Phases of Software Engineering

Marko Schütz-Schmuck

Department of Computer Science and Engineering
University of Puerto Rico at Mayagüez
Mayagüez, PR

August 18, 2020

phases of software engineering

- overview of phases
- phases: ”on-demand” perspective
- phases: mind-set vs periods
- domain description
- requirements prescription
- software design
- implementation
- deployment
- operation
- testing

phases of software engineering

- overview of phases
- phases: ”on-demand” perspective
- phases: mind-set vs periods
- domain description
- requirements prescription
- software design
- implementation
- deployment
- operation
- testing

phases of software engineering

- overview of phases
- phases: ”on-demand” perspective
- phases: mind-set vs periods
- domain description
- requirements prescription
- software design
- implementation
- deployment
- operation
- testing

overview of phases

- phases reflect different mind-sets, not necessarily longer periods of time
- in agile development phases alternate quickly
- which phases should we distinguish?
  - domain description
  - requirements prescription
  - software design
  - implementation
  - deployment
  - operation
  - testing
- triptych: domain description, requirements prescription, software design

overview of phases

- phases reflect different mind-sets, not necessarily longer periods of time
- in agile development phases alternate quickly
- which phases should we distinguish?
  - domain description
  - requirements prescription
  - software design
  - implementation
  - deployment
  - operation
  - testing
- triptych: domain description, requirements prescription, software design

overview of phases

- phases reflect different mind-sets, not necessarily longer periods of time
- in agile development phases alternate quickly
- which phases should we distinguish?
  - domain description
  - requirements prescription
  - software design
  - implementation
  - deployment
  - operation
  - testing
- triptych: domain description, requirements prescription, software design

overview of phases

- phases reflect different mind-sets, not necessarily longer periods of time
- in agile development phases alternate quickly
- which phases should we distinguish?
  - domain description
  - requirements prescription
  - software design
  - implementation
  - deployment
  - operation
  - testing
- triptych: domain description, requirements prescription, software design

overview of phases

- phases reflect different mind-sets, not necessarily longer periods of time
- in agile development phases alternate quickly
- which phases should we distinguish?
  - domain description
  - requirements prescription
  - software design
  - implementation
  - deployment
  - operation
  - testing
- triptych: domain description, requirements prescription, software design

overview of phases

- phases reflect different mind-sets, not necessarily longer periods of time
- in agile development phases alternate quickly
- which phases should we distinguish?
  - domain description
  - requirements prescription
  - software design
  - implementation
  - deployment
  - operation
  - testing
- triptych: domain description, requirements prescription, software design

overview of phases

- phases reflect different mind-sets, not necessarily longer periods of time
- in agile development phases alternate quickly
- which phases should we distinguish?
  - domain description
  - requirements prescription
  - software design
  - implementation
  - deployment
  - operation
  - testing
- triptych: domain description, requirements prescription, software design

overview of phases

- phases reflect different mind-sets, not necessarily longer periods of time
- in agile development phases alternate quickly
- which phases should we distinguish?
  - domain description
  - requirements prescription
  - software design
  - implementation
  - deployment
  - operation
  - testing
- triptych: domain description, requirements prescription, software design

overview of phases

- phases reflect different mind-sets, not necessarily longer periods of time
- in agile development phases alternate quickly
- which phases should we distinguish?
  - domain description
  - requirements prescription
  - software design
  - implementation
  - deployment
  - operation
  - testing
- triptych: domain description, requirements prescription, software design

overview of phases

- phases reflect different mind-sets, not necessarily longer periods of time
- in agile development phases alternate quickly
- which phases should we distinguish?
  - domain description
  - requirements prescription
  - software design
  - implementation
  - deployment
  - operation
  - testing
- triptych: domain description, requirements prescription, software design

overview of phases

- phases reflect different mind-sets, not necessarily longer periods of time
- in agile development phases alternate quickly
- which phases should we distinguish?
  - domain description
  - requirements prescription
  - software design
  - implementation
  - deployment
  - operation
  - testing
- triptych: domain description, requirements prescription, software design

phases: ”on-demand” perspective

- in order to perform some operation on the system the functionality has to be deployed
- … deploy functionality it has to be implemented
- … implement functionality we need a software design
- … design software we need to understand the requirements
- … formulate requirements we need to understand the domain

phases: ”on-demand” perspective

- in order to perform some operation on the system the functionality has to be deployed
- … deploy functionality it has to be implemented
- … implement functionality we need a software design
- … design software we need to understand the requirements
- … formulate requirements we need to understand the domain

phases: ”on-demand” perspective

- in order to perform some operation on the system the functionality has to be deployed
- … deploy functionality it has to be implemented
- … implement functionality we need a software design
- … design software we need to understand the requirements
- … formulate requirements we need to understand the domain

phases: ”on-demand” perspective

- in order to perform some operation on the system the functionality has to be deployed
- … deploy functionality it has to be implemented
- … implement functionality we need a software design
- … design software we need to understand the requirements
- … formulate requirements we need to understand the domain

phases: ”on-demand” perspective

- in order to perform some operation on the system the functionality has to be deployed
- … deploy functionality it has to be implemented
- … implement functionality we need a software design
- … design software we need to understand the requirements
- … formulate requirements we need to understand the domain

phases: mind-set vs periods

- phases complement each other and provide feedback to one-another
  - as soon as even a small aspect belonging to one phase is elaborated, corresponding progress in other phases is possible
  - other phases feed back onto the initial finding
    - example: defining domain term ”cash register” might lead insight that payment is also possible via smartphone app requiring only proof-of-payment to a store clerk at exit
    - therefore: requirement changes
- not necessary nor desirable to continue a phase to ”completion”
- instead fluently move back-and-forth among the phases

requirement in supermarket system

- ”the system charges customers the total for all items taken from the supermarket at the cash register except for items previously paid for”
- needs definitions/descriptions of domain concepts
  - > ”item”, ”pay”, ”charge”, ”cash register”, ...
- has relation to overall goals
  - > profitability, customer satisfaction, data confidentiality, throughput, ...

requirement in supermarket system

- ”the system charges customers the total for all items taken from the supermarket at the cash register except for items previously paid for”
- needs definitions/descriptions of domain concepts
  - ”item”, ”pay”, ”charge”, ”cash register”, ...
- has relation to overall goals
  - profitability, customer satisfaction, data confidentiality, throughput, ...

requirement in supermarket system

- ”the system charges customers the total for all items taken from the supermarket at the cash register except for items previously paid for”
- needs definitions/descriptions of domain concepts
  - ”item”, ”pay”, ”charge”, ”cash register”, ...
- has relation to overall goals
  - profitability, customer satisfaction, data confidentiality, throughput, ...

requirement in supermarket system

- ”the system charges customers the total for all items taken from the supermarket at the cash register except for items previously paid for”
- needs definitions/descriptions of domain concepts
  - ”item”, ”pay”, ”charge”, ”cash register”, …
- has relation to overall goals
  - profitability, customer satisfaction, data confidentiality, throughput, …

requirement in supermarket system

- ”the system charges customers the total for all items taken from the supermarket at the cash register except for items previously paid for”
- needs definitions/descriptions of domain concepts
  - ”item”, ”pay”, ”charge”, ”cash register”, …
- has relation to overall goals
  - profitability, customer satisfaction, data confidentiality, throughput, …

requirement in supermarket system (cont)

- has relation to other requirements ”the system shall provide a scanner for the cashier to scan the items presented at check-out”
- relationships can carry more information: ”contributes strongly to goal XYZ”, ”uses domain definition ABC”, ...
- allows formulation of tests
  - > how can we show, prove, justify that the requirement is satisfied

requirement in supermarket system (cont)

- has relation to other requirements ”the system shall provide a scanner for the cashier to scan the items presented at check-out”
- relationships can carry more information: ”contributes strongly to goal XYZ”, ”uses domain definition ABC”, …
- allows formulation of tests
  - how can we show, prove, justify that the requirement is satisfied

requirement in supermarket system (cont)

- has relation to other requirements ”the system shall provide a scanner for the cashier to scan the items presented at check-out”
- relationships can carry more information: ”contributes strongly to goal XYZ”, ”uses domain definition ABC”, …
- allows formulation of tests
  - how can we show, prove, justify that the requirement is satisfied

requirement in supermarket system (cont)

- has relation to other requirements ”the system shall provide a scanner for the cashier to scan the items presented at check-out”
- relationships can carry more information: ”contributes strongly to goal XYZ”, ”uses domain definition ABC”, …
- allows formulation of tests
  - how can we show, prove, justify that the requirement is satisfied

domain description

- situation as-is and as it is relevant to the project
- not: requirements, design decisions, implementation, ...
- no reference to system-to-be
- (older) viewpoint: domain properties treated as part of requirements
  - > then explicitly labeled as "domain property", "domain assumption", ...
  - > compare Wikipedia entry on domain engineering

domain description

- situation as-is and as it is relevant to the project
- not: requirements, design decisions, implementation, …
- no reference to system-to-be
- (older) viewpoint: domain properties treated as part of requirements
  - > then explicitly labeled as "domain property", "domain assumption", …
  - > compare Wikipedia entry on domain engineering

domain description

- situation as-is and as it is relevant to the project
- not: requirements, design decisions, implementation, …
- no reference to system-to-be
- (older) viewpoint: domain properties treated as part of requirements
  - > then explicitly labeled as "domain property", "domain assumption", …
  - > compare Wikipedia entry on domain engineering

domain description

- situation as-is and as it is relevant to the project
- not: requirements, design decisions, implementation, …
- no reference to system-to-be
- (older) viewpoint: domain properties treated as part of requirements
  - then explicitly labeled as ”domain property”, ”domain assumption”, …
  - compare Wikipedia entry on domain engineering

domain description

- situation as-is and as it is relevant to the project
- not: requirements, design decisions, implementation, …
- no reference to system-to-be
- (older) viewpoint: domain properties treated as part of requirements
  - then explicitly labeled as ”domain property”, ”domain assumption”, …
  - compare Wikipedia entry on domain engineering

domain description

- situation as-is and as it is relevant to the project
- not: requirements, design decisions, implementation, …
- no reference to system-to-be
- (older) viewpoint: domain properties treated as part of requirements
  - then explicitly labeled as ”domain property”, ”domain assumption”, …
  - compare Wikipedia entry on domain engineering

domain description (cont)

- kinds of phenomena/concepts
  - entities
    - "things" concrete or abstract
  - functions
    - actions, operations
    - signature: captures data used & produced
  - events
    - instances in time, mark the start/end of actions
    - can be modeled by messages sent along channels
    - can carry additional information
  - behaviors
    - interleaved sequences of events and actions

domain description (cont)

- kinds of phenomena/concepts
  - entities
    - "things" concrete or abstract
  - functions
    - actions, operations
    - signature: captures data used & produced
  - events
    - instances in time, mark the start/end of actions
    - can be modeled by messages sent along channels
    - can carry additional information
  - behaviors
    - interleaved sequences of events and actions

domain description (cont)

- kinds of phenomena/concepts
  - entities
    - ”things” concrete or abstract
  - functions
    - actions, operations
    - signature: captures data used & produced
  - events
    - instances in time, mark the start/end of actions
    - can be modeled by messages sent along channels
    - can carry additional information
  - behaviors
    - interleaved sequences of events and actions

domain description (cont)

- kinds of phenomena/concepts
  - entities
    - ”things” concrete or abstract
  - functions
    - actions, operations
    - signature: captures data used & produced
  - events
    - instances in time, mark the start/end of actions
    - can be modeled by messages sent along channels
    - can carry additional information
  - behaviors
    - interleaved sequences of events and actions

domain description (cont)

- kinds of phenomena/concepts
  - entities
    - ”things” concrete or abstract
  - functions
    - actions, operations
    - signature: captures data used & produced
  - events
    - instances in time, mark the start/end of actions
    - can be modeled by messages sent along channels
    - can carry additional information
  - behaviors
    - interleaved sequences of events and actions

domain description (cont)

- example: power grid
  - entities
    - e.g. generator, transformer, load, long-distance transmission lines, substations, distribution lines, consumers
  - functions:
    - consume
      consume : PowerGrid × Consumer × Load → PowerGrid
      another view
      consume : Meter × Consumer × Load → Meter
  - events:
    - line repair was just completed
      A distribution line was under repair. That repair has just been completed.
      Marks the end of the repair, the start of providing power of that line again.

domain description (cont)

- example: power grid
  - entities
    - e.g. generator, transformer, load, long-distance transmission lines, substations, distribution lines, consumers
  - functions:
    - consume
      consume : PowerGrid × Consumer × Load → PowerGrid
      another view
      consume : Meter × Consumer × Load → Meter
  - events:
    - line repair was just completed
      A distribution line was under repair. That repair has just been completed. Marks the end of the repair, the start of providing power of that line again.

domain description (cont)

- example: power grid
  - entities
    - e.g. generator, transformer, load, long-distance transmission lines, substations, distribution lines, consumers
  - functions:
    - consume
      consume : PowerGrid × Consumer × Load → PowerGrid
      another view
      consume : Meter × Consumer × Load → Meter
  - events:
    - line repair was just completed
      A distribution line was under repair. That repair has just been completed. Marks the end of the repair, the start of providing power of that line again.

domain description (cont)

- behavior:
  - attend to outage
    - action: determine faulty line segment
    - event: just found faulty line segment
    - action: allocate crew
    - event: crew allocated, ready to leave
    - action: crew transfer to site
    - event: crew just arrived at site
    - action: secure work area
    - event: work area has just been secured
    - ...
    - event: line repair was just completed
  - may use concurrently operating actors

domain description (cont)

- expectations
  - communication platform for all project stakeholders
    - needs to be understandable
  - basis for learning and training in domain
  - basis for large part of requirements
  - basis for business process re-engineering

domain description (cont)

- expectations
  - communication platform for all project stakeholders
    - needs to be understandable
  - basis for learning and training in domain
  - basis for large part of requirements
  - basis for business process re-engineering

domain description (cont)

- expectations
  - communication platform for all project stakeholders
    - needs to be understandable
  - basis for learning and training in domain
  - basis for large part of requirements
  - basis for business process re-engineering

domain description (cont)

- expectations
  - communication platform for all project stakeholders
    - needs to be understandable
  - basis for learning and training in domain
  - basis for large part of requirements
  - basis for business process re-engineering

domain description (cont)

- facets
  - provide generic perspective to focus on specific category of phenomena
  - suggested facets
    - business processes
    - intrinsics
    - support technology
    - management & organization
    - rules & regulations
    - scripts
    - human behavior

domain description (cont)

- facets
  - provide generic perspective to focus on specific category of phenomena
  - suggested facets
    - business processes
    - intrinsics
    - support technology
    - management & organization
    - rules & regulations
    - scripts
    - human behavior

domain description (cont)

- facets
  - provide generic perspective to focus on specific category of phenomena
  - suggested facets
    - business processes
    - intrinsics
    - support technology
    - management & organization
    - rules & regulations
    - scripts
    - human behavior

domain description (cont)

- facets
  - provide generic perspective to focus on specific category of phenomena
  - suggested facets
    - business processes
    - intrinsics
    - support technology
    - management & organization
    - rules & regulations
    - scripts
    - human behavior

domain description (cont)

- facets
  - provide generic perspective to focus on specific category of phenomena
  - suggested facets
    - business processes
    - intrinsics
    - support technology
    - management & organization
    - rules & regulations
    - scripts
    - human behavior

domain description (cont)

- facets
  - provide generic perspective to focus on specific category of phenomena
  - suggested facets
    - business processes
    - intrinsics
    - support technology
    - management & organization
    - rules & regulations
    - scripts
    - human behavior

domain description (cont)

- facets
  - provide generic perspective to focus on specific category of phenomena
  - suggested facets
    - business processes
    - intrinsics
    - support technology
    - management & organization
    - rules & regulations
    - scripts
    - human behavior

domain description (cont)

- facets
  - provide generic perspective to focus on specific category of phenomena
  - suggested facets
    - business processes
    - intrinsics
    - support technology
    - management & organization
    - rules & regulations
    - scripts
    - human behavior

domain description (cont)

- facets
  - provide generic perspective to focus on specific category of phenomena
  - suggested facets
    - business processes
    - intrinsics
    - support technology
    - management & organization
    - rules & regulations
    - scripts
    - human behavior

requirements prescription

- the ”machine”
- the machine’s environment
- features, properties, aspects that the system-to-be must have
- not: description/definition of phenomena in the domain (e.g. ”customers pay at the cash register”)
- not: decision of implementation technology (e.g. ”stored in an SQL database”, ”runs on an 8-core 2GHz server”)

requirements prescription

- the ”machine”
- the machine’s environment
- features, properties, aspects that the system-to-be must have
- not: description/definition of phenomena in the domain (e.g. ”customers pay at the cash register”)
- not: decision of implementation technology (e.g. ”stored in an SQL database”, ”runs on an 8-core 2GHz server”)

requirements prescription

- the ”machine”
- the machine’s environment
- features, properties, aspects that the system-to-be must have
- not: description/definition of phenomena in the domain (e.g. ”customers pay at the cash register”)
- not: decision of implementation technology (e.g. ”stored in an SQL database”, ”runs on an 8-core 2GHz server”)

requirements prescription

- the ”machine”
- the machine’s environment
- features, properties, aspects that the system-to-be must have
- not: description/definition of phenomena in the domain (e.g. ”customers pay at the cash register”)
- not: decision of implementation technology (e.g. ”stored in an SQL database”, ”runs on an 8-core 2GHz server”)

requirements prescription (cont)

- kinds of requirements
  - business process reengineering
  - domain requirements
  - interface requirements
  - machine requirements

requirements prescription (cont)

- kinds of requirements
  - business process reengineering
  - domain requirements
  - interface requirements
  - machine requirements

requirements prescription (cont)

- kinds of requirements
  - business process reengineering
  - domain requirements
  - interface requirements
  - machine requirements

requirements prescription (cont)

- kinds of requirements
  - business process reengineering
  - domain requirements
  - interface requirements
  - machine requirements

requirements prescription (cont)

- kinds of requirements
  - business process reengineering
  - domain requirements
  - interface requirements
  - machine requirements

requirements prescription (cont)

- example: power grid
  - system-to-be must track the current power consumption for each consumer/meter
  - system-to-be must detect & update changes in power consumption within 100ms in 99% of all cases
  - system-to-be must handle 100M transactions per hour at 80% system load

requirements prescription (cont)

- example: power grid
  - system-to-be must track the current power consumption for each consumer/meter
  - system-to-be must detect & update changes in power consumption within 100ms in 99% of all cases
  - system-to-be must handle 100M transactions per hour at 80% system load

requirements prescription (cont)

- example: power grid
  - system-to-be must track the current power consumption for each consumer/meter
  - system-to-be must detect & update changes in power consumption within 100ms in 99% of all cases
  - system-to-be must handle 100M transactions per hour at 80% system load

requirements prescription (cont)

- example: power grid
  - system-to-be must track the current power consumption for each consumer/meter
  - system-to-be must detect & update changes in power consumption within 100ms in 99% of all cases
  - system-to-be must handle 100M transactions per hour at 80% system load

software design

- software architecture
  - assign responsibility for individual requirements to components
  - define and explain the interconnections between the components
- component design
  - select algorithms and data structures

software design

- software architecture
  - assign responsibility for individual requirements to components
  - define and explain the interconnections between the components
- component design
  - select algorithms and data structures

implementation

- write the corresponding source code conforming to the insights from other phases

  obviously much more could be said here…

deployment

- install the system for productive operation

operation

- day-to-day operation of the system

testing

- aim: verification "Are we doing it right?"
- wide variation of approaches, e.g.
  - unit testing - exercise the functioning of some unit (e.g. class) and compare with known correct outcome
  - model checking - exhaustively check a property on a model (e.g. 1 generator, 1 transmission line, 1 substation, 3 consumers)
  - A/B testing - compare user behavior on two different versions
  - proof - rigorously prove properties
  - review - consider how specific scenarios would be dealt with

testing (cont)

- applicable at various phases
  - example:
    - capture some fundamental assumptions of the domain, claim additional property will always hold under assumptions, use logical inference to prove claim
    - correct refinement of requirements: rigorously show that refined requirements together with domain assumptions ensure satisfaction of original requirements