Software-Testing

Integration testing

Autor: Hans Schäfer

Integration testing is necessary to verify interfaces. These can be on any level: components, modules, subsystems and systems. Interfaces must be verified because they may be implemented wrong. Most problematic is interpersonal communication failure if different people implement the interfacing parts. But even the same person implementing two components at different times may introduce problems because of human failure. The risk for problems increases mostly with the organizatorial distance between people implementing the parts. Other factors are complexity, time pressure, bad processes, etc.
Integration testing is concerned about the correct working of the interface.
This means the two parts must understand the data flowing over the interface in the same, or at least a compatible, way.
Another concern is consistency between the parts, i.e. of both parts work in the same way, as seen from the outside. Any outside user of the aggregate should not need to know what is done where: The aggregate should look like a new unit.
Input is given from the outside of both components to be integrated. The output is also observed at the outside. In principle, the interface between the units is not observable nor can it be manipulated. However, test equipment may be used to insert chosen values and monitors may be used to observe what is actually happening at the interface. Such insertion and observations may partly be done (re)using module test environment. Otherwise special monitors, protocol analyzers and insertion software must be bought or designed. Testers should try to be aware of this early and require testability (access points) to be built in.
In principle, integration test should be concerned about interfaces only.
Thus we need a description of all internal interfaces of the system, as well as their details. This information should be contained in an overall design or system architecture.

However, it is often worse in practice, as this information is not always available in the form of a design or architecture.
The list below is thus meant as a checklist for where to look for in order to get interface information.

- Subsystem definitions
- Sequence- and collaboration charts
- APIs
- Communication protocols
- Files and their formats

• Call-pairs, argument lists
• Event lists
• Use and flow of global data (init, write, read, search, change, delete)
• Interaction online - batch - online
• Consistency between interfacing parts
• Synchronization of parallel processes and transactions, especially queue handling.
• Multitasking: capacity of shared ressources (place and speed)
• Handling of failure in other components and recovery
• Special cases at interfaces (not existing data, bad data, empty files, network down, etc.)
• Port assignments and handling
• Internal security rules

The information needed is the syntax and semantics, i.e. WHAT is flowing over the interface and HOW is its meaning. The receiving side of any interface must understand both correctly.