software methodologies

software methodology

- life cycle model
- practices
- principles
- patterns

examples of methodologies

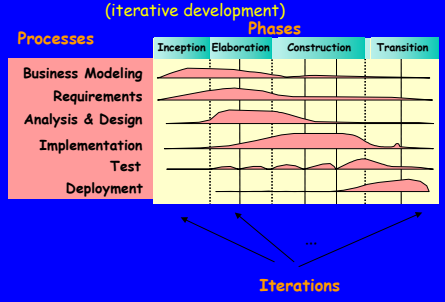
- rational unified process (RUP)
- extreme programming (XP)

Major RUP Principles

- iterative development

Rational Unified Process

(iterative development)



RUP Iteration

- Iteration i
  - Requirements:
    - What are you going to do?
    - How are you going to test it?
  - Design: How will you do it?
  - Implementation: Do it!
  - Test: Does it work?
  - Transition to phase i+1:
    - Integrate results into final project
    - Test integration
    - Acceptance test

Major RUP Principles

- iterative development
- risk-driven

RUP Iteration

- Iteration i
  - Requirements:
    - What are you going to do?
    - How are you going to test it?

    ← choose highest-risk, highest-value issue
  - Design: How will you do it?
  - Implementation: Do it!
  - Test: Does it work?
  - Transition to phase i+1:
    - Integrate results into final project
    - Test integration
    - Acceptance test

Major RUP Principles

- iterative development
- risk-driven
- build core architecture early

RUP Iteration

- Iteration i
  - Requirements:
    - What are you going to do?
    - How are you going to test it?

    core is high risk/high value

    ← choose highest-risk, highest-value issue
  - Design: How will you do it?
  - Implementation: Do it!
  - Test: Does it work?
  - Transition to phase i+1:
    - Integrate results into final project
    - Test integration
    - Acceptance test

Major RUP Principles

- iterative development
- risk-driven
- build core architecture early
- continuously engage users for evaluation and feedback

RUP Iteration

- Iteration i
  - Requirements:
    - What are you going to do?
    - How are you going to test it?

    ← choose highest-risk, highest-value issue
  - Design: How will you do it?
  - Implementation: Do it!

    customers help assign value
  - Test: Does it work?
  - Transition to phase i+1:
    - Integrate results into final project
    - Test integration
    - Acceptance test ← customers write acceptance test

Major RUP Principles

- iterative development
- risk-driven
- build core architecture early
- continuously engage users for evaluation and feedback
- test early and often

RUP Iteration

- Iteration i
  - Requirements:
    - What are you going to do?
    - How are you going to test it?
  - Design: How will you do it?
  - Implementation: Do it!
  - Test: Does it work? unit test
  - Transition to phase i+1:
    - Integrate results into final project
    - Test integration
    - Acceptance test

Major RUP Principles

- iterative development
- risk-driven
- build core architecture early
- continuously engage users for evaluation and feedback
- test early and often
- apply use cases

Definition of "Use Case"

- "The specification of sequences of actions that a system, subsystem, or class can perform by interacting with outside actors"

  (*UML Reference Manual*, Rumbaugh, Jacobson, and Booch).

example use case

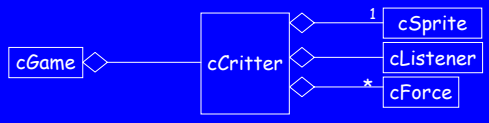
Play Game:

The player moves through a cave trying to kill the wumpus before the wumpus eats him.

Major RUP Principles

- iterative development
- risk-driven
- build core architecture early
- continuously engage users for evaluation and feedback
- test early and often
- apply use cases
- build diagrammatic models (UML)

Example: Class Diagrams



Major RUP Principles

- iterative development
- risk-driven
- build core architecture early
- continuously engage users for evaluation and feedback
- test early and often
- apply use cases
- build diagrammatic models (UML)
- etc.

examples of methodologies

- rational unified process (RUP)
- extreme programming (XP)

Manifesto for agile software development

We are uncovering better ways of developing software by doing it and helping others to do it. Through this work we have come to value:

Individuals and interactions over processes and tools.

Working software over comprehensive documentation.

Customer collaboration over contract negotiation.

Responding to change over following a plan.

That is, while there is value in the items on the right, we value the items on the left more.

life cycle
extreme programming

- short cycles
  - iteration: ~two weeks, ends in minor delivery that may or may not be put in production
  - release plan: ~six iterations, ends in major delivery that can be put into production
- budget is based on accomplishments of previous iteration/release

life cycle
extreme programming

- "requirements"
  - initially customer/developers try to identify the significant *user stories*
  - at the start of each iteration/release the customer prioritizes the user stories, the developers decide how many they can develop in the next iteration/release
  - at the end of each iteration/release the developers demo their work and the customer provides feedback. plans are changed as needed.

user stories
extreme programming

- the player moves through a cave trying to kill the wumpus before the wumpus eats him

Comparison

RUP

- Risk-driven, risks determined by developers
- Establish core architecture early
- Milestones are usually documents
- Test at early and often
- Tool heavy

XP

- Priority-driven, priorities determined by customer
- Build only what you need now
- Milestones are usually code
- Test constantly; build up test base
- Tool light

methodologies

we'll continue our discussion of methodologies throughout the next few weeks

schedule of events

- concept clarification
  - use cases
  - your team will identify major use cases for each game concept
- concept assessment
  - is it a good game?
  - can it be done in POP?
- concept decision