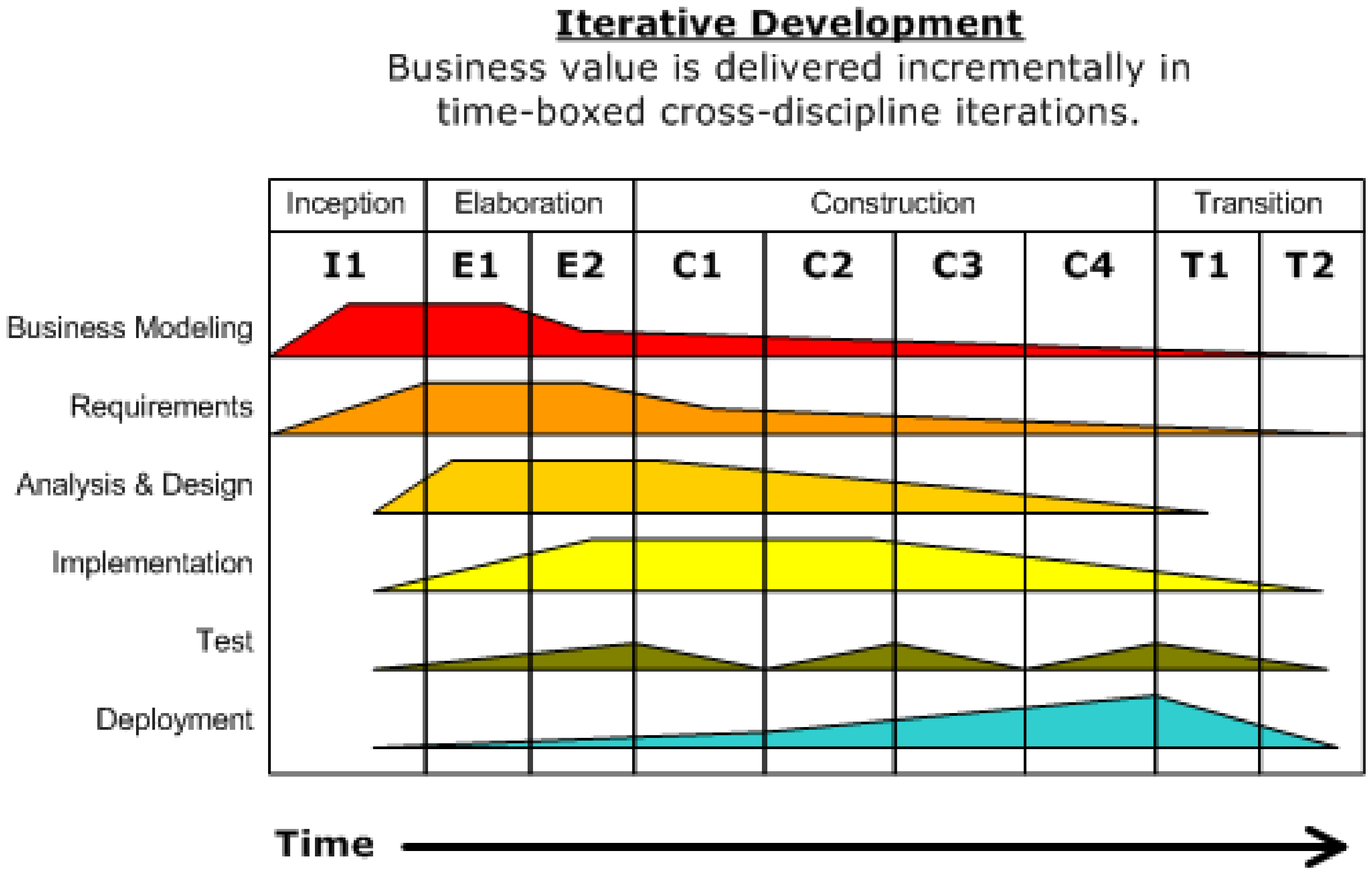
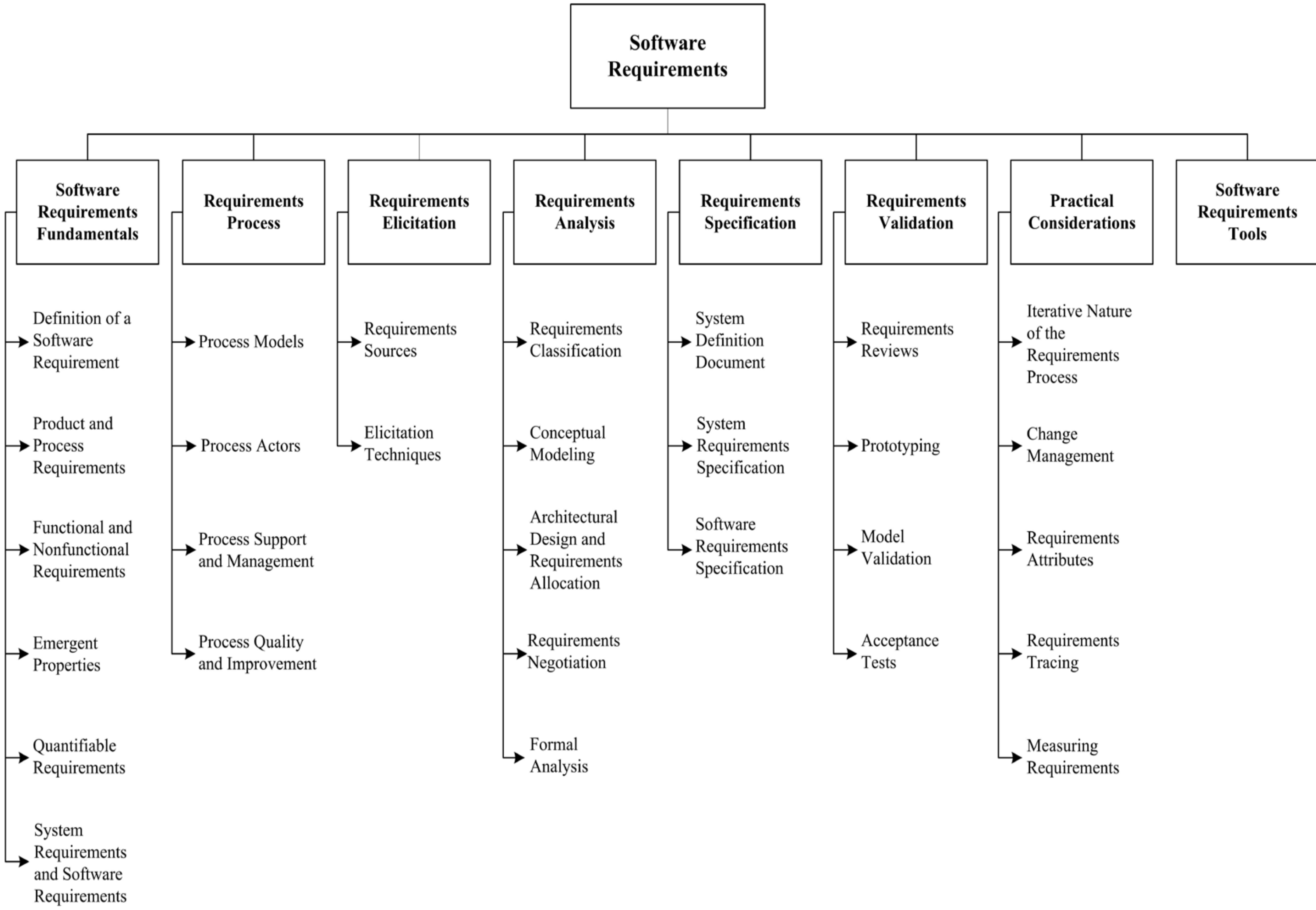
SE
Software Requirements

Karl Wiegers and Joy Beatty. Software Requirements, Third Edition. PUBLISHED BY Microsoft Press. ISBN: 978-0-7356-7966-5
https://www.academia.edu/31604577/Karl_Wiegers_Joy_Beatty_Software_Requirements?auto=download

RUP Model


Iterative Development
Business value is delivered incrementally in time-boxed cross-discipline iterations.
Inception
Elaboration
Construction
Transition
I1
E1
E2
C1
C2
C3
C4
T1
T2
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Time


Software Requirements
Software Requirements Fundamentals
Definition of a Software Requirement
Product and Process Requirements
Functional and Nonfunctional Requirements
Emergent Properties
Quantifiable Requirements
System Requirements and Software Requirements
Requirements Process
Process Models
Process Actors
Process Support and Management
Process Quality and Improvement
Requirements Elicitation
Requirements Sources
Elicitation Techniques
Requirements Analysis
Requirements Classification
Conceptual Modeling
Architectural Design and Requirements Allocation
Requirements Negotiation
Formal Analysis
Requirements Specification
System Definition Document
System Requirements Specification
Software Requirements Specification
Requirements Validation
Requirements Reviews
Prototyping
Model Validation
Acceptance Tests
Practical Considerations
Iterative Nature of the Requirements Process
Change Management
Requirements Attributes
Requirements Tracing
Measuring Requirements
Software Requirements Tools

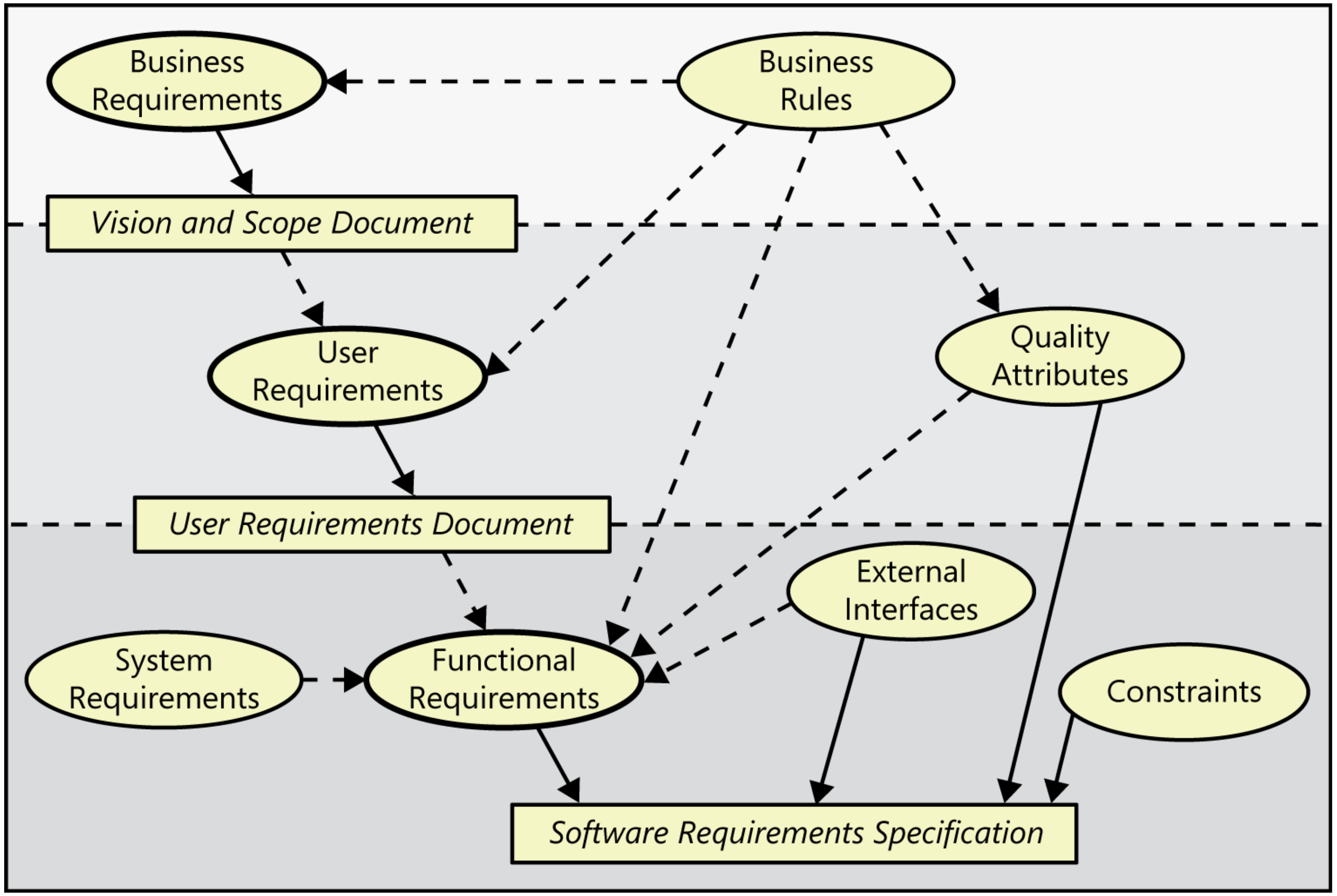
Software requirements

- Requirements are:
  - a specification of what should be implemented
  - descriptions of how the system should behave
  - descriptions of a system property or attribute
  - a constraint on the development process of the system
- Requirements include
  - **both** the **user's view** of the external system behavior
    and the **developer's view** of some internal characteristics
  - **both** the behavior of the system under **specific** conditions
    and those properties that make the system suitable for use by its **intended** operators

Types of requirements

Term	Definition
Business requirement	A high-level business objective of the organization that builds a product or of a customer who procures it.
Business rule	A policy, guideline, standard, or regulation that defines or constrains some aspect of the business. Not a software requirement in itself, but the **origin** of several types of software requirements.
Constraint	A restriction that is imposed on the choices available to the developer for the design and construction of a product.
External interface requirement	A description of a connection between a software system and a user, another software system, or a hardware device.
Feature	One or more logically related system **capabilities** that **provide value** to a user and are described by a set of functional requirements.
Functional requirement	A description of a **behavior** that a system will exhibit under specific conditions.
Nonfunctional requirement	A description of a **property or characteristic** that a system must exhibit or a constraint that it must respect.
Quality attribute	A kind of nonfunctional requirement that describes a service or performance characteristic of a product.
System requirement	A top-level requirement for a product that contains multiple subsystems, which could be all software or software and hardware.
User requirement	A goal or task that **specific classes of users** must be able to perform with a system, or a desired product attribute.

Relationships among several types of requirements



Solid arrows mean "are stored in"
Dotted arrows mean "are the origin of" or "influence."

Levels of requirements

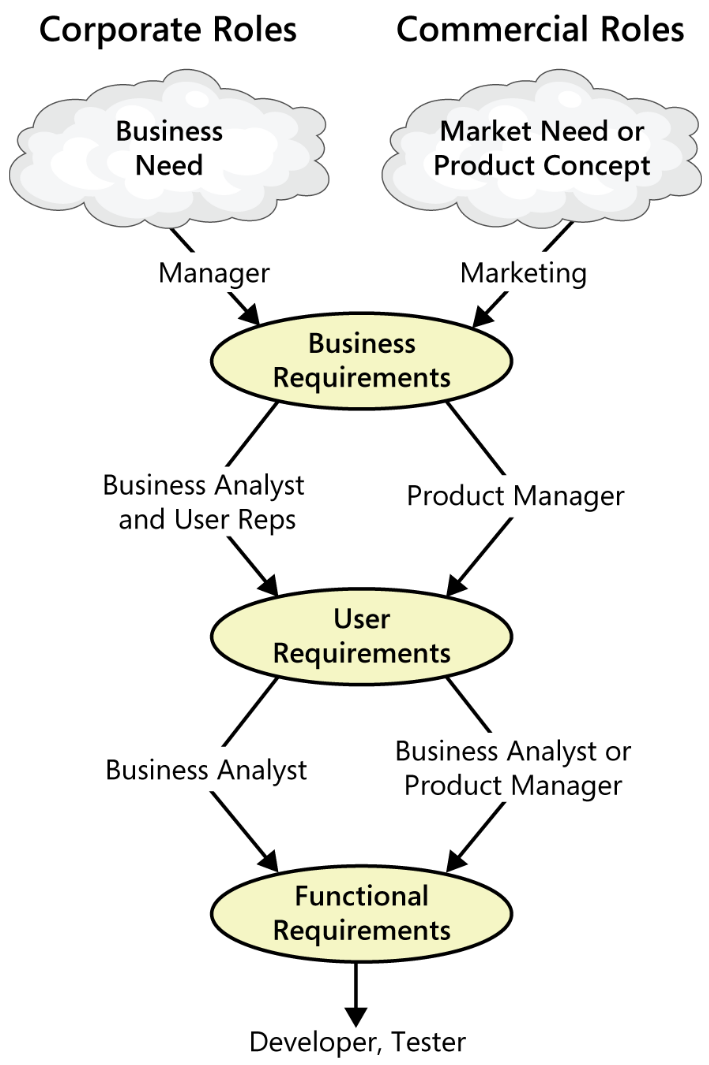
- *Business requirements* describe ***why*** the organization is implementing the system—the business benefits the organization hopes to achieve. The focus is on the business objectives of the organization or the customer who requests the system.

- *User (*stakeholder) *requirements* describe ***what*** the user will be able to do with the system:
  - describe goals or tasks the users must be able to perform with the product that will provide value to someone;
  - the domain of user requirements also includes descriptions of product attributes or characteristics that are important to user satisfaction;
  - include use cases, user stories, and event-response tables.

- *Functional requirements* specify the **behaviors** the product will exhibit under specific conditions. They describe **what** the developers must implement to enable users to accomplish their tasks (user requirements), thereby satisfying the business requirements:
  - alignment among the three levels of requirements is essential for project success;
  - often are written in the form of the traditional "shall" ;
  - business analyst (**BA**) documents functional requirements in a *software requirements specification* (**SRS**), which describes **as fully as necessary** the expected behavior of the software system;
  - SRS is used in development, testing, quality assurance, project management, and related project functions.

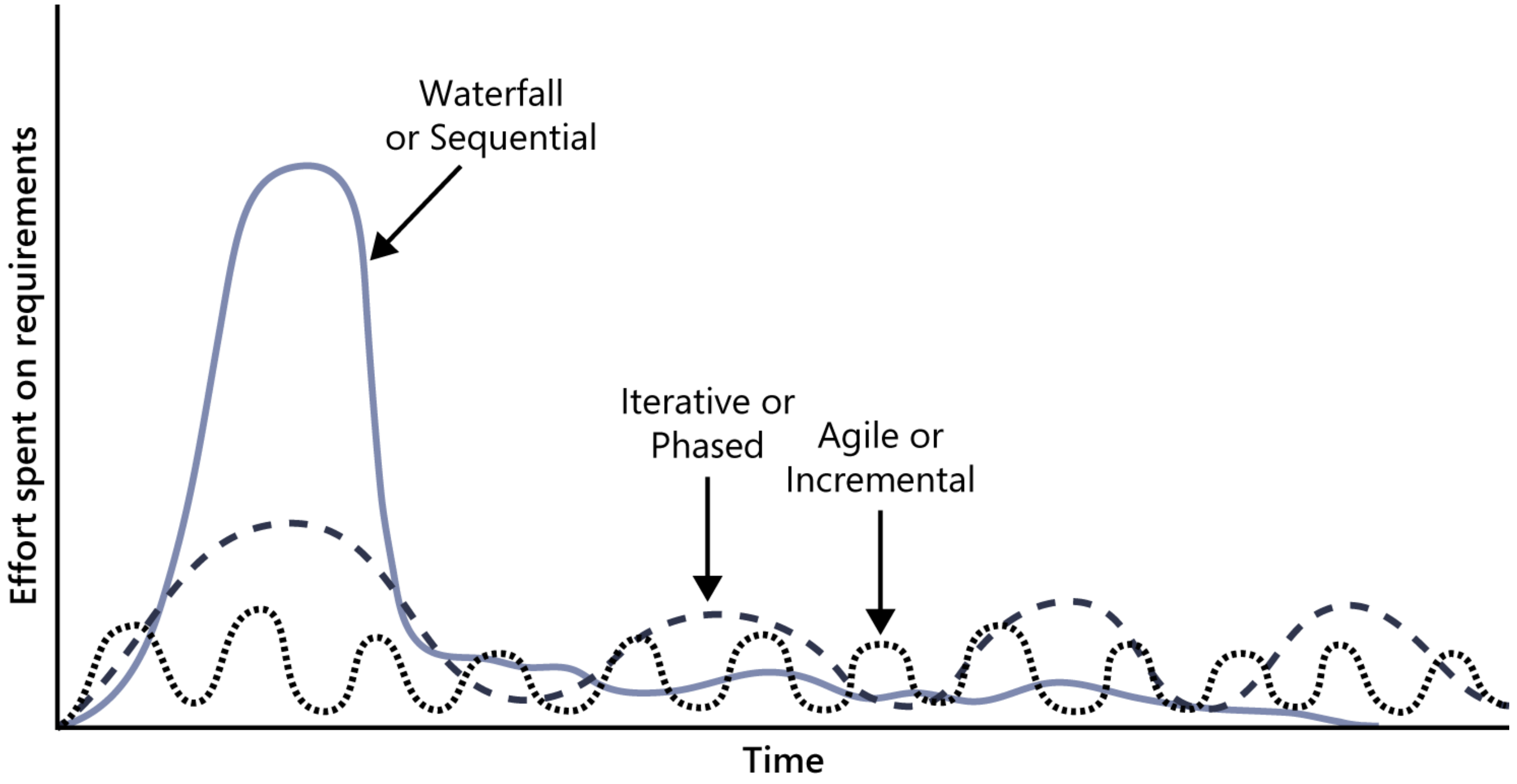
- *System requirements* describe the requirements for a product that is composed of multiple components or subsystems. A system can be all software or it can include both software and hardware subsystems.

Stakeholders participating in requirement elicitation

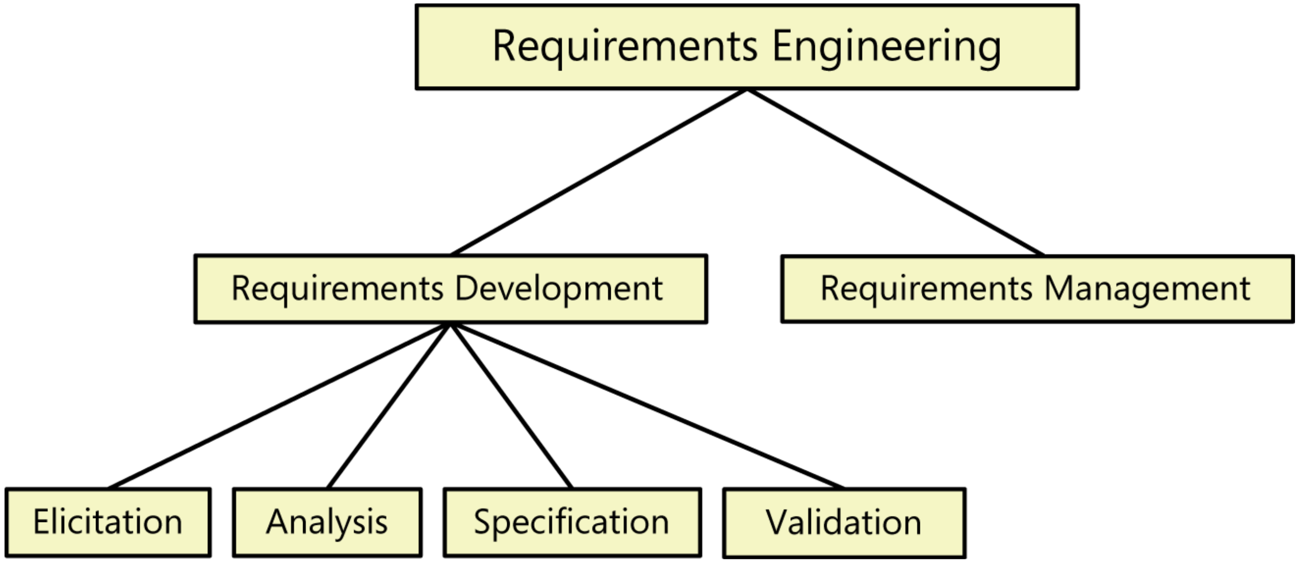


- Requirements, hosted in SRS that describe properties of a software system to be built, are **product requirements**
- **Project requirements** include:
  - design or implementation details, project plans, test plans, or similar information
  - physical resources the development team needs
  - staff training needs
  - user documentation, including training materials, tutorials, reference manuals, and release notes

The distribution of requirements development effort in different development life cycles



Requirements development and management



Requirements elicitation

- **Elicitation** -- gathering and discovery of the system requirements from users, customers and **other stakeholders**.
- Elicitation encompasses **all** of the activities involved with discovering requirements, such as interviews, workshops, document analysis, prototyping, and others.
- Usage-centric and product-centric strategies.
- The key actions :
  - Identifying user classes and other stakeholders.
  - Understanding user tasks and goals and the business objectives with which those tasks align.
  - Learning about the environment in which the new product will be used.
  - Working with individuals who represent each user class to understand their functionality needs and their quality expectations

Practices of elicitation

- interviews
- questionnaires
- user observation
- use cases
- workshops
- brainstorming
- role playing
- prototyping

Problem of elicitation

- **Problems of scope**
  - boundary of the system is ill-defined
  - customers/users specify unnecessary technical detail that may confuse overall system objectives
- **Problems of understanding**
  - customers/users are not completely sure of **what is needed**
  - poor understanding of the capabilities and limitations of computing environment
  - don’t have a full understanding of the **problem domain**
  - omit information that is believed to be “obvious”
  - specify requirements that conflict with the needs of other customers/users
  - specify requirements that are ambiguous or untestable
- **Problems of volatility**
  - requirements change over time
  - level of requirement volatility

Analysis

- Analyzing requirements involves :
  - reaching a richer and more precise understanding of each requirement
  - representing sets of requirements in **multiple** ways

- Principal activities:
  - Analyzing the information received from users to distinguish their task goals from functional requirements, quality expectations, business rules, suggested solutions, and other information
  - Decomposing high-level requirements into an appropriate level of detail
  - Deriving functional requirements from other requirements information
  - Understanding the relative importance of quality attributes
  - Allocating requirements to software components defined in the **system architecture**
  - Negotiating **implementation priorities**
  - Identifying gaps in requirements or unnecessary requirements as they relate to the defined scope
  - development of **models**: use case diagrams, data flow, state, goal-based …
  - determining whether the stated requirements are clear, complete, consistent and unambiguous, and resolving any apparent **conflicts**
  - discover **bounds** of software and how it must interact with its organizational and operational environment
  - representing sets of requirements in multiple, complementary ways

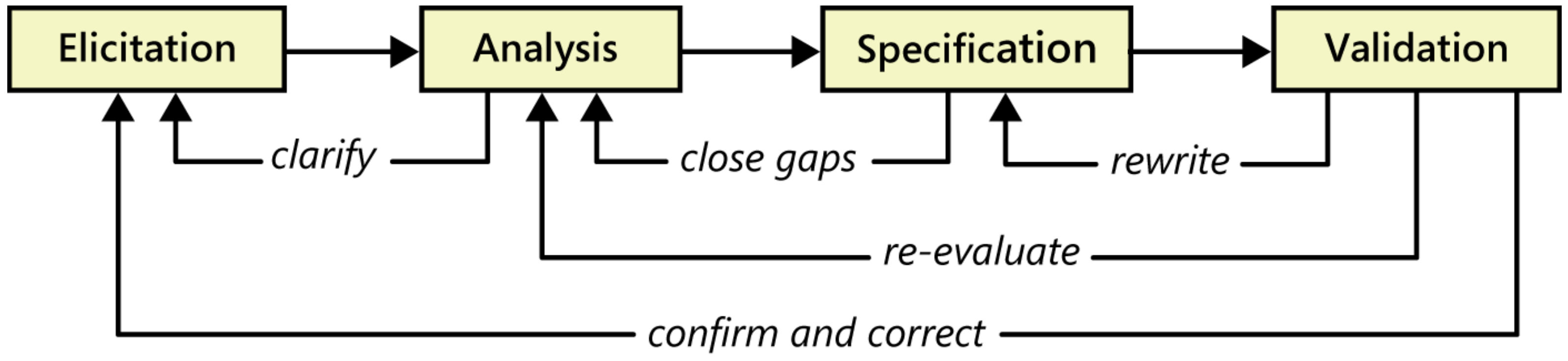
Specification

- **Specification** (SRS) -- output of the requirements analysis process, representing and storing the collected requirements knowledge in a persistent and well-organized fashion
  - system definition
  - system requirements
  - software requirements
- Translating the collected user needs into **written** requirements and diagrams **suitable** for comprehension, review, and use by their intended audiences.

Validation

- The goal of requirements development is to accumulate a shared understanding of requirements:
  - that is **good enough** (**not perfect!**) to allow construction of the next portion of the product
  - development can be proceeded at an **acceptable** level of risk
  - avoid the danger of " paralysis from analysis"
- **Validation** -- involves techniques to confirm that the **correct** set of requirements has been specified to build a solution that **satisfies** the project's business objectives
  - requirements reviews to correct any problems before the development group accepts them
  - prototyping
  - model validation
  - developing **acceptance tests** and **criteria** to confirm that a product based on the requirements would meet customer needs and achieve the business objectives

A requirements development process framework



Requirements management activities

- Defining the requirements baseline, a snapshot in time that represents an agreed-upon, reviewed, and approved set of functional and nonfunctional requirements, often for a specific product **release** or development **iteration**
- Evaluating the impact of proposed requirements changes and incorporating approved changes into the project in **a controlled way**
- Keeping project plans current with the requirements as they evolve
- Negotiating new commitments based on the estimated impact of requirements changes
- Defining the relationships and dependencies that exist between requirements
- **Tracing** individual requirements to their corresponding designs, source code, and tests
- Tracking requirements **status** and change activity throughout the project

Requirements Bill of Rights for Software Customers

1. To expect BAs to speak your language
2. To expect BAs to learn about your business and your objectives
3. To expect BAs to record requirements in an appropriate form
4. To receive explanations of requirements practices and deliverables
5. **To change your requirements**
6. To expect an environment of **mutual respect**
7. To hear ideas and alternatives for your requirements and for their solution
8. To describe characteristics that will make the product easy to use
9. To hear about ways to adjust requirements to accelerate development through reuse
10. !!! To receive a system that meets your functional needs and quality expectations

Requirements Bill of Responsibilities for Software Customers

1. To educate BAs and developers about your business
2. To dedicate the time that it takes to provide and clarify requirements
3. To be **specific and precise** when providing input about requirements
4. To make timely decisions about requirements when asked
5. To respect a developer's assessment of the cost and feasibility of requirements
6. To set **realistic requirement priorities** in collaboration with developers
7. To review requirements and evaluate prototypes
8. To establish acceptance criteria
9. To promptly communicate changes to the requirements
10. To respect the requirements development process

Good practices (GP): Requirements elicitation 1

- **Define product vision and project scope**
  - vision statement gives all stakeholders a common understanding of the product's outcome
  - scope defines the boundary between **what's in** and **what's out** for a specific release or iteration
- **Identify user classes and their characteristics**
  - they might differ in frequency of use, features used, privilege levels, or experience
  - describe their job tasks, attitudes, or personal characteristics that might influence product design
  - create user personas, who will represent particular user classes
- **Select a product champion for each user class**
  - identify an individual who can accurately serve as the literal voice of the customer for each user class
  - product champion presents the needs of the user class and makes decisions on its behalf

GP: Requirements elicitation 2

- **Distribute questionnaires**
  - a way to survey large groups of users to determine analytical information about needs
  - are useful with any large user population but are particularly helpful with distributed groups
- **Perform document analysis**
  - can help reveal how systems **currently** work or what they are supposed to do
  - includes any written information about current systems, business processes, requirements specifications
- **Examine problem reports of current systems for requirement ideas**
  - provide **a rich source** of ideas for capabilities to include in a later release or in a new product
  - help desk and support staff can provide valuable input into the requirements for future development work

GP: Requirements elicitation 3

- **Identify system events and its expected response to each event**
  - signal events are control signals or data received from external hardware devices
  - temporal, or time-based, events trigger a response, such as an external data feed
  - business events trigger use cases in business applications
- **Hold elicitation interviews and workshops**
  - Interviews one-on-one or with a small group of stakeholders for discussing only the specific requirements that are important to them
  - workshops permit collaboration between analysts and customers
  - a powerful way to draft requirements documents
- **Observe users performing their jobs**
  - establishing a **context** for potential use of a new application
  - **flow diagrams** can depict the steps and decisions involved and show how different user groups interact

GP: Requirements elicitation 4

- **Distribute questionnaires**
  - a way to survey large groups of users to determine analytical information about needs
  - are useful with any large user population but are particularly helpful with distributed groups
- **Perform document analysis**
  - can help reveal how systems **currently** work or what they are supposed to do
  - includes any written information about current systems, business processes, requirements specifications
- **Examine problem reports of current systems for requirement ideas**
  - provide **a rich source** of ideas for capabilities to include in a later release or in a new product
  - help desk and support staff can provide valuable input into the requirements for future development work

GP: Requirements analysis 1

- **Model the application environment**
  - The context diagram is a simple analysis model that shows how the new system fits into its environment, defines the **boundaries and interfaces** between the system and external entities, such as users, hardware devices, and other systems
  - An ecosystem map shows the various systems in the solution space that interact with each other and the nature of their interconnections
- **Create user interface and technical prototypes**
  - construct a prototype—a partial, possible, or preliminary implementation—to make the concepts and possibilities more tangible
  - allow developers and users to achieve a **mutual understanding** of the problem being solved, as well as helping to validate requirements

GP: Requirements analysis 2

- **Analyze requirement feasibility**
  - evaluate the feasibility of implementing each requirement **at acceptable cost and performance** in the intended operating environment
  - **reveal the risks** associated with implementing each requirement, including conflicts and dependencies with other requirements, dependencies on external factors, and technical obstacles
  - Requirements that are technically infeasible or overly expensive to implement can perhaps be simplified and still contribute to achieving the project's business objectives
- **Prioritize the requirements**
  - ensure implementing the highest value or most timely functionality first
  - apply an analytical approach to determine the relative implementation priority of product features, use cases, user stories, or functional requirements
  - based on priority, determine which release or increment will contain each feature or set of requirements
  - adjust priorities throughout the project as new requirements are proposed

GP: Requirements analysis 3

- **Create a data dictionary**
  - cover the data items and structures associated with the system
  - this enables everyone working on the project to use consistent data definitions
  - data dictionary should define data items from the problem domain to facilitate communication between the customers and the development team
- **Model the requirements by diagrams that depicts requirements visually**
  - can reveal incorrect, inconsistent, missing, and superfluous requirements
  - include data flow diagrams, entity-relationship diagrams, state-transition diagrams, state tables, dialog maps, decision trees

GP: Requirements analysis 4

- **Analyze external interfaces between system and the outside world**
  - user interfaces
  - interfaces for data exchanging with other software systems
  - interconnections between software and hardware components for embedded systems
  - communication interfaces for network-connected applications
- **Allocate requirements to subsystems**
  - The requirements for a complex product that contains multiple subsystems must be apportioned among the various software, hardware, and human subsystems and components

GP: Requirements specification 1

- **Adopt standard requirement document templates --** a consistent structure for recording various groups of requirements-related information:
  - **vision** and scope document template
  - use case template
  - SRS template
- **Identify requirement origins**
  - trace every requirement back to its origin and to the stakeholders to contact when a change is requested
  - requirement origins can be identified through traceability links or by defining a requirement attribute for this purpose
- **Uniquely label each requirement**
  - define a convention for giving each requirement a unique identifying label
  - the convention must be robust enough to withstand additions, deletions, and changes made in the requirements over time
  - labeling the requirements permits requirements traceability and the recording of changes made

GP: Requirements specification 2

- **Record business rules** (corporate policies, government regulations, standards, and computational algorithms)
  - document your business rules **separately** from a project’s requirements because they typically have an existence beyond the scope of a specific project
  - treat business rules as an enterprise-level asset, not a project-level asset
  - if some rules will lead to functional requirements that enforce them, so define traceability links between those requirements and the corresponding rules
- **Specify nonfunctional requirements**
  - understand the quality characteristics that are important to success (performance, reliability, usability, modifiability)
  - specify external interface requirements, design and implementation constraints, internationalization issues, and other nonfunctional requirements

Vision statement

- Vision summarizes the long-term purpose and intent of the product
- Vision should reflect a balanced view that will satisfy the expectations of diverse stakeholders
- Vision should be grounded in the realities of existing or anticipated markets, enterprise architectures, corporate strategic directions, and resource limitations
- Keyword template
  - ***For*** [target customer]
  - ***Who*** [statement of the need or opportunity]
  - ***The*** [product name]
  - ***Is*** [product category]
  - ***That*** [major capabilities, key benefit, compelling reason to buy or use]
  - ***Unlike*** [primary competitive alternative, current system, current business process]
  - ***Our product*** [statement of primary differentiation and advantages of new product]

GP: Requirements validation 1

- **Review the requirements**
  - Peer review, particularly inspection, is one of the highest-value software quality practices available
  - Assemble a small team of reviewers who represent different perspectives (such as analyst, customer, developer, and tester), and carefully examine the written requirements, analysis models, and related information for defects.
  - Informal preliminary reviews during requirements development are also valuable
- **Test the requirements**
  - derive tests from the user requirements to document the expected behavior of the product under specified conditions
  - walk through the tests with customers to ensure that they reflect user expectations
  - map the tests to the functional requirements to make sure that **no requirements have been overlooked** and that all have corresponding tests
  - use the tests to verify the correctness of analysis models and prototypes

GP: Requirements validation 2

- **Define acceptance criteria as a set of acceptance tests**
  - based on user requirements
  - demonstrating satisfaction of specific nonfunctional requirements
  - tracking open defects and issues
- **Simulate the requirements using special tools**
  - interact with the simulated system to validate requirements and make design choices, making the requirements come to life before they are cast into the concrete of code