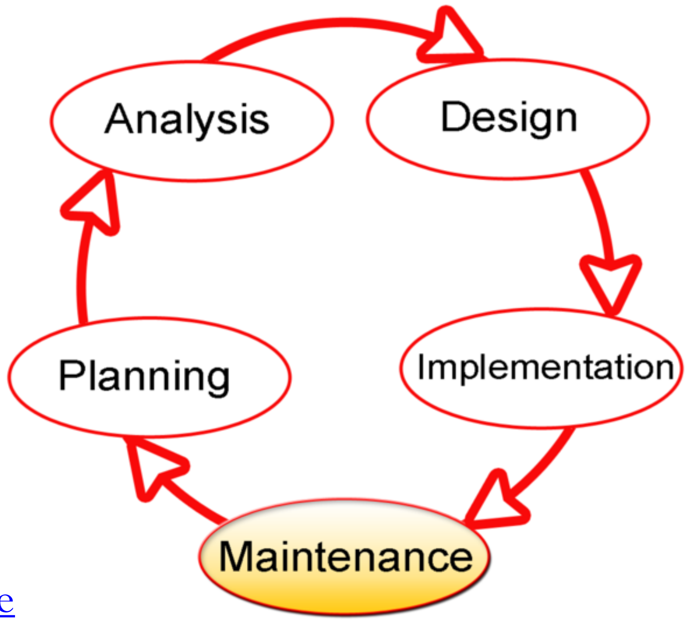
System Development Life Cycle (SDLC)

Dr. Animesh Chaturvedi
Assistant Professor: IIIT Dharwad
Young Researcher Alumni: Heidelberg Laureate Forum
Postdoc: King's College London & The Alan Turing Institute
PhD: IIT Indore MTech: IIITDM Jabalpur

General SDLC stages

- Six stages are
  - requirement analysis: define project goals into defined functions and operations of the intended application.
  - design, desired features and operations are described in detail, including screen layouts,
  - development and testing, the real cost of project
  - implementation,
  - documentation, and
  - evaluation.



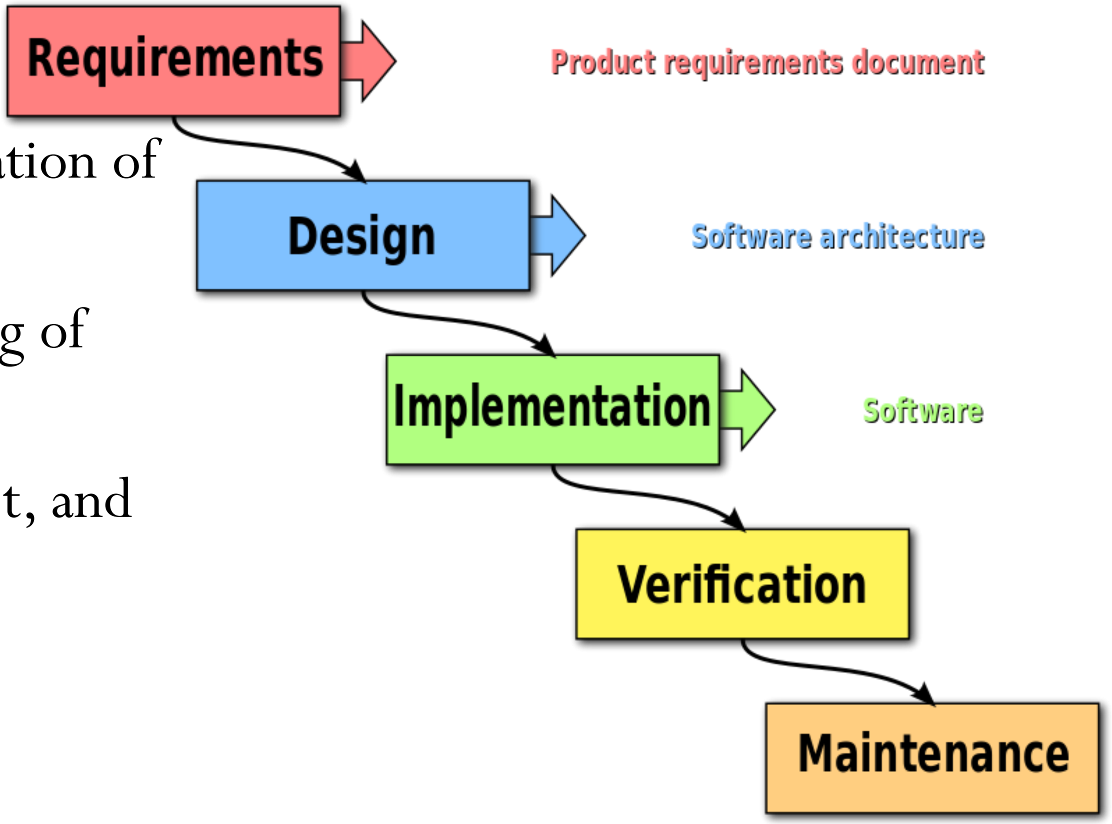
https://en.wikipedia.org/wiki/Systems_development_life_cycle

SDLC Models

- Methodologies are created,
  - Waterfall,
  - Spiral,
  - Agile development,
  - Rapid prototyping, and
  - Incremental.

Waterfall model

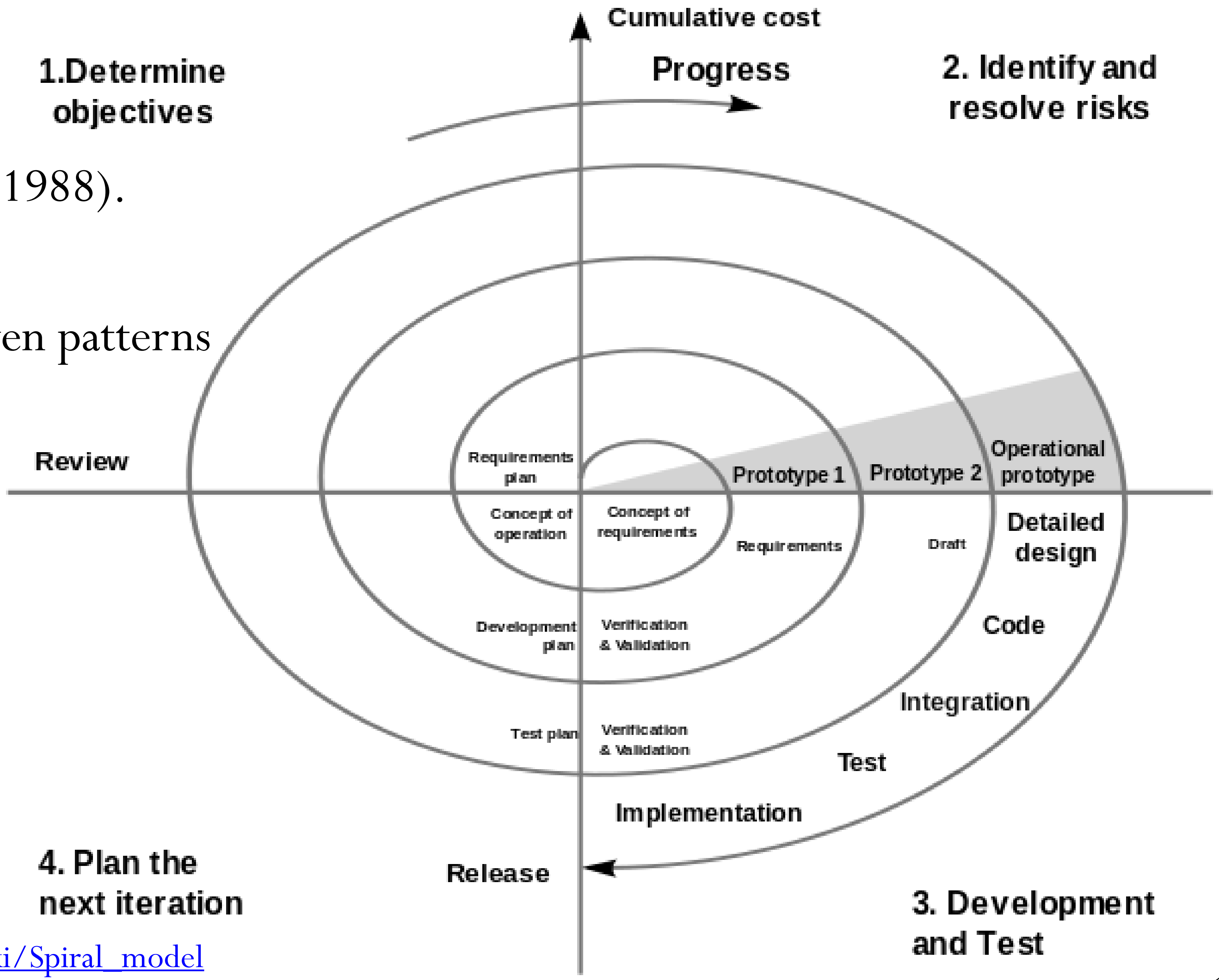
- Royce's original waterfall model have the following phases:
- System and software requirements document
- Analysis: resulting in models, schema, and business rules
- Design: resulting in the software architecture
- Coding: the development, proving, and integration of software
- Testing: the systematic discovery and debugging of defects
- Operations: the installation, migration, support, and maintenance of complete systems



Spiral Model

- Spiral model (Boehm, 1988).

- Based on the Risk-driven patterns of a given project,



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

Agile development

- Discover requirements and develop solutions with collaborative effort of self-organizing and cross-functional teams and customers (end users). Promotes
  - adaptive planning,
  - evolutionary development,
  - early delivery, and
  - continual improvement, and
  - it encourages flexible responses to change.

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

Rapid Prototyping

- Involves creating a working model of various parts of the system at a very early stage, after a relatively short investigation
- The Dynamic Systems Development (DSDM) lifecycle of a prototype is to:
  - Identify prototype
  - Agree to a plan
  - Create the prototype
  - Review the prototype

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

Incremental build model

- The product is designed, implemented and tested incrementally (a little more is added each time) until the product is finished.
- Involves both development and maintenance.
- The product is defined as finished when it satisfies all of its requirements.
- This model combines the elements of the waterfall model with the iterative philosophy of prototyping.

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

Extreme project management

- XPM: manages complex and uncertain projects
- Differs from traditional project management: it is open, elastic, and indeterministic approach.
- Focus: Human side of project management (e.g. managing project stakeholders).
- It corresponds to extreme programming (intended to improve software quality and responsiveness to changing customer requirements).

Development Phases

- ***Development*** activities required to evolve the system
- ***Manufacturing/Production/Construction*** of large or unique systems or subsystems
- ***Deployment*** activities (e.g. environment, assemble, install, train) to achieve full operational capability
- ***Operation*** is the user functions, objectives, and tasks

[4] *Systems Engineering Fundamentals.* Defense Acquisition University Press, 2001

Maintenance Phase

- ***Verification*** activities evaluate progress and effectiveness, and measure specification compliance
- ***Support*** activities provide logistics management
- ***Training*** activities like knowledge and skill levels functions.
- ***Disposal*** activities decommissioned, destroyed

[4] *Systems Engineering Fundamentals.* Defense Acquisition University Press, 2001

System Development Life Cycle (SDLC)

- ❑ Systems Analysis,
- ❑ Systems Design,
- ❑ Systems Modelling,
- ❑ Systems Architecture,
- ❑ System Development and Testing,
- ❑ System Maintenance and Evolution,
- ❑ SDLC example (Cloud Service life cycle)

Systems Development Life Cycle (SDLC)

- In systems engineering, SDLC is a view of a system that addresses all phases:
- conception, design, development, production and/or construction, distribution, operation, maintenance, support, retirement, phase-out, and disposal.
- SDLC is a process for planning, creating, testing, and deploying a system.
- Systems engineers and Systems developers define the working phases: plans, design, build, test, and deliver.

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

Systems Development Life Cycle (SDLC) Phases

- **Preliminary analysis:** propose alternative solutions, describe costs and benefits, and submit a preliminary plan with recommendations.
- **Systems analysis and requirements definition:** Define project goals into defined functions and operations of the intended application.
- **Systems design:** desired features and operations, including screen layouts, business rules, process diagrams, pseudo-code, and other documentation
- **Development**: depend of system's domain.
- **Integration and testing:** modules are brought together into a special testing environment, then checked for errors, bugs, and interoperability.

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

Systems Development Life Cycle (SDLC) Phases

- **Acceptance, installation, deployment**: This is the final stage of initial development, where the software is put into production and runs actual business.
- **Maintenance**: During the maintenance stage of the SDLC, the system is assessed/evaluated to ensure it does not become obsolete. This is also where changes are made to initial software.
- **Evaluation**: extension of the maintenance stage, post-implementation review, evaluate entire processes, business requirements, objectives, reliability, fault-tolerance, functional requirements.
- **Disposal:** Plans are developed for discontinuing the use of system information and making the transition to a new system.

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

Systems Analysis

Systems Analysis

- Process to identify goals and purposes that efficiently create and operate systems
- Breaks down a system into its component to study their interactions
- Relates to requirements analysis or to operations research
- Helps a decision maker identify a better action and decision
- Helps to produce and mange the data model and the database
- Feasibility study: determining whether a project is economically, socially, technologically and organizationally feasible
- Fact-finding measures: ascertain the requirements of the system's end-users
- How the end-users would operate the system

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

Systems Analysis: Phases

- **Scope Definition:** Clearly defined objectives and requirements necessary to meet a project's requirements as defined by its stakeholders
- **Problem analysis:** the process of understanding problems and needs and arriving at solutions that meet them
- **Requirements analysis:** determining the conditions that need to be met
- **Logical design:** looking at the logical relationship among the objects
- **Decision analysis:** making a final decision
- **System analysis professionals** include: system analyst, business analyst, manufacturing engineer, system architect, enterprise architect, software architect,

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

Systems Analysis

- It allows developers to carry out quantitative assessments of systems
  - to select and/or update systems efficient and
  - to generate system data
- During engineering, assessments should be performed every time technical choices or decisions are made to determine compliance with system requirements.
- Systems Analysis provides a rigorous approach to technical decision-making.
- It is used to perform trade-off studies, and includes modeling and simulation, cost analysis, technical risks analysis, and effectiveness analysis.

https://www.sebokwiki.org/wiki/System_Analysis

Systems Analysis: Evaluation criteria

- System Analysis Processes
  - Trade-Off Studies
  - Effectiveness Analysis
  - Cost Analysis
  - Technical Risks Analysis (or Risk Management)
- to define assessment criteria based on system requirements;
- to assess design properties of each candidate solution in comparison to these criteria;
- to score the candidate solutions globally and to justify the scores; and
- to decide on the appropriate solution(s)

https://www.sebokwiki.org/wiki/System_Analysis

Systems Analysis: General principles

- **Assessment criteria** are based upon a problem or opportunity system description
  - Criteria assumes a **hard system** problem context can be defined.
  - **Soft system** descriptions from which additional "soft" criteria can be defined.
  - Criteria must consider required system behavior and properties of the complete solution.
  - Must consider non-functional issues such as system safety, security, etc.
  - For example, a stakeholder preference for or against certain kinds of solutions, relevant social, political or cultural conventions to be considered, etc.
- **Assessment criteria:** constraints on cost and time scales acceptable to stakeholders
- **Trade studies** provide a mechanism for conducting analysis of alternative solutions

https://www.sebokwiki.org/wiki/System_Analysis

Systems Analysis: System of Interest (SoI)

- SoI is understanding the socio-economic and technological context in which potential problems or opportunities reside.
- Enterprise strategic goals and stakeholder needs, expectations, and requirements represent the problem
- Opportunity from the viewpoint of business or enterprise decision makers while also taking into account the views of users, acquirers, and customers.

https://www.sebokwiki.org/wiki/Business_or_Mission_Analysis

Business or Mission Analysis

- It is part of the larger set of concept definition activities
  - the set of systems engineering activities in which the problem space and
  - the needs of the business or enterprise and stakeholders are closely examined
  - identification, characterization, and assessment of operational problem or opportunity
- Stakeholder Needs and Requirements activities
  - Stakeholder's desired mission and performance of certain aspects of the solution.
- Mission or Business function in a problem space frames the solution
- 'Concept of Operations' (ConOps) and 'Operational Concept' (OpsCon)
  - to analyze and define system operation, and operational scenarios

https://www.sebokwiki.org/wiki/Business_or_Mission_Analysis

Systems Design

Systems Design

- Systems design is the process of defining the architecture, product design, modules, interfaces, and data for a system to satisfy specified requirements.
- Systems design could be seen as the application of systems theory to product development.
- There are some overlaps with the disciplines of systems analysis, systems architecture, and systems engineering.
- Provides sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture. (ISO/IEC/IEEE 15288 [ISO 2015]).

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

Systems Design: levels

- **Architectural design:** the design of the system architecture to describe the structure, behavior, analysis, and views of system.
- **Logical design:** an abstract representation of the data flows, inputs and outputs of the system with modelling, graphical model of the actual system, and entity-relationship diagrams (ER diagrams).
- **Physical design:** the actual input and output processes of the system.
  - how input-output in a system works,
  - how data/input/output are verified/authenticated, processed, and displayed
  - Input requirements, Output requirements,
  - Storage requirements, Processing requirements,
  - System control and backup or recovery.
  - User Interface Design, Data Design, Process Design

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

Systems Design

- to supplement the system architecture by providing information and data useful and necessary for implementation of the system components
- the process of developing, expressing, documenting, and communicating the realization of the architecture
- System deals with complexity
  - (interconnections level, multi-techno, emergence, etc.).
  - structuring the components that comprise a system
  - structure explains the functional, behavioral, temporal, physical, and other aspects of a system
  - activities to conceive a set of system components

https://www.sebokwiki.org/wiki/System_Design

Design Characteristics and Enablers

- Every technological discipline owns its laws, rules, theories, and enablers concerning transformational, structural, behavioral, and temporal properties.
- Design characteristics include dimensions, shapes, materials, and data processing structures.
- Design enablers include formal expressions, equations, drawings, diagrams, tables of metrics with their values, margins, patterns, algorithms, and heuristics.
- Examples in mechanics: shape, geometrical pattern, dimension, volume, surface, curves, resistance to forces, distribution of forces, weight, velocity of motion etc.
- Examples in software: distribution of processing, data structures, data persistence, procedural/data/control abstraction, encapsulation, and structural patterns

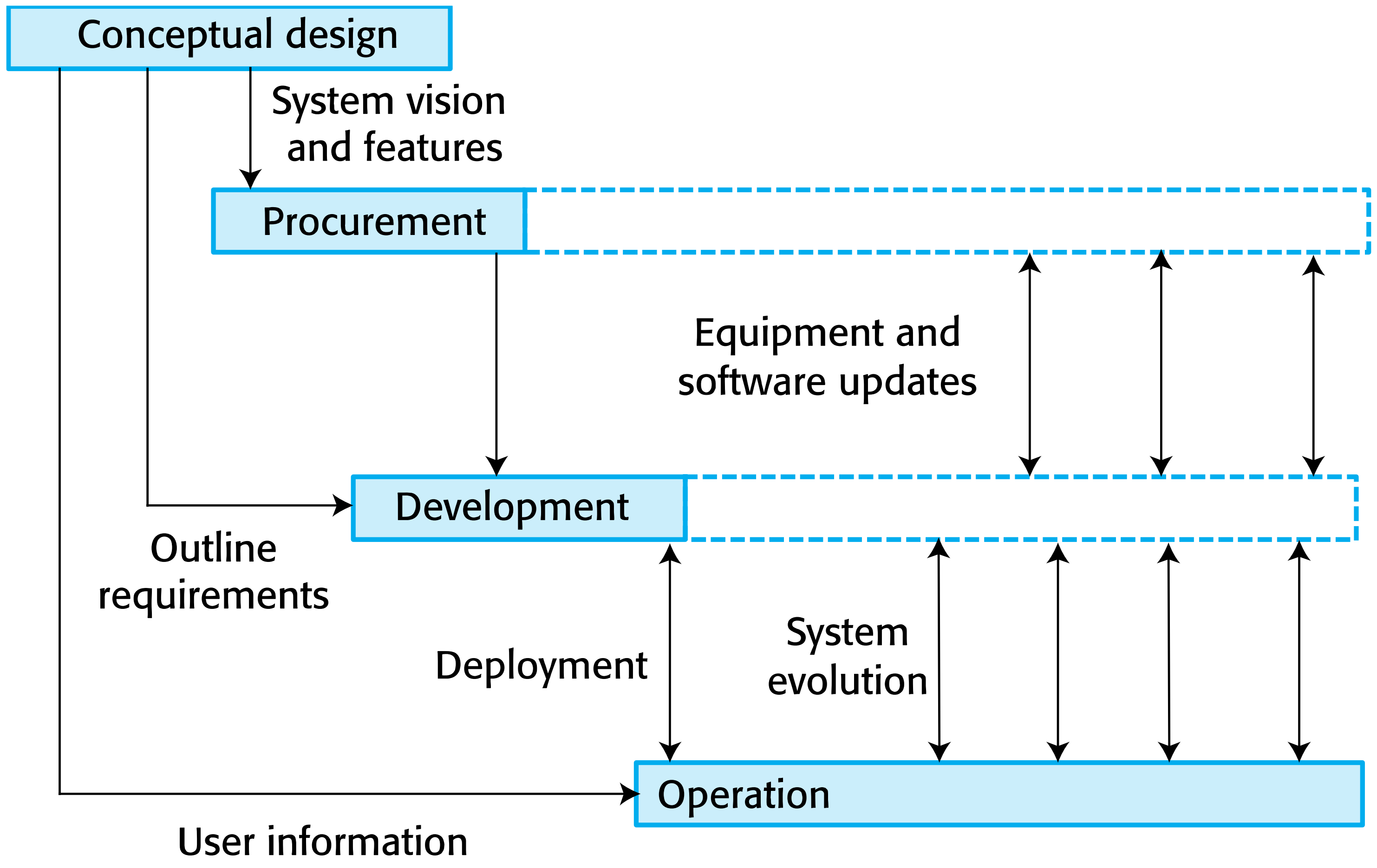
https://www.sebokwiki.org/wiki/System_Design

System Design Process

1. Initialize design definition
2. Establish design characteristics and design enablers related to each system component
3. Assess alternatives for obtaining system component
4. Manage the design

https://www.sebokwiki.org/wiki/System_Design

Stages of systems engineering

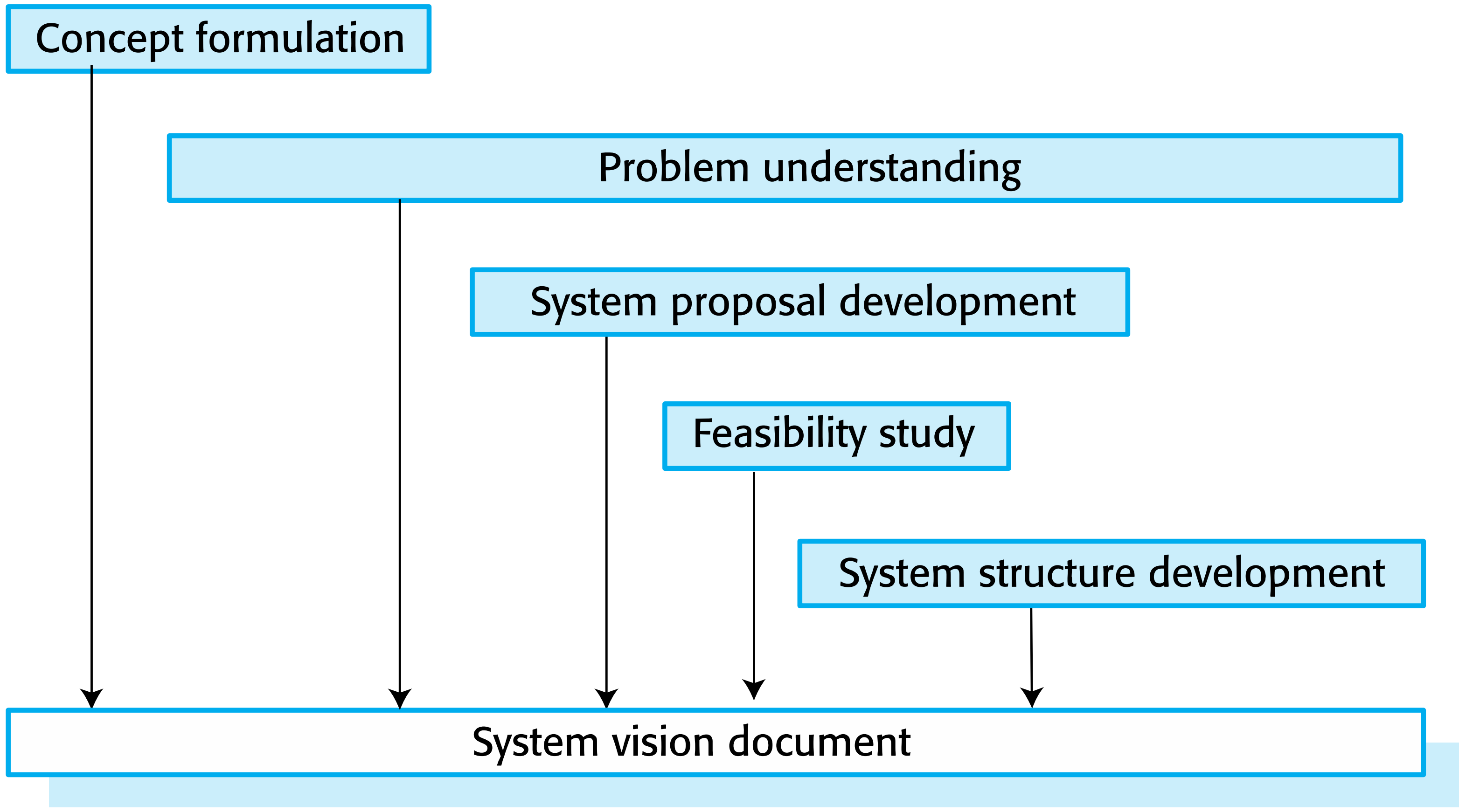


Sommerville, Ian. "Software engineering 10th Edition." (2015).

Stages of systems engineering

- Conceptual design
  - Sets out the purpose of the system, why it is needed and the high-level features that users might expect to see in the system
- Procurement or acquisition
  - The conceptual design is developed so that decisions about the contract for the system development can be made.
- Development
  - Engineered and operational processes defined.
- Operation
  - The system is deployed and used for its intended purpose.

Conceptual design activities



Sommerville, Ian. "Software engineering 10th Edition." (2015).

Conceptual design activities

- Concept formulation
  - Refine an initial statement of needs and work out what type of system is most likely to meet the needs of system stakeholders
- Problem understanding
  - Discuss with stakeholders how they do their work, what is and isn’t important to them, what they like and don’t like about existing systems
- System proposal development
  - Set out ideas for possible systems (maybe more than one)

Conceptual design activities

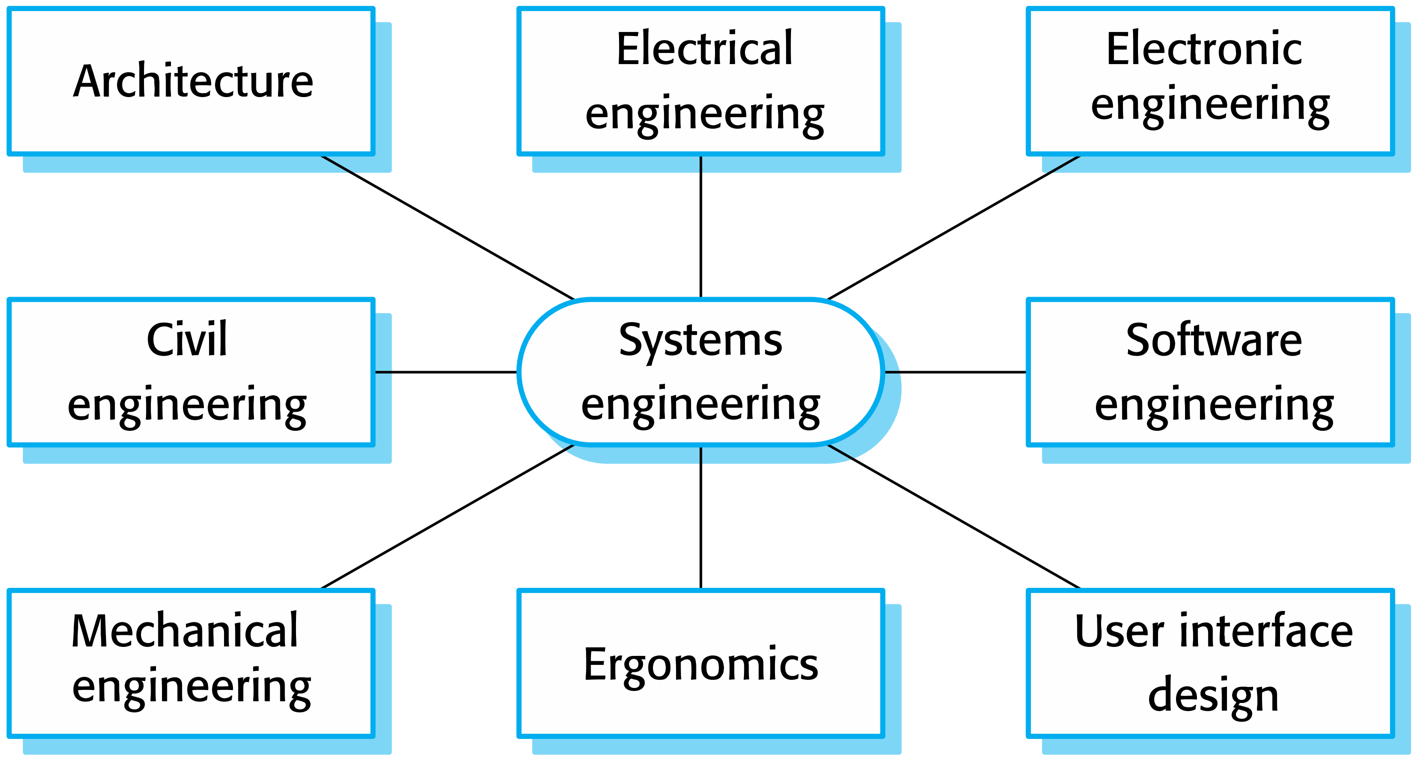
- Feasibility study
  - Look at comparable systems that have been developed elsewhere (if any) and assess whether or not the proposed system could be implemented using current technologies
- System structure development
  - Develop an outline architecture for the system, identifying (where appropriate) other systems that may be reused
- System vision document
  - Document the results of the conceptual design in a readable, non-technical way. Should include a short summary and more detailed appendices.

Systems Modelling

Conceptual model

- Conceptual systems design is a key activity where
  - high level system requirements and
  - a vision of the operational system is developed.
- System procurement covers all of the activities involved in deciding
  - what system to buy and
  - who should supply that system.

Professional disciplines involved



Sommerville, Ian. "Software engineering 10th Edition." (2015).

Inter-disciplinary working

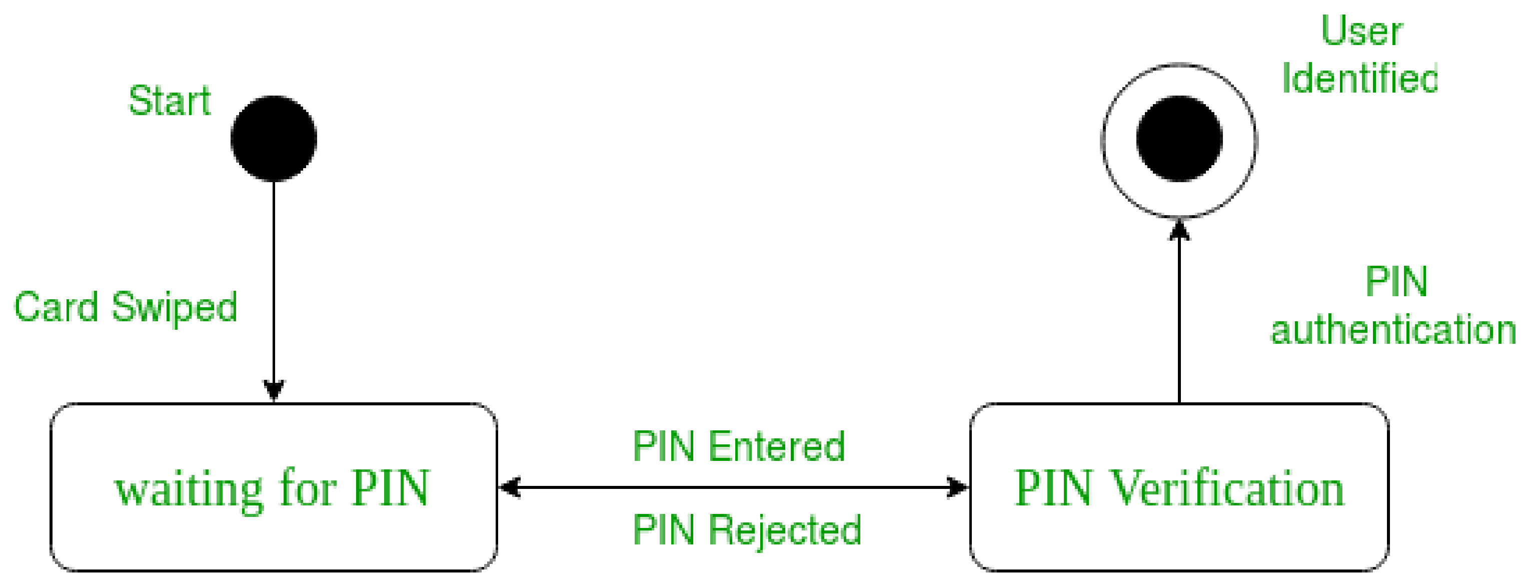
- Communication difficulties
  - Different disciplines use the same terminology to mean different things. This can lead to misunderstandings about what will be implemented.
- Differing assumptions
  - Each discipline makes assumptions about what can and can't be done by other disciplines.
- Professional boundaries
  - Each discipline tries to protect their professional boundaries and expertise and this affects their judgments on the system.

Modelling Techniques

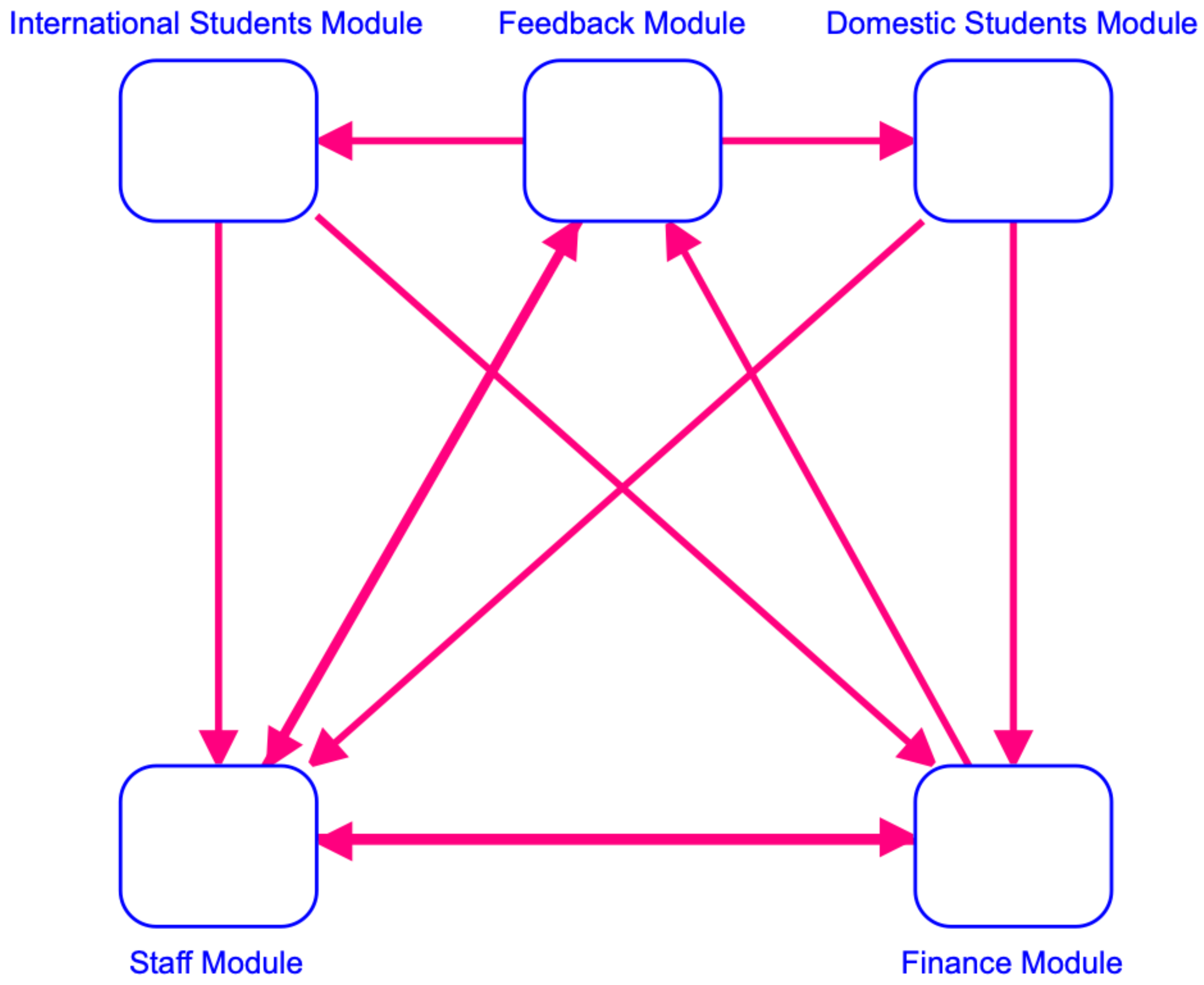
- State-charts models
- Scenarios modelling
- Simulations, prototyping
- Quality Function Deployment
- Functional Flow Block Diagram for operational scenarios
- Sequence diagrams, Activity diagrams, Use-cases, State-machine diagrams, Requirements diagrams
- Systems Modelling Language (SysML),

https://www.sebokwiki.org/wiki/System_Requirements

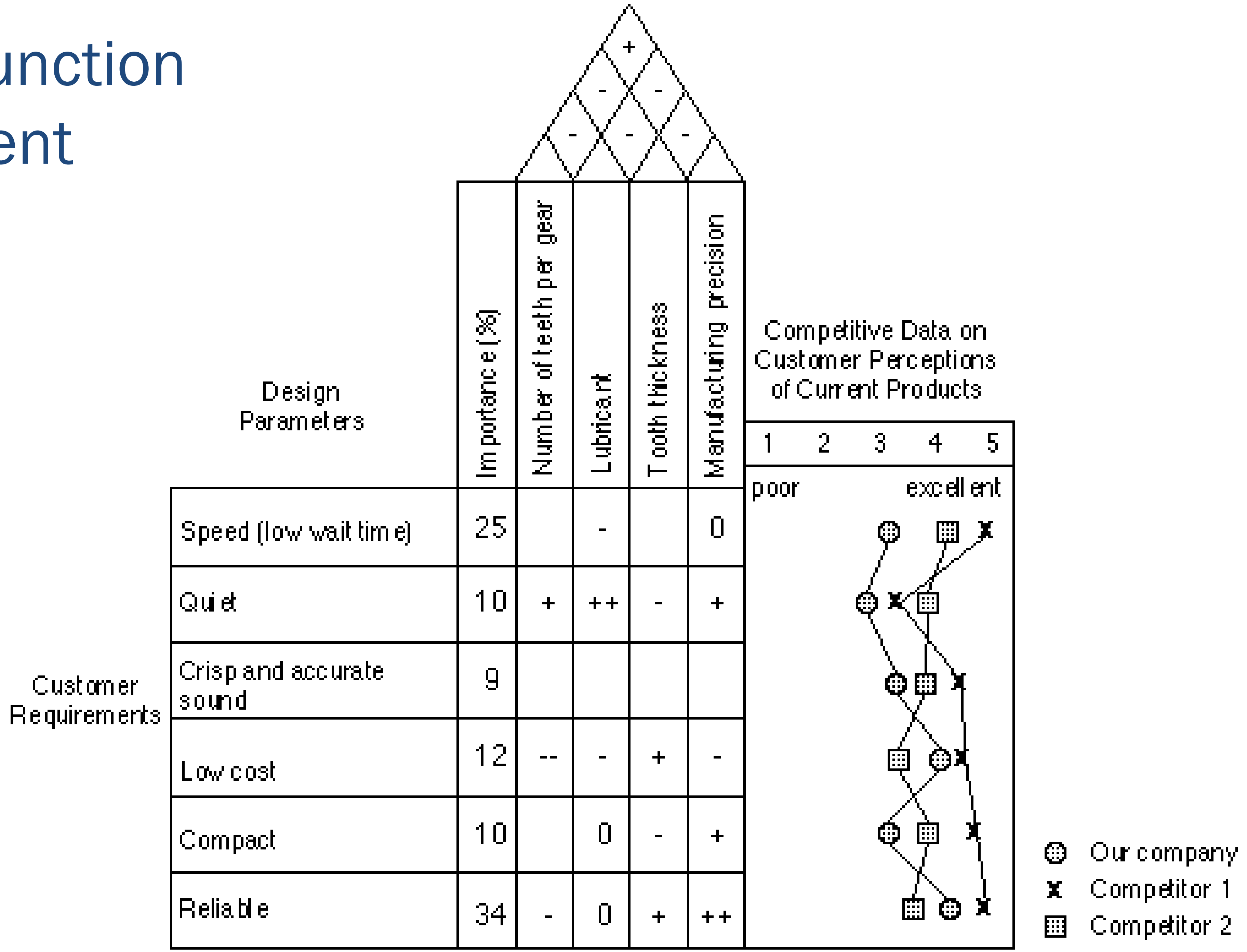
State-chart models



Scenarios modelling



Quality Function Deployment

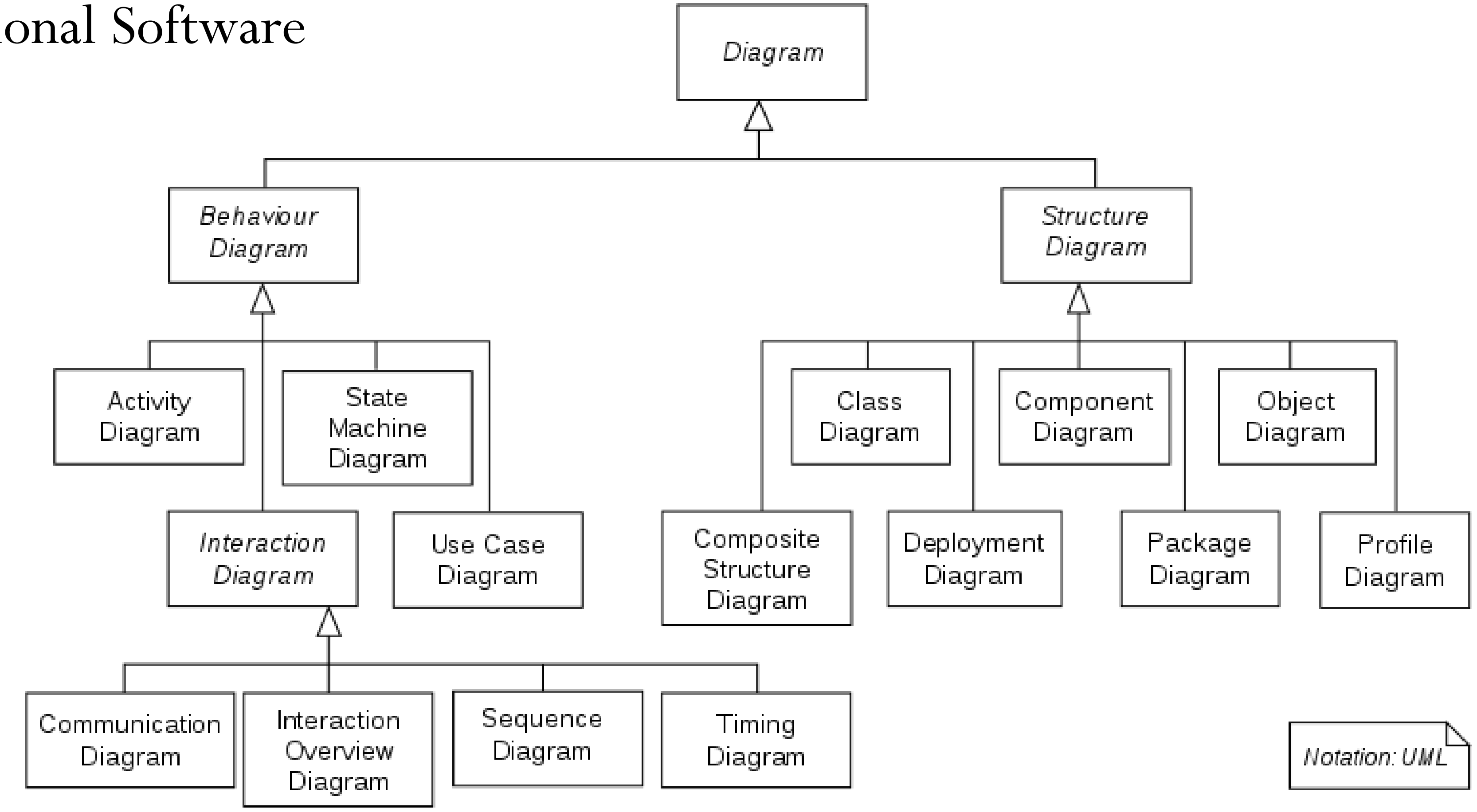


+
-
-
-
-
-
Competitive Data on Customer Perceptions of Current Products
1
2
3
4
5
poor
excellent
Our company
Competitor 1
Competitor 2

Customer Requirements / Design Parameters	Importance (%)	Number of teeth per gear	Lubricant	Tooth thickness	Manufacturing precision
Speed (low wait time)	25		-		0
Quiet	10	+	++	-	+
Crisp and accurate sound	9				
Low cost	12	--	-	+	-
Compact	10		0	-	+
Reliable	34	-	0	+	++

Unified Modeling Language (UML)

- A general-purpose, developmental, modeling language in software engineering
- Developed at Rational Software

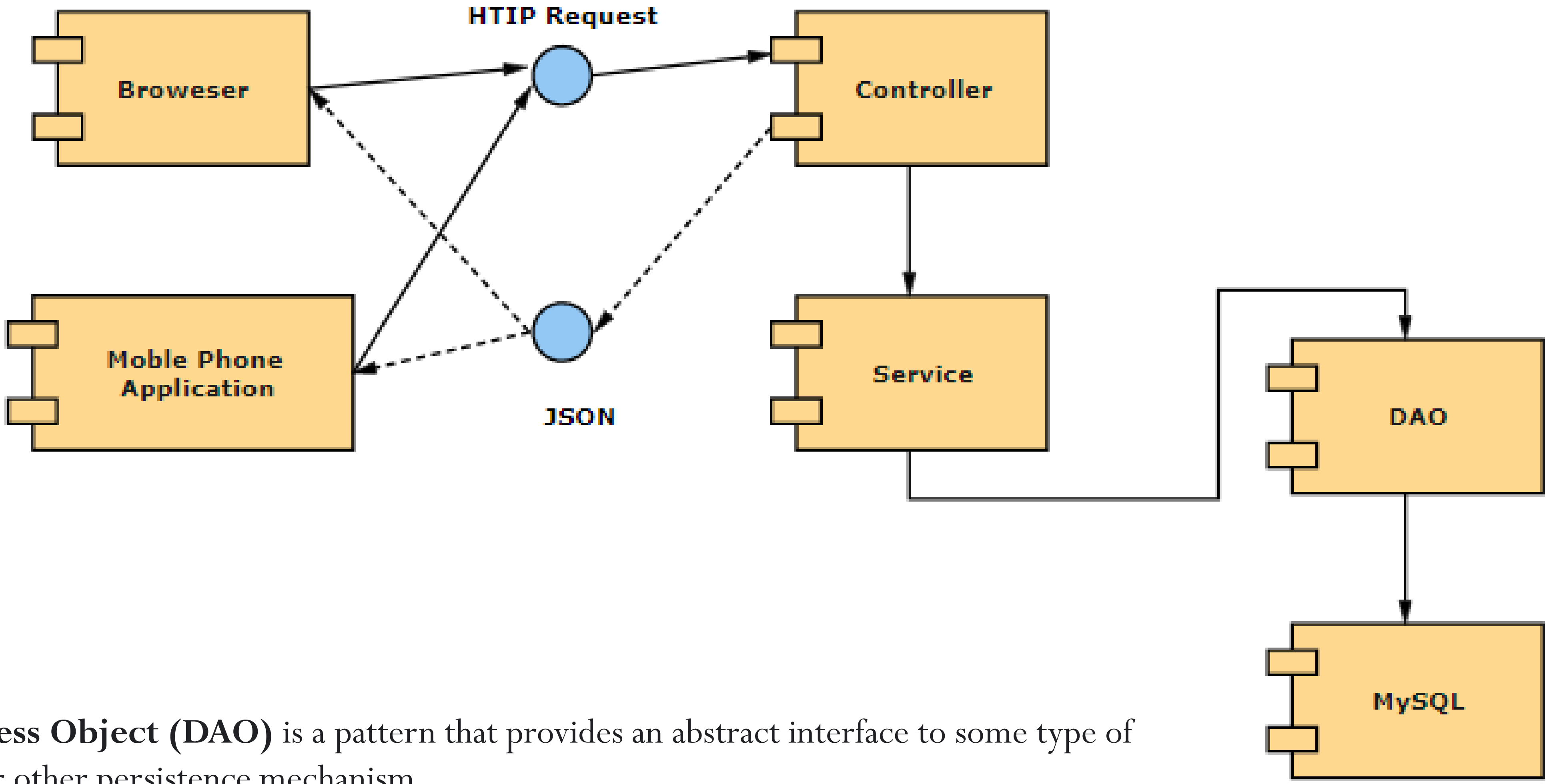


https://en.wikipedia.org/wiki/Unified_Modeling_Language

UML diagram types

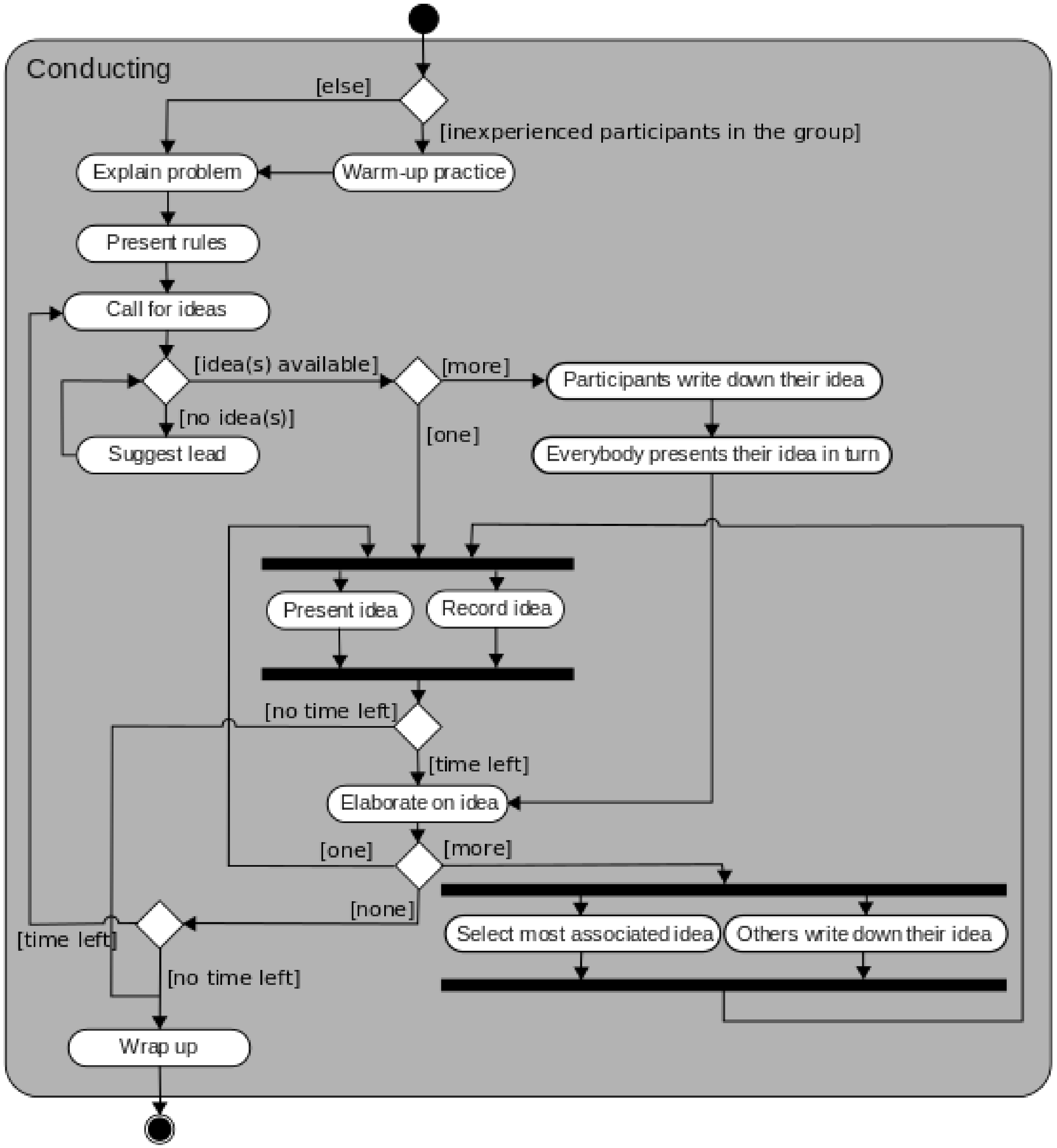
- **Structure diagrams**
  - **Component diagrams** and **Class diagrams**
- **Behaviour diagrams**
  - **Activity diagrams**, which show the activities involved in a process or in data processing.
  - **Use case diagrams**, which show the interactions between a system and its environment.
  - **State diagrams**, which show how the system reacts to internal and external events.
  - **Interaction diagrams**
    - **Sequence diagrams**, which show interactions between actors and the system and between system components.
    - **Communication diagrams**

Component diagrams



Data Access Object (DAO) is a pattern that provides an abstract interface to some type of database or other persistence mechanism.

Activity diagrams



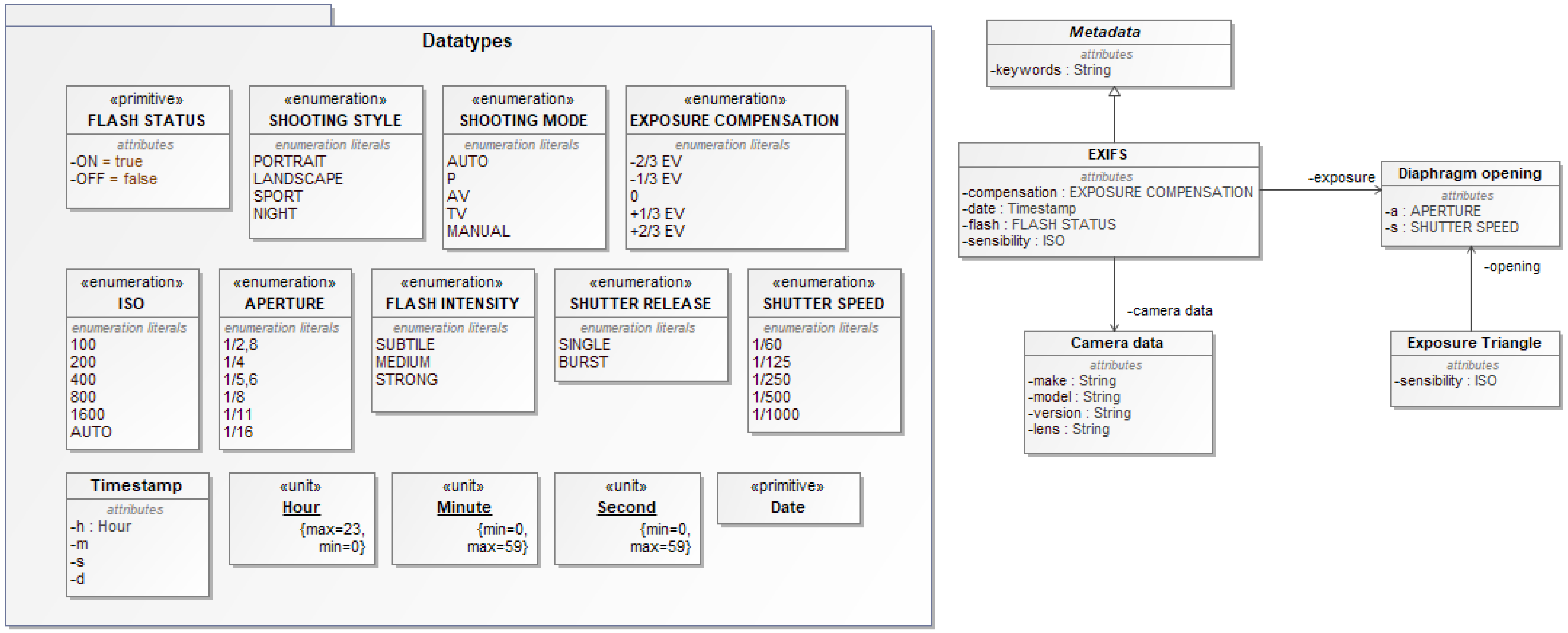
https://en.wikipedia.org/wiki/Unified_Modeling_Language

Systems Modeling Language (SysML)

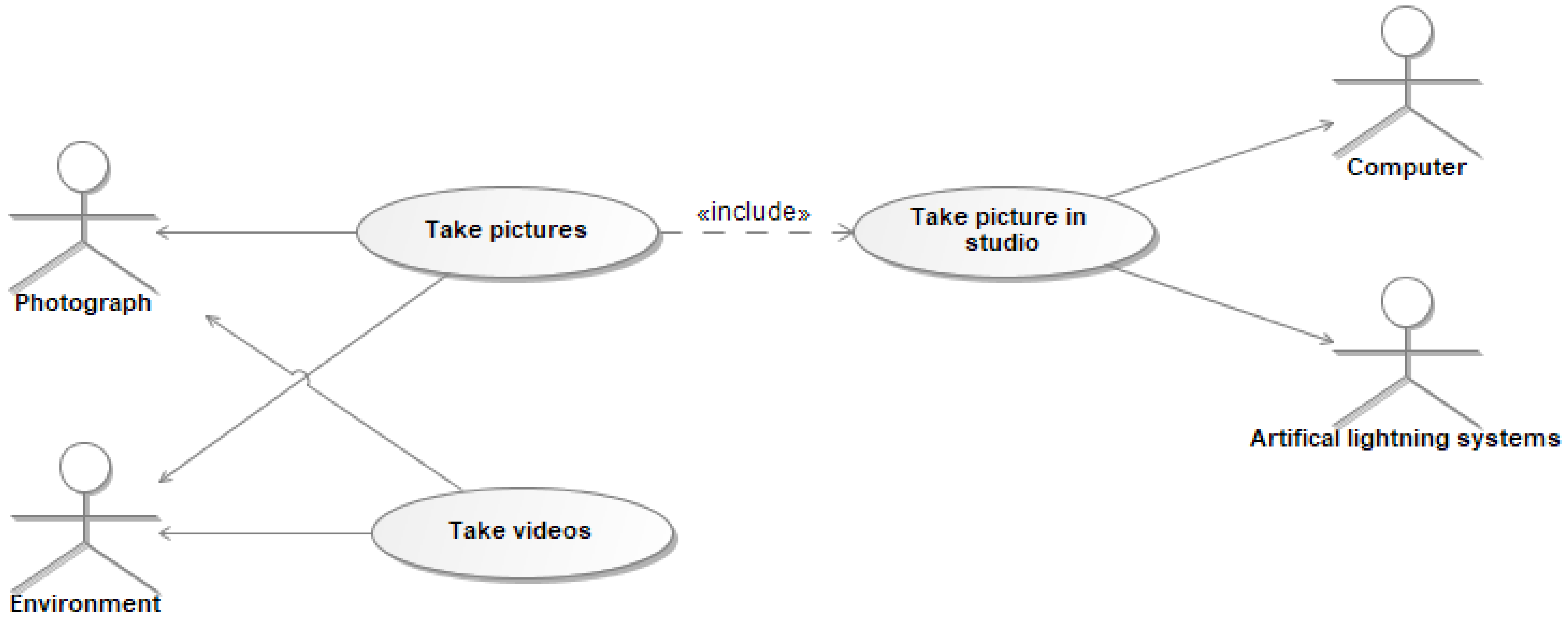
- It is a general-purpose modeling language for systems engineering applications.
- It supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems.
- SysML is defined as an extension of a subset of the Unified Modeling Language (UML).
- SysML's semantics are more flexible and expressive.
- SysML reduces UML's software-centric restrictions
- SysML adds two new diagram types, **requirement and parametric diagrams**.
- SysML is a comparatively small language i.e. easier to learn and apply (w.r.t. UML's software-centric constructs)

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

SysML Class diagrams

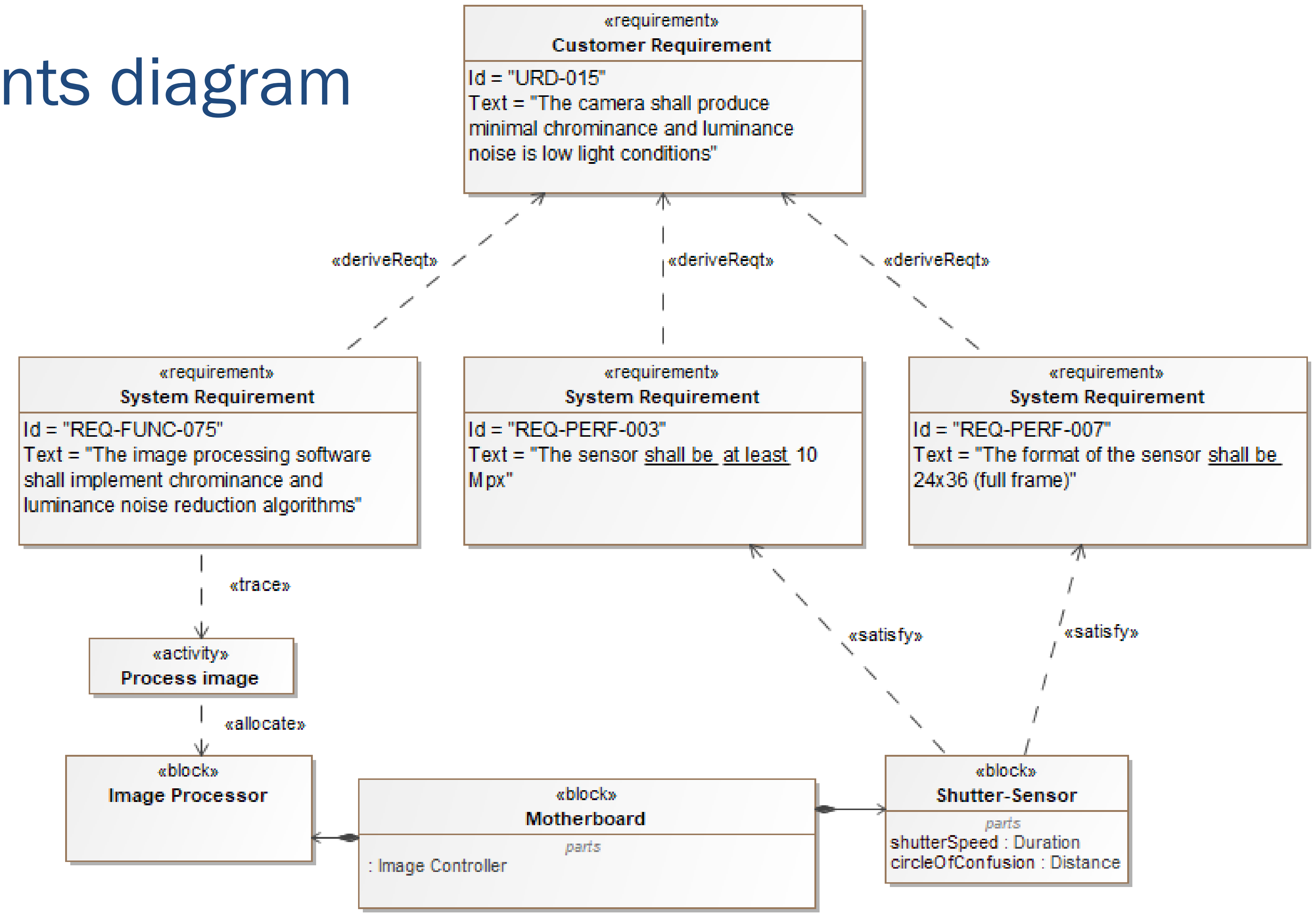


SysML Use case



SysML
Requirements diagram

SysML



«requirement»
Customer Requirement
Id = "URD-015"
Text = "The camera shall produce minimal chrominance and luminance noise is low light conditions"

«requirement»
System Requirement
Id = "REQ-FUNC-075"
Text = "The image processing software shall implement chrominance and luminance noise reduction algorithms"

«requirement»
System Requirement
Id = "REQ-PERF-003"
Text = "The sensor shall be at least 10 Mpx"

«requirement»
System Requirement
Id = "REQ-PERF-007"
Text = "The format of the sensor shall be 24x36 (full frame)"

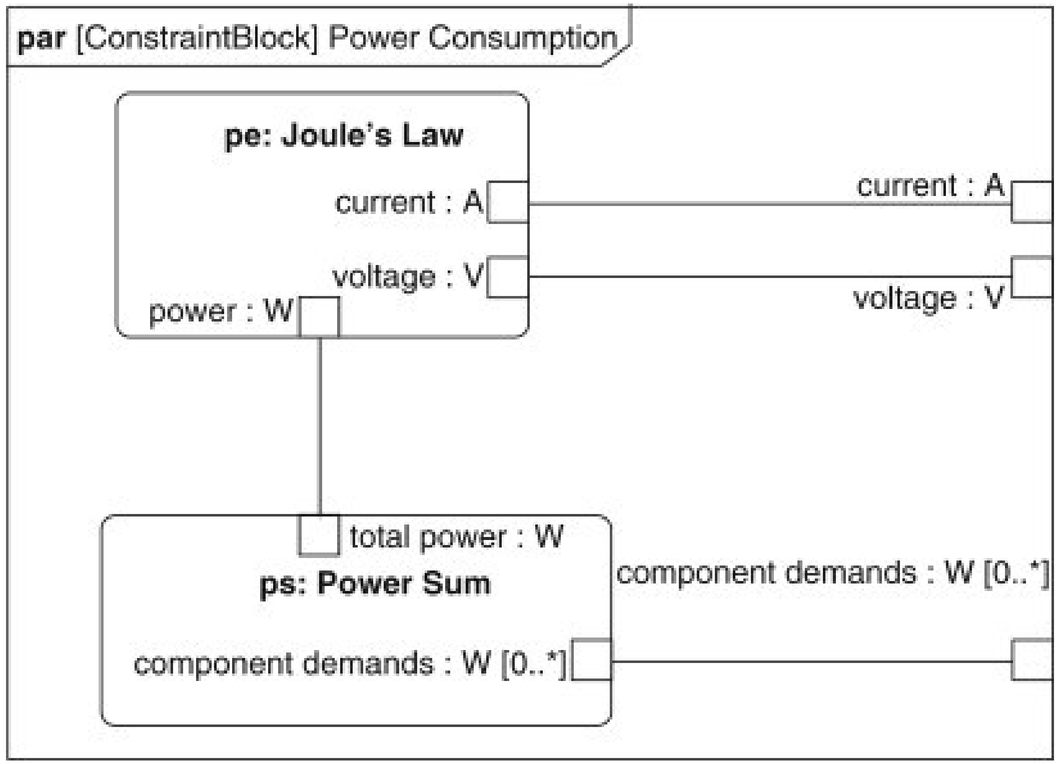
«activity»
Process image

«block»
Image Processor

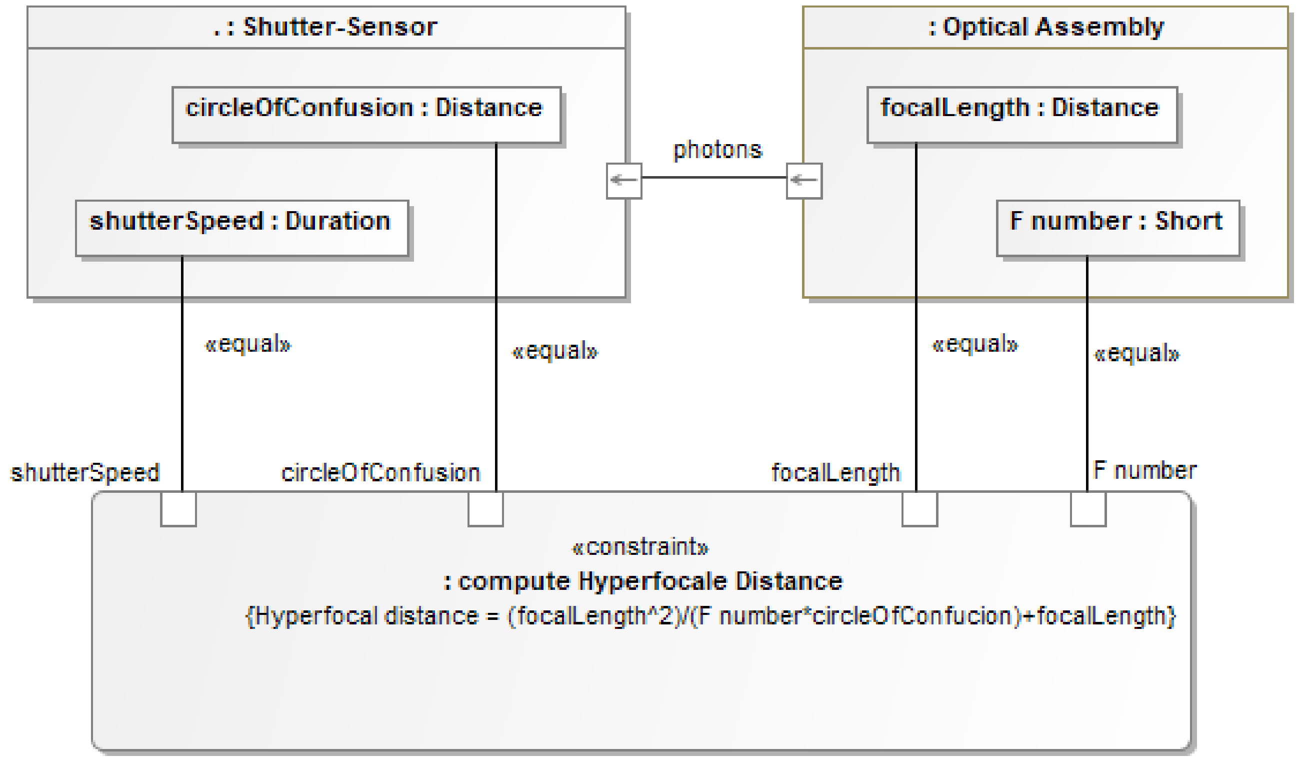
«block»
Motherboard
parts
: Image Controller

«block»
Shutter-Sensor
parts
shutterSpeed : Duration
circleOfConfusion : Distance

SysML
Parametric Diagram



SysML



Systems Architecture

System Architecture and System Design

- System design links the system architecture and the implementation of components
- Explains the functional, behavioral, temporal, physical, and other aspects of a system
- Creates a blueprint for the design with the necessary structure and behavior specifications
- Design definition is driven by specified requirements, the system architecture, and analysis of performance and feasibility.
- Design addresses the implementation technologies and their assimilation.
- Design provides the "how-" or "implement-to" level of the definition.
- Design concerns every system component composed of implementation technologies, such as mechanics, electronics, software, chemistry, human operations and services

https://www.sebokwiki.org/wiki/System_Design

System Architecture

- to define a comprehensive solution based on principles, concepts, and properties logically related to and consistent with each other.
- solution architecture has features, properties, and characteristics which satisfy the problem or opportunity expressed by
  - a set of system requirements (traceable to mission/business and stakeholder requirements) and
  - life cycle concepts (e.g., operational, support) and
  - implementable through technologies (e.g., mechanics, electronics, hydraulics, software, services, procedures, human activity).

https://www.sebokwiki.org/wiki/System_Architecture

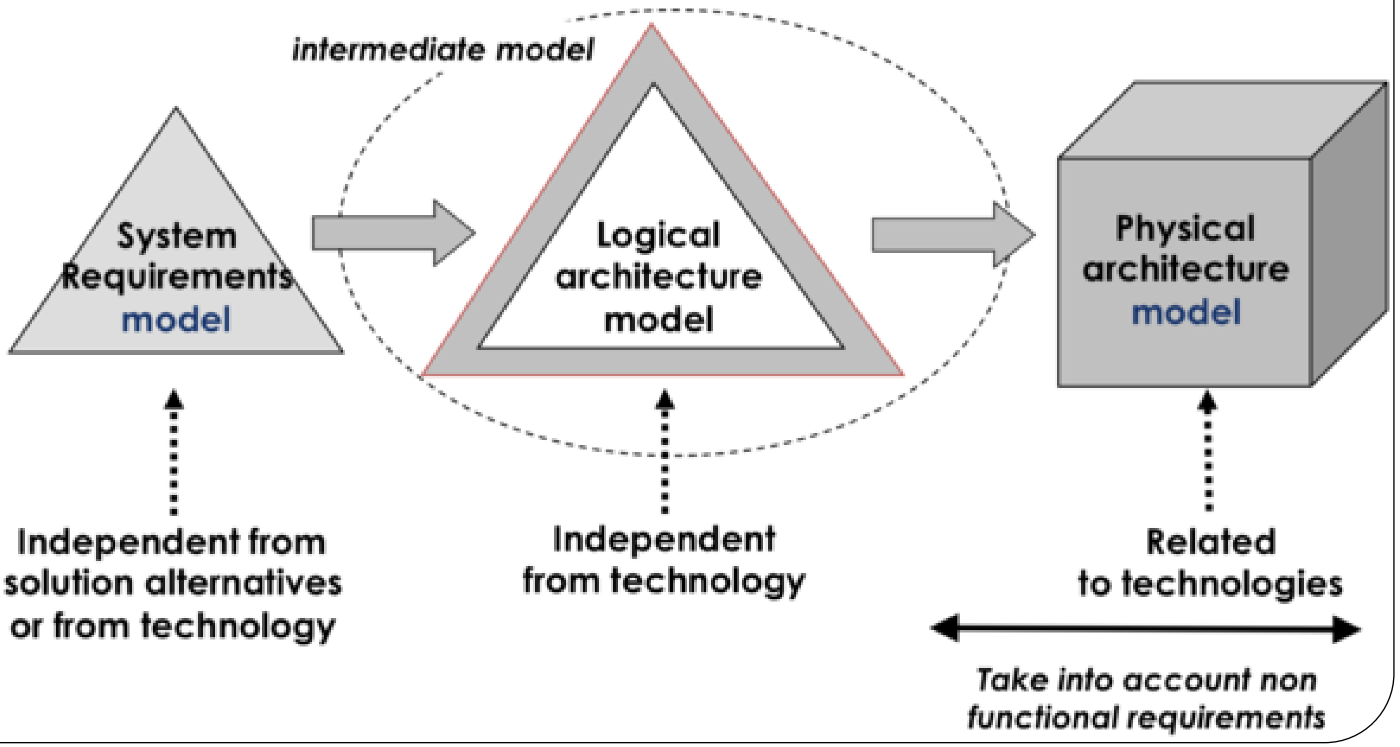
System Architecture

- abstract, conceptualization-oriented, global, and focused to achieve the mission and life cycle concepts of the system.
- focuses on high-level structure in systems and system elements.
- addresses the architectural principles, concepts, properties, and characteristics of the system-of-interest.
- applied to more than one system, in some cases forming the common structure, pattern, and
- set of requirements for classes or families of similar or related systems

https://www.sebokwiki.org/wiki/System_Architecture

Transition from System Requirements to Architecture

- System requirements to an intermediate model of logical architecture and physical architecture models.


intermediate model
System Requirements model
Logical architecture model
Physical architecture model
Independent from solution alternatives or from technology
Independent from technology
Related to technologies
Take into account non functional requirements

https://www.sebokwiki.org/wiki/System_Architecture
Faisandier 2012

Logical Architecture

- The logical architecture of a system is composed of a set of related technical concepts and principles that support the logical operation of the system.
- It includes a functional architecture, a behavioral architecture, and a temporal architecture.
- System boundary and functions, from which more detailed system requirements can be derived.
- To identify functional requirements from the stakeholder requirements and
- To use this to start the architectural definition, or
- To begin with a high-level functional architecture view and use this as the basis for structuring system requirements.

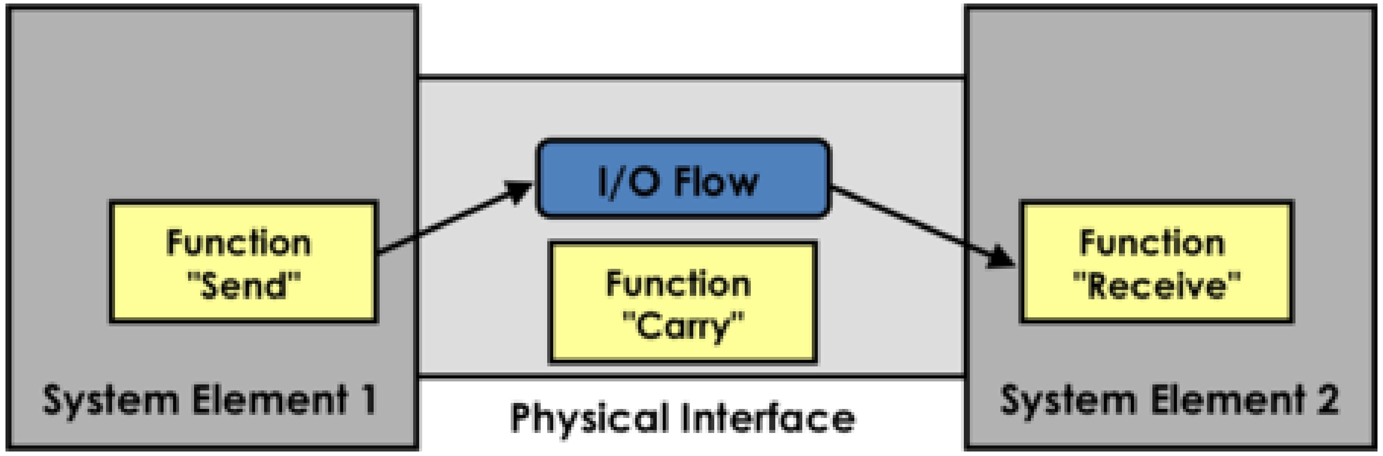
https://www.sebokwiki.org/wiki/Logical_Architecture_(glossary)
https://www.sebokwiki.org/wiki/System_Requirements

Physical Architecture

- An arrangement of physical elements (system elements and physical interfaces) which provides the design solution for a product, service, or enterprise, and is intended to satisfy logical architecture elements and system requirements. It is implementable through technologies. (ISO/IEC 2010)
- An arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirement baseline. (ISO/IEC 2007)

System Architecture activities of the process

1. Initialize the definition of the system architecture
2. Define necessary architecture viewpoints
3. Develop candidate architectures models and views
4. Relate system architecture to system design
5. Assess architecture candidates and select one
6. Manage the selected architecture



https://www.sebokwiki.org/wiki/System_Architecture

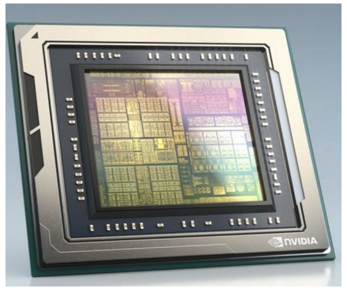
Autonomous Driving Cars

- aka. Self-driving car
- a vehicle that is capable of sensing its environment and moving safely with little or no human input.
- Self-driving cars combine a variety of sensors to perceive their surroundings, such as radar, sonar, GPS, odometry, and inertial measurement units.
- Advanced control systems interpret sensory information to identify appropriate navigation paths, as well as obstacles and relevant signage.

- https://www.youtube.com/watch?v=tlThdr3O5Qo
- https://www.youtube.com/watch?v=0GnysB0rO3s

https://en.wikipedia.org/wiki/Self-driving_car

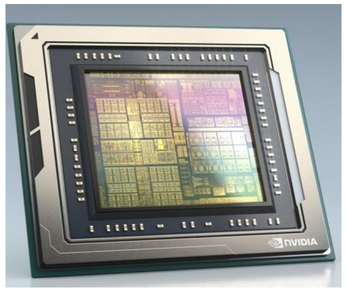
NVIDIA DRIVE AGX



- Software-Defined Platform for Autonomous Machines
  - deep neural networks run simultaneously in autonomous vehicles and robots
- The platform is powered by a new system-on-a-chip (SoC) called Orin, which consists of 17 billion transistors and is the result of four years of R&D investment.
- The Orin SoC integrates NVIDIA's next-generation GPU architecture and Arm Hercules CPU cores.
- deliver 200 trillion operations per second — nearly 7x the performance of NVIDIA's previous generation Xavier SoC.
- Orin is designed to handle the large number of applications (e.g. deep learning and computer vision)

https://nvidianews.nvidia.com/news/nvidia-introduces-drive-agx-orin-advanced-software-defined-platform-for-autonomous-machines

NVIDIA DRIVE AGX



- Developed to enable architecturally compatible platforms that scale full self-driving vehicle, enabling to develop large-scale and complex families of software products.
- "Creating a safe autonomous vehicle is perhaps society's greatest computing challenge," said Jensen Huang, founder and CEO of NVIDIA.
- "The amount of investment required to deliver autonomous vehicles has grown exponentially, and the complexity of the task requires a scalable, programmable, software-defined AI platform like Orin."

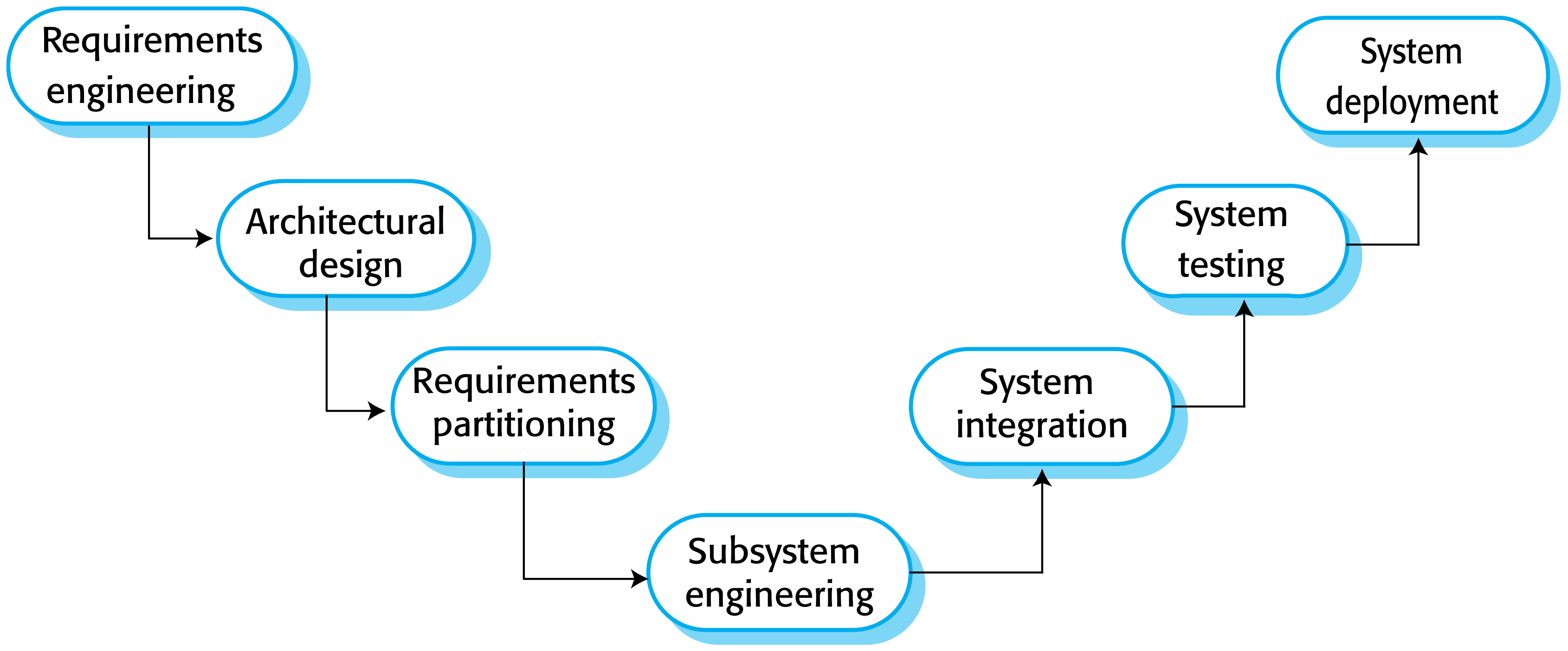
https://nvidianews.nvidia.com/news/nvidia-introduces-drive-agx-orin-advanced-software-defined-platform-for-autonomous-machines

System Development and Testing

System Development and Testing

- Systems engineering is concerned with all aspects of specifying, buying, designing and testing complex sociotechnical systems.
- The fundamental systems engineering processes are conceptual systems design, system procurement, system development and system operation.
- System development processes include requirements specification, design, construction, integration and testing.

System Development and Testing



Sommerville, Ian. "Software engineering 10th Edition." (2015).

System Development and Testing

- Requirements engineering
  - The process of refining, analysing and documenting the high-level and business requirements identified in the conceptual design
- Architectural design
  - Establishing the overall architecture of the system, identifying components and their relationships
- Requirements partitioning
  - Deciding which subsystems (identified in the Systems Architecture) are responsible for implementing the system requirements

System Development and Testing

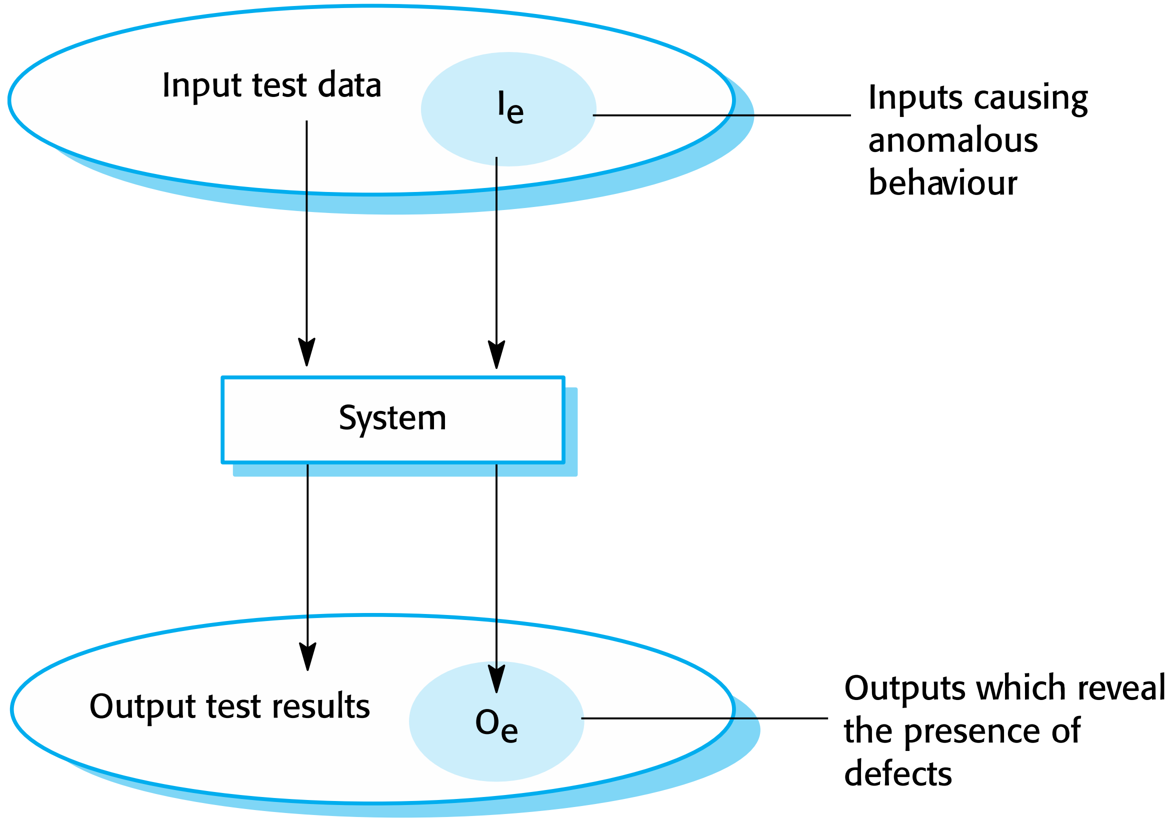
- Subsystem engineering
  - Developing the system components, configuring components, defining the operational processes for the system and re-designing business processes
- System integration
  - Putting together system components to create a new system
- System testing
  - The whole system is tested to discover problems
- System deployment
  - the process of making the system available to its users, transferring data from existing systems and establishing communications with other systems in the environment

Sommerville, Ian. "Software engineering 10th Edition." (2015).

Verification vs validation

- **Verification:**
  "Are we building the product right".
- The system should conform to its specification.

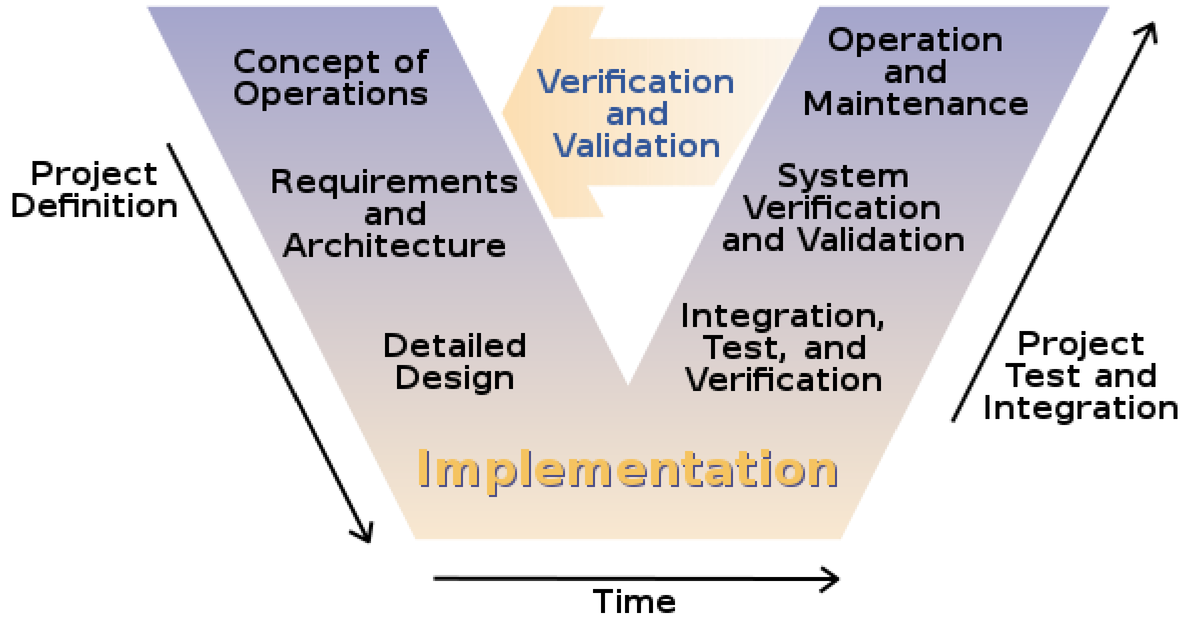
- **Validation:**
  "Are we building the right product".
- The system should do what the user really requires.



Input test data
I_e
Inputs causing anomalous behaviour
System
Output test results
O_e
Outputs which reveal the presence of defects

V-model

- Demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.
- V-model is a graphical representation of a systems development lifecycle.
- Left side represents requirements, and creation of system specifications.
- Right side represents integration of parts and their validation.



Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

https://en.wikipedia.org/wiki/V-Model

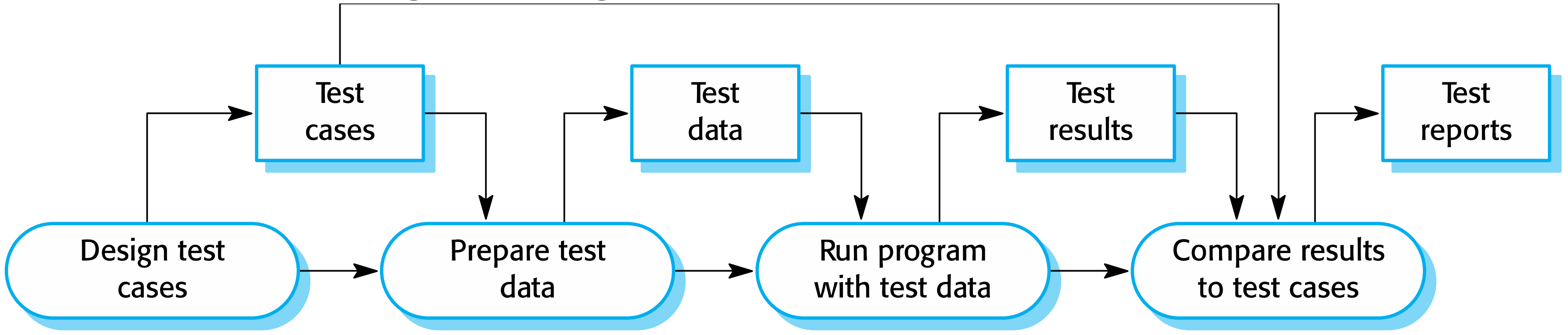
Verification vs validation

- Verification and validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it fulfills its intended purpose.
- “Verification. The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with *validation*.” PMBOK guide,
- “Validation. The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with *verification*.” PMBOK guide,

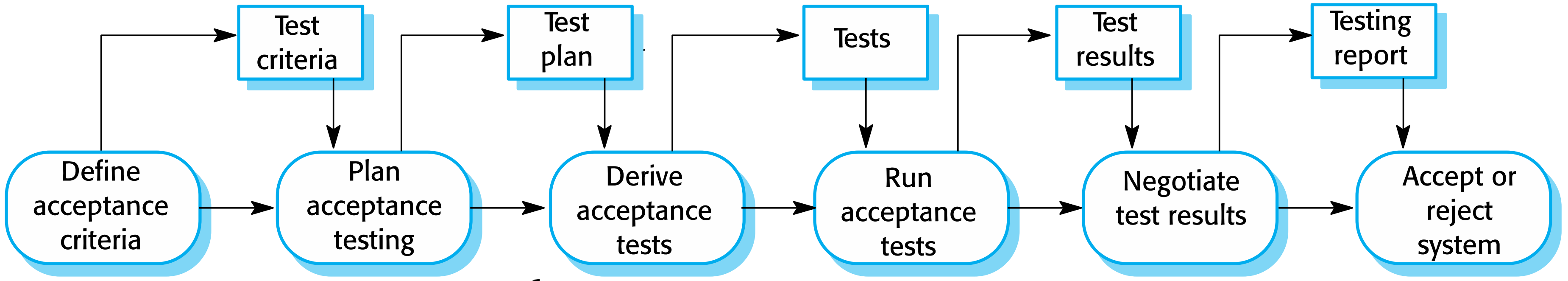
https://en.wikipedia.org/wiki/Verification_and_validation

System Testing

- evaluates the system's actual functionality in relation to expected or intended functionality, including all integration aspects.



An example of software testing process



The acceptance testing process

Sommerville, Ian. "Software engineering 10th Edition." (2015).

Inspections and testing

- Inspections and testing are complementary and not opposing verification techniques.
- Both should be used during the V & V process.
- Inspections can check conformance with a specification but not conformance with the customer's real requirements.
- Inspections cannot check non-functional characteristics such as performance, usability, etc.

Development testing

- Development testing includes all testing activities that are carried out by the team developing the system.
  - Unit testing, where individual program units or object classes are tested. Unit testing should focus on testing the functionality of objects or methods.
  - Component testing, where several individual units are integrated to create composite components. Component testing should focus on testing component interfaces.
  - System testing, where some or all of the components in a system are integrated and the system is tested as a whole. System testing should focus on testing component interactions.

Unit testing and Integration Testing

- Unit testing is the process of testing individual components in isolation.
- It is a defect testing process.
- Units may be:
  - Individual functions or methods within an object
  - Object classes with several attributes and methods
  - Composite components with defined interfaces used to access their functionality.
- Integration Test Plans are developed during the Architectural Design Phase.
- These tests verify that units created and tested independently can coexist and communicate among themselves.
- Test results are shared with customer's team.

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

Sommerville, Ian. "Software engineering 10th Edition." (2015).

Component testing

- System components are often composite components that are made up of several interacting objects.
  - For example, in the weather station system, the reconfiguration component includes objects that deal with each aspect of the reconfiguration.
- You access the functionality of these objects through the defined component interface.
- Testing composite components should therefore focus on showing that the component interface behaves according to its specification.
  - You can assume that unit tests on the individual objects within the component have been completed.

Interface testing

- Objectives are to detect faults due to interface errors or invalid assumptions about interfaces.
- Interface types
  - Parameter interfaces Data passed from one method or procedure to another.
  - Shared memory interfaces Block of memory is shared between procedures or functions.
  - Procedural interfaces Sub-system encapsulates a set of procedures to be called by other sub-systems.
  - Message passing interfaces Sub-systems request services from other sub-systems

System testing

- System testing during development involves integrating components to create a version of the system and then testing the integrated system.
- The focus in system testing is testing the interactions between components.
- System testing checks that components are compatible, interact correctly and transfer the right data at the right time across their interfaces.
- System testing tests the emergent behaviour of a system.

Regression testing

- The tests run every time a change is made to a system.
- Regression testing is testing the system to check that changes have not 'broken' previously working code.
- In a manual testing process, regression testing is expensive but, with automated testing, it is simple and straightforward. All tests are rerun every time a change is made to the program.
- Tests must run 'successfully' before the change is committed.

System Maintenance and Evolution

System operation

- Operational processes are the processes involved in using the system for its defined purpose.
- For new systems, these processes may have to be designed and tested and operators trained in the use of the system.
- Operational processes should be flexible to allow operators to cope with problems and periods of fluctuating workload.

System evolution

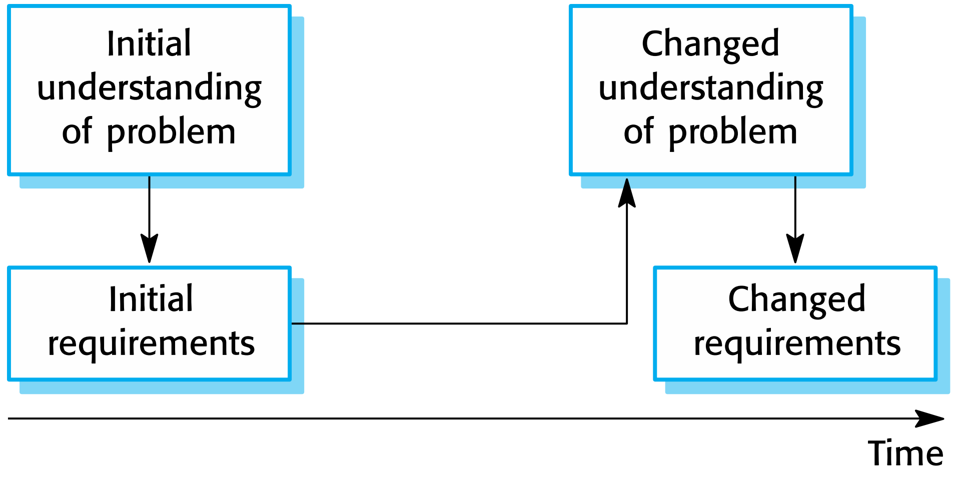
- Large systems have a long lifetime. They must evolve to meet changing requirements.
- Evolution is inherently costly
  - Changes must be analysed from a technical and business perspective;
  - Sub-systems interact so unanticipated problems can arise;
  - There is rarely a rationale for original design decisions;
  - System structure is corrupted as changes are made to it.
- Existing systems which must be maintained are sometimes called legacy systems or evolving systems.
- When a system is put into use, the operational processes and the system itself inevitably change to reflect changes to the business requirements and the system's environment.

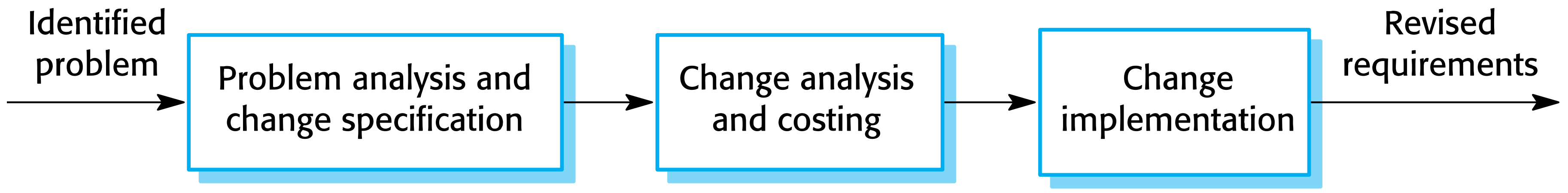
Cost factors in system evolution

- Proposed changes have to be analysed very carefully from a business and a technical perspective.
- Subsystems are never completely independent so changes to a subsystem may have side-effects that adversely affect other subsystems.
- Reasons for original design decisions are often unrecorded. Those responsible for the system evolution have to work out why these decisions were made.
- As systems age, their structure becomes corrupted by change so the costs of making further changes increases.

Requirements evolution and change management

- Deciding if a requirements change should be accepted
  - *Problem analysis and change specification*
  - *Change analysis and costing*
  - *Change implementation*


Initial understanding of problem
Changed understanding of problem
Initial requirements
Changed requirements
Time


Identified problem
Problem analysis and change specification
Change analysis and costing
Change implementation
Revised requirements

Regression Testing

- **Regression test suites** (that contains regression test cases) tend to grow with changes
- **Change impact analysis** is performed to determine an appropriate subset of tests
- Regression testing techniques:
- **Retest all:** Checks all the test cases to check its integrity. It is expensive as it needs to re-run all the cases, it ensures that there are no errors.
- **Regression test selection:** Runs a part of the test suite (owing to the cost of retest all), the cost of selecting the part of the test suite is less.
- **Test case prioritization:** Prioritize the test cases schedule, the test cases that are higher in priority are executed before the test cases that have a lower priority.

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

System of Systems Engineering (SoSE)

System of Systems Engineering (SoSE)

- Maier (1998) key characteristics of SoS:
  - operational independence, managerial independence,
  - geographical distribution, emergent behavior, and
  - evolutionary development processes

Systems tend to …	Systems of systems tend to …
Clear set of stakeholders	Multiple levels of stakeholders with mixed and possibly competing interests
Clear objectives and purpose	Multiple, and possibly contradictory, objectives and purpose
Clear operational priorities, with escalation to resolve priorities	Multiple, and sometimes different, operational priorities with no clear escalation routes
A single lifecycle	Multiple lifecycles with elements being implemented asynchronously
Clear ownership with the ability to move resources between elements	Multiple owners making independent resourcing decisions

https://www.sebokwiki.org/wiki/Systems_of_Systems_(SoS)

Systems of Systems (SoS)

ISO/IEC/IEEE 21839 (ISO, 2019)

- **System of Systems (SoS)** — Set of systems or system elements that interact to provide a unique capability that none of the constituent systems can accomplish on its own. Note: Systems elements can be necessary to facilitate the interaction of the constituent systems in the system of systems
- **Constituent Systems** — Constituent systems can be part of one or more SoS. Note: Each constituent is a useful system by itself, having its own development, management goals and resources, but interacts within the SoS to provide the unique capability of the SoS.

https://www.sebokwiki.org/wiki/Systems_of_Systems_(SoS)

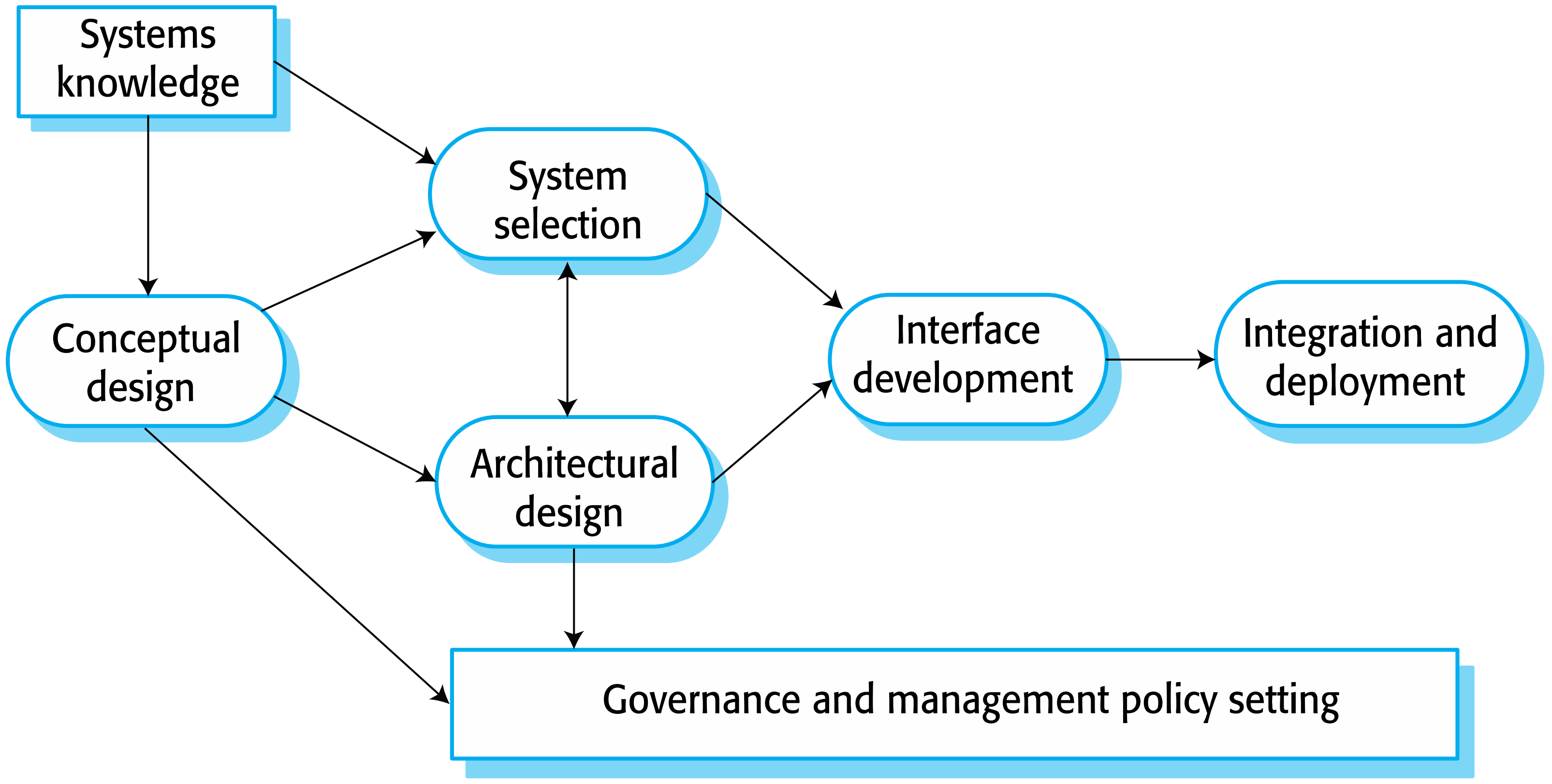
System of Systems Engineering (SoSE)

- Systems of systems are systems where two or more of the constituent systems are independently managed and governed.
- Reductionism as an engineering method breaks down because of the inherent complexity of systems of systems.
- Reductionism assumes clear system boundaries, rational decision making and well-defined problems. None of these are true for systems of systems.
- Governance and management policies must be designed in parallel.

System of Systems Engineering (SoSE)

- Three types of complexity in systems of systems –
  - technical complexity,
  - managerial complexity and
  - governance complexity.
- The key stages of the SoS development process are
  - conceptual design,
  - system selection,
  - architectural design,
  - interface development, and
  - integration and deployment.

SoSE development processes



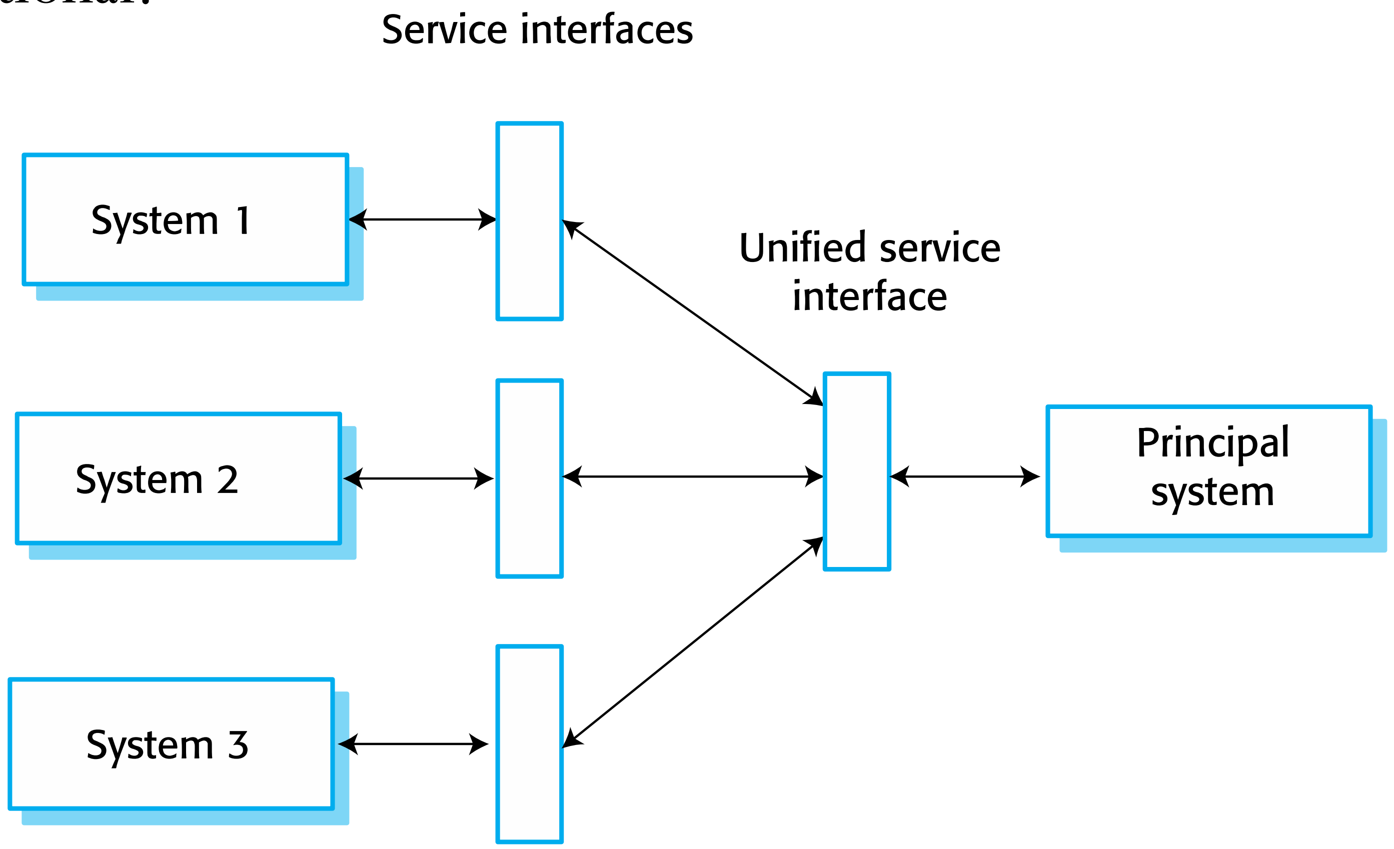
Sommerville, Ian. "Software engineering 10th Edition." (2015).

SoSE development processes

- *Conceptual design* is the activity of creating a high-level vision for a system, defining essential requirements and identifying constraints on the overall system.
- *System selection*, where a set of systems for inclusion in the SoS is chosen.
  - Political imperatives and issues of system governance and management are often the key factors that influence what systems are included in a SoS.
- *Architectural design* where an overall architecture for the SoS is developed.
- *Interface development* – the development of system interfaces so that the constituent systems can interoperate.
- *Integration and deployment* – making the different systems involved in the SoS work together and interoperate through the developed interfaces.

SoSE development processes

- System deployment means putting the system into place in the organizations concerned and making it operational.

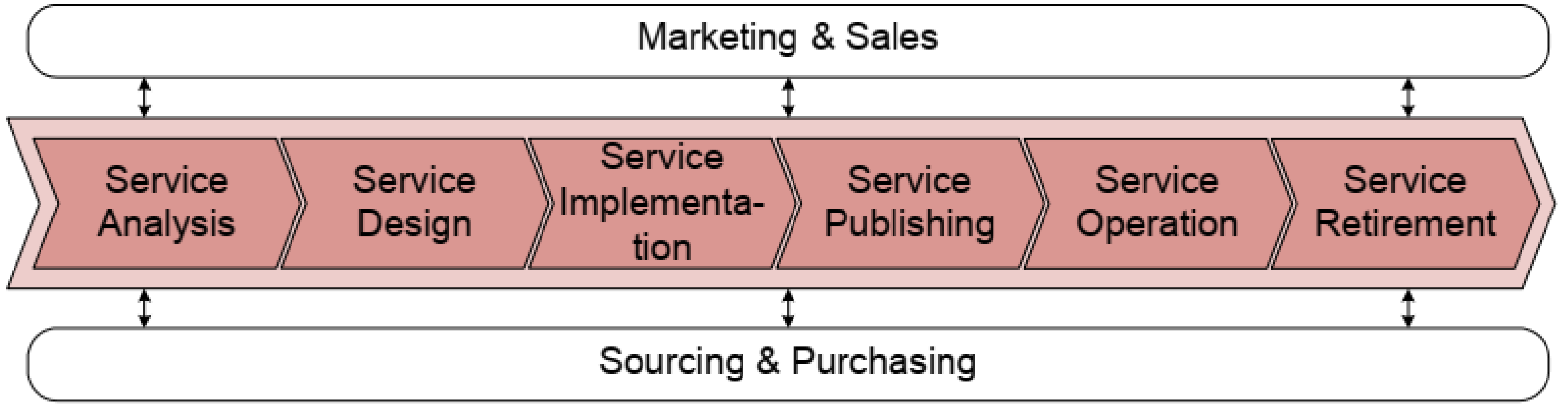


Sommerville, Ian. "Software engineering 10th Edition." (2015).

SDLC example (Cloud Service Life Cycle)

Cloud Service Life Cycle

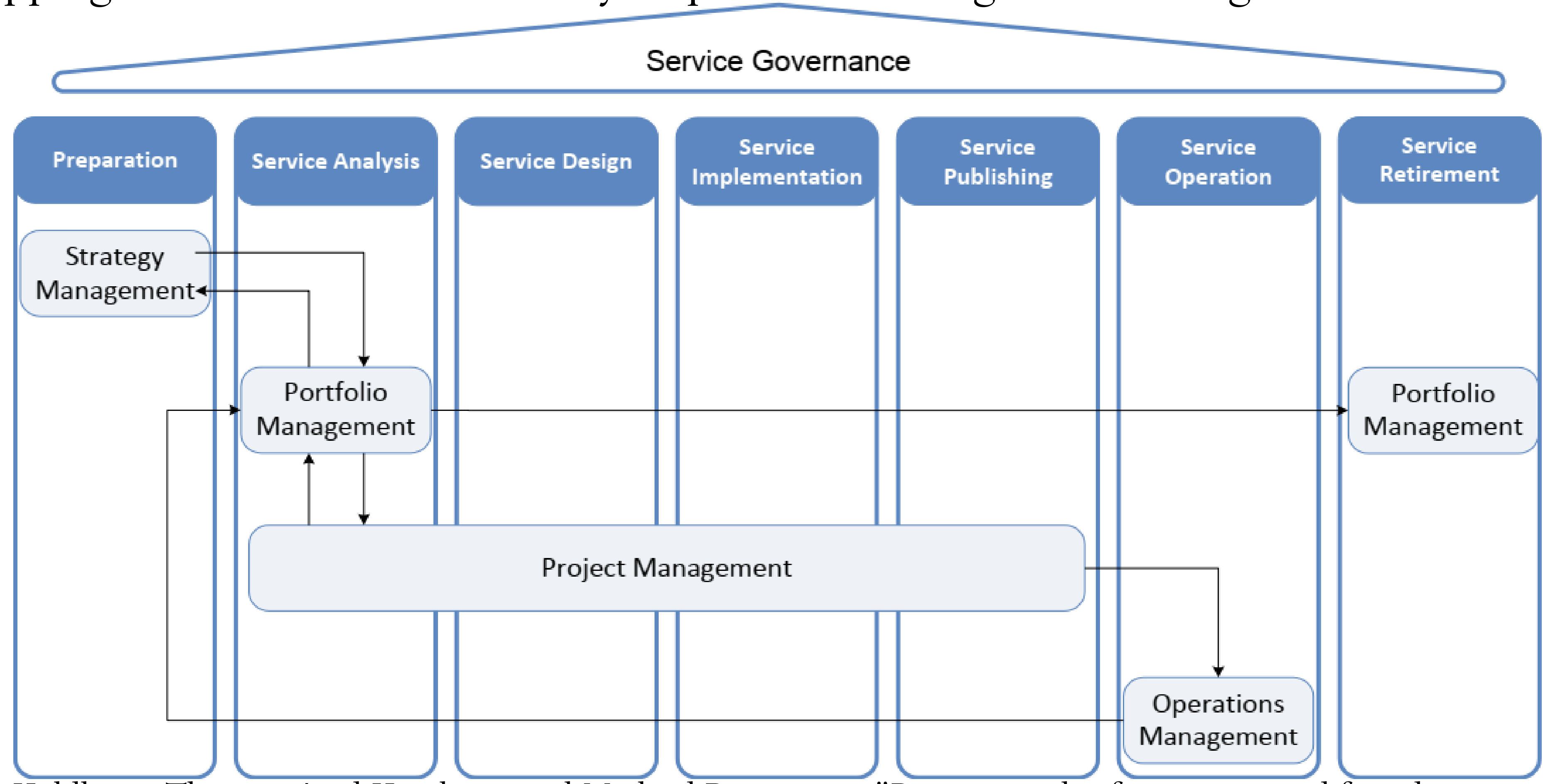
- Service Analysis: Identification and contextualisation - market requirements
- Service Design: specification for development and reuse,
- Service Implementation: either software with technical service characteristics
- Service Publishing: dissemination of the service,
- Service Operation: monitored for contract management,
- Service Retirement: reached end of economic or technical competitiveness,



Kohlborn, Thomas, Axel Korthaus, and Michael Rosemann. "Business and software service lifecycle management." *2009 IEEE International Enterprise Distributed Object Computing Conference*. IEEE, 2009.

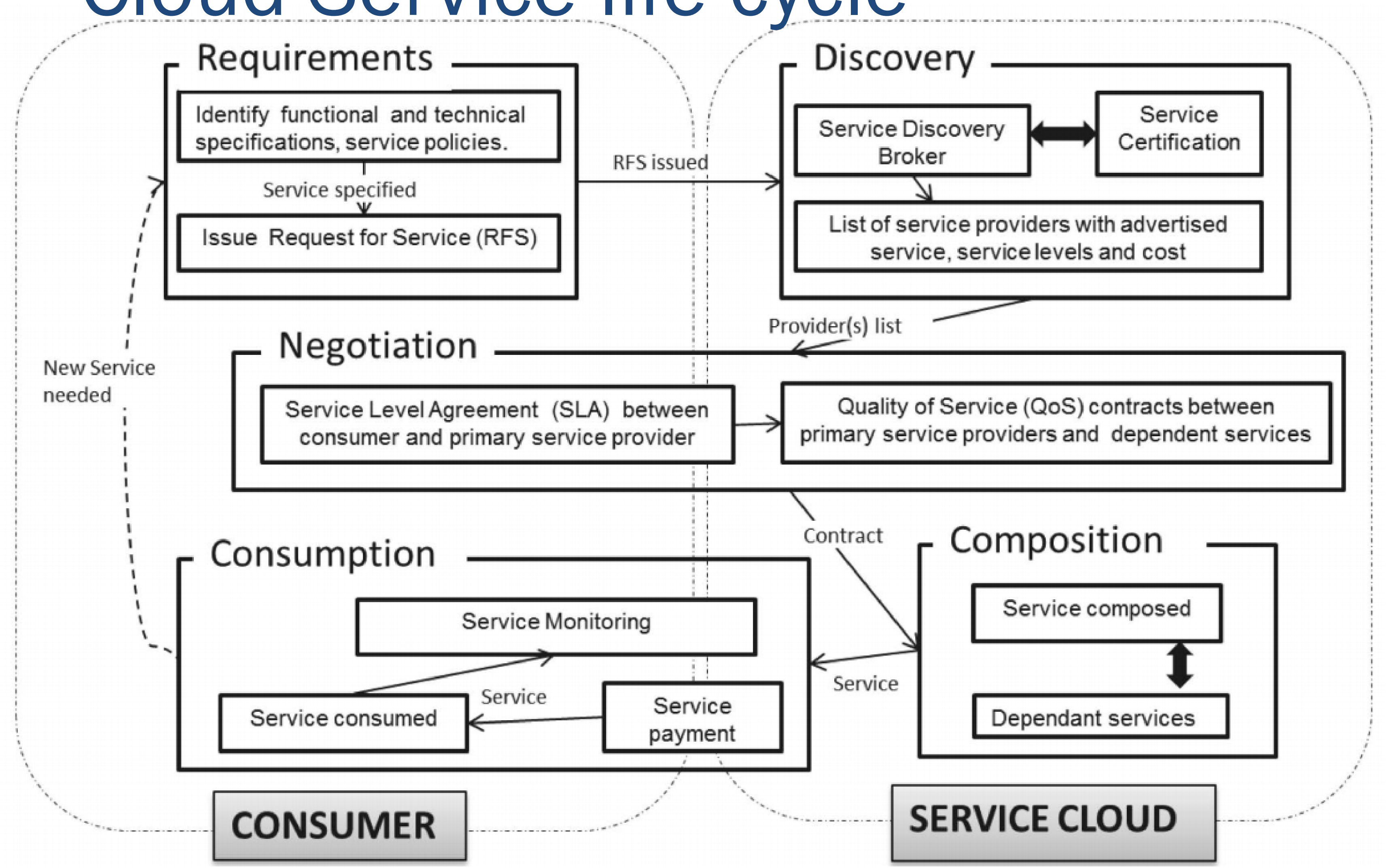
Cloud Service Life Cycle

- Mapping between the service lifecycle phases and the generic management



Kohlborn, Thomas, Axel Korthaus, and Michael Rosemann. "Business and software service lifecycle management." *2009 IEEE International Enterprise Distributed Object Computing Conference*. IEEE, 2009.

Cloud Service life cycle


Requirements
Identify functional and technical specifications, service policies.
Service specified
Issue Request for Service (RFS)
RFS issued
Discovery
Service Discovery Broker
Service Certification
List of service providers with advertised service, service levels and cost
Provider(s) list
Negotiation
Service Level Agreement (SLA) between consumer and primary service provider
Quality of Service (QoS) contracts between primary service providers and dependent services
Contract
Composition
Service composed
Dependant services
Service
Consumption
Service Monitoring
Service consumed
Service
Service payment
New Service needed
CONSUMER
SERVICE CLOUD

Joshi, Karuna P., Yelena Yesha, and Tim Finin. "Automating cloud services life cycle through semantic technologies." *IEEE Transactions on Services Computing* 7.1 (2012): 109-122.

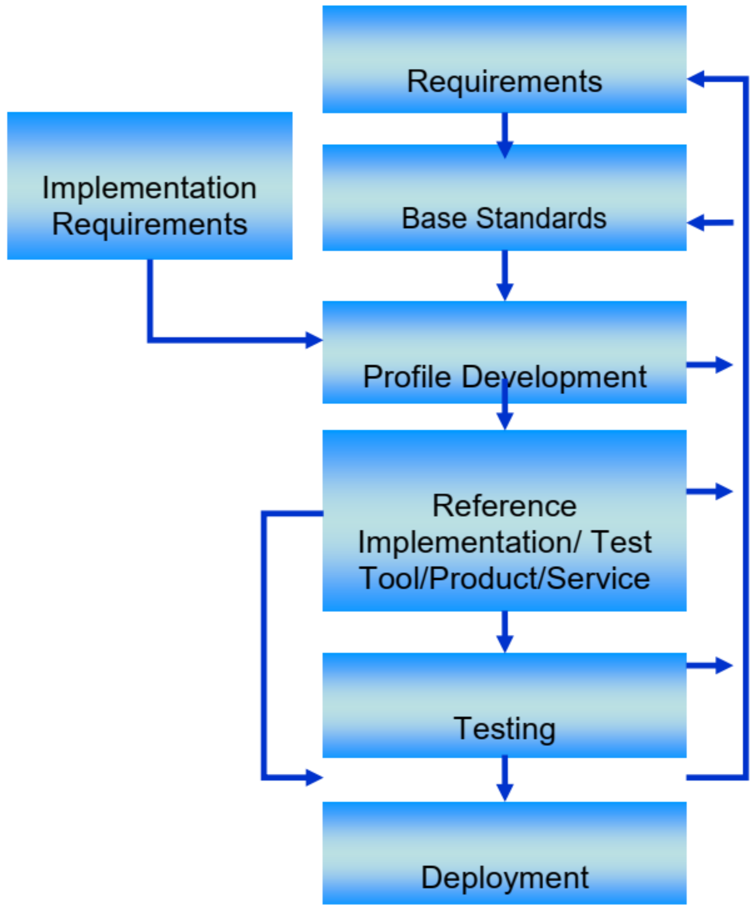
Cloud Service life cycle

- The processes and data flow of the five phases:
  - Requirements,
  - Discovery,
  - Negotiation,
  - Composition, and
  - Consumption
- Represent the concepts and relationships for each phase

Joshi, Karuna P., Yelena Yesha, and Tim Finin. "Automating cloud services life cycle through semantic technologies." *IEEE Transactions on Services Computing* 7.1 (2012): 109-122.

IT Standards Life Cycle

- High-level conceptualization of IT standards with development, products, processes, and services are deployed processes can occur concurrently


Requirements
Implementation Requirements
Base Standards
Profile Development
Reference Implementation/ Test Tool/Product/Service
Testing
Deployment

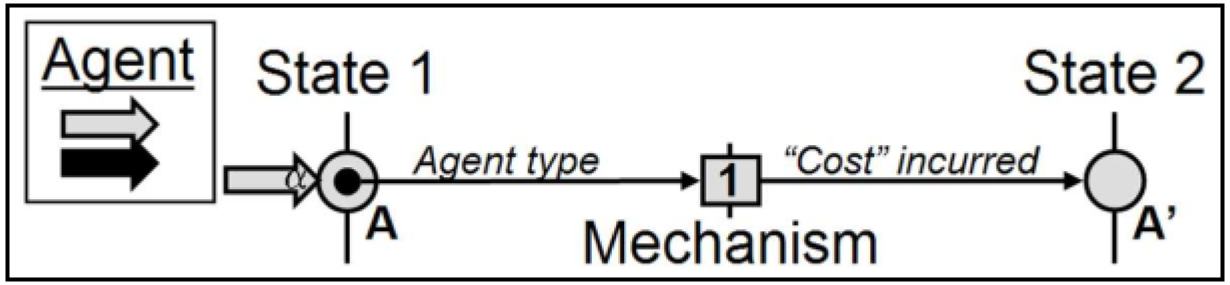
Sokol, Annie W., and Michael D. Hogan. *NIST Cloud Computing Standards Roadmap*. No. Special Publication (NIST SP)-500-291r2. 2013.

System “ilities”
(Reliability, Availability, Maintainability, & Changeability)

Emergent properties

- The emergent properties of a system are characteristics of the system as a whole rather than of its component parts.
- They include properties such as
  - performance,
  - reliability,
  - usability,
  - safety,
  - security,
  - changeability etc

Changeability



- System "changeability" (Ross et al.) define it as system ilities
- **Change Agent:** Instigator, or force, which employs a given change mechanism in order to achieve a desired change effect
- **Change Effect:** The difference in system states (performance or value) before and after a change has taken place
- **Change Objective:** The specific approach / plan / goal / strategies employed to achieve a desired change effect
- **Change Enablers:** (e.g. design elements) enable desired objective
- **Change Considerations:** Design considerations (e.g. conditions, resources, constraints, etc.) applied to design / operational approaches

Ross, A.M., Rhodes, D.H., and Hastings, D.E., "Defining Changeability: Reconciling Flexibility, Adaptability, Scalability, Modifiability, and Robustness for Maintaining Lifecycle Value," *Systems Engineering*, Vol. 11, No. 3, pp. 246-262, Fall 2008.

System "ilities"

- ☐ Accessibility
- ☐ Accountability
- ☐ Adaptability
- ☐ Administrability
- ☐ Affordability
- ☐ Auditability
- ☐ Availability
- ☐ Credibility
- ☐ Compatibility
- ☐ Configurability
- ☐ Correctness
- ☐ Customizability
- ☐ Debugability
- ☐ Degradability
- ☐ Determinability
- ☐ Demonstrability
- ☐ Dependability
- ☐ Deployability
- ☐ Distributability
- ☐ Durability
- ☐ Effectiveness
- ☐ Evolvability
- ☐ Extensibility
- ☐ Fidelity
- ☐ Flexibility
- ☐ Installability
- ☐ Integrity
- ☐ Interchangeability
- ☐ Interoperability
- ☐ Learnability
- ☐ Maintainability
- ☐ Manageability
- ☐ Mobility
- ☐ Modifiability
- ☐ Modularity
- ☐ Operability
- ☐ Portability
- ☐ Predictability
- ☐ Provability
- ☐ Recoverability
- ☐ Reliability
- ☐ Repeatability
- ☐ Reproducibility
- ☐ Resilience
- ☐ Responsiveness
- ☐ Reusability
- ☐ Robustness
- ☐ Safety
- ☐ Scalability
- ☐ Sustainability
- ☐ Serviceability
- ☐ Supportability
- ☐ Securability
- ☐ Simplicity
- ☐ Stability
- ☐ Survivability
- ☐ Sustainability
- ☐ Tailorability
- ☐ Testability
- ☐ Timeliness
- ☐ Traceability
- ☐ Ubiquity
- ☐ Understandability
- ☐ Upgradability
- ☐ Usability
- ☐ Versatility

Examples of emergent properties

Property	Description
Reliability	System reliability depends on component reliability but unexpected interactions can cause new types of failures and therefore affect the reliability of the system.
Repairability	This property reflects how easy it is to fix a problem with the system once it has been discovered. It depends on being able to diagnose the problem, access the components that are faulty, and modify or replace these components.
Security	The security of the system (its ability to resist attack) is a complex property that cannot be easily measured. Attacks may be devised that were not anticipated by the system designers and so may defeat built-in safeguards.
Usability	This property reflects how easy it is to use the system. It depends on the technical system components, its operators, and its operating environment.
Volume	The volume of a system (the total space occupied) varies depending on how the component assemblies are arranged and connected.

Sommerville, Ian. "Software engineering 10th Edition." (2015).

Reliability, Availability, & Maintainability (RAM)

- RAM are system attributes that affect system life-cycle costs
- RAM are interest to engineers, logisticians, and users.
  - Systems Engineering, Product Management (Life Cycle Cost and Warranty), Quality Assurance
- **Reliability:** measures probability of a system or its component performing their expected function for the requirements without failure for a defined period of time.
- **Maintainability:** measures probability of a system or its component can be repaired in a defined environment within a specified period of time. Increased maintainability implies shorter repair times
- **Availability:** the probability of a repairable system or its component is operational at a given point in time under a given set of environmental conditions. Availability depends on reliability and maintainability

https://www.sebokwiki.org/wiki/Reliability,_Availability,_and_Maintainability

Reliability

- Collectively, they affect economic life-cycle costs of a system and its utility.
- Description of the function, the environment, the time scale, and what constitutes a failure.
- Emergent property: because of component inter-dependencies, faults can be propagated through the system.
- System failures often occur because of unforeseen inter-relationships between components.
- It is practically impossible to anticipate all possible component relationships.

https://www.sebokwiki.org/wiki/Reliability,_Availability,_and_Maintainability

Availability

- Availability can be calculated from the total operating time and the downtime, or in the alternative, as a function of MTBF and MTTR (Mean Time To Repair.)
  - $T_{op,Tot}$ is the total operating time
  - $T_{down,tot}$ is the total down time
  - Mean Time To Failure (MTTF) for non-restorable systems, or Mean Time Between Failures (MTBF for restorable systems are used)
  - MTTR (Mean Time To Repair.)

$$A = \frac{T_{op,Tot}}{T_{down,tot} + T_{op,tot}} = \frac{MTBF}{MTBF + MTTR}$$

https://www.sebokwiki.org/wiki/Reliability,_Availability,_and_Maintainability

Maintainability

- correct defects or their cause, repair or replace faulty components,
- prevent unexpected working conditions, maximize a product's useful life,
- maximize efficiency, reliability, and safety, meet new requirements,
- make future maintenance easier, or cope with a changing environment.
- involves continuous improvement of a system
- Maintainability is often characterized in terms of the exponential distribution and the mean time to repair and be similarly calculated, i.e.,
  - where $T_{down,Tot}$ is the total down time and $n_{outages}$ is the number of outages.

$$MTTR = \frac{T_{down,Tot}}{n_{outages}}$$

https://en.wikipedia.org/wiki/Maintainability
https://www.sebokwiki.org/wiki/Reliability,_Availability,_and_Maintainability

ขอบคุณ

Thai

Grazie

Italian

תודה רבה

Hebrew

धन्यवादः

Sanskrit

ಧನ್ಯವಾದಗಳು

Kannada

Thank You

English

Ευχαριστώ

Greek

Gracias

Spanish

Спасибо

Russian

Obrigado

Portuguese

شكراً

Arabic

https://sites.google.com/site/animeshchaturvedi07

Merci

French

多謝

Traditional Chinese

धन्यवाद

Hindi

Danke

German

多谢

Simplified Chinese

நன்றி

Tamil

Tamil

ありがとうございました

Japanese

감사합니다

Korean