Foundations for Systems Development:

Other Approaches

Prepared by:

Hiranya Prasad Bastakoti

Contents

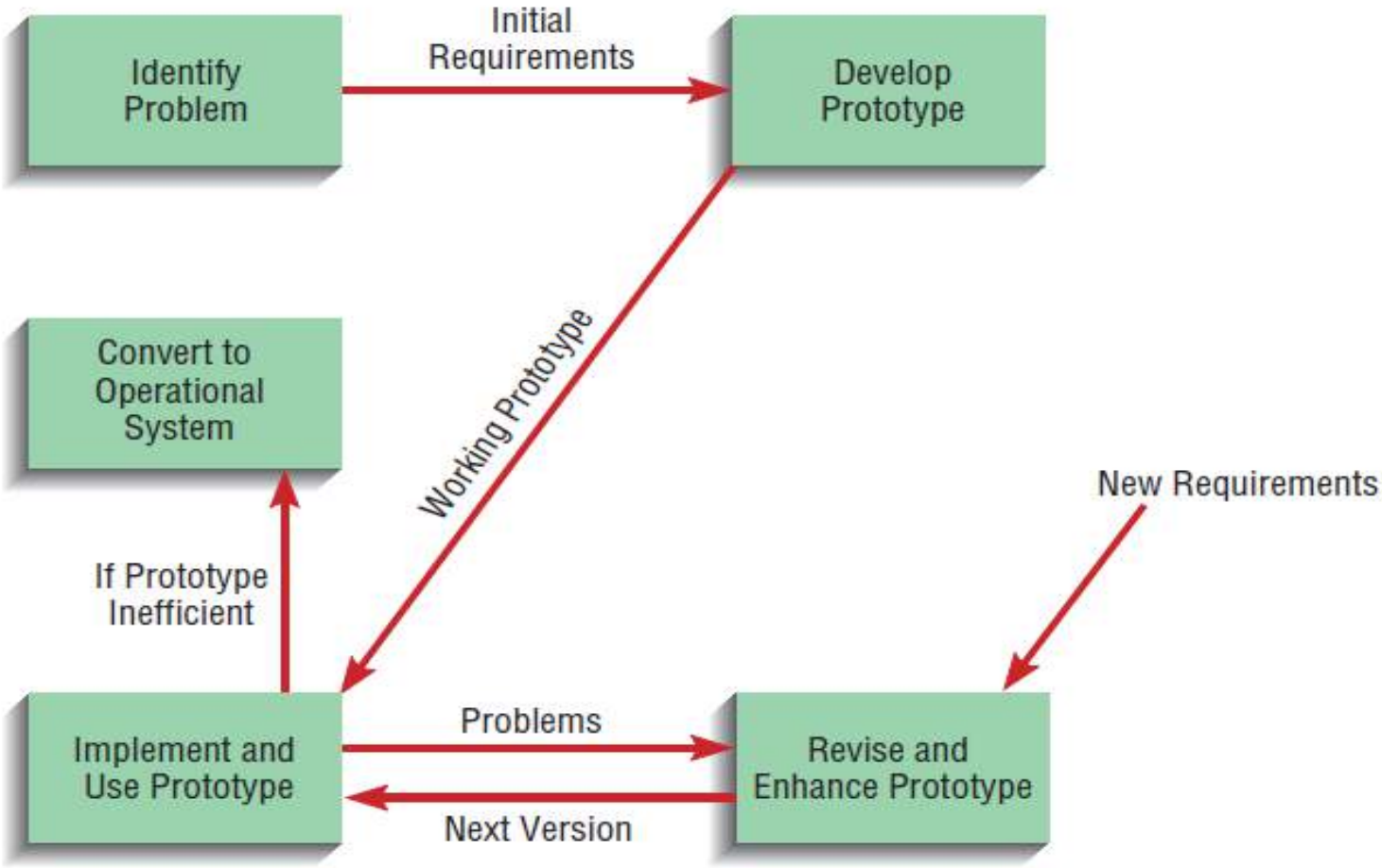
- Prototyping
- Spiral
- Rapid Application Development
- Introduction to Agile Development

Prototyping

- Iterative development process:
- Requirements quickly converted to a working system.
- System is continually revised.
- Close collaboration between users and analysts.
- **Is a form of Rapid Application Development.**
- **Building a scaled-down working version of the system**
- **Advantages:**
  - **Users are involved in design**
  - **Captures requirements in concrete form**

Contd..

- Designing and building a scaled-down but working version of a desired system is known as **prototyping.**
- A prototype can be developed with a computer-aided software engineering (CASE) tool, a software product that automates steps in the systems development life cycle.
- CASE tools make prototyping easier and more creative by supporting the design of screens and reports and other parts of a system interface.
- CASE tools also support many of the diagramming techniques you will learn, such as data-flow diagrams and entity-relationship diagrams.


Identify Problem
Initial Requirements
Develop Prototype
Working Prototype
Convert to Operational System
If Prototype Inefficient
New Requirements
Implement and Use Prototype
Problems
Next Version
Revise and Enhance Prototype

- It can be defined as an interactive process for *systems development* in which users and analysts are in close collaboration for *converting requirements* to a working system that is continuously revised.
- Prototyping can be used as:
  - Alternative to the 'traditional' SDLC approach
  - Technique for gathering information in the requirements analysis phase of the SDLC (help to find what the user really wants)
- Prototyping is a complex technique and to apply it successfully, detailed knowledge of the SDLC is required.
- Prototyping helps to set priorities and adapt the planning to changes in requirements with minimum disruption

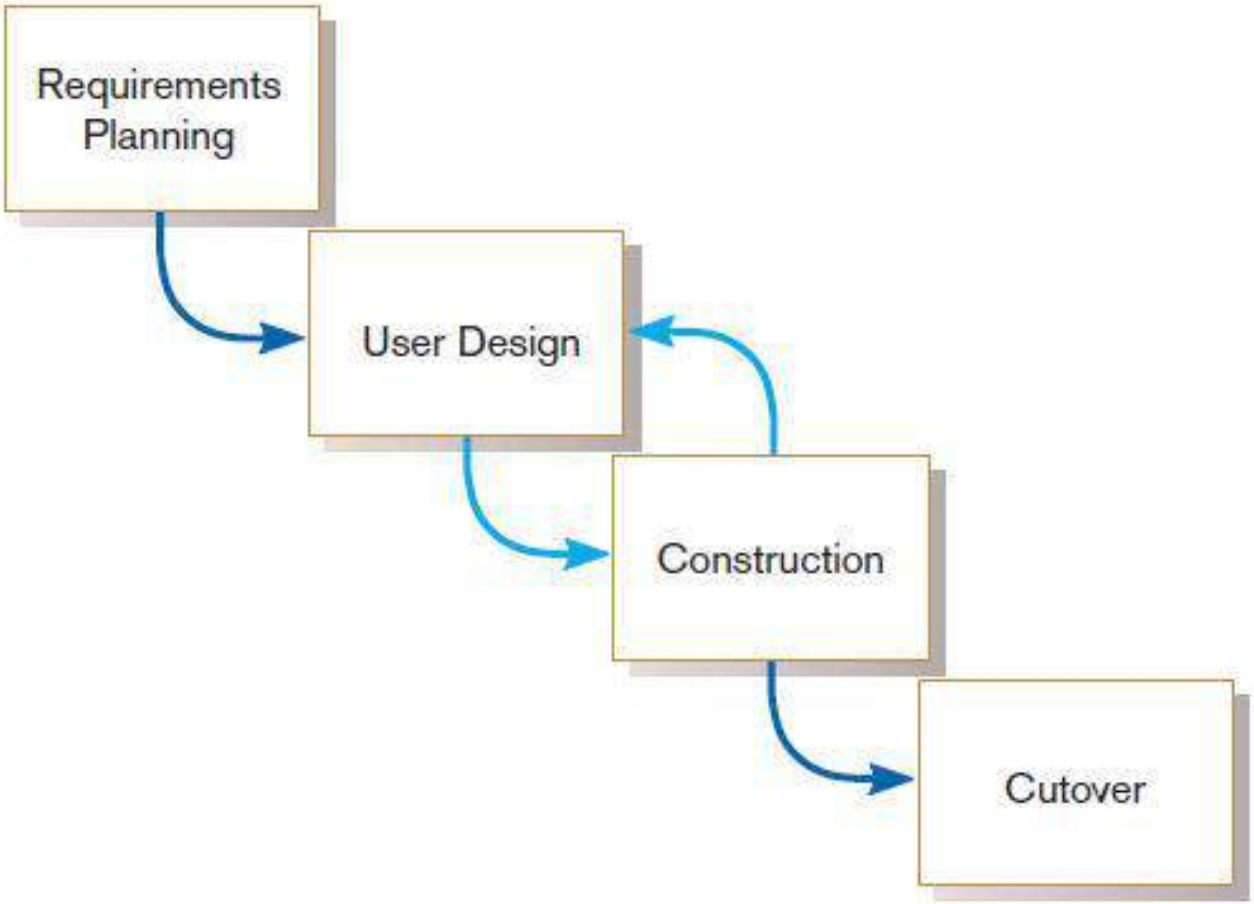
- Advantages of prototypes:
  - Potential for changes to the system early in the development.
  - Opportunity to stop developing a non-working system.
  - Possibility of developing system that closely addresses users' requirements and expectations.

- Disadvantages of prototypes:
  - Prototyping is difficult to manage.
  - Increase cost

Rapid Application Development

- **RAD Model** or Rapid Application Development Methodology is a system/software development process based on prototyping without any specific planning.
- In RAD model, there is less attention paid to the planning and more priority is given to the development tasks.
- It targets at developing software/system in a short span of time.
- **System methodology to radically decrease the time needed to design and implement information systems.**
- **It relies on extensive user involvement.**
- **RAD cycle is limited to the design, construction, and development phases.**
- Systems development methodology created to radically decrease the time needed to design and implement information systems.
- RAD relies on extensive user involvement, prototyping, integrated CASE tools, and code generators

RAD



- Requirement Planning: Involves identifying and analyzing project objectives and requirements.
- It focuses on gathering input from stakeholders to establish a clear understanding of the syetem's scope and purpose.
- User Design: The development team creates a prototype or mock-up of the system's user interface and experience.
- Feedback from users is actively sought and incorporated to ensure the design meets their expectations and needs.
- Construction: The construction phase is where the actual development of the software takes place.
- Developers utilize the feedback received during the user design phase to rapidly build the application, prioritizing the delivery of working features.
- Cutover: The fully functional system is thoroughly tested, integrated, and deployed to the production environment.
- This phase involves transitioning from the development environment to the live environment, ensuring the system is stable and ready for end-users.

Joint Application Design (JAD)

- JAD (Joint Application Design) is a collaborative and structured process that brings together key stakeholders, including users, managers, and analysts, for several-day intensive workgroup sessions to specify or review system requirements.
- During these sessions, JAD facilitates open communication and active participation among stakeholders, aiming to elicit, analyze, and document detailed system requirements.
- The purpose of JAD is to ensure a comprehensive and accurate understanding of stakeholders' needs, leading to the successful development and implementation of technology solutions.

1.11

- Joint application design (JAD) is a group-based method for collecting user requirements and creating system designs.
- It is used within the systems analysis and design stages of the SDLC.
- Unlike the traditional SDLC, where the analysts interview individual users of the new information system to understand their needs JAD has a meeting in which all users meet simultaneously with analysts.
- During the meeting, all users jointly define and agree upon systems requirements.

- **Advantages:**
  - Saves time
  - Greater support for, and acceptance of new systems
  - Produces higher quality systems
  - Easier implementation
  - Lower training costs
- **Disadvantages:**
  - Very difficult to get all users to JAD meetings
  - All the problems that may be caused by any group process

AGILE Methodologies

- Agile methodologies challenge conventional engineering-based approaches by acknowledging the unique nature of system/software development compared to civil engineering.
- System/Software requirements are less defined, and the development process is fluid and unpredictable.
- The core principles of Agile emphasize adaptability over predictability, prioritizing the importance of people over rigidly defined roles.
- Agile promotes self-adaptive processes that evolve as the software is developed and reviewed by project participants.
- Agile is particularly well-suited for projects with dynamic or uncertain requirements. It thrives with the contribution of responsible and motivated developers who collaborate in cross-functional teams.
- The active involvement of engaged customers is essential for the success of Agile projects.
- Customers who understand and actively participate in the Agile process provide valuable feedback throughout the development cycle

AGILE

- Agile methodologies, it argues that software development methodologies adapted from engineering generally do not fit with real world software development.
- In civil engineering requirements tends to be well understood, construction become very predictable.
- Motivated by recognition of software development as fluid, unpredictable, and dynamic.
- Three key principles
  - Adaptive rather than predictive.
  - Emphasize people rather than roles.
  - Self-adaptive processes, as software is developed, the process used should be refined and improved of course after reviewed by people working on the project.

Agile is not for every project, it is for:

- unpredictable or dynamic requirement.
- responsible and motivated developers.
- customers who understand the process and will get involved.

TABLE 1-3 The Agile Manifesto

The Manifesto for Agile Software Development

Seventeen anarchists agree:

We are uncovering better ways of developing software by doing it and helping others 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 we value the items on the right, we value the items on the left more. We follow the following principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Businesspeople and developers work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Continuous attention to technical excellence and good design enhances agility.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

—Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas (*www.agileAlliance.org*)

[*Source: http://agilemanifesto.org/* © 2001, the above authors. This declaration may be freely copied in any form, but only in its entirety through this notice.]

The Agile Methodologies group argues that software development methodologies adapted from engineering generally do not fit with real-world software development.

When to use Agile Methodologies

- If your project involves:
  - Unpredictable or dynamic requirements
  - Responsible and motivated developers
  - Customers who understand the process and will get involved

TABLE 1-4 Five Critical Factors That Distinguish Agile and Traditional Approaches to Systems Development

Factor	Agile Methods	Traditional Methods
Size	Well matched to small products and teams. Reliance on tacit knowledge limits scalability.	Methods evolved to handle large products and teams. Hard to tailor down to small projects.
Criticality	Untested on safety-critical products. Potential difficulties with simple design and lack of documentation.	Methods evolved to handle highly critical products. Hard to tailor down to products that are not critical.
Dynamism	Simple design and continuous refactoring are excellent for highly dynamic environments but a source of potentially expensive rework for highly stable environments.	Detailed plans and Big Design Up Front, excellent for highly stable environment but a source of expensive rework for highly dynamic environments.
Personnel	Requires continuous presence of a critical mass of scarce experts. Risky to use non-agile people.	Needs a critical mass of scarce experts during project definition but can work with fewer later in the project, unless the environment is highly dynamic.
Culture	Thrives in a culture where people feel comfortable and empowered by having many degrees of freedom (thriving on chaos).	Thrives in a culture where people feel comfortable and empowered by having their roles defined by clear practices and procedures (thriving on order).

(*Source:* Boehm, Barry; Turner, Richard, *Balancing Agility and Discipline: A Guide for the Perplexed*, 1st Ed., © 2004. Reprinted and electronically reproduced by permission of Pearson Education, Inc.)

Extreme Programming (XP)

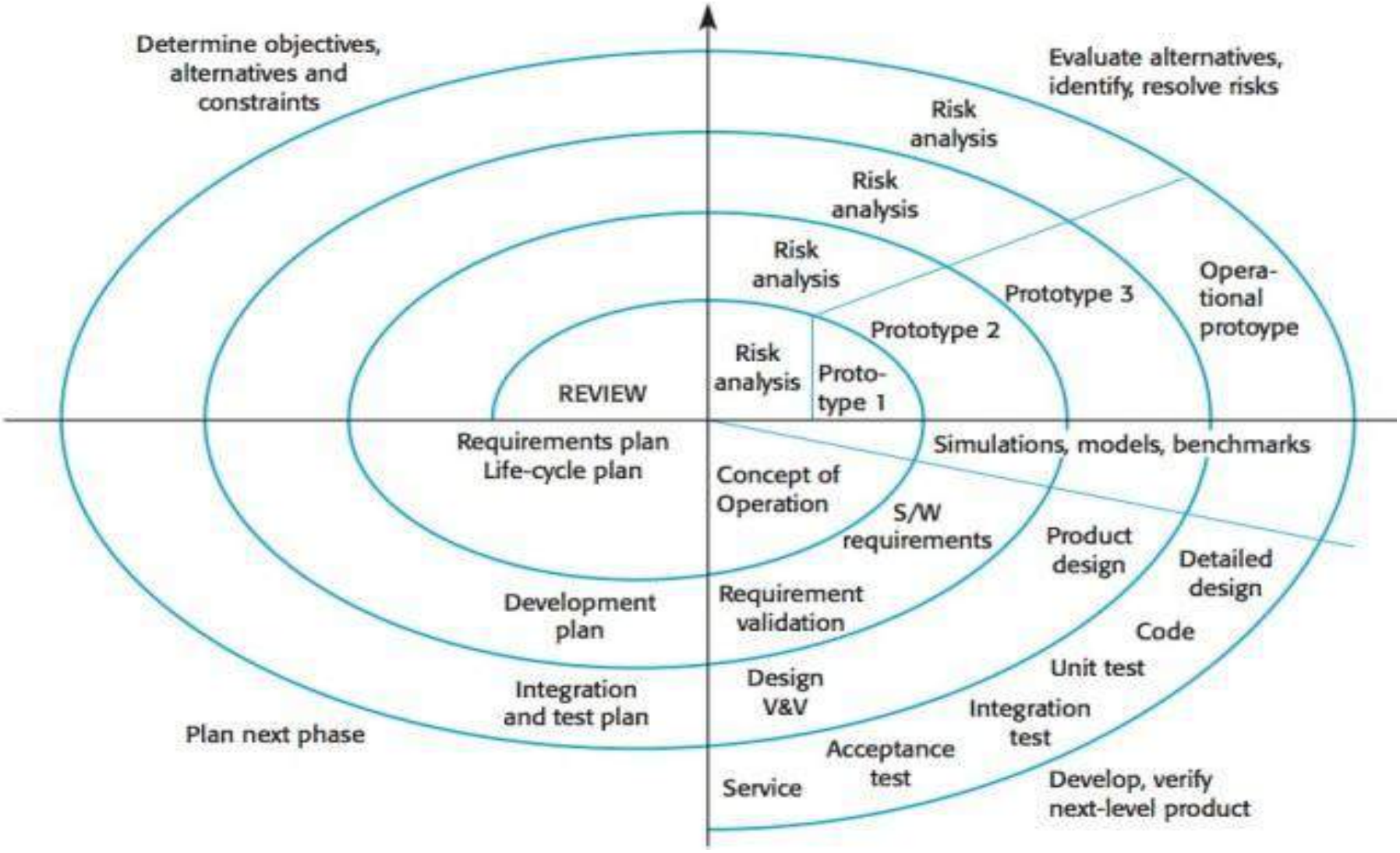
- Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, and higher quality of life for the development team.
- XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.

When Applicable

- The general characteristics where XP is appropriate were described by Don Wells on www.extremeprogramming.org:
- Dynamically changing software requirements
- Risks caused by fixed time projects using new technology
- Small, co-located extended development team
- The technology you are using allows for automated unit and functional tests

Spiral Development

- The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model.
- This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis.
- It allows incremental releases of the product or incremental refinement through each iteration around the spiral.
- The exact number of loops of the spiral is unknown and can vary from project to project. **Each loop of the spiral is called a Phase of the software development process.**


Determine objectives, alternatives and constraints
Evaluate alternatives, identify, resolve risks
Risk analysis
Risk analysis
Risk analysis
Risk analysis
Prototype 3
Prototype 2
Opera-tional protoype
Proto-type 1
REVIEW
Requirements plan
Life-cycle plan
Simulations, models, benchmarks
Concept of Operation
S/W requirements
Product design
Detailed design
Development plan
Requirement validation
Code
Unit test
Integration and test plan
Design V&V
Integration test
Plan next phase
Acceptance test
Service
Develop, verify next-level product

- The functions of these four quadrants are discussed below-
- **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
- **Identify and resolve Risks:** During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution

- **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
- **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started

Advantages:

- **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
- **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
- **Flexibility in Requirements:** Change requests in the Requirements at later phase can be incorporated accurately by using this model.
- **Customer Satisfaction:** Customer can see the development of the product at the early phase of the software development and thus, they familiar with the system by using it before completion of the total product.

Disadvantages

- **Complex:** The Spiral Model is much more complex than other SDLC models.
- **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
- **Too much dependable on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced expertise, it is going to be a failure to develop a project using this model.
- **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.

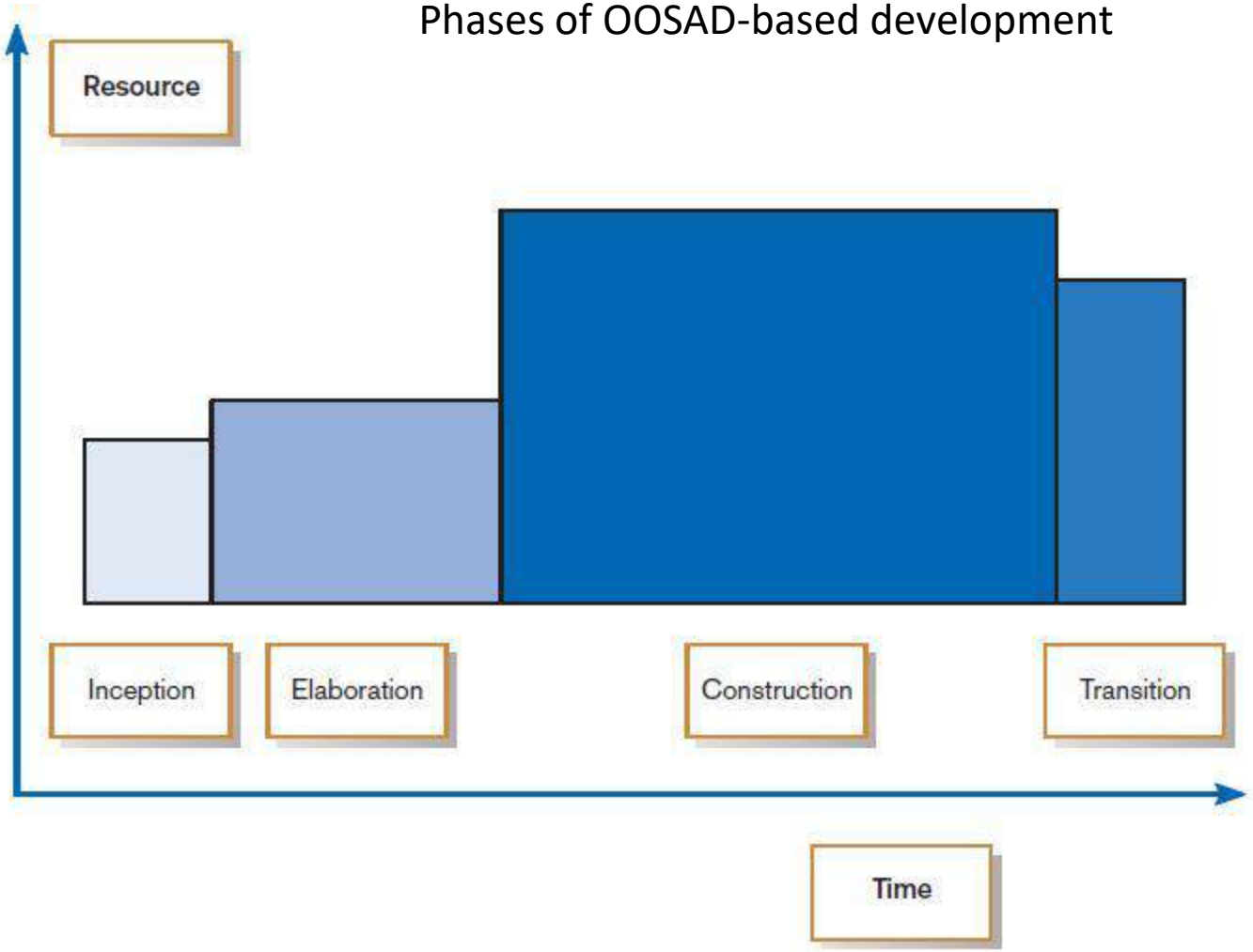
Object-Oriented Analysis and Design (OOAD) (Cont.)

- **Object class**: a logical grouping of objects sharing the same attributes and behaviors
- **Inheritance**: hierarchical arrangement of classes enable subclasses to inherit properties of superclasses

Rational Unified Process (RUP)

- An object-oriented systems development methodology
- RUP establishes four phase of development: inception, elaboration, construction, and transition.
- Each phase is organized into a number of separate iterations.

FIGURE 1-13
Phases of OOSAD-based development



Review Questions

1. Explain what is meant by Prototyping approches.Write its advantages and disadvantages.
2. What are the Phases of RAD?
3. What are the major activities of spiral model?
4. Explain what is meant by Agile Methodologies.
5. What is eXtreme Programming?
6. When would you use Agile Methodologies Vs Traditional Methodologies to System development?