System Integration

Mini Case Studies

System Integration Life Cycles

Shawn A. Butler, Ph.D.
Senior Lecturer, Executive Education Program
Institute for Software Research
Carnegie Mellon University

Lecture Objectives

- Describe the system integration life cycles and potential red flags in each phase
- Learn how to select a system development life cycle for a system integration project
- Understand how system integration life cycles differ from traditional software development life cycles
- See the common phases across all life cycles

SDLC Phases

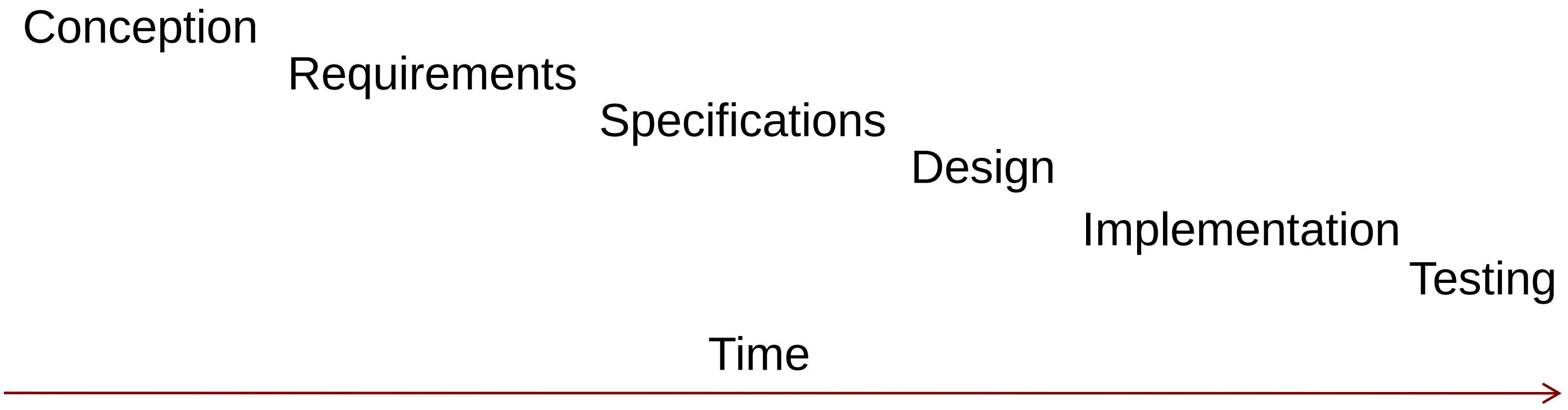
- Conception – What are the goals and objectives of the integrated system? What is the vision?
- Requirements Analysis – What are the requirements of the integrated system
- Specification – What are the system detailed specifications?
- Design – What are the components and how will they communicate?

SDLC Phases cont’d

- Implementation – Integrate!
- Testing – testing and more testing
- Training and Support – Make sure the users know how to properly use what they paid for!
- Maintenance – Fix bugs and make changes as necessary
- Retirement – This almost never happens!

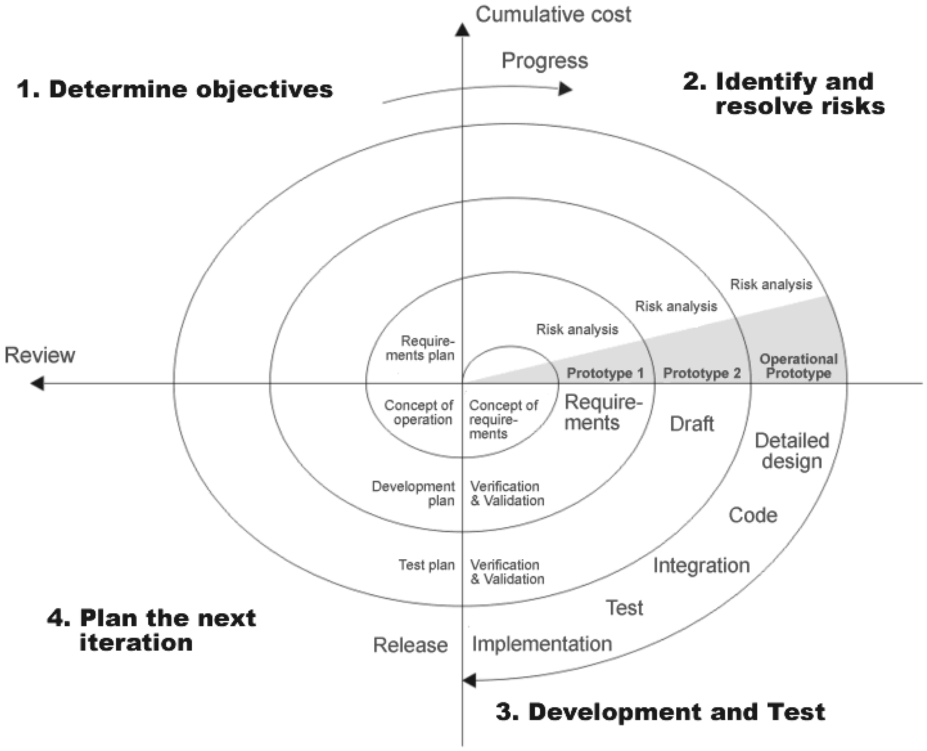
SILC: Waterfall

- User role
- Very structured approach
- Still strong in USA government system integration tasks
- Rarely works in practice



SILC: Spiral

- Emphasis on backtracking and iteration
- Eventually prototype developed
- Prototype seldom thrown away



SILC: Rapid Application Development

- Small highly collaborative teams develop increasingly functional prototypes
- Prototype used to develop user’s vision and usefulness
- Thrown away after completed
- Prototype never seems to really get thrown away
- Prototype often lacks complete functionality

SILC: Agile

- Emphasis on user involvement
- Very short increments develop
- Requirements and vision may not be well developed
- Highly flexible
- Design into a corner
- Difficult to integrate security
- Often no documentation

SILC

All SDLC's go through every phase –
the order, emphasis, and implementation may differ

System and Software Development Life Cycles

- Requirements and Specification:
  - Must determine which legacy systems to access
  - Determine how to access the legacy system
  - Identify or coordinate changes to legacy systems
  - Resolve redundancy issues, i.e., which system is the data source
  - Identify impact of updates if applicable
  - Identify process changes
  - Identify maintenance and change processes

System and Software Development Life Cycles

- Testing
  - Far more complicated than developing a single application
  - Design tests that are within system integrator’s control
  - Data does not adhere to specifications
  - Impossible to test all scenarios – must select carefully
  - Can’t always replicate system in a test environment
  - Difficult to problem solve errors because of complexity

System and Software Development Life Cycles

- Maintenance and Retirement
  - Maintenance complex and requires careful coordination with legacy owners
  - Testing bug fixes and adding functionality makes testing more challenging
  - Changes in legacy systems often affect overall system and aren’t usually thoroughly tested
  - Legacy systems may need to remain in place until new systems are completely integrated

Reality and SDLC's

- Waterfall is still used by the USA government and very large software development projects in commercial companies – despite an abundance of evident failures
- Process is emphasized over, and as a substitution, for engineering – both are needed
- Metrics do not always let you know that you have met the user's vision
- Strict adherence to any SDLC will get you in just as much trouble as too much process
- Prototypes are rarely thrown away

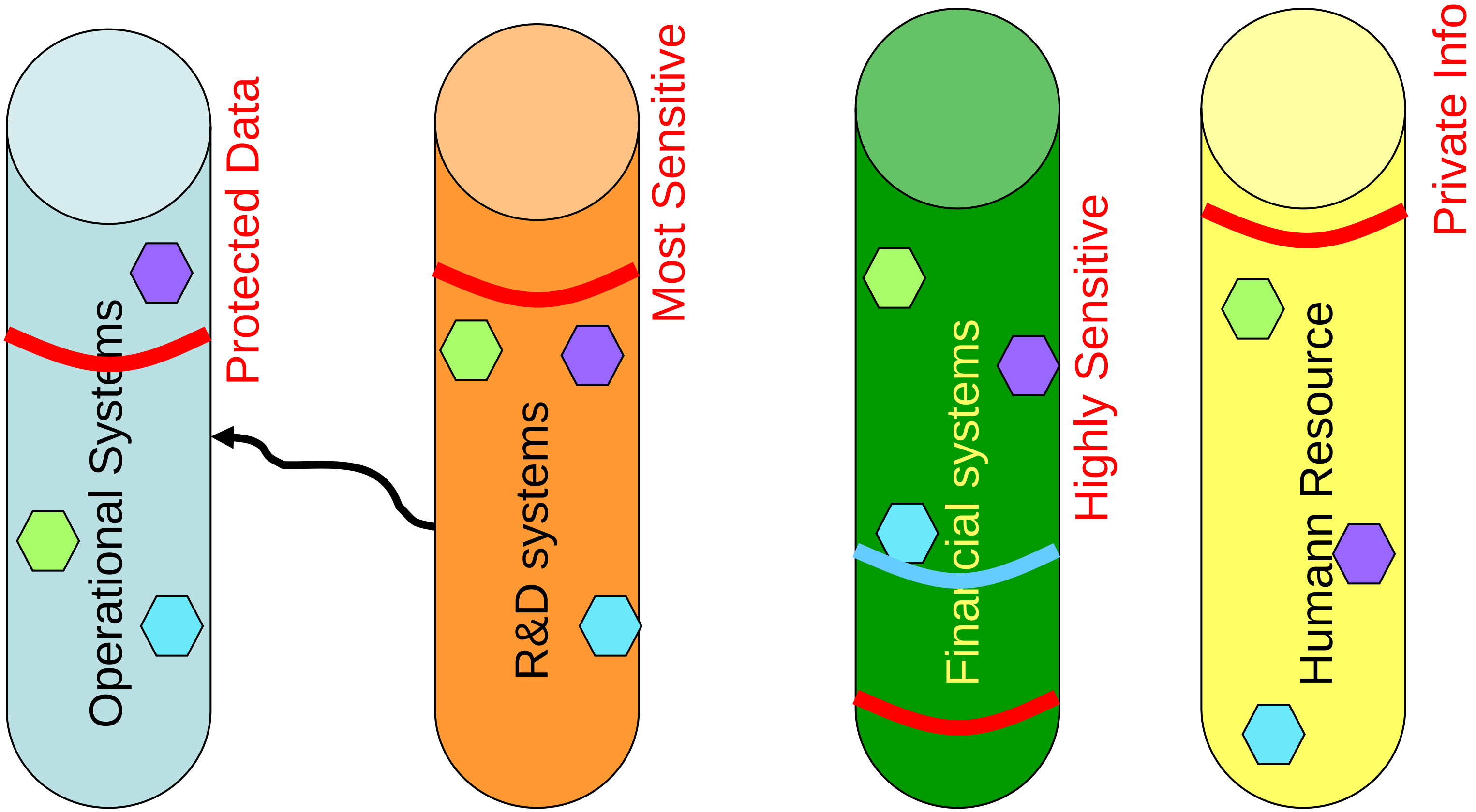
Selecting a SDLC

- Ability to handle rapidly changing (unstable) set of requirements
- Ability to manage change orders in a cost effective manner (cost of refactoring)
- Emphasis on quality measurement (unit testing/defect)
- Complexity/Size of product
- Customizability/Flexibility of approach
- Suitability for small development team

Selecting a SDLC cont'd

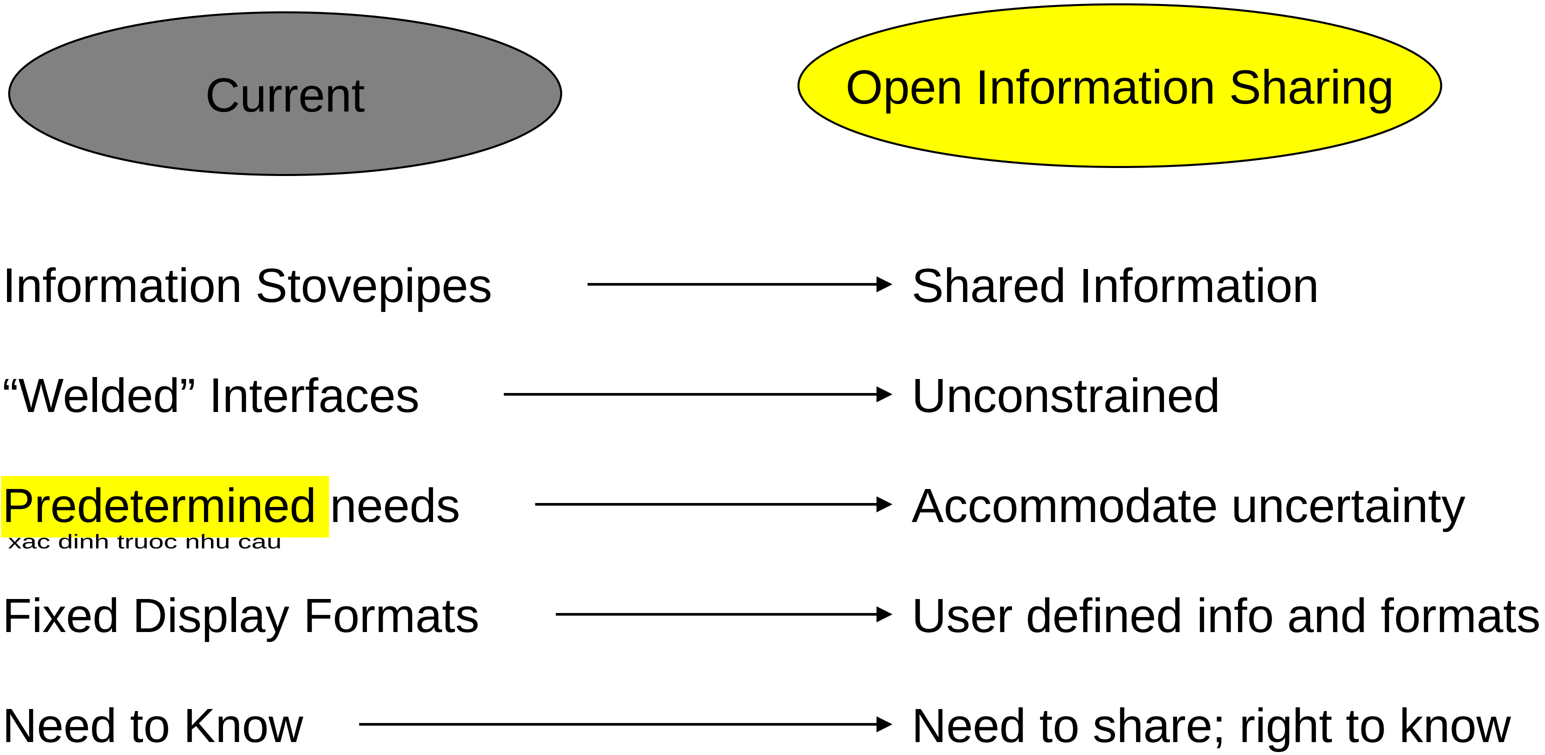
- Compatibility with distributed teams
- Built-in support for prototyping
- Learning curve
- Availability of tools
- Implementation cost
- Market adoption
- Non-reliance on external elements/resources

Current Enterprise Systems



© 2010 CMU-ISR

Now and Then



Current → Open Information Sharing

Information Stovepipes → Shared Information

“Welded” Interfaces → Unconstrained

Predetermined needs → Accommodate uncertainty

xac dinh truoc nhu cau

Fixed Display Formats → User defined info and formats

Need to Know → Need to share; right to know

Heuristic 2

Build and maintain options as long as possible in the design and implementation of complex systems. You will need them.

Summary

- Different types of system integration life cycles:
  - Waterfall
  - Rapid Application Development
  - Spiral
  - Agile
- All life cycles have same stages, only differ in emphasis, duration, and timing
- System integration life cycle differs from Software Development Life Cycle:
  - Legacy system analysis
  - Accessibility of legacy information constrains the design
  - Testing!!!!!
  - Dependencies