1. Beri penjelasan tentang CMM level.
2. Apa pentingnya memahami konsep CMM level dalam menganalisis dan merancang sistem?

Jawaban

1. Penjelasan Tentang CMM Level.



a. Initial

Ciri-ciri dari fungsi initial adalah tidak ada manajemen proyek, tidak adanya quality assurance, tidak adanya mekanisme manajemen perubahan (change management), tidak ada dokumentasi, adanya seorang ahli yang tau segalanya tentang perangkat lunak yang dikembangkan, dan sangat bergantung pada kemampuan individual

b. Repeatable

Ciri-ciri dari fungsi repeatable adalah kualitas perangkat lunak mulai bergantung pada proses bukan pada oprang, ada manajemen proyek sederhana, ada quality assurance

sederhana, ada dokumen sederhana, ada software configuration management sederhana, tidak adanya knowledge management, tidak adanya komitmen untuk selalu mengikuti SDLC dalam kondisi apapun, tidak adanya stastikal control untuk estimasi proyek dan rentan perubahan struktur organisasi.

c. Defined

   Ciri-ciri dari level Defined adalah SDLC sudah ditentukan, ada komitmrn untuk mengikuti SDLC dalam keadaan apapun, kualitas proses dan produk masih bersifat kualitatif atau hanya perkiraan saja, idak menerapkan Activity Based Costing, dan tidak adanya mekanisme umpan balik yang baku.

d. Managed

   Ciri-cirinya adalah sebagai berikut, sudah ada Activity Based Costing dan digunakan untuk estimasi proyek berikutnya, proses penilaian kualitas perangkat lunak dan proyek bersifat kuantitatif, terjadi pemborosan biaya untuk pengumpulan data karena proses pengumpulan data masih dilakukan secara manual, cenderung belum jelas disebabkan karena masnusia ketika diperhatikan perilakuknya cenderung berubah, tidak ada mekanisme pencegahan defect dan adanya mekanisme umpan balik

e. Optimized

   Pengumpulan data secara automatis, adanya mekanisme pencegahan defect, adanya mekanisme umpan balik yang sangat baik, dan adanya peningkatan kualitas dari SDM dan juga peningkatan kualitas proses.

2. Pentingnya memahami konsep CMM level dalam menganalisis dan merancang sistem. Sangat penting untuk membantu pemahaman kematangan kapabilitas proses organisasi di berbagai bidang. Misalnya rekayasa perangkat lunak, rekayasa sistem, manajemen proyek, manajemen risiko, teknologi informasi, serta manajemen sumber daya manusia. Dan CMM juga dapat didefinisikan sebagai sebuah penyederhanaan yang representatif yang digunakan untuk mengukur tingkat kematangan sebuah software development house dalam menyajikan/membuat/mengembangkan perangkat lunak sebagaimana telah dijanjikan secara tertulis dalam perjanjian kerjasama. Sehingga bisa dikatakan bahwa CMM adalah mengukur.

Nama : Theo Vhaldino
Nim : 192420058
Angkatan/Reguler : 22 / A R1
Mata Kuliah : Advanced is Analysis and Design (MTIK113)

Tugas E-L 3 (Rumusan berdasarkan topik diskusi forum E-L 3)

Capability Maturity Model (CMM) adalah model kematangan kemampuan (kapabilitas) untuk membantu pendefinisian dan pemahaman proses-proses pada suatu organisasi. CMM awalnya ditujukan sebagai suatu alat untuk secara objektif menilai kemampuan kontraktor pemerintah untuk menangani proyek perangkat lunak yang diberikan. Walaupun berasal dari bidang pengembangan perangkat lunak, model ini dapat juga diterapkan sebagai suatu model umum yang membantu pemahaman kematangan kapabilitas proses organisasi di berbagai bidang. Misalnya rekayasa perangkat lunak, rekayasa sistem, manajemen proyek, manajemen risiko, teknologi informasi, serta manajemen sumber daya manusia.

Capability Maturity Model terdapat 5 level/skala kematangan yaitu :

1. initial : proses tidak dapat diprediksi, kurang terkontrol dan reaktif
2. repeatable : proses yang diperuntukkan untuk proyek dan seringkali reaktif
3. defined : proses yang dikarakterisasi untuk organisasi dan proaktif (proyek menyesuaikan proses mereka dari standar organisasi)
4. managed : proses diukur dan dikendalikan
5. optimizing : fokus pada peningkatan proses yang berkelanjutan

Mengapa pentingnya memahami konsep CMM level dalam menganalisis dan merancang sistem karena CMM level banyak memiliki kegunaan diantara lain sebagai berikut :

1. Menilai tingkat kematangan sebuah organisasi pengembang perangkat lunak
2. Memfilter kontraktor yang akan menjadi pengembang perangkat lunak
3. Memberikan arah untuk peningkatan organisasi bagi top managemen di dalam sebuah organisasi pengembang perangkat lunak
4. Sebagai alat bantu untuk menilai keunggulan kompetitif yang dimiliki sebuah perusahaan dibandingkan perusahaan pesaingnya.

Nama : Arpa Pauziah

Mata Kuliah : Advanced Is Analysis And Design

Pertanyaan:

Beri penjelasan tentang CMM level.

Apa pentingnya memahami konsep CMM level dalam menganalisis dan merancang sistem?

Tanggapan:

Capability Maturity Model (CMM) - kerangka kerja standar untuk menilai tingkat kematangan pengembangan sistem informasi dan proses dan produk manajemen organisasi. Terdiri dari lima tingkat kematangan:

Level 1 — Awal : Proyek pengembangan sistem tidak mengikuti proses yang ditentukan.

Level 2 — Berulang : proses dan praktik manajemen proyek ditetapkan untuk melacak biaya, jadwal, dan fungsi proyek.

Level 3 — Ditetapkan : Proses pengembangan sistem standar (metodologi) dibeli atau dikembangkan. Semua proyek menggunakan versi proses ini.

Level 4 — Dikelola : Sasaran yang terukur untuk kualitas dan produktivitas ditetapkan.

Level 5 — Optimalisasi : Proses pengembangan sistem terstandardisasi terus dimonitor dan ditingkatkan berdasarkan pengukuran dan analisis data yang ditetapkan di Level 4.

Jadi dengan menggunakan konsep CMM, Model tingkat kematangan dapat digunakan sebagai tolok ukur untuk perbandingan dalam menganalisa dan merancang sistem.

misalnya, untuk penilaian perbandingan dari berbagai organisasi di mana ada sesuatu yang sama yang dapat digunakan sebagai dasar untuk perbandingan. Dalam kasus *Capability Maturity Models (CMM),* misalnya, dasar untuk perbandingan adalah proses pengembangan perangkat lunak organisasi.

Nama : Elpina Sari

Nim : 192420050

Tugas 3 (EL)

Beri penjelasan tentang CMM level.

Apa pentingnya memahami konsep CMM level dalam menganalisis dan merancang sistem?

Jawab:

Pentingnya memahami konsep CMM (Capability Maturity Model) level dalam menganalisis dan merancang sistem merupakan mekanisme kualifikasi sebuah Software Development House yang dapat memberikan gambaran tentang kemampuan perusahaan tersebut dalam melakukan development software untuk mengukur tingkat kematangan sebuah software development house dalam menyajikan, membuat atau mengembangkan perangkat lunak sebagaimana telah dijanjikan secara tertulis dalam perjanjian kerjasama, Sehingga bisa dikatakan bahwa CMM adalah mengukur nilai – nilai parameter, metode, skala penilaian dan interprestasinya bagi manusia.

(DMM) Capability Maturity Model terdapat 5 level/skala kematangan yaitu :

1. Initial
   Ciri-ciri dari fungsi initial adalah tidak ada manajemen proyek, tidak adanya quality assurance, tidak adanya mekanisme manajemen perubahan (change management), tidak ada dokumentasi, adanya seorang ahli yang tau segalanya tentang perangkat lunak yang dikembangkan, dan sangat bergantung pada kemampuan individual.
2. Repeatable
   Ciri-ciri dari fungsi repeatable adalah kualitas perangkat lunak mulai bergantung pada proses bukan pada oprang, ada manajemen proyek sederhana, ada quality assurance sederhana, ada dokumen sederhana, ada software configuration management sederhana, tidak adanya knowledge management, tidak adanya komitmen untuk selalu mengikuti SDLC dalam kondisi apapun, tidak adanya stastikal control untuk estimasi proyek dan rentan perubahan struktur organisasi.
3. Defined
   Ciri-ciri dari level Defined adalah SDLC sudah ditentukan, ada komitmrn untuk mengikuti SDLC dalam keadaan apapun, kualitas proses dan produk masih bersifat kualitatif atau hanya perkiraan saja, idak menerapkan Activity Based Costing, dan tidak adanya mekanisme umpan balik yang baku.
4. Managed
   Ciri-cirinya adalah sebagai berikut, sudah ada Activity Based Costing dan digunakan untuk estimasi proyek berikutnya, proses penilaian kualitas perangkat lunak dan proyek bersifat kuantitatif, terjadi pemborosan biaya untuk pengumpulan data karena proses pengumpulan data masih dilakukan secara manual, cenderung belum jelas disebabkan karena masnusia ketika diperhatikan perilakuknya cenderung berubah, tidak ada mekanisme pencegahan defect dan adanya mekanisme umpan balik

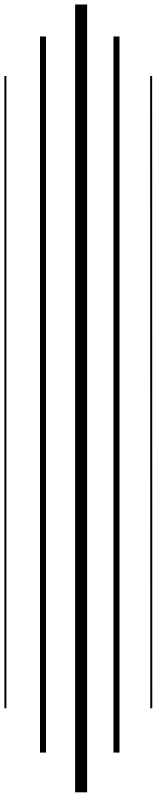
5. Optimized

   Pengumpulan data secara automatis, adanya mekanisme pencegahan defect, adanya mekanisme umpan balik yang sangat baik, dan adanya peningkatan kualitas dari SDM dan juga peningkatan kualitas proses.

Dimana 5 level yang ada dalam konsep DMM level memiliki hubangan kerja sama dengan Programming in small (coding red) ekivalen dengan CMM level 1. Programming in large (proyek managemen, documentasi, dll) ekivalen dengan CMM level 2. Keduanya dapat dikelompokan menjadi programming as art proccess karena tidak memiliki unsur engineering. Unsur engineering yang perlu ditambahkan adalah standarisasi (pembakuan) dan pengukuran. Jika sudah dilakukan standarisasi maka ekivalen dengan CMM level 3. Jika sudah ada pengukuran maka ekivalen dengan CMM level 4. Jika sudah sampai di level 4 maka programming dapat dianggap sebagai engineering process. Keseluruhan level dari 1-4 dapat dipandang sebagai programming as discreet process dimana tidak ada pengembangan berkelanjutan (life time quality improvment) . Baru pada level 5 programming dapat dianggap sebagai continues process dimana peningkatan kualitas sumber daya manusia dan proses dilakukan secara terus menerus.

TUGAS ADVANCE IS ANALYSIS AND DESIGN

Information System Developments



DISUSUN OLEH:

FADEL MUHAMMAD MADJID

192420052

PROGRAM PASCA SARJANA MAGISTER TEKNIK INFORMATIKA

UNIVERSITAS BINA DARMA

Capability Maturity Model (CMM)

Definisi CMM

CMM adalah sebuah penyederhanaan representatif yang digunakan untuk mengukur tingkat kematangan sebuah *software development house* dalam menyajikan, membuat, dan mengembangkan perangkat lunak sebagaimana telah dijanjikan secara tertulis dalam perjanjian kerja sama.

CMM awalnya ditujukan sebagai suatu alat untuk secara objektif menilai kemampuan kontraktor pemerintah untuk menangani proyek perangkat lunak yang diberikan. Walaupun berasal dari bidang pengembangan perangkat lunak, model ini dapat juga diterapkan sebagai suatu model umum yang membantu pemahaman kematangan kapabilitas proses organisasi di berbagai bidang. Misalnya rekayasa perangkat lunak, rekayasa sistem, manajemen proyek, manajemen risiko, teknologi informasi, serta manajemen sumber daya manusia.

Secara umum, ***maturity model*** biasanya memiliki ciri sebagai berikut:

1. Proses pengembangan dari suatu organisasi disederhanakan dan dideskripsikan dalam wujud tingkatan kematangan dalam jumlah tertentu.
2. Tingkatan kematangan tersebut dicirikan dengan beberapa persyaratan tertentu yang harus diraih.
3. Tingkatan-tingkatan yang ada disusun secara sekuensial, mulai dari tingkat inisial sampai pada tingkat akhir.
4. Selama pengembangan, entitas bergerak maju dari satu tingkatan ke tingkatan berikutnya tanpa boleh melewati salah satunya secara bertahap dan berurutan.

Tujuan CMM

Tujuan penggunaan CMM adalah membuat ujian saringan masuk bagi kontraktor yang mendaftarkan diri untuk menjadi konsultan.

Nilai-nilai yang dilihat dalam pengukuran CMM

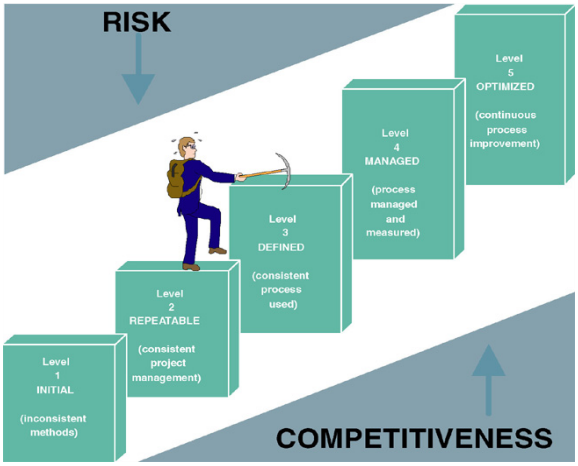
1. Apa yang diukur (parameter)
2. Bagaimana cara mengukurnya (metode)
3. Bagaimana standar penilaiannya (skala penilaian)
4. Bagaimana interpretasinya (bagi manusia)

Kegunaan CMM

1. Untuk menilai tingkat kematangan sebuah organisasi pengembang sistem.
2. Untuk menyaring kontraktor yang akan menjadi pengembang sistem.
3. Untuk memberikan arah akan peningkatan organisasi bagi top management di dalam sebuah organisasi pengembang sistem.
4. Sebagai alat bantu untuk menilai keunggulan kompetitif yang dimiliki sebuah perusahaan dibandingkan perusahaan pesaingnya.

Tahapan CMM

Terdapat 5 tahapan dalam CMM yaitu, sebagai berikut:



1. ***Initial Level***

   Level ini hiasa disebut *anarchy* atau *chaos*. Pada pengembangan sistem ini masing – masing *developer* menggunakan peralatan dan metode sendiri. Berhasil atau tidaknya tergantung dari *project* teamnya. *Project* ini seringkali menemukan saat – saat krisis, kadang kelebihan budget dan di belakang rencana. Dokumen sering tersebar dan tidak konsisten dari satu *project* ke *project* lainnya. Level initial bercirikan sebagai berikut :

   - Tidak adanya manajemen proyek.
   - Tidak adanya quality assurance.
   - Tidak adanya mekanisme manajemen perubahan (change management).
   - Tidak ada dokumentasi.
   - Adanya seorang guru/dewa yang tahu segalanya tentang perangkat lunak yang dikembangkan.
   - Sangat bergantung pada kemampuan individual.

2. ***Repeatable level***

   Proses *project management* dan prakteknya telah membuat aturan tentang biaya projectnya, *schedule*, dan funsionalitasnya. Fokusnya adalah pada *project management* bukan pada pengembangan sistem. Proses pengembangan sistem selalu diikuti, tetapi akan berubah dari *project* ke *project*. Sebuah konsep upaya dibuat untuk mengulang kesuksesan *project* dengan lebih cepat. *Level Repeatable* bercirikan sebagai berikut :

   - Kualitas perangkat lunak mulai bergantung pada proses bukan pada orang
   - Ada manajemen proyek sederhana
   - Ada quality assurance sederhana
   - Ada dokumentasi sederhana
   - Ada software configuration managemen sederhana
   - Tidak adanya knowledge managemen
   - Tidak ada komitment untuk selalu mengikuti SDLC dalam kondisi apapun
   - Tidak ada statiskal control untuk estimasi proyek
   - Rentan terhadap perubahan struktur organisasi.

3. ***Defined level***

   Standard proses pengembangan sistem telah dibeli dan dikembangkan dan ini telah digabungkan seluruhnya dengan unit sistem informasi dari organisasi. Dari hasil penggunaan proses standard, masing – masing project akan mendapatkan hasil yang konsisten dan dokumentasi dengan kualitas yang baik dan dapat dikirim. Proses akan bersifat stabil, terprediksi, dan dapat diulang. Level Defined bercirikan :

   - SDLC sudah dibuat dan dibakukan

- Ada komitmen untuk mengikuti SDLC dalam keadaan apapun
- Kualitas proses dan produk masih bersifat kwalitatif bukan kualitatif (tidak terukur hanya kira-kira saja)
- Tidak menerapkan *Activity Based Costing*
- Tidak ada mekanisme umpan balik yang baku

4. ***Managed level***

   Tujuan yang terukur untuk kualitas dan produktivitas telah dibentuk. Perhitungan yang rinci dari standard proses pengembangan sistem dan kualitas produk secara rutin akan dikumpulkan dan disimpan dalam database. Terdapat suatu usaha untuk mengembangkan individual project management yang didasari dari data yang telah terkumpul. Level Managed bercirikan :
   - Sudah adanya Activity Based Costing dan dan digunakan untuk estimasi untuk proyek berikutnya
   - Proses penilaian kualitas perangkat lunak dan proyek bersifat kuantitatif.
   - Terjadi pemborosan biaya untuk pengumpulan data karena proses pengumpulan data masih dilakukan secara manual
   - Cenderung bias. Ingat efect thorne, manusia ketika diperhatikan maka prilakunya cenderung berubah.
   - Tidak adanya mekanisme pencegahan defect
   - Ada mekanisme umpan balik

5. ***Optimized level***

   Proses pengembangan sistem yang distandardisasi akan terus dimonitor dan dikembangkan yang didasari dari perhitungan dan analisis data yang dibentuk pada level 4. Ini dapat termasuk perubahan teknologi dan praktek – praktek terbaik yang digunakan untuk menunjukkan aktivitas yang diperlukan pada standard proses pengembangan sistem . Level Optimized bercirikan :
   - Pengumpulan data secara automatis
   - Adanya mekanisme pencegahan defect
   - Adanya mekanisme umpan balik yang sangat baik
   - Adanya peningkatan kualitas dari SDM dan peningkatan kualitas proses.

Daftar Pustaka

Whitten, Jeffrey L. System analysis and design methods – 7th ed. 2007

https://syafrudinmtop.blogspot.com/2015/03/capability-maturity-model-cmm.html

http://agraandhyka.blogspot.com/2014/03/capability-maturity-model-capability.html

http://harumindripermata.blogspot.com/2014/03/capability-maturity-model-cmm.html

https://itgid.org/mengukur-kinerja-ti-menggunakan-maturity-level-dari-framework-cobit-5/

Nama : Isti Maátun Nasichah
NPM : 192420051
Program : Magister Teknik Informatika

TUGAS

Beri penjelasan tentang CMM level. Apa pentingnya memahami konsep CMM level dalam menganalisis dan merancang sistem?

Jawab :

Capability Maturity Model disingkat CMM merupakan mekanisme kualifikasi sebuah software development house yang dapat memberikan gambaran tentang kemampuan perusahaan tersebut dalam melakukan development software. Dalam arti lain, Capability Maturity Model disingkat CMM adalah suatu model kematangan kemampuan (kapabilitas) proses yang dapat membantu pendefinisian dan pemahaman proses-proses suatu organisasi.

Nilai-nilai yang dilihat dalam pengukuran CMM :

1. Apa yang diukur (Parameter)
2. Bagaimana cara mengukurnya (Metode)
3. Bagaimana standar penilaiannya (Skala Penilaian)
4. Bagaimana Interpretasinya (Bagi Manusia)

Karakteristik CMM :

1.Initial

Level ini biasa disebut *anarchy* atau *chaos*. Pada pengembangan sistem ini masing – masing *developer* menggunakan peralatan dan metode sendiri. Berhasil atau tidaknya tergantung dari *project* teamnya

2.Repeatable

Proses *project management* dan prakteknya telah membuat aturan tentang biaya projectnya, *schedule*, dan funsionalitasnya. Fokusnya adalah pada *project management* bukan pada pengembangan sistem. Proses pengembangan sistem selalu diikuti, tetapi akan berubah dari *project* ke *project*. Sebuah konsep upaya dibuat untuk

mengulang kesuksesan *project* dengan lebih cepat.

3.Defined

Standard proses pengembangan sistem telah dibeli dan dikembangkan dan ini telah digabungkan seluruhnya dengan unit sistem informasi dari organisasi. Dari hasil penggunaan proses standard, masing – masing project akan mendapatkan hasil yang konsisten dan dokumentasi dengan kualitas yang baik dan dapat dikirim. Proses akan bersifat stabil, terprediksi, dan dapat diulang.

4.Managed

Tujuan yang terukur untuk kualitas dan produktivitas telah dibentuk. Perhitungan yang rinci dari standard proses pengembangan sistem dan kualitas produk secara rutin akan dikumpulkan dan disimpan dalam database. Terdapat suatu usaha untuk mengembangkan individual project management yang didasari dari data yang telah terkumpul.

5.Optimizing

Proses pengembangan sistem yang distandardisasi akan terus dimonitor dan dikembangkan yang didasari dari perhitungan dan analisis data yang dibentuk pada level 4. Ini dapat termasuk perubahan teknologi dan praktek – praktek terbaik yang digunakan untuk menunjukkan aktivitas yang diperlukan pada standard proses pengembangan sistem.

Kegunaan CMM meliputi:

1. Untuk menilai tingkat kematangan sebuah organisasi pengembang perangkat lunak.
2. Untuk menyaring kontraktor yang akan menjadi pengembang perangkat lunak.
3. Untuk memberikan arah akan peningkatan organisasi bagi top management di dalam sebuah organisasi pengembang perangkat lunak.
4. Sebagai alat bantu untuk menilai keunggulan kompetitif yang dimiliki sebuah perusahaan dibandingkan perusahaan pesaingnya.

Pentingnya memahami konsep CMM level dalam menganalisis dan merancang sistem adalah untuk memfilter perusahaan mana yang berhak mengikuti tender serta menentukan arah pengembangan software house (yang dikembangkan software housenya bukan software) yang paling efektif dalam meningkatkan kualitas proses (bukan kualitas product).

Implementasinya = kebijakan manajemen.

Contohnya :

1.Standarisasi

2.Pelatihan

3.Penggunaan source control dan bug management

4.Pembuatan dokument perencanaan dan design

5.Pengukuran

6.Penerapan sdlc