

---

# Software Quality Assurance: SOFT3302

## Tutorial – Week 10

---

### Objectives

This tutorial deals with the development of a risk registers. By the end of this tutorial you should have produced a risk register using quality attributes.

### Prework

Read Assignment 3. Form groups. Register your group with the course coordinator according to the instructions in the assignment.

Familiarise yourself with the SUT for this tutorial. Download and run it. View the sources.

### Labwork

Work in groups of 2 or 3. The SUT is *JBooks-0.1.1* (<http://sourceforge.net/projects/jbooks>) a Java-based personal finance application.

**We're well into the realm of 'no single right answer but plenty of wrong ones' now and it will be necessary to be comfortable with testing concepts as a whole to be able to answer questions and give sensible answers.**

**This tutorial will focus on discussions of the functionality of the SUT and the identification of various risks.**

1. With reference to the quality attributes table supplied, extend the risk register below. Perform the risk evaluation by selecting quality attributes one at a time. For each attribute discuss how that attribute should be interpreted for the SUT. For example, 'Correctness' should be interpreted with respect to values supplied as numerical values in transaction; 'Authorisation' would only relate to user permissions supported by the operating system. Once the interpretation has been agreed, list all relevant risks. (Some examples have been included in the proto-register to help you get started.)

*JBooks may have been used previously<sup>1</sup> for some of the tutorial exercises. If you have not done this week's pre-work then you will find that you will still need to spend an amount of time familiarising yourself with the SUT.*

The SDLC-phase you can focus on here is the 'operational' aspects of the SUT, *i.e.* the actual behaviour of the installed programme as captured in the 'Programme' phase, although you might like to give some thought to which phases the defects you are trying to guard against might creep in and what mitigating activities might exist in which previous and subsequent phases. (The items listed in the second factor-table under 'Maintain' are for preservation of those factors during a maintenance activity as compared to satisfaction of those factors during operation.)

2. For each risk evaluate the severity and likelihood. Use a four-value scale consisting of Very High, High, Low and Very Low.

---

<sup>1</sup> In previous years.

- Once the risk register has been completed, check the register for completeness by reviewing it with regard to the quality categories listed below.

## Quality Attributes/Test Factors

From An Introduction to Software Testing, *Ostfold University College (HiØ)*, 2003.

Test factors are the objective of testing, when designing a test strategy. Even if test factors are only the attributes of software, we can refer to them as risks if they are not present, but wanted in the software system. An example is if the access control is not working, and the data of a software system are being misused. These test factors should be reduced to a level specified in the specification, acceptable to an organisation. The table below shows the test factors, with some general examples.

The examples are general examples of the sort of things one might consider for each factor. They are not *JBooks*-specific.

Test factor	Description	Examples
<u>Correctness</u>	Assurance that the data entered, processed, and outputted by the application system is accurate and complete. Accuracy and completeness is achieved through controls over transactions and data elements, which should commence when transactions is originated and conclude when the transaction data has been used for its intended purpose.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Products are priced correctly on invoices</li> <li>- Gross pay is properly calculated</li> <li>- Inventory-on-hand balances are correctly accumulated</li> </ul>
<u>Authorisation</u>	Assurance that the data is processed in accordance with the intents of management. In an application system, there is both general and specific authorisation for the processing of transactions. General authorisation governs the authority to conduct different types of business, while specific authorisation provides authority to perform a specific act.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Price overrides are authorised by management</li> <li>- Credits for product returns have been approved by management</li> <li>- Employee overtime pay is authorised by the employee's supervisor</li> </ul>
<u>File/data integrity</u>	Assurance that the data entered into the application system will be returned unaltered.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- The amounts in the detail records of a file support the control totals</li> <li>- Customer addresses are correct</li> <li>- Employee pay rates are correct</li> </ul>
<u>Audit trail</u>	The capability to substantiate the processing that has occurred. The processing of data can be supported through the retention of sufficient evidential matter to substantiate the accuracy, completeness, timeliness, and authorisation of data. The process of saving the supporting evidential matter is frequently called an audit trail.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Employee gross can be substantiated by supporting documentation</li> <li>- Sales tax paid to a specific state can be substantiated by the supporting invoices</li> <li>- Payments made to vendors can be substantiated should the vendor disavow receiving the payment</li> </ul>

Test factor	Description	Examples
<u>Continuity of processing</u>	The ability to sustain processing in the event problems occur. Continuity of the processing assures that the necessary procedures and backup information are available to recover operations should integrity be lost due to problems. Continuity of processing includes the timeliness of recovery operations and the ability to maintain processing periods when the computer is inoperable.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Banking transactions can continue if computer becomes inoperational</li> <li>- Recovery of an online-system can occur within the predetermined tolerances</li> </ul>
<u>Service levels</u>	Assurance that the desired result will be available within a timeframe acceptable to the user. To achieve the desired service level, it is necessary to match user requirements with the available resources. Resources include input/output capabilities, communication facilities, processing, and system software capabilities.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Response time in a online-system is within the time span tolerance</li> <li>- Application workload can be completed in accordance with the application schedule</li> <li>- Changes to the system can be incorporated within the agreed upon schedule</li> </ul>
<u>Access control</u>	Assurance that the application system resources will be protected against accidental and intentional modification, destruction, misuse and disclosure. The security is the totality of the steps taken to ensure the integrity of application data and programs from unintentional and unauthorised acts.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Programmers will not be given access to data</li> <li>- Access will be restricted to predetermined system resources</li> <li>- Automated access mechanism will be current</li> </ul>
<u>Compliance</u>	Assurance that the system is designed in accordance with organisational strategy, policies, procedures, and standards. These requirements need to be identified, implemented, and maintained in conjunction with other application requirements.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Information standards are compiled with</li> <li>- System strategy is followed</li> <li>- System in developed in accordance with budget and schedules</li> </ul>
<u>Reliability</u>	Assurance that the application will perform its function with the required precision over an extended period of time. The correctness of processing deals with the ability of the system to process valid transactions correctly, while reliability relates to the system's being able to perform correctly, while reliability relates to the system's being able to perform correctly over an extended period of time when placed into production.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- User can enter correct information on a day-to-day-basis</li> <li>- Errors can be correctly reprocessed</li> <li>- Appropriate action will be taken on system reports</li> </ul>

Test factor	Description	Examples
<u>Ease of use</u>	The extent of effort required to learn, operate, prepare input for, and interpret output from the system. This test factor deals with the usability of the system to the people interfacing with the application system.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Input form minimise input errors</li> <li>- Flow of work will be optimised in order to process work quickly</li> <li>- Reporting procedures will be written in an ease-to-use terminology</li> </ul>
<u>Maintainability</u>	The effort required to locate and fix an error in an operational system. Error is used in the broad context to mean both a defect in the system and a misinterpretation of user requirements.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Programme documentation will be up-to-date</li> <li>- Programme segments will point to other segments that need to be changed concurrently with that segment</li> <li>- Segments of programs will be identified with appropriate identifiers</li> </ul>
<u>Portability</u>	The effort required to transfer a programme from one hardware configuration and/or software system environment to another. The effort includes data conversion, programme changes, operating system, and documentation changes.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Computer programme will only use common language features</li> <li>- System will be hardware independent</li> <li>- System will be independent of system software special features</li> </ul>
<u>Coupling</u>	The effort required to interconnect components within an application system and with all other application systems in their processing environment.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Segments in one application requiring concurrent changes in other applications will be properly identified</li> <li>- Common documentation will be up-to-date</li> <li>- Changes will be coordinated</li> </ul>
<u>Performance</u>	The amount of computing resources and code required by a system to perform its stated functions. Performance includes both manual and automated segments involved in fulfilling system functions	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- System is completed within time and budget constraints</li> <li>- System achieves performance acceptance criteria</li> </ul>
<u>Ease of operation</u>	The amount of effort required to integrate the system into the operating environment and then to operate the application system. The procedures can be both manual and automated.	<u>Assurance that:</u> <ul style="list-style-type: none"> <li>- Operation documentation is up-to-date</li> <li>- Operators are trained in any special application operating procedures</li> <li>- Correct version of programs run in production</li> </ul>

All in all, the fifteen test factors shown in the table above give a description of the broad objectives of system testing. Still, in every SDLC-phase there are certain *risk related perspectives* associated with each chosen test factor that needs to be focused on in that specific SDLC-phase. This is shown in the table below, where the test factors are listed again, but now in accordance with the phases in the SDLC.

**This table has been supplied for students to think about the application of various factors to the various SDLC phases. It may or may not be useful to students in this tutorial. They should be thinking about the ‘Programme’ phase.**

Test factor	SDLC-phase					
	Requirements	Design	Programme	Test	Installation	Maintain
Correctness	Functional specifications designed	Design conforms to requirements	Programme conform to design	Functional testing	Proper programs and data are placed into production	Update requirements
Authorisation	Authorisation rules defined	Authorisation rules designed	Authorisation rules implemented	Compliance testing	Prohibit data changes during installation	Preserve authorisation rules
File/data integrity	File integrity requirements defined	File integrity controls designed	File integrity controls implemented	Functional testing	Verify integrity of production files	Preserve file integrity
Audit trail	Reconstruction requirements defined	Audit trail designed	Implement audit trail	Functional testing	Record installation audit trail	Update audit trail
Continuity of processing	Impact on failure defined	Contingency plan designed	Write contingency plan and procedures	Recovery testing	Assure integrity of previous testing	Update contingency plan
Service level	Desired service level defined	Method to achieve service level designed	Design system to achieve service level	Stress testing	Implement fail-safe installation plan	Reserve service level
Access control	Access defined	Access procedure designed	Implement security procedures	Compliance testing	Control access during integration	Preserve security

Test factor	SDLC-phase					
	Requirements	Design	Programme	Test	Installation	Maintain
Reliability	Tolerance established	Data integrity controls designed	Data integrity controls implemented	Manual regression and functional testing	Accuracy and completeness of installation verified	Update accuracy requirements
Ease of use	Usability specifications determined	Design facilitates use	Programs conform to design	Manual support testing	Usability instructions disseminated	Preserve ease of use
Maintainable	Maintenance specifications determined	Design is maintainable	Programs are maintainable	Inspection	Documentation complete	Preserve maintainability
Portable	Portability needs determined	Design is portable	Programs conform to design	Disaster testing	Documentation complete	Preserve portability
Coupling	Systems interface defined	Interface design complete	Programs conform to design	Functional and regression testing	Interface coordinated	Assure proper interface
Performance	Performance criteria established	Design achieves criteria	Programs achieve criteria	Compliance test	Integration performance monitored	Preserve level of performance
Ease of operation	Operation needs defined	Communicate needs to operations	Develop operating procedures	Operations test	Implement operating procedures	Update operating procedures
Methodology	Requirements comply with methodology	Design complies with methodology	Programs comply with methodology	Testing complies with methodology	Integration complies with methodology	Maintenance complies with methodology

## Risk Register

Factor	Issue	Severity	Likelihood
Correctness	Transaction values not recorded correctly	VH	L
	Transaction amount incorrect	H	L
	...		
Authorisation	Database modified by another user	VH	VL
...			

## Quality Categories

Quality Category	Description
Complex	Anything disproportionately large, intricate or convoluted
New	Anything that has no history in the project
Changed	Anything that has been changed recently, tampered with or 'improved'
Upstream dependency	Anything whose failure will cause cascading failure in the rest of the system
Downstream dependency	Anything that is especially sensitive to failures in the rest of the system
Critical	Anything whose failure could cause substantial damage
Precise	Anything that must meet its requirements exactly
Popular	Anything that will be used a lot
Strategic	Anything that has special importance to the business-at-hand, such as feature that set the product apart from competitors
Third-party	Anything used in the product but developed outside the product
Distributed	Anything spread out in time or space yet whose elements must work together
Buggy	Anything known to have lots of problems
Recent failure	Anything with a recent history of failure