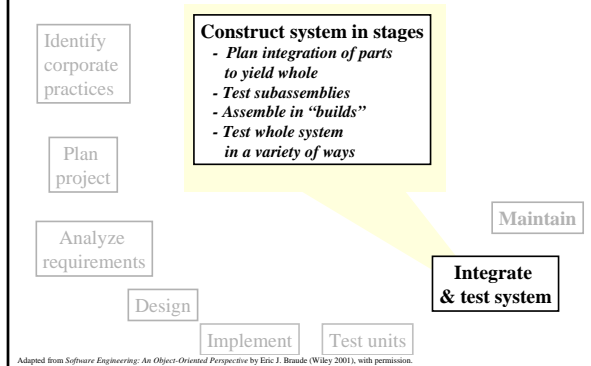


Lecture 10: System Integration

Software Engineering Roadmap: Chapter 9 Focus



Chapter Learning Goals

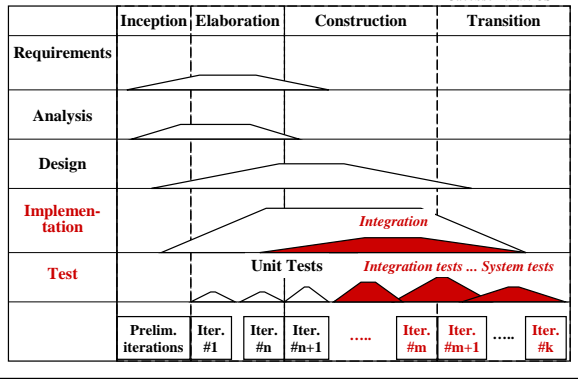
- Be able to plan the integration of modules
- Understand types of testing required
- Be able to plan and execute testing
 - beyond the unit level

Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

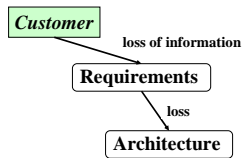
1. Introduction to system integration

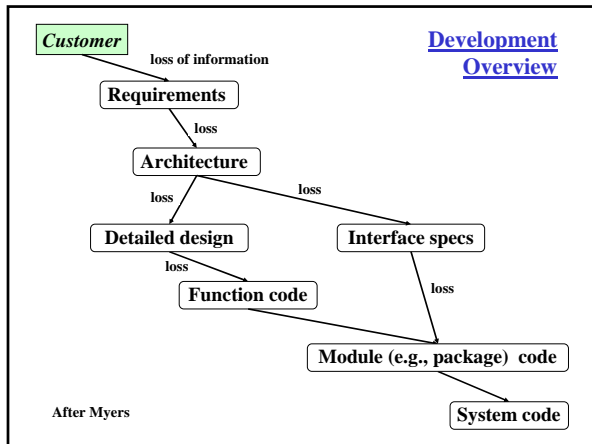
Unified Process for Integration & Test

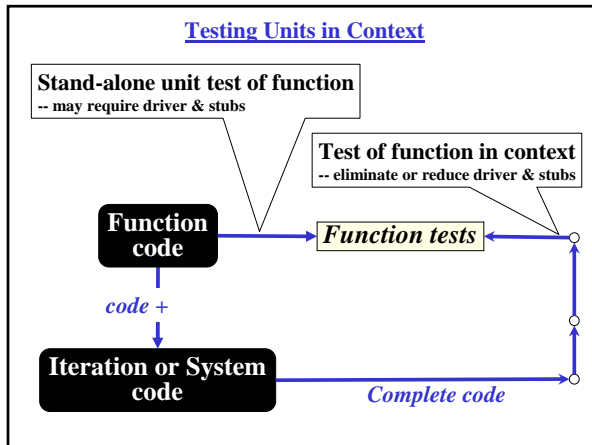
Jacobson et al: USDP

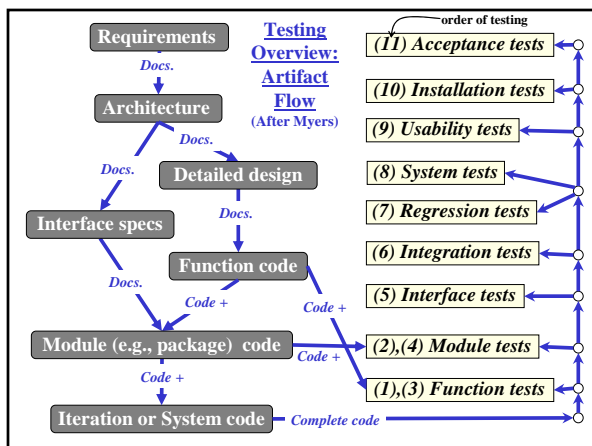


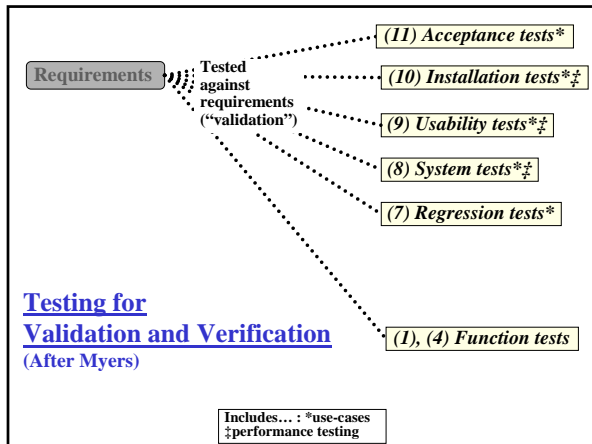
Development Overview

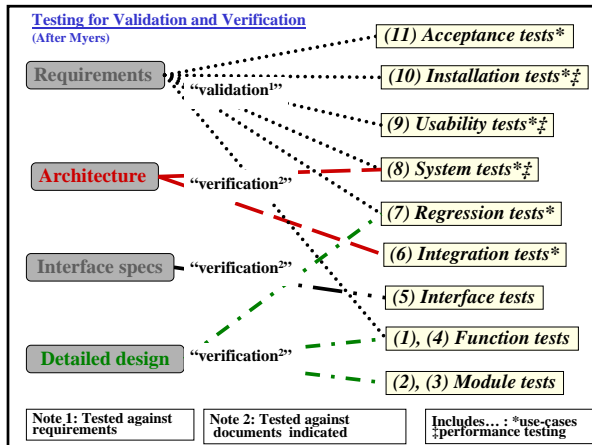




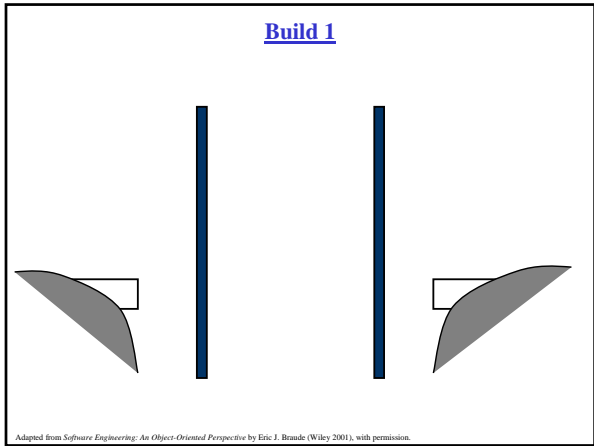


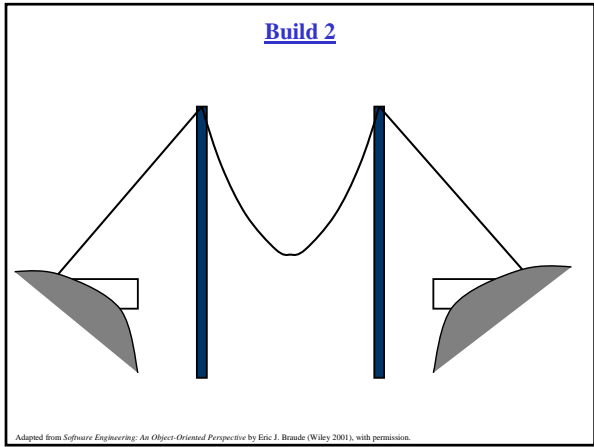


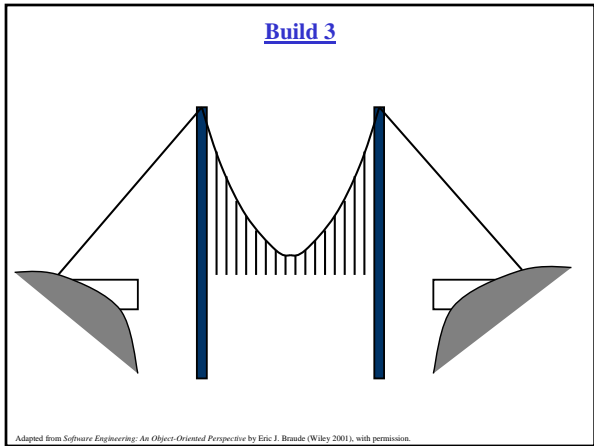


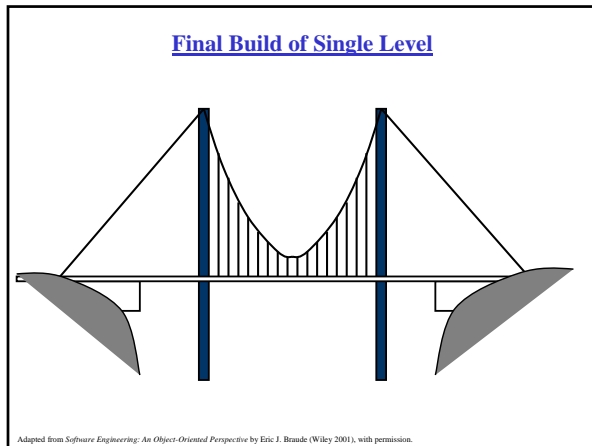


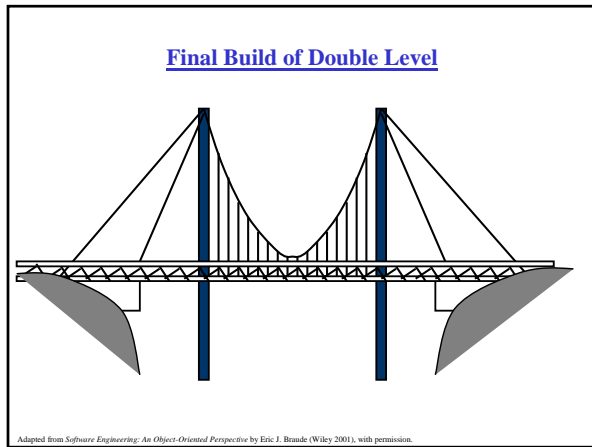
2. The integration process

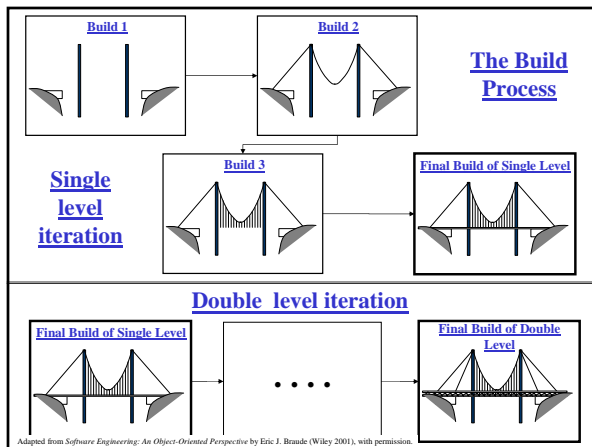


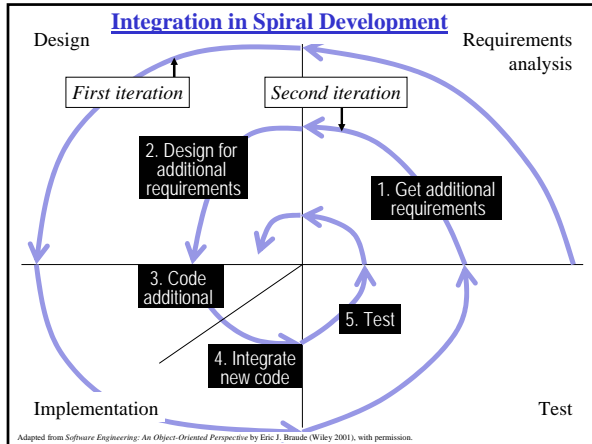










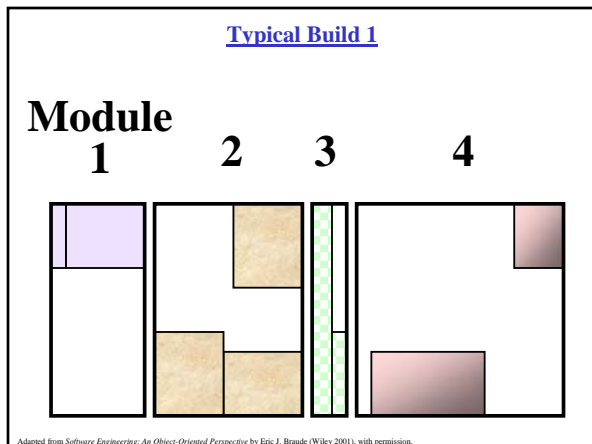


Relating Builds and Iterations in the Unified Process

Jacobson *et al.*: USDP

	Inception	Elaboration	Construction				Transition		
Requirements									
Analysis									
Design									
Implementation									
Test									
	Prelim. iterations	Iter. #1	Iter. #n	Iter. #n+1	Iter. #i	Iter. #m	Iter. #m+1	Iter. #k

First build for iteration i
Last build for iteration i



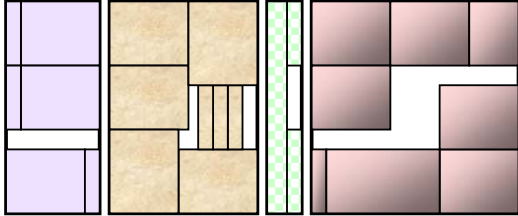
Typical Build 2

Module
1

2

3

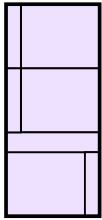
4



Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

Unit-Oriented Build 1

Module
1



Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

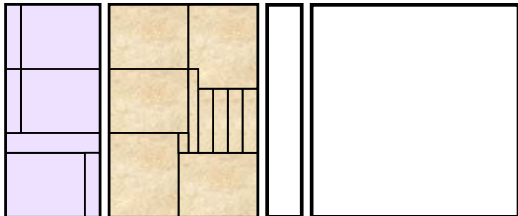
Unit-Oriented Build 2

Module
1

2

3

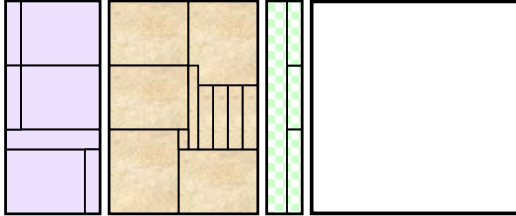
4



Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

Unit-Oriented Build 3

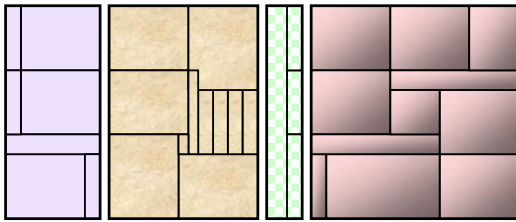
Module
1 2 3 4



Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

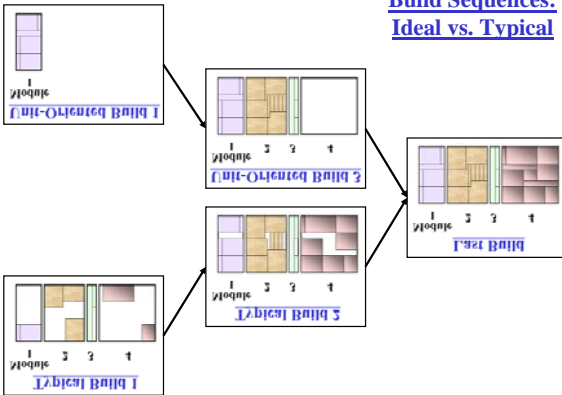
Last Build

Module
1 2 3 4



Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

Build Sequences:
Ideal vs. Typical



Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

1. Understand the architecture decomposition.
 - try to make architecture simple to integrate
2. Identify the parts of the architecture that each iteration will implement.
 - build framework classes first, or in parallel
 - if possible, integrate "continually"
 - build enough UI to anchor testing
 - document requirements for each iteration
 - try to build bottom-up
 - so the parts are available when required
 - try to plan iterations so as to retire risks
 - biggest risks first
 - specify iterations and builds so that each use case is handled completely by one
3. Decompose each iteration into builds if necessary.
4. Plan the testing, review and inspection process.
 - see section tbd.
5. Refine the schedule to reflect the results.

One way to
**Plan
 Integration
 & Builds**

Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

1. Decide extent of all tests.

2. For each iteration ... **Roadmap for Integration and System Test**

2.1 **For each build** ...

2.2 Perform iteration system and usability tests -- see section tbd

3. Perform installation tests -- see section tbd

4. Perform acceptance tests -- see section tbd

1. Decide extent of all tests.

2. For each iteration ...

2.1 For each build ...	2.1.1 Perform regression testing from prior build
	2.1.2 Retest functions if required
	2.1.3 Retest modules if required
	2.1.4 Test interfaces if required
	2.1.5 Perform build integration tests -- section 3.1

Development of iteration complete

2.2 Perform iteration system and usability tests -- sections 3.4, 3.5

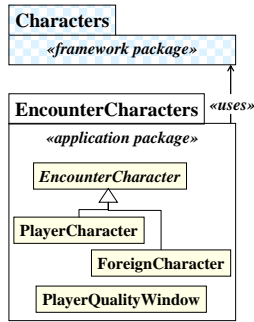
System implemented

3. Perform installation tests -- section 3.8
System installed

4. Perform acceptance tests -- section 3.7
Job complete

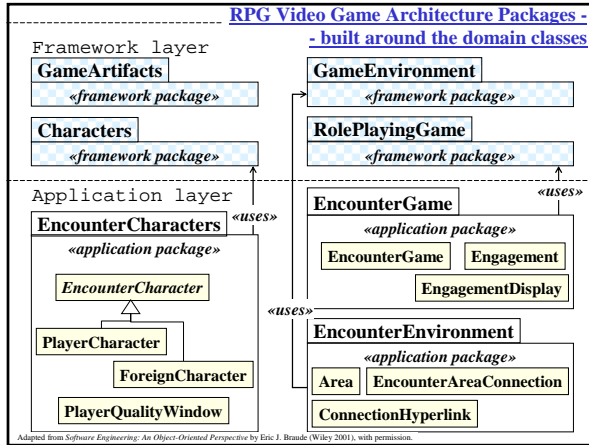
Roadmap for Integration and System Test

RPG Video Game Architecture Packages -- built around the domain classes



Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

RPG Video Game Architecture Packages - built around the domain classes



Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

Factors Determining the Sequence of Integration

- | | |
|------------------------|--|
| technical
(factors) | <ul style="list-style-type: none"> • Usage of modules by other modules <ul style="list-style-type: none"> – build and integrate modules <i>used</i> before modules that <i>use</i> them • Defining and using framework classes |
| risk
reduction | <ul style="list-style-type: none"> • Exercising integration early • Exercising key risky parts of the application as early as possible |
| requirements | <ul style="list-style-type: none"> • Showing parts or prototypes to customers |

Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.

Integration Schedule

Milestones	
Iterations	Inception iteration Elaboration iterations →
	<i>Iteration 1</i> "view characters in areas" <i>Iteration 2</i> "elementary interaction"
Builds	
Modules	

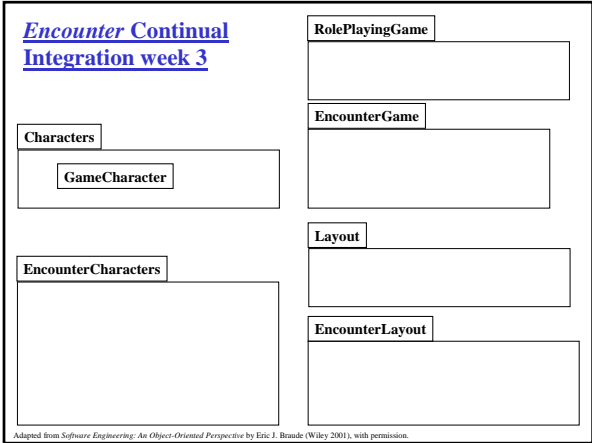
Integration Schedule	Month 1	Month 2	Month 3	Month 4	Month 5
	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4 1
Milestones	Proto. requirements		▲ Complete prototype ▲		
Iterations	Inception iteration			Elaboration iterations →	
	<i>Iteration 1</i> "view characters in areas"			<i>Iteration 2</i> "elementary interaction"	
Builds	build 1	build 2	build 3		
Modules				

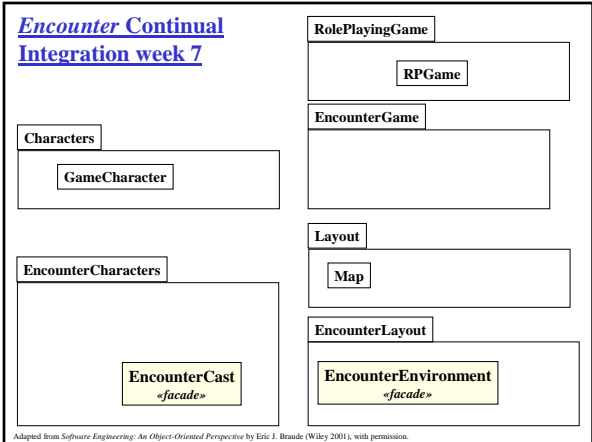
Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

Integration Schedule	Month 1	Month 2	Month 3	Month 4	Month 5
	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4 1
Milestones	Proto. requirements		▲ Complete prototype ▲		
Iterations	Inception iteration			Elaboration iterations →	
	<i>Iteration 1</i> "view characters in areas"			<i>Iteration 2</i> "elementary interaction"	
Builds	build 1	build 2	build 3		
Modules	GameEnvironment package EncounterEnvironment package Integrate & test	Characters package Encounter Characters package Integrate & test	RolePlaying Game package EncounterGame package Integrate & test		

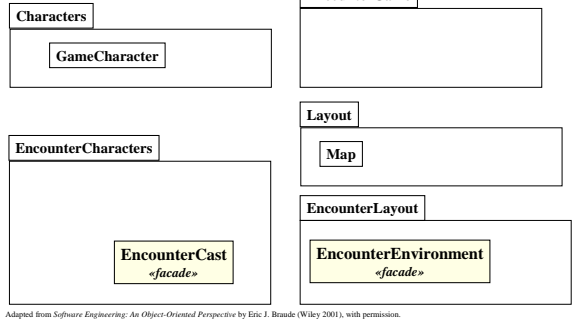
Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

3. The testing process



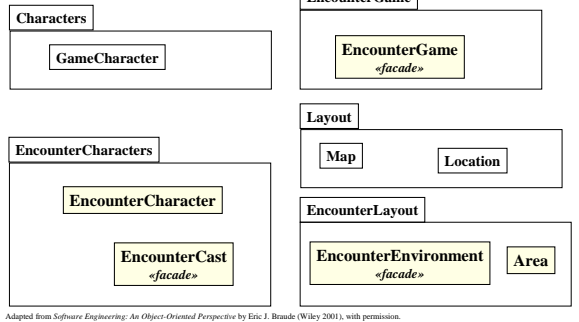


Encounter Continual
Integration week 11



Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

Encounter Continual
Integration week 15



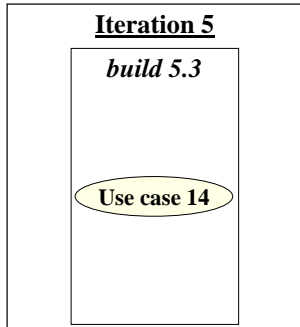
Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

***One way to* Plan and Execute Integration Tests**

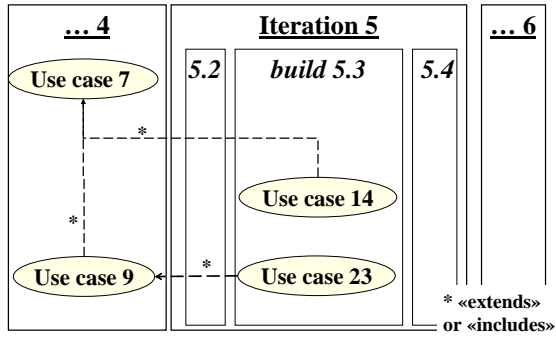
- 1. Decide how and where to store, reuse and code the integration tests.**
 - show this in the project schedule
- 2. Execute as many unit tests (again) as time allows**
 - this time in the context of the build
 - no drivers or stubs required this time
 - prioritize by those most likely to uncover defects
- 3. Exercise regression tests**
 - to ensure existing capability has not been compromised
- 4. Ensure build requirement are properly specified**
- 5. Exercise use cases that the build should implement**
 - test against the SRS
- 6. Execute the system tests supported by this build**

Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

Relationship Between Use Cases, Iterations and Builds

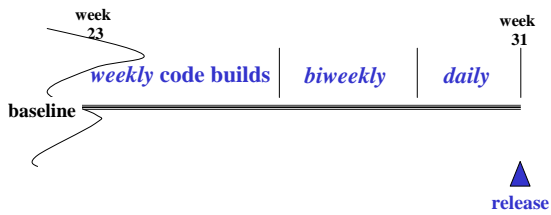


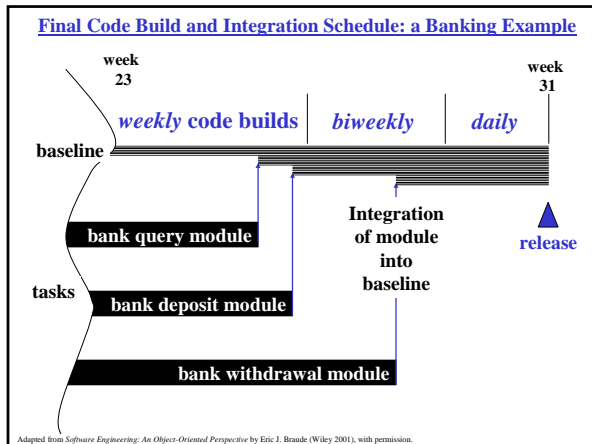
Relationship between Use Cases, Iterations and Builds

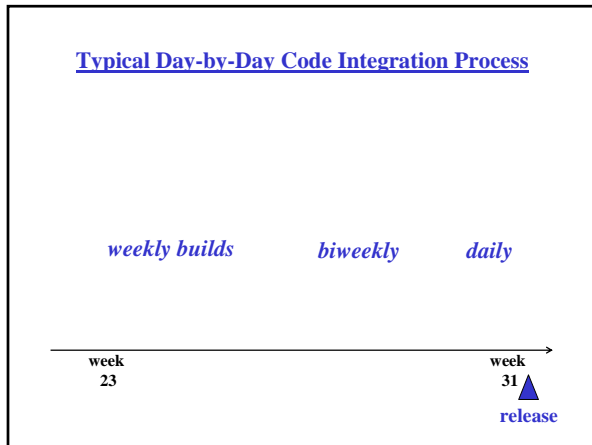


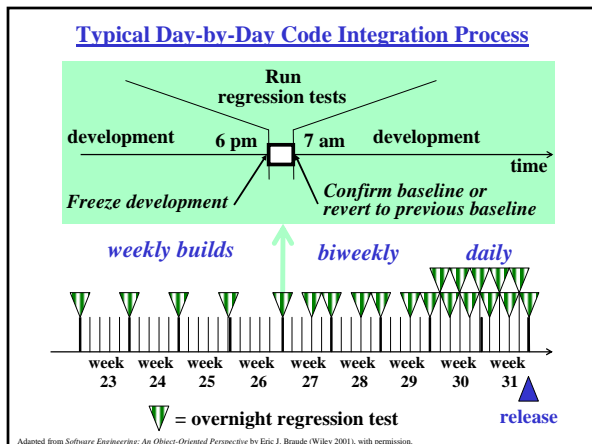
Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

Final Code Build and Integration Schedule: a Banking Example

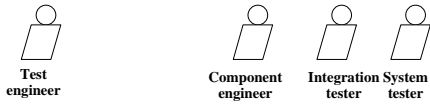






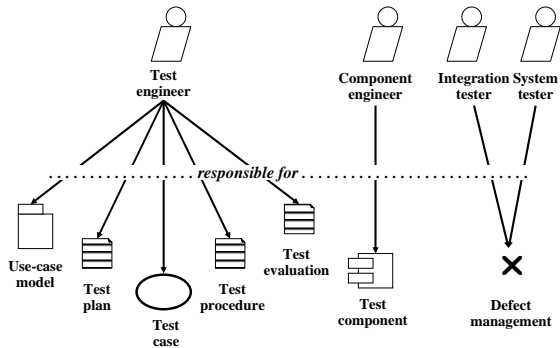


Artifacts and Roles for Integration Testing*



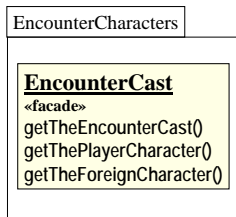
..... responsible for

Artifacts and Roles for Integration Testing*

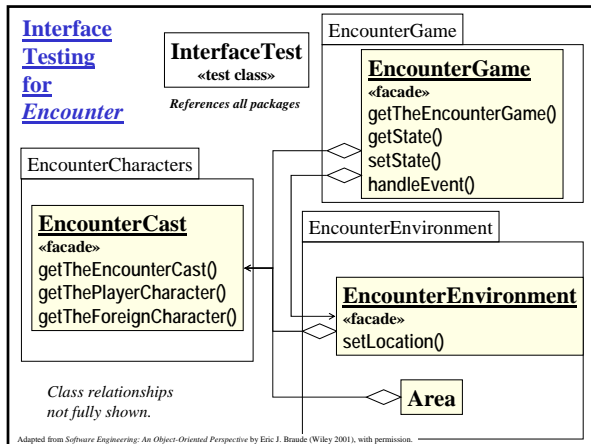


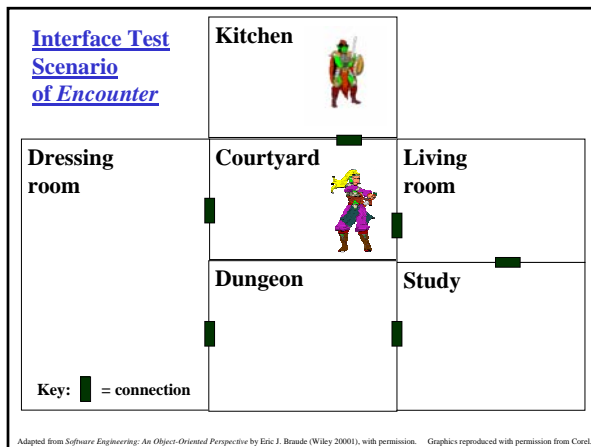
*Jacobson et al: USDP

Interface Testing for Encounter



Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.





- Types of System Tests* 1**
- **Volume**
Subject product to large amounts of input.
 - **Usability**
Measure user reaction (e.g., score 1-10).
 - **Performance**
Measure speed under various circumstances.
 - **Configuration**
Configure to various hardware / software
e.g., measure set-up time.
 - **Compatibility**
-- with other designated applications
e.g., measure adaptation time.
 - **Reliability / Availability**
Measure up-time over extended period.
- *see Kit [Ki]

- **Security**
Subject to compromise attempts.
 - e.g., measure average time to break in.
- **Resource usage**
Measure usage of RAM and disk space etc.
- **Install-ability**
Install under various circumstances.
 - measure time to install.
- **Recoverability**
Force activities that take the application down.
 - measure time to recover
- **Serviceability**
Service application under various situations.
 - measure time to service
- **Load / Stress**
Subject to extreme data & event traffic

* see Kit [Ki]

Types of
System
Tests* 2

- **Accessibility**
How easily can users enter, navigate & exit?
 - e.g., measure by average time taken to . . .
- **Responsiveness**
How quickly does the application allow the user to accomplish specified goals?
 - e.g., measure by average time taken
- **Efficiency**
Degree to which the number of required steps for selected functionality is minimal
 - “minimal” deduced in theory
 - e.g., measure by minimal time / average time
- **Comprehensibility**
How easy is the product to understand and use with documentation and *help*?
 - e.g., measure time taken for standard queries

* Adapted from Kit [Ki].

Key
Attributes
for
Usability
Testing*

4. Documenting integration and tests

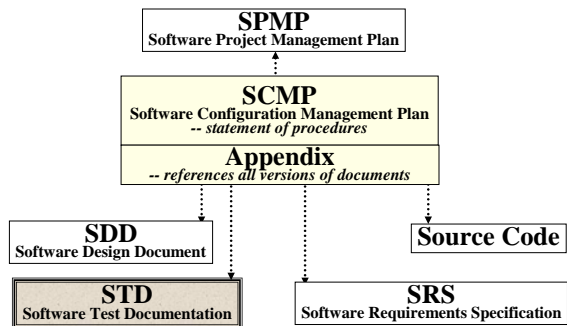
ANSI/IEEE 829-1983 Software Test Documentation
(reaff. 1991)

- 1. Introduction**
- 2. Test plan**
items under test, scope, approach, resources, schedule, personnel
- 3. Test design**
items to be tested, the approach, the plan in detail
- 4. Test cases**
sets of inputs and events
- 5. Test procedures**
steps for setting up and executing the test cases

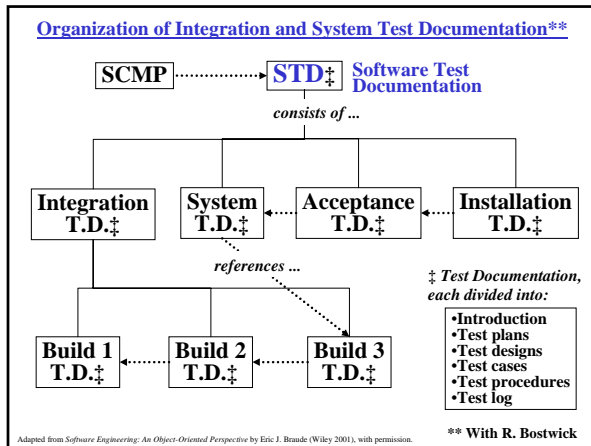
ANSI/IEEE 829-1983 Software Test Documentation
(reaff. 1991)

- | | |
|--|--|
| <ul style="list-style-type: none"> 1. Introduction 2. Test plan
<i>items under test, scope, approach, resources, schedule, personnel</i> 3. Test design
<i>items to be tested, the approach, the plan in detail</i> 4. Test cases
<i>sets of inputs and events</i> 5. Test procedures
<i>steps for setting up and executing the test cases</i> | <ul style="list-style-type: none"> 6. Test item transmittal report
<i>items under test, physical location of results, person responsible for transmitting</i> 7. Test log
<i>chronological record, physical location of test, tester name</i> 8. Test incident report
<i>documentation of any event occurring during testing which requires further investigation</i> 9. Test summary report
<i>summarizes the above</i> |
|--|--|

System Test Documentation in Context



Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.
Key:> "Refers to"








5. The Transition iterations

Goals of the Transition Iterations

	Transition			
Requirements	██████████			
Analysis	██████████			
Design	██████████			
Implementation	██████████			
Test	██████████			
	<table border="1"> <tr> <td style="text-align: center;">Iter. #m+1</td> <td style="text-align: center;">...</td> <td style="text-align: center;">Iter. #k</td> </tr> </table>	Iter. #m+1	...	Iter. #k
Iter. #m+1	...	Iter. #k		

- Find defects through customer use
- Test user documentation and help

Goals of the *Transition Iterations*

	Transition			
Requirements				
Analysis				
Design				
Implementation				
Test				
	<table border="1"> <tr> <td>Iter. #m+1</td> <td>...</td> <td>Iter. #k</td> </tr> </table>	Iter. #m+1	...	Iter. #k
Iter. #m+1	...	Iter. #k		

- Find **defects** through customer use
- Test **user documentation** and help
- Determine realistically whether application **meets customer requirements**
- Retire **deployment risks**
- Satisfy miscellaneous **marketing goals**

Alpha- and Beta- Releases

In-house and highly trusted users

- Multiplies testing
- Previews customer reaction
- Benefits third-party developers
- Forestalls competition

Alpha

Alpha- and Beta- Releases

In-house and highly trusted users

- Multiplies testing
- Previews customer reaction
- Benefits third-party developers
- Forestalls competition

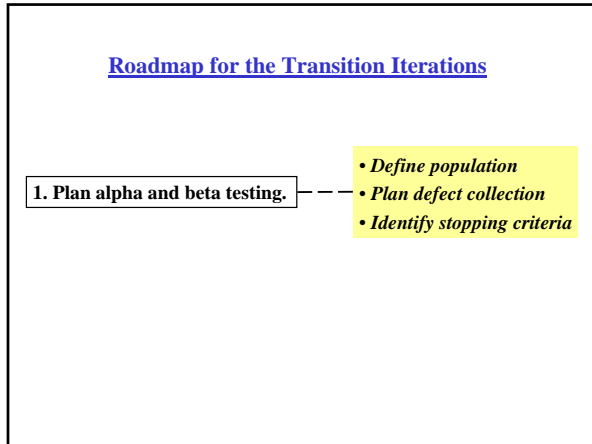
Alpha

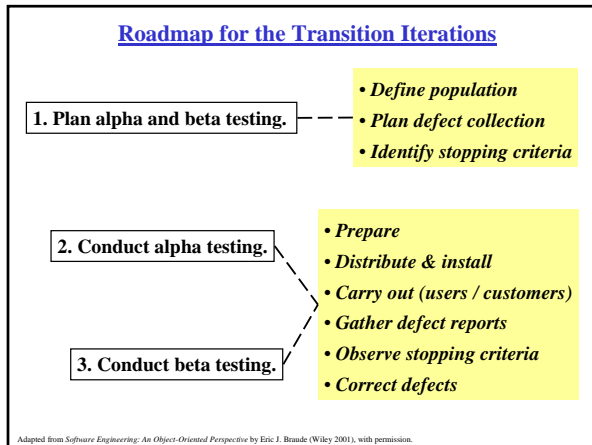
Selected customers

- Multiplies testing activity
- Gets customer reaction

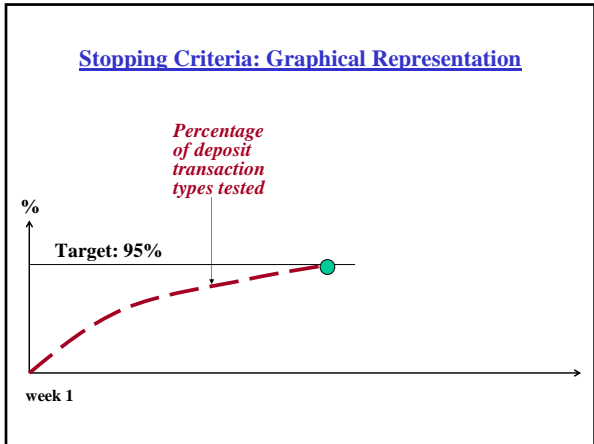
Beta

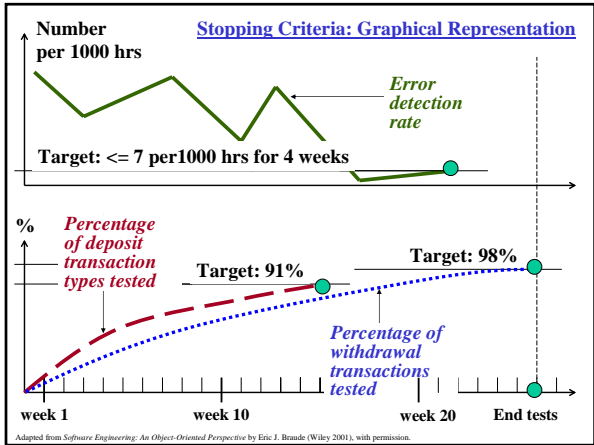
Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Brande (Wiley 2001), with permission.





- **Completing a particular test methodology**
 - Complete the procedures of a method or tool.
 - **Estimated percent coverage for each category**
 - predetermine percent of each & how to calculate
 - e.g., “95% statement coverage”
 - **Error detection rate**
 - predetermine rate with given severity level
 - e.g., “2 medium severity defects or less per 100 hours of operation”
 - **Total number of errors found**
 - (if possible) computed from a percentage of remaining defects
 - predetermine percent
 - e.g., “95% of estimated existing defects found ”
- * Kit [Ki]





7. Tools for integration and system testing

Capabilities of Automated System Test Tools

1. Record mouse and keyboard actions to enable repeated playback
2. Run test scripts repeatedly
3. Enable recording of test results
4. Record execution timing
5. Record runtime errors

Capabilities of Automated System Test Tools

1. Record mouse and keyboard actions to enable repeated playback
2. Run test scripts repeatedly
3. Enable recording of test results
4. Record execution timing
5. Record runtime errors
6. Create and manages regression tests
7. Generate test reports
8. Generate test data
9. Record memory usage
10. Manage test cases
11. Analyze coverage

Types of Capture/Playback Tests*

- *Native / software intrusive*
test software intermingled with software under test
 - could compromise software under test
 - least expensive
- *Native / hardware intrusive*
test hardware intermingled with software under test
 - could compromise software under test
- *Non-intrusive*
uses separate test hardware
 - does not compromise software under test
 - most expensive

* adapted from Kit [Ki]

Memory Usage Test Tools

- *Memory leaks*
 - detect growing amounts of unusable memory
 - inadvertently caused by the implementation
- *Memory usage behavior*
 - confirm expectations
 - identify bottlenecks
- *Data bounds behavior*
 - e.g., confirm integrity of arrays
 - e.g., detect attainment of limiting values
- *Variable initialization*
 - indicate un-initialized variables
- *Overwriting of active memory*

Test Case Management*

- *Provide user interface*
 - to manage tests
- *Organize tests*
 - for ease of use
 - for maintenance
- *Manage test execution sessions*
 - to run tests selected by user
- *Integrate with other test tools*
 - to capture and play back
 - to analyse coverage
- *Provide reporting*
- *Provide documentation*

* adapted from Kit [Ki]

8. Summary

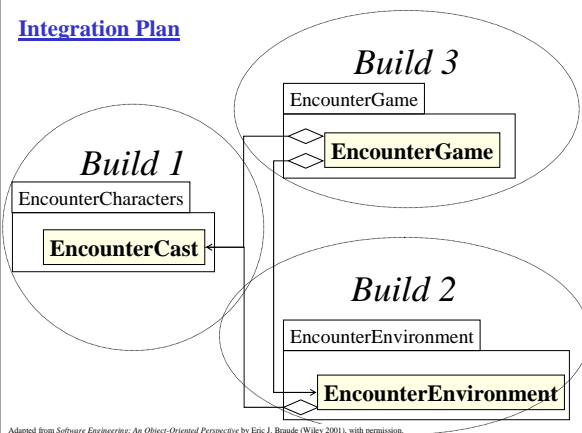
Summary of
System Integration and Verification:

- *Integration process* executed in carefully planned builds
- *Integration testing*: each build
- *System testing*: whole application
- *Regression testing*: verify that changes do not compromise pre-existing capabilities

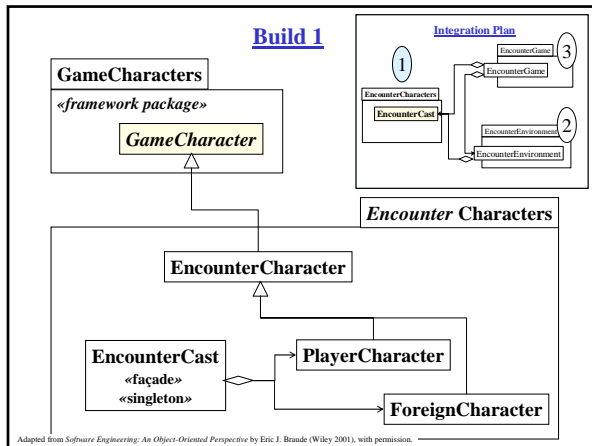
Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

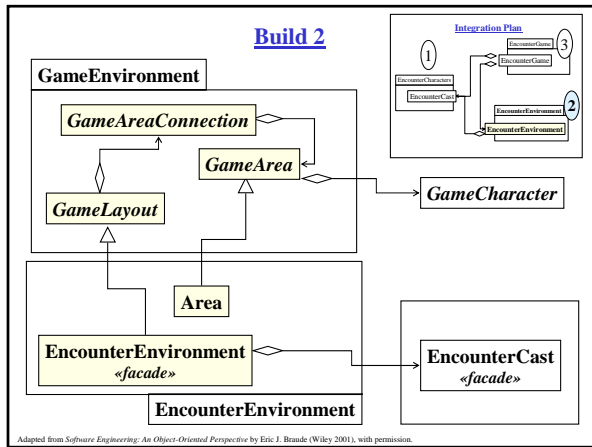
Case study -
SCMP: Appendix A
Plan for Integration Baselines

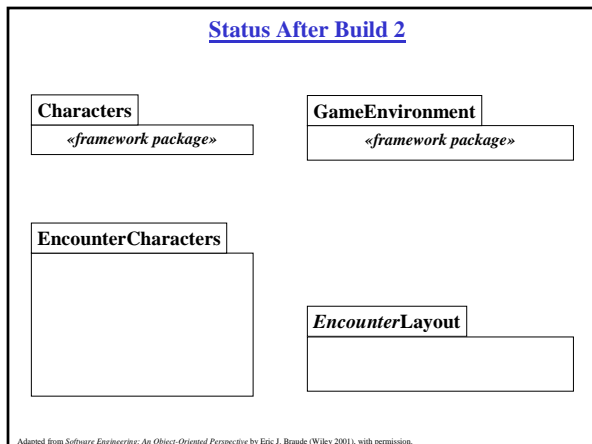
Integration Plan

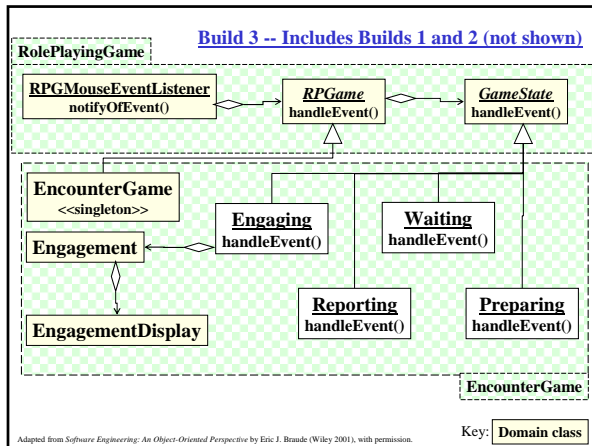


Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.









Software Test Documentation for
Encounter

Test Type	Approach	Corresponding document sections
Unit	White- and black box; method and class tests; test against D-requirements and design.	SRS section(s): 3.2 <i>Classes/Objects</i> SDD section(s): 6. <i>Detailed design</i>
Integration	Gray box; mostly package-level; oriented to builds (1, 2, and 3); test against architecture and C-requirements.	SRS section(s): 2. <i>Overall description</i> , 3.1 <i>External interfaces</i> , validate representative requirements in 3.2 <i>Classes/Objects</i> SDD section(s): 3. <i>Decomposition description</i> , 4. <i>Dependency description</i> , 5. <i>Interface description</i> .
<u>Approaches and Documentation for Test Types</u>		
System	Black box; all packages; whole system (Build 3); test against non-functional requirements, architecture and C-requirements.	SRS section(s): 2. <i>Overall description</i> , 3.1 <i>External interfaces</i> , validate representative requirements in 3.2 <i>Classes/Objects</i> , 3.3 <i>Performance requirements</i> , 3.4 <i>Design constraints</i> , 3.5 <i>Software system attributes</i> , 3.6 <i>Other requirements</i> SDD section(s): 3. <i>Decomposition description</i> , 4. <i>Dependency description</i> , 5. <i>Interface description</i> ; validate representative requirements in 6. <i>Detailed design</i>

Approaches and Documentation for Test Types

Acceptance	Black box; all packages; whole system (Build 3); test against C-requirements and D-requirements.	SRS section(s): 2. Overall description, 3.2 Classes/Objects
Installation	Black box; all packages; whole system (Builds for customer specific configurations); test against C-requirements and D-requirements.	SRS section(s): 2. Overall description, 3.2 Classes/Objects

Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

Document	Section	Requirement Title
SRS	2.1.2.2	User interface for setting quality values
	3.1.1.2	User interface for setting quality values
	3.2.EC	Encounter characters
	3.2.FC	Foreign characters
	3.2.P	Player characters
	3.2.PQ	The player quality window
SDD for RPG framework	3.1.2	Characters package
	5.0	Interface description
SDD for Encounter	3.1.2	EncounterCharacters package
	4.2	Inter-process dependencies
	5.1.2	Interface to the EncounterCharacters package

Features to be Tested in Build 1

Test Document	Document Identifier
Build 1 Test Plan	Build1_TP
Build 1 Test Design Specification	Build1_TD
Build 1 Test Case Specifications	Build1_TC1 to Build1_TC...
Build 1 Test Procedure Specifications	Build1_TP1 to Build1_TP...
Build 1 Test Logs	Build1_LOG1 to Build1_LOG...
Build 1 Test Incident Report	Build1_InRep1 to Build1_InRep...
Build 1 Test Summary Report	Build1_SumRep1 to Build1SumRep...

Adapted from Software Engineering: An Object-Oriented Perspective by Eric J. Braude (Wiley 2001), with permission.

Test Document Identifiers

