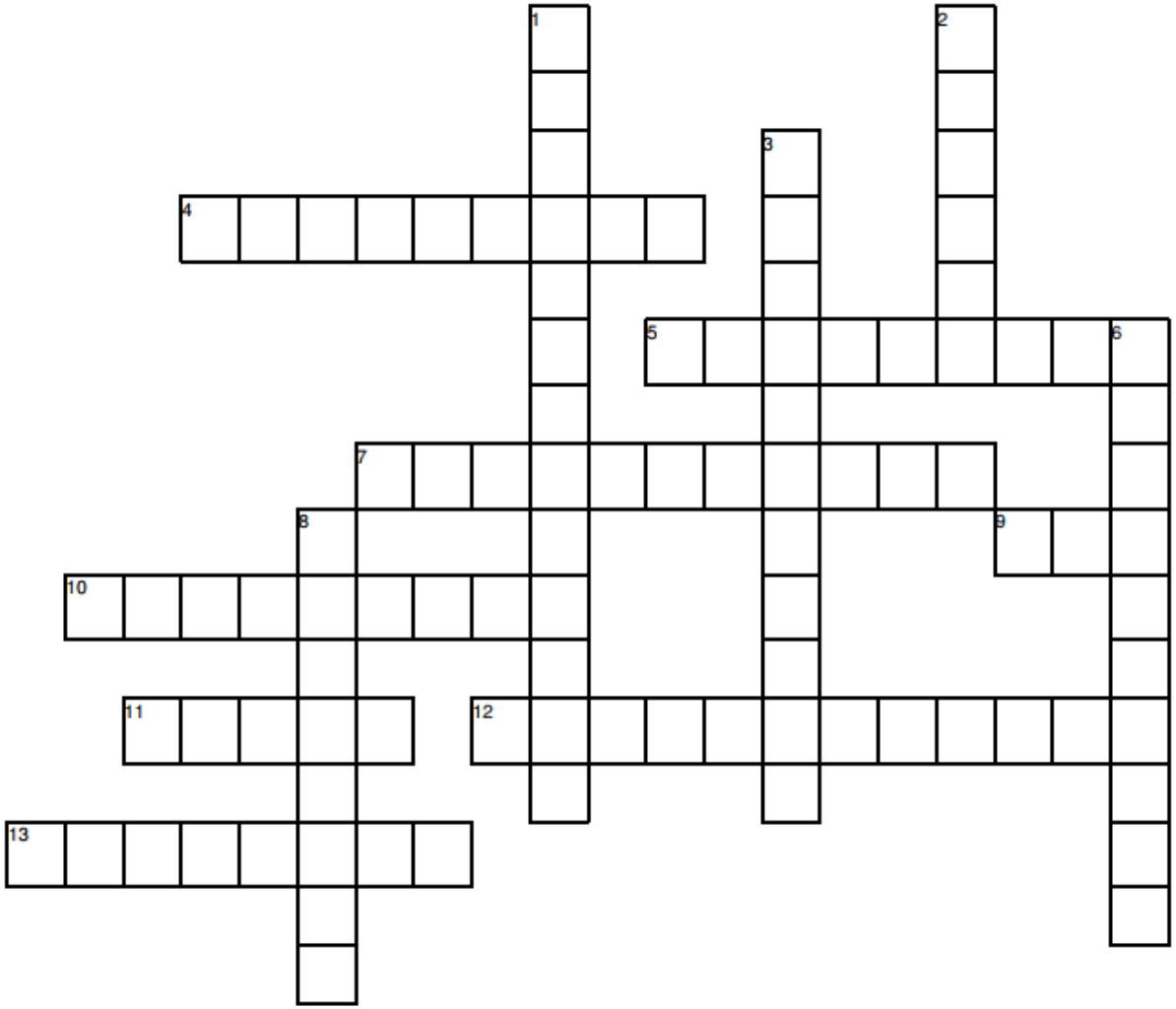
Pragmatics

Object Orientated Analysis and Design

Benjamin Kenwright

"That's a great question. Come to think of it, I'm not sure what it is I'm trying to design."

Crossword Challenge



Across

4. A collection of operations that have no implementation but specify a particular service of a class or a component

5. Visibility access only by operations within the class or within children of the class

7. Classes specialized so their instances can manage a collection of other objects

9. An open extensible industry standard visual modelling language

10. A modular and replaceable part of a system that encapsulates its contents

11. A complete description of a system from a particular perspective

12. Different classes having methods of the same name that perform the same conceptual task

13. The degree to which classes within our system are dependent on each other

Down

1. The data inside an object is hidden and can only be manipulated by invoking one of the object's function

2. Visibility access by any element that can access the class

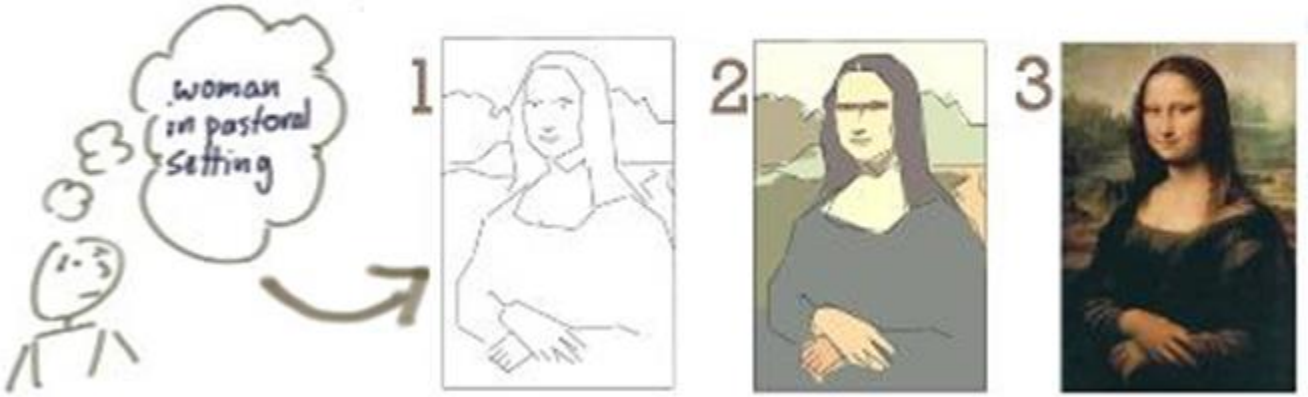
3. When classes are connected together conceptually

6. Diagram that maps system software to artifacts to the physical hardware that executes them

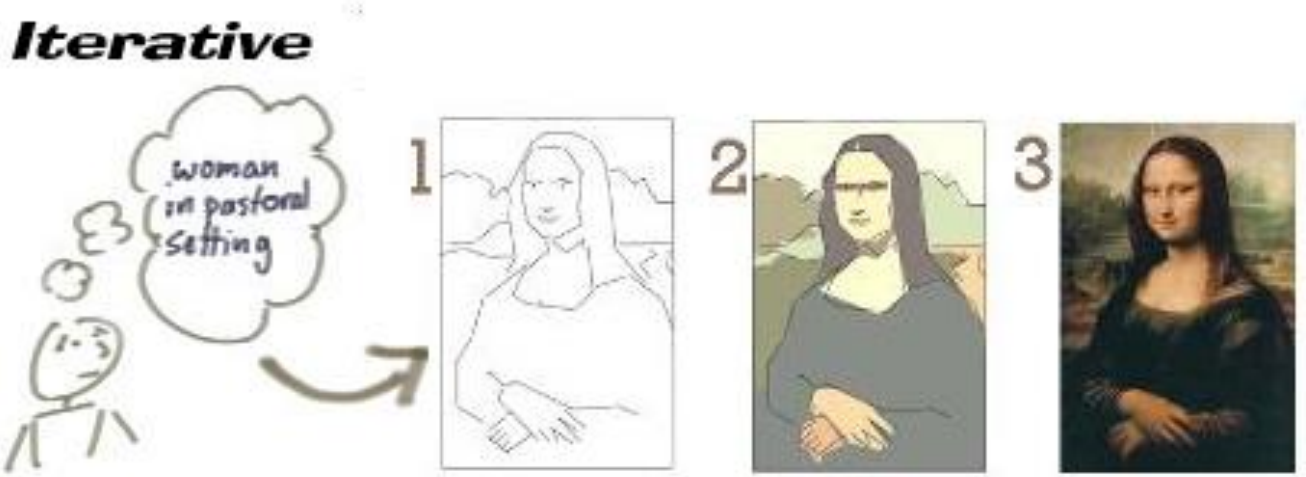
8. The measure of how much an entity supports a singular purpose within a system

Question

- Which is incremental and iterative?



Answer



Question

- The difference between V-shaped model and waterfall model is the early test planning in the V-shaped model

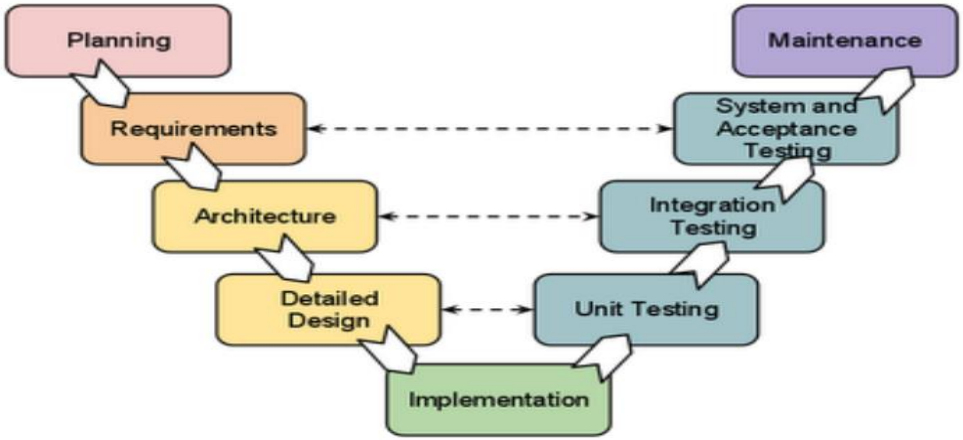
a) True
b) False

Answer

- a) True

V-Shaped Model

- Difference between V-shaped model and waterfall model is the early test planning in the V-shaped model



Question

- Which development approach is the waterfall model?

a) iterative development approach

b) static development approach

c) behavioral development approach

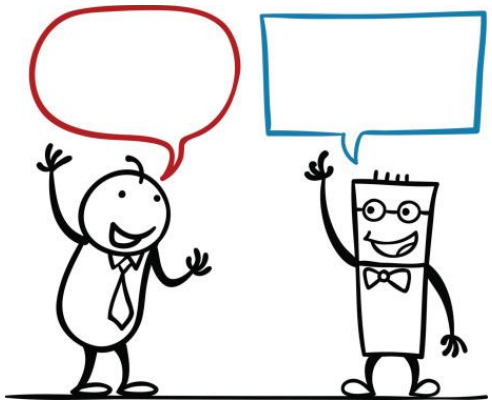
d) incremental development approach

Answer

- d) incremental development approach

Question

- What are the four lifecycle phases for SCRUM?



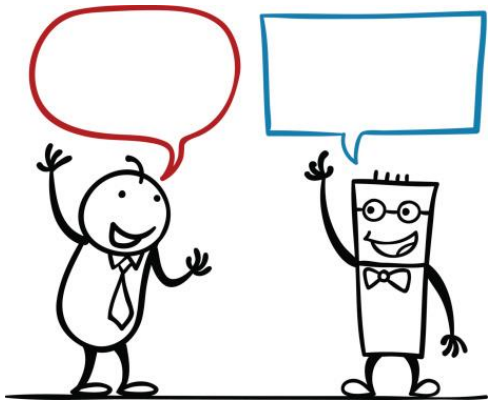
Answer

- SCRUM lifecycle includes four phases:

1. *Planning*
2. *Staging*
3. *Development*
4. *Release*

Question

- Write down the differences between Agile and Plan-Driven development



Agile Answer Plan-Driven

- Project is small
- Experienced teams with a wide range of abilities take part
- Teams are self-starters, independent leaders and others who are self-directing
- Project is an in-house project and the team co-located
- System is new with lots of unknowns
- Requirements must be discovered
- Requirements and environment are volatile with high change rates
- End-user environment is flexible
- Relationship with customer is close and collaborative
- Customer is readily available dedicated and co-located
- High trust environment exists within the development teams and customer
- Rapid value and high-responsiveness are required

- Project is large
- Teams include varied capabilities and skill sets
- Teams are geographically distributed and/or outsourced
- Project is of strategic importance
- System is well understood (scope and features set)
- Requirements are fairly stable
- System is large and complex (critical safety/high reliability requirements)
- Project stakeholders have a weak relationship with the development team
- External legal concerns
- Focus is on a strong, quantitative process improvement
- Definition and management of process are important
- Predictability and stability of process are important

Outline

- Review
- What do we mean by Pragmatics?
- Management and Planning
- Benefits and Risks of Object Orientated Methods
- Testing
- Conclusion & Discussion

Management and Planning

- Strong project leadership
- Actively manages and directs a project's activities
- Avoid project going astray
  - ▷lack of focus
  - ▷Ignoring problems

Risk Management

- Software development is responsible for non-technical risks
  - supervising the timely delivery of software from a third-party vendor
- Technical risks are typically the responsibility of the project architect
  - selection of an inheritance structure that offers the best compromise between usability and flexibility
  - choice of mechanisms that yield acceptable performance while simplifying the system's architecture

Task Planning

- Periodic team meetings
  - discuss work completed
  - activities for the coming work period
- Too many meetings
  - destroys productivity
  - sign that the project has lost its way
- Developers require unscheduled time
  - allows them to think, innovate, develop, share knowledge with other teams/individuals

Schedules

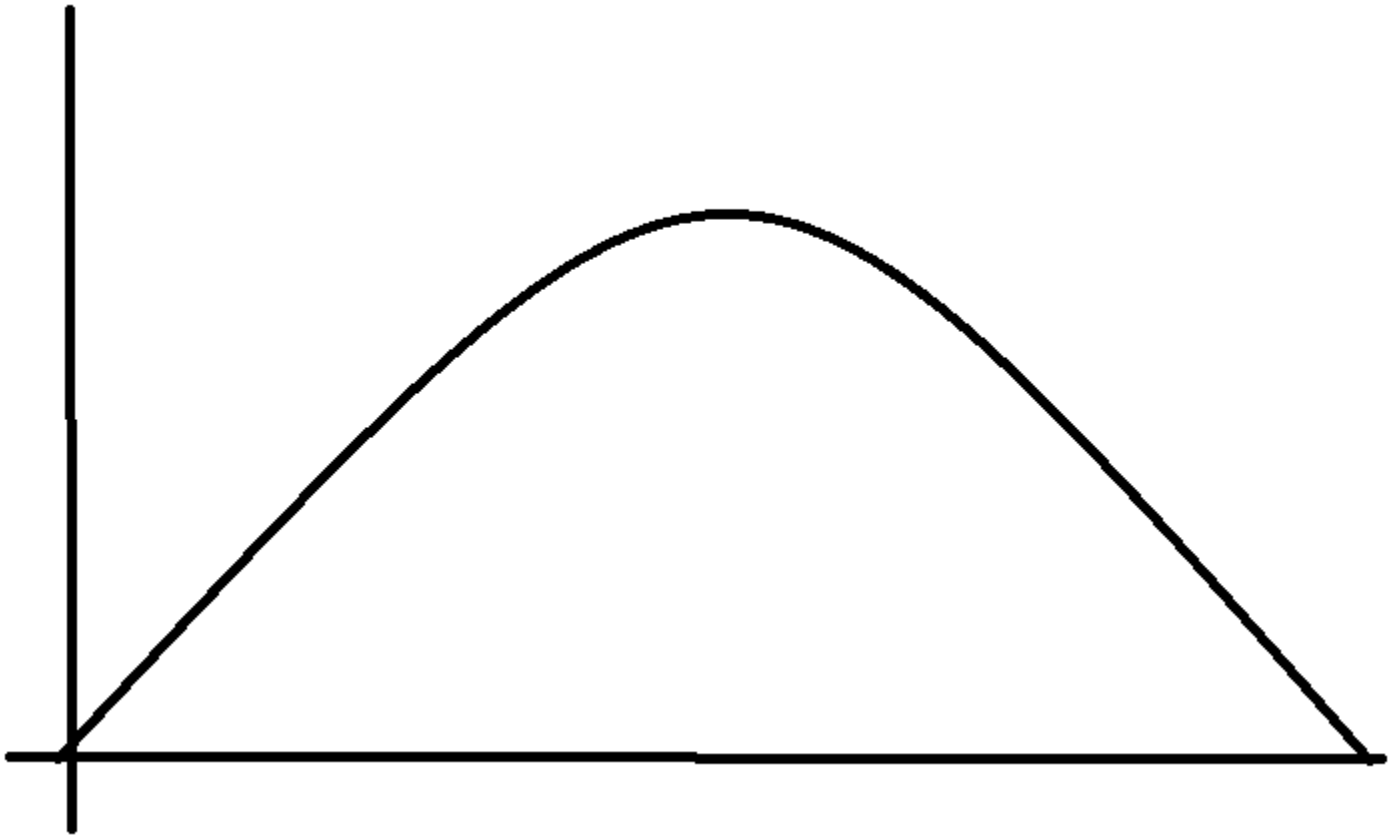
- Avoid overly optimising the schedule/plan
  - ▷at the mercy of overly optimistic planning
- Schedules/tasks change
- Require fine-tuning
- Milestones within the schedule to monitor the project

Staffing

- Timing of resources within the development cycle
- Different skillsets/abilities at different times during the project
- Account for ‘change’
  - Staff leaving/changing/training

Staff Resource Allocation

- Design, Development, Testing, …
- Lifecycle Costs



Development Team Roles

- Software development is ultimately a human endeavour
- Developers are not interchangeable parts
- Complex system requires the unique and varied skills of a focused team of people

Central

- Three roles to be central to the technical development team for an object-oriented project
  1. Project architect
  2. Component lead
  3. Application engineer

Challenges

- Breakdown of skills addresses the staffing problem faced by most software development organizations
- Handful of really skilled experienced individuals and many more less-experienced ones
- Opportunity to learn/gain experience

Configuration Management and Version Control

- Interfacing components
- Keeping track of different versions
  - stable, debug, testing, …
  - features, …
  - compatible components
  - build 03929, 03930, ..
- Avoid ‘halting’ work while newer versions of components are developed

Integration

- Integration events
- Each marking the creation of another prototype or architectural release
- Generally incremental in nature
  - (i.e., not big-bang)

Testing

- Testing should encompass at least three dimensions:
  1. *Unit testing*
  2. Component testing
  3. *System testing*

Unit Testing

- Involves testing individual classes and mechanisms and is the responsibility of the application engineer who implemented the structure

Component Testing

- Involves integration testing a complete component and is the responsibility of the component lead.
- Component tests can be used as regression tests for each newly released version of the component
- Note that the term component is generic and can mean a single component in a small project or a collection of components, sometimes referred to as a subsystem, in a larger project

System Testing

- Involves integration testing the system as a whole and is the responsibility of the quality assurance team
- System tests are also typically used as regression tests by the integration team when assembling new releases

Reuse

- Acclaimed benefits of object-oriented development is reuse
- Depends on management
- Realize the benefits of reusing the many artifacts of the development process

Elements of Reuse

- Any artifact of software development can be reused
- Any stage, e.g., requirement, development and testing
- For example, design, code and documentation
  - reuse patterns of classes, objects, and designs in the form of idioms, mechanisms, and frameworks

Question

- Documentation should drive the development process

a) True
b) False

Answer

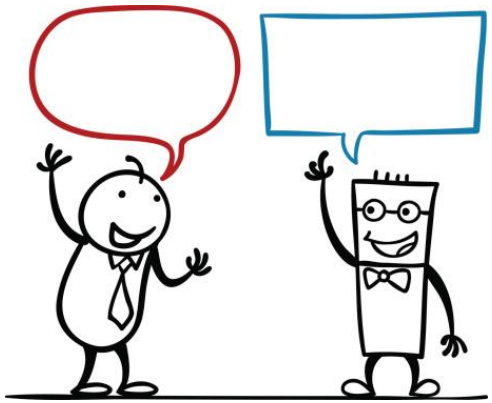
- b) False

Documentation should never drive the development process

Institutionalizing Reuse

- Reuse within a project or even within an entire organization doesn't just happen

WHY?



Institutionalizing Reuse

- Encouraged
- Opportunities for reuse must be actively sought out and rewarded
- Pattern scavenging as an explicit activity in the development process
- Making specific individuals responsible for leading the reuse activity

Long Term

- Reuse costs resources in the short term but pays off in the long term
- Organization that takes a long-term view of software development and optimizes resources

Quality Assurance and Metrics

- Systematic activities providing evidence of the fitness for use of the total software product

Software Quality

- Software quality as the fitness for use of the total software product
- Software quality doesn't just happen: It must be engineered into the system

Object-Oriented Metrics

- Lord Kelvin:

"When you can measure what you are speaking about, and express it into numbers, you know something about it

Process & Product Metrics

- Metrics to assist us in this endeavor fall into one of two categories, process metrics or product metrics
- Process metrics, sometimes called project metrics, assist the management team in assessing progress with respect to the object-oriented development process being used

Process Metrics

- Application size
  - Number of scenario scripts (NSS)
  - Number of key classes (NKC)
  - Number of support classes (NSC)
  - Number of subsystems (NOS)
- Staffing size
  - Person-days per class (PDC)
  - Classes per developer (CPD)
- Scheduling
  - Number of major iterations (NMI)
  - Number of contracts completed (NCC)

Documentation

- Development artifacts that are critical to the complete lifecycle of a software system (other than code)
- Example, requirements and design, must be documented to support the development process and the operation and maintenance of the system

Tools

- Trying to build a large software system with a minimal tool set is equivalent to building a multi-story building with stone hand tools
- Examples
  - visual modeling tool supporting the UML notation
  - version control

Benefits of Object-Oriented Development

- Appeals to the working of human cognition
- Leads to systems that are more resilient to change
- Encourages the reuse of software components
- Reduces development risk
- Exploits the expressive power of object-oriented programming languages

Risks of Object-Oriented Development

- A unique way to define architecture and data structure instances
- Information hiding through abstraction and encapsulation
- Inheritance to organize related elements
- Polymorphism to perform operations that can automatically adapt to the type of structure they operate on
- Specialized analysis and design methods
- Object-oriented languages
- Environments that facilitate the creation of object-oriented systems
- Design by contract, a powerful technique to circumvent module boundary and interface problems

8 Software Risks

1. Personnel shortfalls
2. Unrealistic schedules, budgets, or processes
3. Shortfalls in commercial off-the-shelf products, external components, or legacy software
4. Mismatches in requirements or user interface
5. Shortfalls in architecture, performance, or quality
6. Continuing stream of requirements changes
7. Shortfalls in externally performed tasks
8. Straining computer science

Summary

- Clear idea of Pragmatics in Object Orientated Analysis and Design
- See bigger picture (i.e., much more than just generating code)
- Reuse must be institutionalized to be successful

This Week

- Review Slides
- Read Chapter 8

Questions/Discussion

“Would you like the technical or non-technical explanation?”

Question

- There are many benefits to object-oriented technology and no risks

a) True
b) False

Answer

- b) False