CS3215: Software Engineering Project

CS3215, LN set #3: SDLC for the project

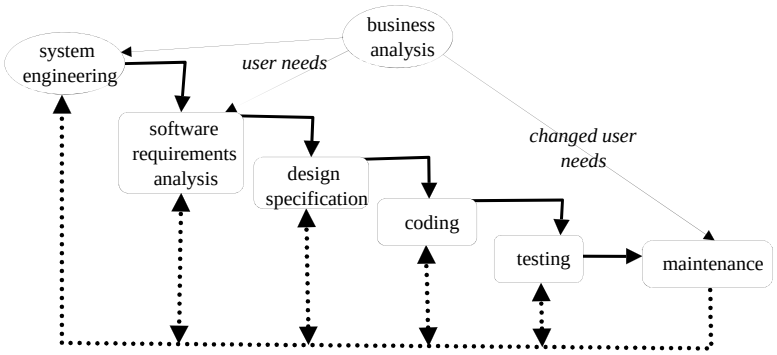
SDLC: Software Development Lifecycle

- SDLC is a framework for project planning and executing
  - who is doing what, when and how?
  - for managers: to tell how things should be done, define project tasks
  - for developers: to follow the plan, provide feedback to managers
- SDLC addresses people, product, process, technology
  - organization of the project team, communication channels
  - project phases, activities, milestones
  - software system architecture
  - methods and tools to be used
- SDLC helps us manage project complexities

How can SDLC help?

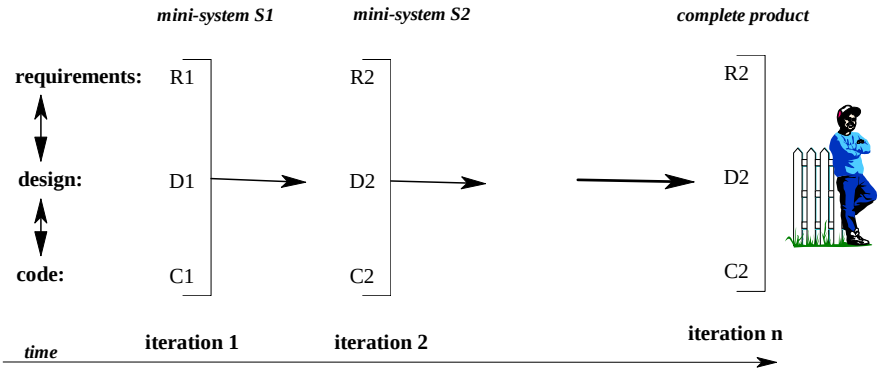
- Benefit from 50 years of project experiences
  - learn from project management experiences that work
  - learn from development strategies
  - avoid common mistakes
- Standardize routine tasks
  - use creativity to solve novel problems rather than to reinvent the wheel
- Reduce risk of failure, increase project predictability
  - 25% of large projects fails
  - 20% implemented on time but 70% experience 100% cost overruns
  - 50% of projects run over planned schedule and budget
- Types of SDLC:
  - waterfall, spiral, incremental, RAD (rapid application development)

Waterfall SDLC with feedback loops



- Problems with the waterfall SDLC:
  - fuzzy and changing requirements
  - late detection of errors
  - easy to say, difficult to do
  - high risk, high failure rate

Iterative, incremental SDLC



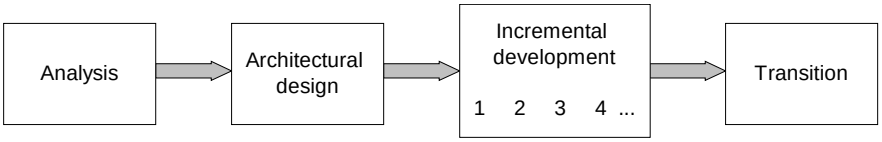
- System is developed in pieces rather than in one big-bang
- Project is divided into mini-projects (iterations)
- Each iterations delivers fully tested and integrated mini-system, implementing a subset of the complete product

Why the incremental SDLC works better?

- You can gradually learn about the problem and program solution
  - you can better deal with complex problems - decomposition
  - you can identify major problems early and work on them
  - in each iteration you check if all basic project elements are in sync
- Early validation of requirements
- Early validation of design decisions
- Incremental development encourages “design for change”
- Incremental SDLC effectively reduces project risks related to:
  - requirements, technical issues, personnel and politics.

Phases in incremental SDLC

- You must perceive a problem as a whole before you can plan development iterations



- Analysis: understand the problem, scope the project
- Architectural design: create a blueprint for a system
- Incremental development through iterations
- Transition:
  - beta-testing, performance tuning, user training, packaging the product

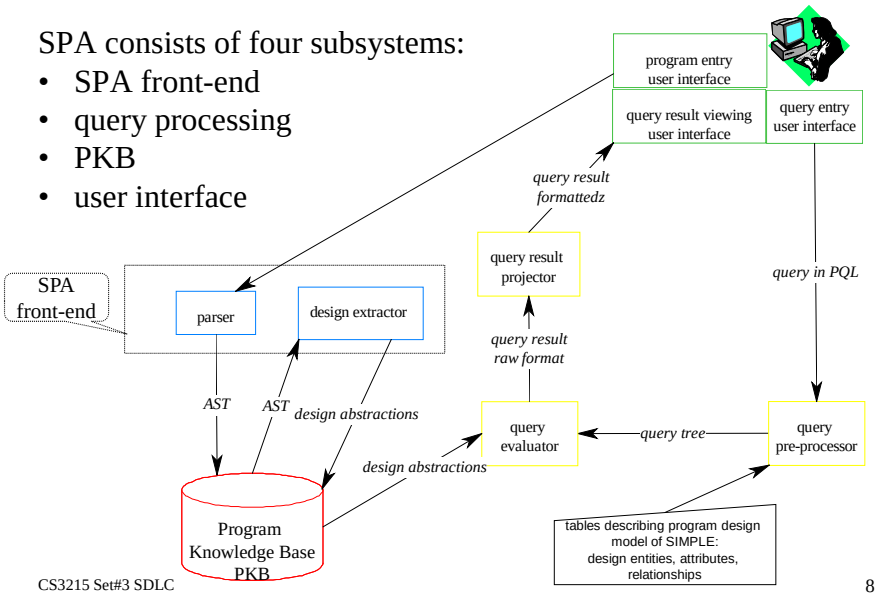
An SDLC for SPA project

- First, we must understand the problem and solution
- *Analysis*: understand the problem
  - play with models on paper
  - draw AST and CFG for sample programs
  - compute other design abstractions to be derived from sources
  - write program queries and evaluate queries by hand
- *Architectural design*: understand the solution
  - identify major SPA components and their interactions
  - sketch design abstractions in PKB as ADTs:
    - model associations among ADTs
    - define interface operations
  - look into a strategy for evaluating program queries

A sketch of the SPA architecture

SPA consists of four subsystems:

- SPA front-end
- query processing
- PKB
- user interface



Team structure

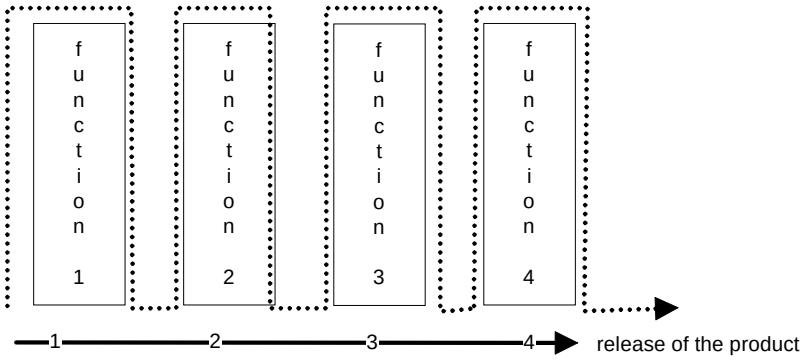
Group-PKB

- design of the SPA front-end to parse a SIMPLE program and build an AST
- design of the algorithms to derive procedure call, control flow and other program design information, as described in the program design model
- design of the data structures to store the program design abstractions in the PKB

- Group-PQL
  - validating a query
  - design a query evaluation mechanism
- Both groups:
  - design the PKB API
  - design algorithms to compute program design abstractions on demand, such as Calls*, Next*, Affects and Affects*.

Strategy 1: Depth-first iterations

- take one function at a time
- develop solution for that problem in a given iteration

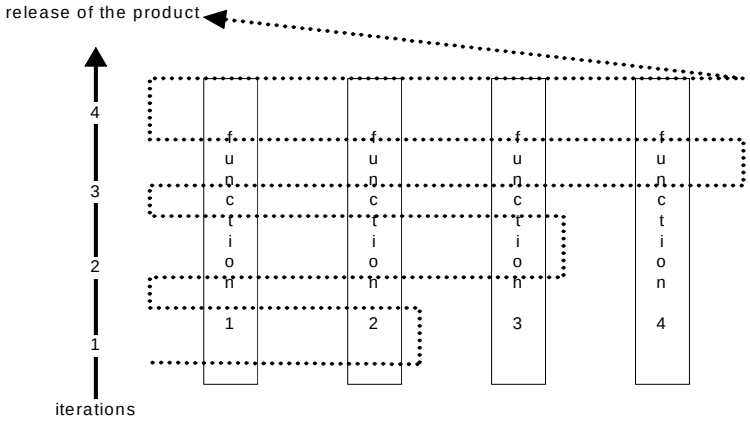


Example

1. develop a parser for SIMPLE
2. implement ADT for AST, CFG, etc.
3. generate an AST
4. generate CFG
5. develop parser for program queries
6. develop query evaluator

Strategy 2: Breadth-first iterations

- in a given iteration, work on a set of related functions, but in a simplified form
- select iterations so that the whole team works closely together



Example

First iteration:

- select a small subset of SIMPLE and PQL

Group-PKB:

- develop a parser for that subset
- generate simplified AST, CFG

Group-PQL:

- work on queries in a fixed, simple format only
- consider queries that mostly require program design information computed by Group-PKB

Subsequent iteration:

2. extend the subset of SIMPLE
3. refine PKB API
4. develop query evaluator for more query formats

Which strategy is better for the SPA project?

Evaluate strategies, plan and justify the choice of your development strategy in assignment #2

--- The end ---