SE 555 Software Requirements & Specification

Prototyping

Prototyping

- Prototyping involves building simple & quick implementations of parts of the product
- The major reason for prototyping is to reduce some type of risk associated with the project

Project Risks

- A risk is a possibility that could significantly impact the success of the project if it occurs
- Projects may have various sources of risk e.g.
  - Requirements risk: Missing key requirements and preferences that impact customer satisfaction
  - Technical risk: Not sure if the product will be able to deliver on functionality / performance / convenience
  - Technology risk: Will the team be able to use the development technologies properly? (if new technologies being used)
  - Design risk: Missing key design issues / wrong decisions
  - Market risk: Will it sell? Will customers like it?
  - Schedule risk: Will it get completed on time?

Risk Mitigation

- A proactive way to deal with risk is to mitigate it
  - Put in some additional effort now to reduce the impact of the risk if it occurs
    - E.g. Take backups to deal with risk of data loss
  - (or) Put in additional effort to make it less likely that the risk will occur
    - E.g. Do market surveys to check if the product is likely to sell
- For several types of risk, prototyping is an effective way to mitigate the risk

Types of prototype

- Several types of prototypes, depending on objectives:
  - Interface prototypes (most common) mitigate requirements risk
    - Build a mockup of product interface to get user feedback
    - Minimal or dummy functionality
  - Implementation prototypes mitigate technical risk
    - Core product functionality to demonstrate feasibility of product
    - May have trivial interfaces and few additional features
    - Variation: Design prototypes, to understand design issues, study behavior
  - Technology prototypes try out use of new technologies
    - Build only a part of the product, or even something totally different
  - Demos are prototypes built to mitigate market risk

Throw-away Prototypes

- Discard prototype, build product from scratch
  - \+ Can use special “prototyping technologies” to build quick prototypes e.g. GUI builders, code generators
  - \+ Don’t have to design prototype carefully
  - \+ Coding can be more freestyle and unconstrained
  - – Need to redo all the work

Evolutionary Prototypes

- Modify prototype into final product
  - + Prototype is just first version in incremental development
  - • But need to be careful to use full product-style development process when building prototype
  - – Prototypes change a lot early on, so design & code quality may deteriorate. Need to re-factoring / reimplementation as needed

Horizontal Prototypes

- Horizontal prototypes are a partial or possible implementation of a user interface for a software system
- Used to evaluate usability and to assess requirements
- Also called a behavioral prototype or a mock-up

Vertical Prototypes

- A vertical prototype is a partial implementation of a software system that slices through all layers of the architecture.
- Used to evaluate technical feasibility and performance.
- Also called a structural prototype or a proof of concept

Prototyping for req elicitation

- Build prototype of interface
  - Minimal or dummy functionality
- Demonstrate / let user use the prototype
- Obtain feedback and suggestions
  - If possible (esp. with GUI builders), make changes immediately and get further feedback
- Iterate until customer satisfied with interface and behavior

- Advantage: Customer knows exactly what to expect
- Danger: Customers don't know why it takes so long to go from prototype to product!

Prototyping Success Factors

- Include prototyping tasks in project plan
- State the purpose of each prototype
- Plan to possibly develop multiple prototypes to get requirements correct – you will probably do several iterations of the prototype
- Create throwaway prototypes quickly & cheaply
- Don’t include extensive coding practices in a throwaway prototype – keep the effort to a minimum
- Don’t prototype requirements you already understand
- Use as realistic data and displays as possible in order to focus on functionality.
- Don’t expect a prototype to replace a complete SRS

Prototype Planning

- Do you need a to build a prototype from scratch? Could an existing one be adapted to serve the intended purpose?
- Would it be best to build a separate prototype for this event/use case, or combine with others?
- Rather than a high-fidelity (automated) prototype, might a low-fidelity (pencil, paper, whiteboard) prototype serve the purpose?
- Choose prototyping tools that require the least effort to accomplish the objective
- Make sure to include appropriate time to test the prototype with users in your schedule

Prototype Testing with Users

- Some questions to ask:

  - Does this model behave as you expected?

  - Can you imagine yourself using a product that works like this to do useful work?

  - Does anything about this prototype irritate you?

  - Is anything missing from this prototype?

- Develop test scenarios, don’t just evaluate in an ad hoc manner

- Make sure people evaluating the prototype are representative of actual users