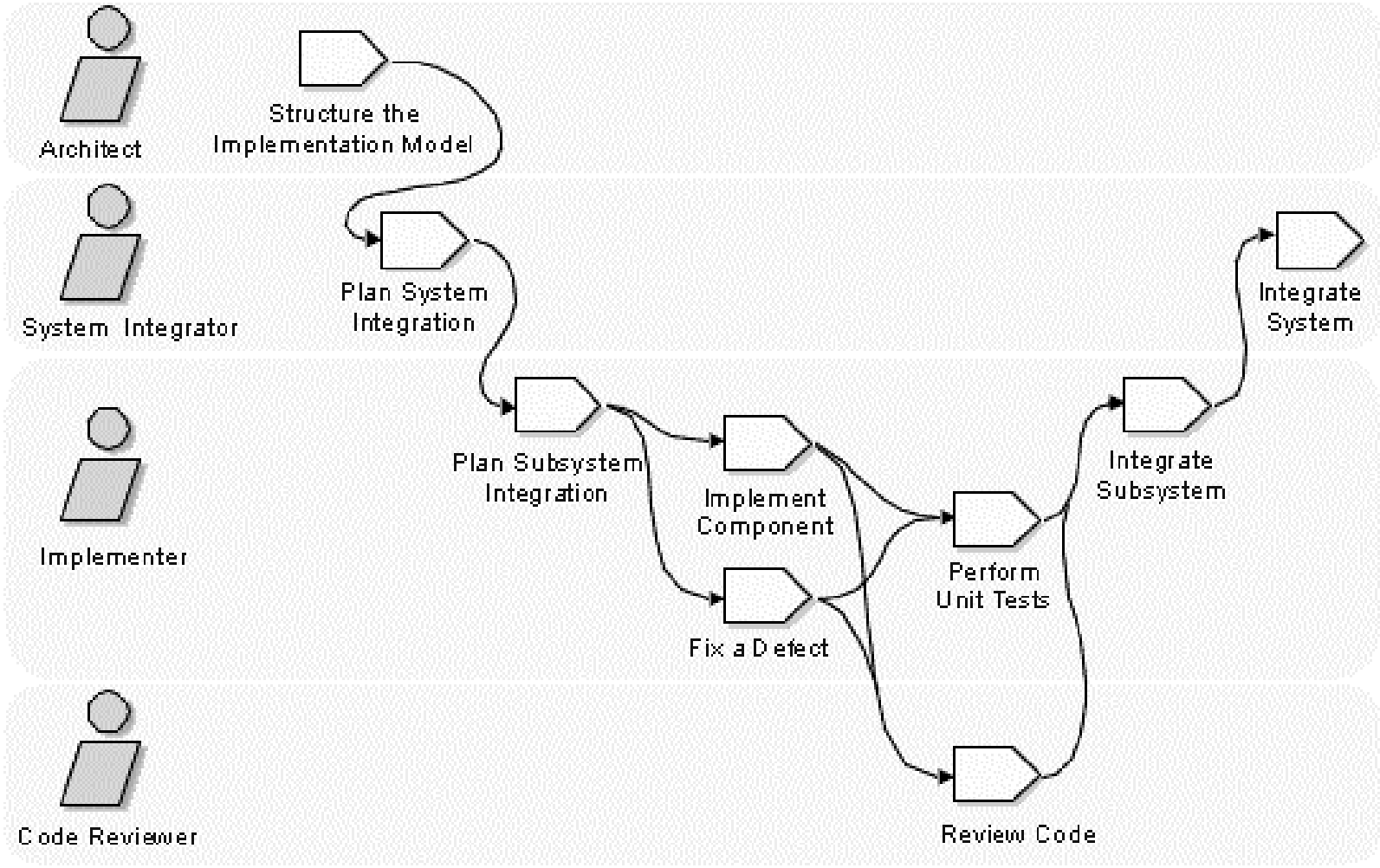
Implementation

Mar 19, 2000 | v1.0

Goals

- To define the organization of the code, in terms of implementation subsystems organized in layers
- To implement classes and objects in terms of components (source files, binaries, executables, and others)
- To test the developed components as units
- To integrate the results produced by individual implementers (or teams), into an executable system

Workflow



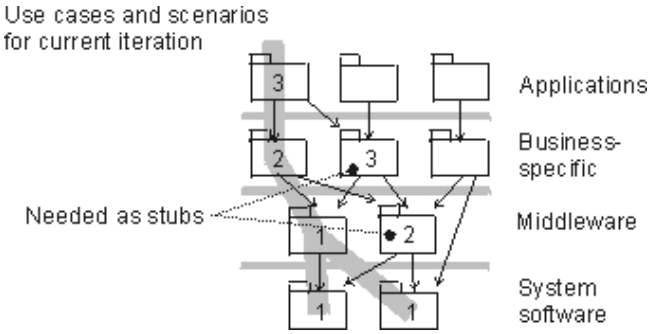
Structure the Implementation Model

- Create the Initial Implementation Model Structure
  - → to establish an initial structure for the Implementation Model - component diagrams
- Adjust Subsystems
  - → to adapt the structure of the model to reflect team organization or implementation language constraints
- Decide Placement of Executables
  - → to add main executables to the Implementation Model structure
- Define Imports for Each Subsystems
  - → to define dependencies between subsystems
- Decide Placement of Test Subsystems and Test Components
  - → to add test artifacts (automated test procedures) to the Implementation Model
- Update the Implementation View
  - → to update the Implementation View of the Software Architecture Document
- Evaluate the Implementation View

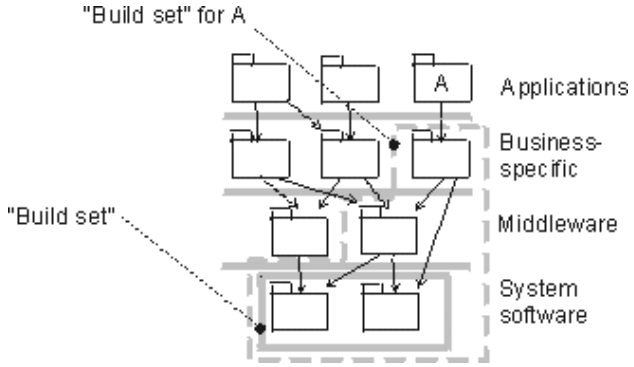
Plan System Integration

- Identify Subsystems [1]
  - → to identify which implementation subsystems participate in the use cases and scenarios for the current iteration
- Define "Builds Sets" [2]
  - → to define meaningful sets of subsystems (*build sets* or *towers*), that belong together from an integration point of view
- Define a Series of Builds [3]
  - → to define a series of builds (subsystems implemented/integrated at once) to incrementally integrate the system; bottom-up in the layered structure of subsystems
- Evaluate the Integration Build Plan

1, 3

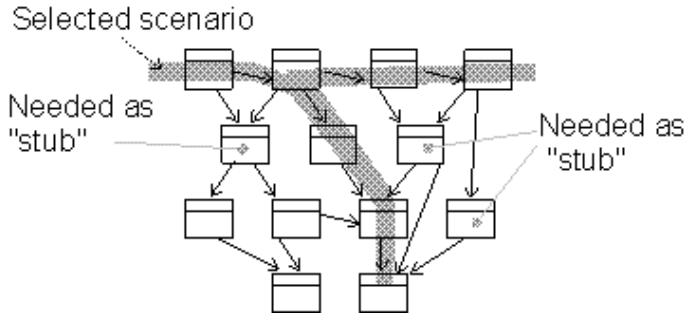


2



Plan Subsystem Integration

- Define the Builds
  - → to select one, or several scenarios, that will be the goal for each increment of the integration
- Identify the Classes*
  - → to identify the classes that participate in the selected scenarios
- Update the Subsystem's Imports
  - → to identify which other implementation subsystems (which versions) are needed for this build

* 

Implement Component

- Implement Operations
  - → to choose an algorithm and data structures, define new classes and operations as necessary and code the operation
- Implement States
  - → to implement an object's states, if needed
- Use Delegation to Reuse Implementation
  - → to use delegation for classes that can be implemented by reusing of existing classes
- Implement Associations
  - → to implement associations as pointers, collection of pointers (possibly ordered or sorted), attributes in both directions, special classes, etc.

Implement Component (cont.)

- Implement Attributes
  - → to implement attributes either as a variable of a built-in primitive, a reusable component class, or a new class
- Provide Feedback to Design
  - → to provide the rework feedback to the design if a design error is discovered in any of the steps
- Evaluate the Code
  - → to check the code before unit testing; setting the compiler's warning level to the most detailed level, mental checking the operations, using of tools (e.g. a static code rule checker)

Fix a Defect

- Stabilize the Defect
  - → to stabilize the defect, to make it occur reliably and identify which of the factors in the test case, make the defect occur
- Locate the Fault
  - → to execute the test cases that cause the defect to occur and try to identify where in the code the source of the fault is
- Fix the Fault
  - → to fix the fault and add a special test case that verifies this particular fault

Perform Unit Tests

- Identify the Required Types of Tests
  - → to identify the appropriate types of tests necessary to test the unit
- Identify and Describe the Test Cases
  - → to identify and describe the test conditions to be used for testing, to identify the specific data necessary for testing, and to identify the expected results of test
- Structure Test Procedures
  - → to identify and describe the setup, execution steps, and evaluation methods for implementing and executing unit test
- Review and Assess Test Coverage
  - → to identify and describe the measures of test that will be used to identify the completeness of testing

Perform Unit Tests (cont.)

- Implement Unit Tests
  - → to create appropriate test scripts (for automated testing) which implement (and execute) the test cases as desired
- Execute Unit Tests
  - → to execute the test procedures (or test scripts if testing is automated)
  - 1 set-up the test environment to ensure that all the needed components
  - 2 initialize the test environment to ensure all components are in the correct initial state for the start of testing
  - 3 execute the test procedures
- Evaluate Execution of Test
  - → to determine whether the tests completed successfully, as desired and to determine if corrective action is required

Review Code

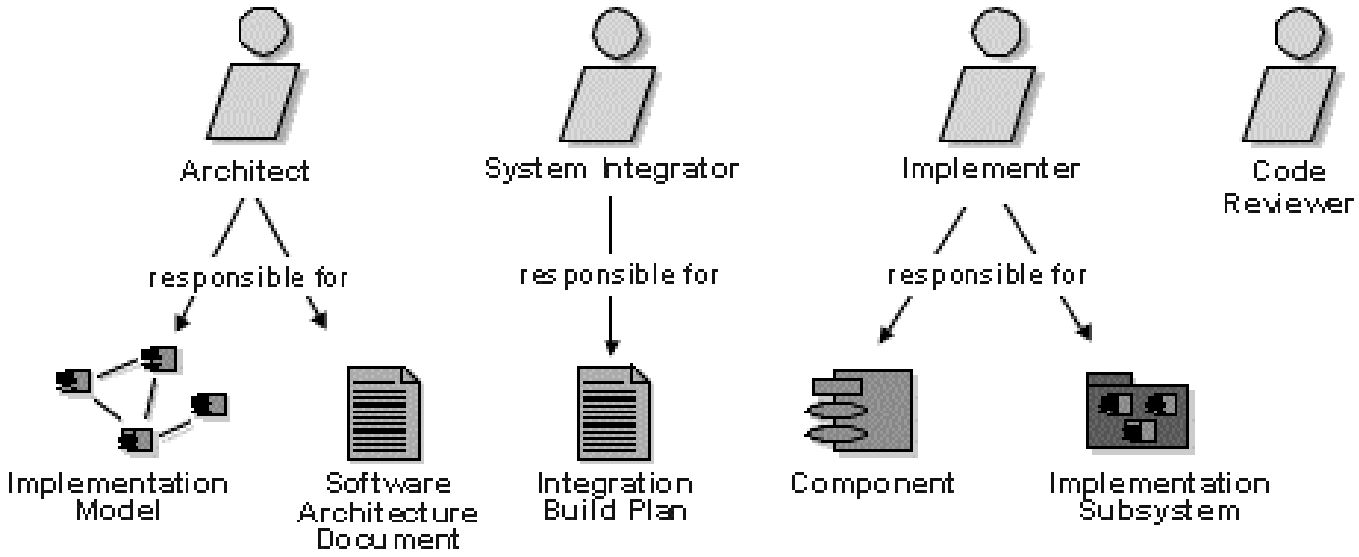
Techniques

- Inspection
  - → a formal evaluation technique in which the code is examined in detail by a person or group other than the author to detect errors, violations of development standards, and other problems.
- Walkthrough
  - → the author of the code, leads one or more reviewers through the code. The reviewers ask questions, and make comments regarding technique, style, possible error, violation of coding standards, and so on.
- Code reading
  - → one or two persons read the code. When the reviewers are ready, they can meet and present their comments and questions. The meeting can be omitted, however, and reviewers can give their comments and questions to the author in written form instead. Code reading is recommended to verify small modifications, and as a "sanity check”.

Integrate Subsystem and Integrate System

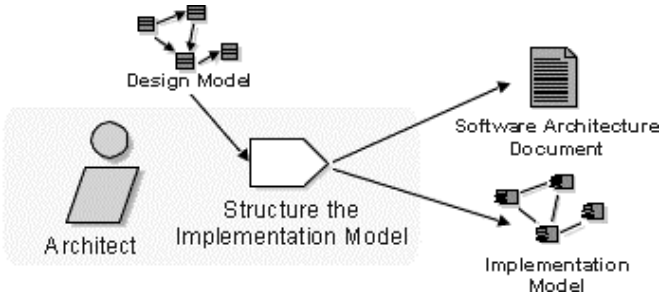
- Integrate Components
  - → to integrate the implemented classes (components); incrementally bottom-up in the compilation-dependency hierarchy; care about more implementers working in parallel
- Release the Subsystem
  - → to notify the system integrators that a new version of the subsystem is ready for integration after the final component is integrated
- Incrementally Add Subsystems
  - → to integrate the subsystems one-by-one, into an integration area; bottom-up approach is recommended in the layered structure
- Release a Build Internally
  - → to release the system internally after it has passed the System Test, that make the build available to the implementers

Artifacts

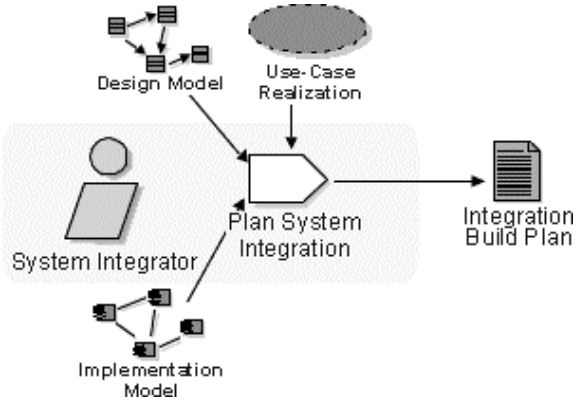

Architect
System Integrator
Implementer
Code Reviewer
responsible for
responsible for
responsible for
Implementation Model
Software Architecture Document
Integration Build Plan
Component
Implementation Subsystem

Workflow Details

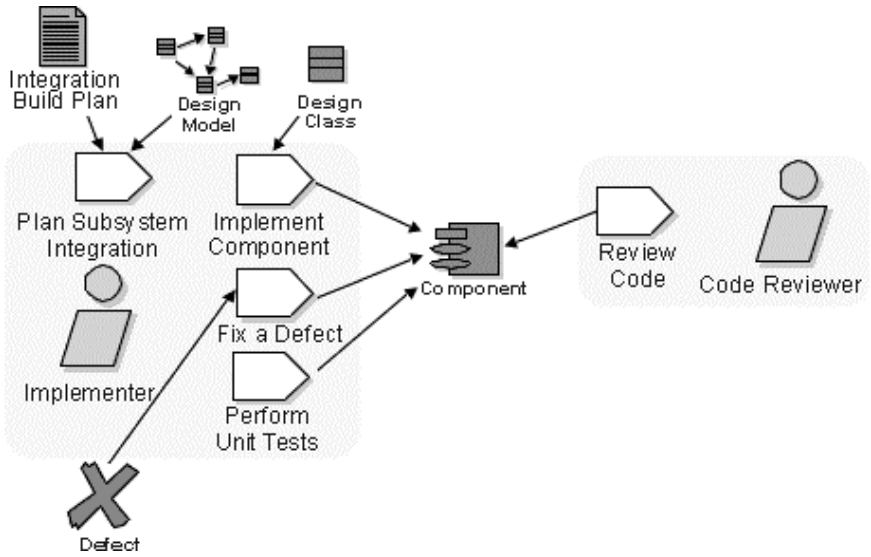
Structure the Implementation Model



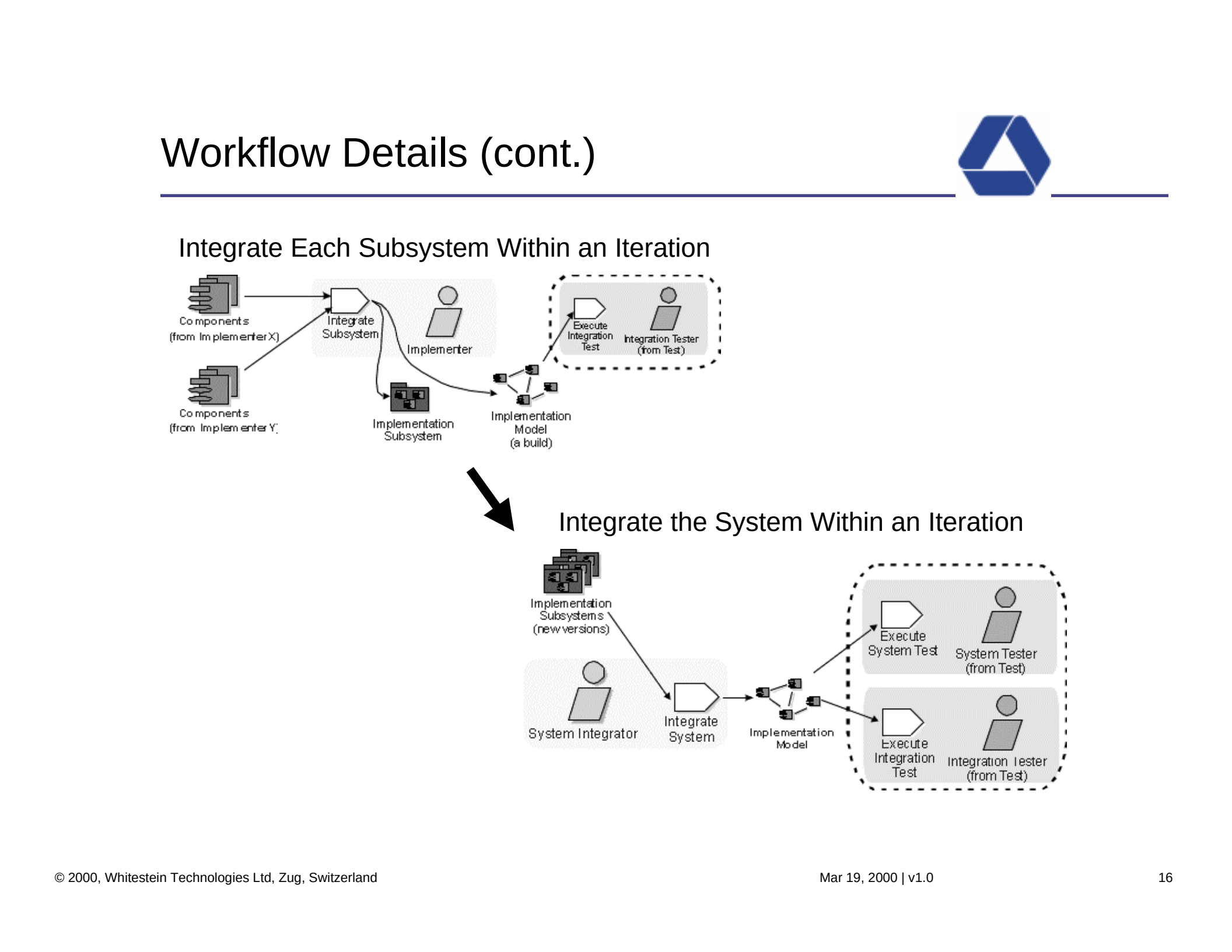
Plan the Integration within an Iteration



Implement Classes Within an Iteration



Workflow Details (cont.)



Integrate Each Subsystem Within an Iteration

Integrate the System Within an Iteration