Ideally, software is developed as described in Chapter 1 (Figure 2.1)
- Linear
- Starting from scratch

In the real world, software development is totally different
- We make mistakes
- The client’s requirements change while the software product is being developed

Set up public transportation system to reduce traffic congestion
- Use bus-only lanes and park-and-ride lots
- Each bus fare machine accepts only dollar bills
- Response must be fast (avg. < 1sec) and accurate (avg. 98% correct)

Episode 1: The first version is implemented
Episode 2: A fault is found
- The product is too slow (avg. 10sec) because of an implementation fault (double precision vs single precision)
- Changes to the implementation are begun
Episode 3: A new design is adopted
- Change will still be too slow. A faster algorithm is used

Episode 4: The requirements change
- Want to sell as a product to defray cost overrun
- Accuracy must be increased to 99.5% 
- Finally, installation for city is complete
Epilogue: A few years later, these problems recur
- New hardware => new software
- New software => new design and programming language to be “better”

The linear life cycle model with feedback loops (Figure 2.3)
- The waterfall model cannot show the order of events
Evolution-Tree Model
Winburg Mini Case Study (Figure 2.2)

- The explicit order of events is shown
- At the end of each episode
  - We have a baseline, a complete set of artifacts (constituent components)
- Example:
  - Baseline at the end of Episode 3:
    - Requirements, Analysis, Design, Implementation

Lessons of the Winburg Mini Case Study

- In the real world, software development is more chaotic than the Winburg mini case study
- Changes are always needed
  - A software product is a model of the real world, which is continually changing
  - Software professionals are human, and therefore make mistakes

Teal Tractors Mini Case Study

- While its ERP software product is being constructed, the requirements change: The company is expanding into Canada
- Changes needed include:
  - Additional sales regions must be added
  -Must be able to handle Canadian taxes and other business aspects that are handled differently
  - Must be extended to handle two different currencies, US$ and CDN$
- These changes may be
  - Great for the company; but
  - Disastrous for the software product

Moving Target Problem

- A change in the requirements while the software product is being developed
- Even if the reasons for the change are good, the software product can be adversely impacted
  - Dependencies will be induced

Moving Target Problem (2)

- Any change made to a software product can potentially cause a regression fault
  - A fault in an apparently unrelated part of the software
- If there are too many changes
  - The entire product may have to be redesigned and reimplemented
Moving Target Problem (3)

- Change is inevitable
  - Growing companies are always going to change
  - If the individual calling for changes has sufficient clout, nothing can be done about it
- There is no solution to the moving target problem

Iteration and Incrementation

- In real life, we cannot talk about “the analysis phase”
  - Instead, the operations of the analysis phase are spread out over the life cycle
- The basic software development process is iterative
  - Each successive version is intended to be closer to its target than its predecessor

Iteration and Incrementation

- Iteration and incrementation are used in conjunction with one another
  - There is no single “requirements phase” or “design phase”
  - Instead, there are multiple instances of each phase
- The number of increments will vary — e.g., Winburg Mini Case Study has four

Miller’s Law

- At any one time, we can concentrate on only approximately seven chunks (units of information)
- To handle larger amounts of information, use stepwise refinement
  - Concentrate on the aspects that are currently the most important
  - Postpone aspects that are currently less critical
  - Every aspect is eventually handled, but in order of current importance
- This is an incremental process

Iteration and Incrementation (2)

- Iteration and incrementation are used in conjunction with one another
  - There is no single “requirements phase” or “design phase”
  - Instead, there are multiple instances of each phase
- The number of increments will vary — e.g., Winburg Mini Case Study has four

Classical Phases versus Workflows

- Sequential phases do not exist in the real world
- Instead, the five core workflows (activities) are performed over the entire life cycle
  - Requirements workflow
  - Analysis workflow
  - Design workflow
  - Implementation workflow
  - Test workflow

Iteration and Incrementation (3)

![Diagram showing the comparison between phases and workflows](image-url)
Workflows

- All five core workflows are performed over the entire life cycle
- However, at most times one workflow predominates
- Examples:
  - At the beginning of the life cycle
    - The requirements workflow predominates
  - At the end of the life cycle
    - The implementation and test workflows predominate
- Planning and documentation activities are performed throughout the life cycle

Iteration is performed during each incrementation (Figure 2.5)

Winburg Mini Case Study Revisited

- Consider the next slide (Figure 2.6)
- The evolution-tree model has been superimposed on the iterative-and-incremental life-cycle model
- The test workflow has been omitted — the evolution-tree model assumes continuous testing

Winburg Mini Case Study Revisited (2)

- More on Incrementation
  - Each episode corresponds to an increment
  - Not every increment includes every workflow
  - Increment B was not completed
  - Dashed lines denote maintenance
    - Episodes 2, 3: Corrective maintenance
    - Episode 4: Perfective maintenance

Other Aspects of Iteration and Incrementation

- We can consider the project as a whole as a set of mini projects (increments)
- Each mini project extends the
  - Requirements artifacts
  - Analysis artifacts
  - Design artifacts
  - Implementation artifacts
  - Testing artifacts
- The final set of artifacts is the complete product
Other Aspects of Iteration and Incrementation

- During each mini project we
  - Extend the artifacts (incrementation);
  - Check the artifacts (test workflow); and
  - If necessary, change the relevant artifacts (iteration)
- Each iteration can be viewed as a small but complete waterfall life-cycle model

Strengths of the Iterative-and-Incremental Model

- There are multiple opportunities for checking that the software product is correct
  - Every iteration incorporates the test workflow
  - Faults can be detected and corrected early
- The robustness of the architecture can be determined early in the life cycle
  - Architecture — the various component modules and how they fit together
  - Robustness — the property of being able to handle extensions and changes without falling apart

Strengths of the Iterative-and-Incremental Model (2)

- We can mitigate (resolve) risks early
  - Risks are invariably involved in software development and maintenance
- We have a working version of the software product from the start
  - The client and users can experiment with this version to determine what changes are needed
- Variation: Deliver partial versions to smooth the introduction of the new product in the client organization

Strengths of the Iterative-and-Incremental Model (3)

- There is empirical evidence that the life-cycle model works
- The CHAOS reports show that the percentage of successful products increases (Figure 2.7)

Strengths of the Iterative-and-Incremental Model (4)

- Reasons given for the decrease in successful projects in 2004 include:
  - More large projects in 2004 than in 2002
  - Use of the waterfall model
  - Lack of user involvement
  - Lack of support from senior executives
Managing Iteration and Incrementation

- The iterative-and-incremental life-cycle model is as regimented as the waterfall model ...
- ... because the iterative-and-incremental life-cycle model *is* the waterfall model, applied successively
- Each increment is a waterfall mini project