MSC Testing: Principles and Practice

Andrey Boytsov, MSC Testing Engineer
Alexander Frolkin, MSC Testing Engineer
Fedor Malakhov, MSC Testing Technical Leader
Tatiana Golitsyna, MSC Testing Project Leader

Content

- **MSC & SDL – What is it?**
- **Conception**
- **Benefits**
- **Constraints**
- **Efficiency Tracking**
- **Usage Statistics**
- **FMEA-based Methods**
- **Summary**

Acronyms

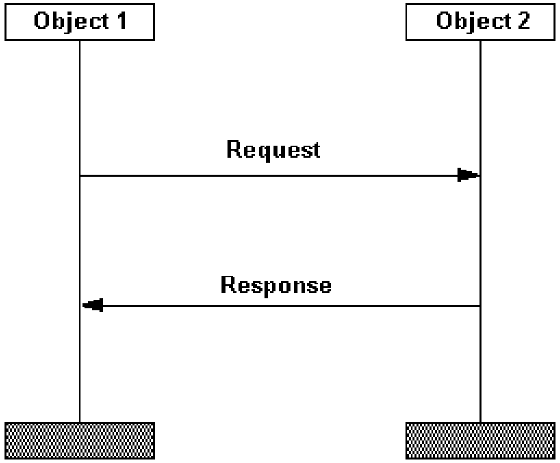
- **SDL – Specification & Description Language**
- **MSC – Message Sequence Chart**
- **GR – Graphical Representation**
- **PR – Phrasal Representation**
- **EFSM – Extended Finite State Machine**
- **FMEA – Failure Mode & Effect Analysis**

MSC and SDL – What is it? (1/2)

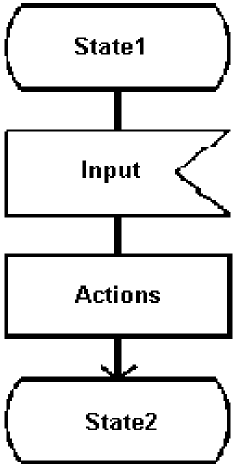
- **Specification & Description Language (SDL)**
  - SDL defines the system in terms of communicating EFSMs
  - Messages are discrete and carry information
  - The communication paths are explicitly defined
  - Can be represented either in GR or PR form
- **Message Sequence Chart (MSC)**
  - It is used to document the interactions between finite state machines
  - It is included in most SDL toolsets
  - Can be represented either in GR or PR form

MSC and SDL – What is it? (2/2)

MSC GR Example



SDL GR Example



MSC PR Example

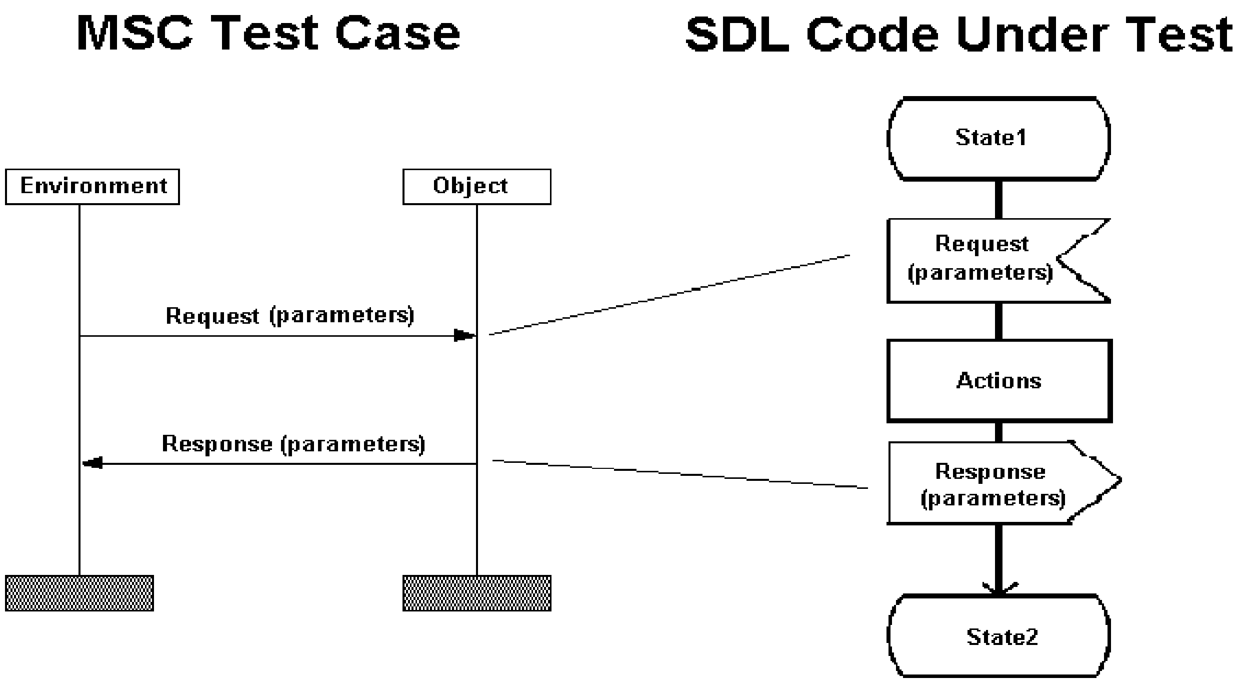
```
msc Example;
Object1: instance;
Object2: instance;
Object1: out Request,1 to Object2;
Object2: in Request,1 from Object1;
Object2: out Response,2 to Object1;
Object1: in Response,2 from Object2;
Object1: endinstance;
Object2: endinstance;
endmsc;
```

SDL PR Example

```
state State1;
input SwMgrCpa_FilterIpPortAck
task
{
Actions;
}
nextstate State2;
endstate;
```

MSC Testing Conception (1/2)

- **MSC should represent the exact scenario of how the SDL model should behave in some situation**
- **Engineer is able to run the scenarios on SDL model and automatically track, whether such a set of stimulus will cause expected responses or not**



MSC Testing Conception (2/2)

- **Tests are mostly developed on coding & maintenance phase**
- **All the SDL models should be provided with test sets**
- **Engineer’s job is to create & then maintain the test set (update the tests and exclude the outdated ones)**

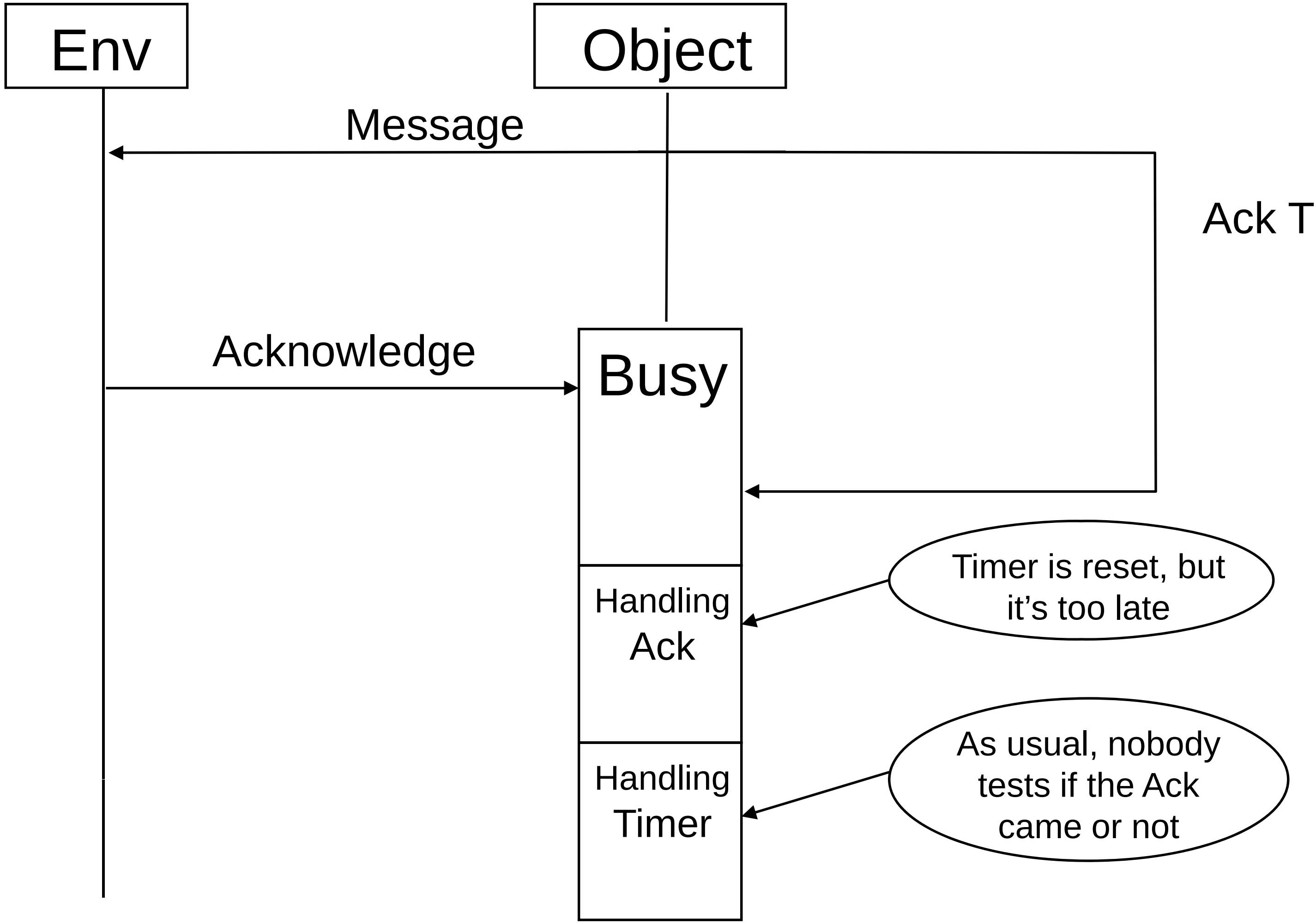
MSC Testing Benefits & Constraints

- **MSC is most useful for testing:**
  - Accurate timing
  - Queue management
  - Unexpected signals handling
  - Message sequences
  - Complex Message structure
- **MSC is not suitable for testing:**
  - Messages pack/unpack
  - Large message sequences
  - Complex platform interaction
  - Interaction with non-SDL external functions

Accurate Timing

- **Sometimes the test scenario requires the message to come at a very narrow timeframe (counted in milliseconds)**
- **On MSC testing all the transitions and actions are instant**
- **It must be explicitly specified, that the next signal comes after a period of time**
  - Of course, if it is not a timer signal
  - However, the signal order of arrival is preserved.
- **That feature brings up the ability to send the message in a narrow timeframe very accurately**

Queue Management (1/2)



Timer is reset, but it’s too late

As usual, nobody tests if the Ack came or not

Queue Management (2/2)

- **How to test it using MSC:**
  - SDL model sends the message to be acknowledged
  - Waiting for the guard timer to expire
  - As soon as it expired, the message is in the queue
  - Sending the acknowledge signal to the model
  - Swapping acknowledge & timer signal in a queue
  - Looking if the situation is handled correctly

Handling Unexpected Signals

- **The reasons of unexpected signals**
  - Serious message delay (most probable)
  - Effect of the other system issue
- **What to do? (suggestions)**
  - Handle gracefully
  - Continue work
  - Report the situation
- **Unexpected signals are easily sent via MSC**

Message Sequences

- **Some message sequences are easily triggered on box testing, while the others are not**
- **Some message sequences do not cause any output**
  - Filtering takes much more effort

Message Structure Testing (1/2)

- **Incoming messages**
  - Developers are able to track the messages and see byte stream
  - It takes a lot of efforts to decode
  - On MSC all the message borders, structure borders inside the message etc. are clearly visible
- **Developers leave the debug means to inject the message for testing purpose**
  - The message is injected in the format of raw bytes
  - The message structure is visible, and using the message in test scenario is not a problem

Messages Pack/Unpack

- **Pack/unpack involves translating the information structures to the real byte stream to be sent to the network**
- **SDL & MSC deal with information presented in structured way**
  - Translation of the message into byte stream is completely out of SDL coding scope
  - SDL just gives the command “Send message”, but does not specify, what exactly “Send message” means
- **MSC testing is completely inapplicable here**

Large Message Sequences

- **MSC test creation process:**
  1. Create the test: the incoming messages & expected output
  2. See whether the test passes or fails on SDL code
  3. If it passes – test development is complete (or both SDL code & MSC test errors set off the effect of each other)
  4. If it fails – investigate the issue more close & determine, whether it is a SDL code or MSC test error
  5. If it is an MSC test error – correct the test & go to point 2
- **The larger the test is, than more such cycles will be held and more time each cycle will take**
- **The test becomes very difficult to read and understand**
  - That greatly reduces the maintainability as well

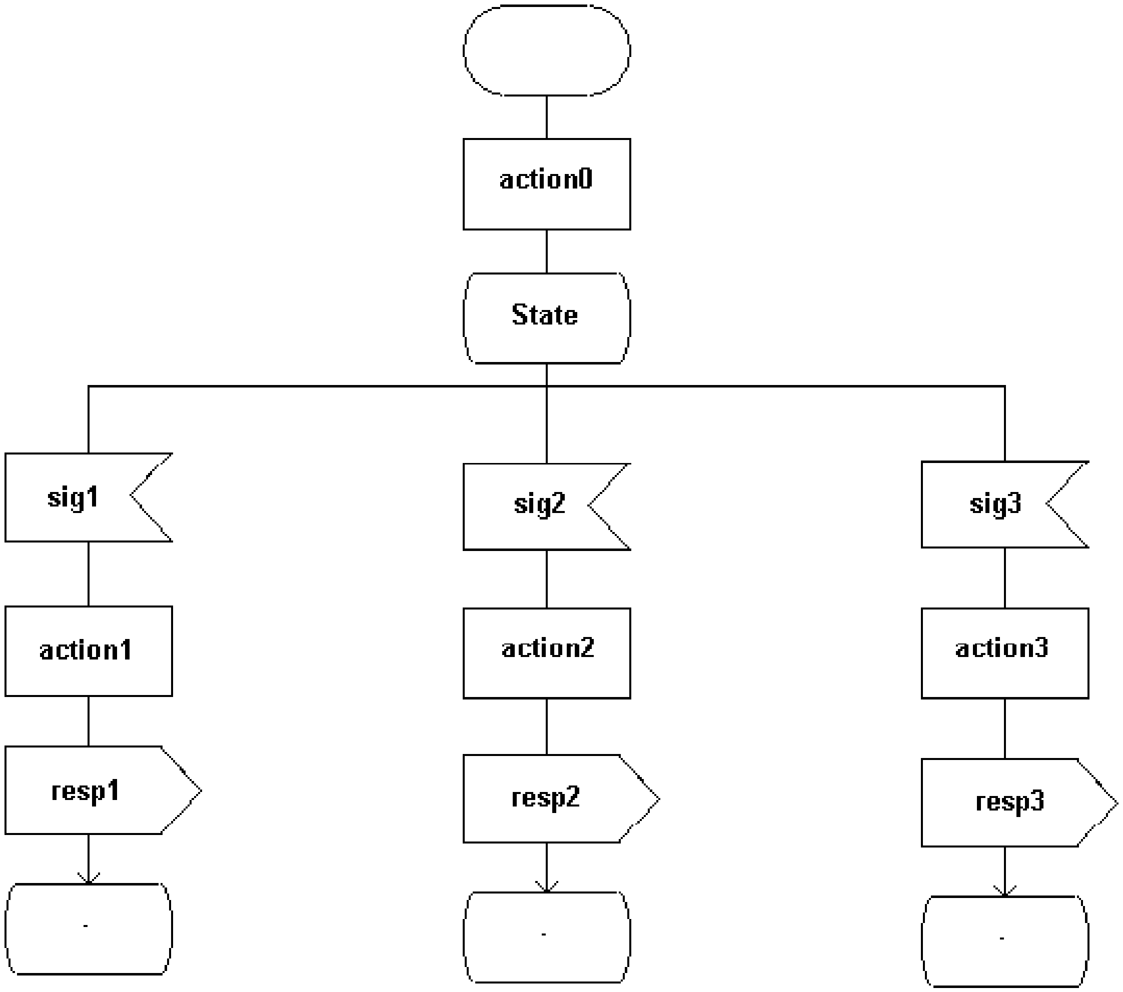
Complex Platform Interactions

- **SDL-developed code resides on some platform and use its services**
  - Most of the services are provided in the terms of function calls
- **Engineer runs the MSC tests using SDL code only**
  - Entire environment is simulated
  - No real platform interaction is possible
  - For MSC testing the real platform functions are replaced with special stubs
  - The stubs directly tell what the platform function result is
- **Complex platform interaction scenarios require too much MSC testing effort and are prone to errors**
  - Box testing might be more effective in that case

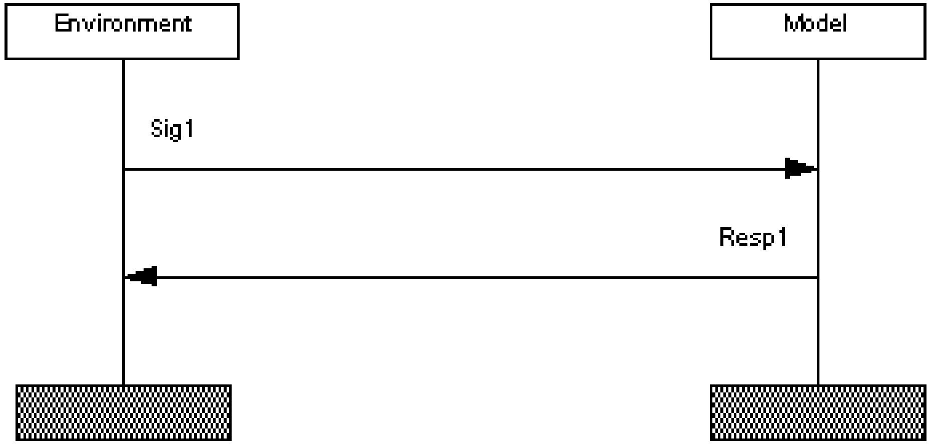
Interactions with External Functions

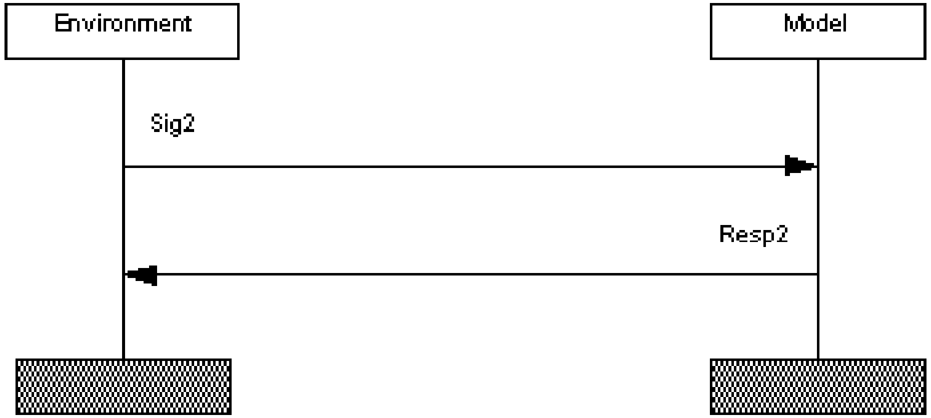
- **Sometimes, it is more applicable not to write the entire code in SDL, but to use C-code injections instead. Reasons:**
  - Code reuse
  - Platform interactions
  - If SDL code is translated to C before compilation to the target platform
- **The function is replaced by the stub during SDL code compilation for testing**
  - Real non-SDL external function is never involved
- **MSC testing is completely inapplicable**

MSC testing Efficiency Tracking – Example (1/2)



MSC testing Efficiency Tracking – Example (2/2)

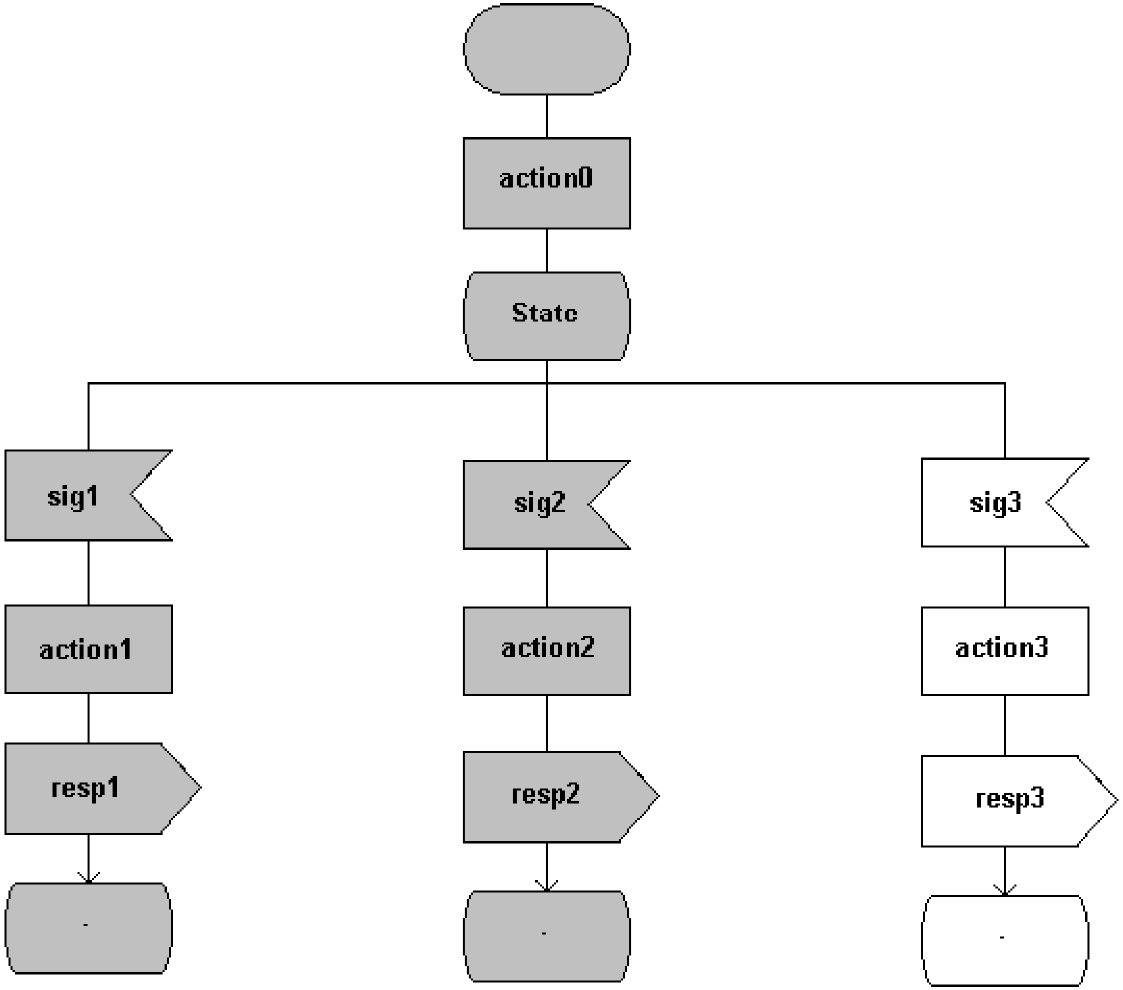




Symbol Coverage (1/2)

- **Symbol Coverage**
  - All the actions, incoming signals, outgoing signals, procedure calls etc. have the joint name “symbols”
  - If the symbol is reached even once during the test suite run, it is considered being covered
  - It does not matter, whether it is reached once or several times
  - Practically 80% symbol coverage is considered good

Symbol Coverage (2/2)



Transition coverage

- **Transition coverage collection requires:**
  - Consider all the possible combinations of state and incoming signal that may appear
  - Track, whether it is reached at least once or not.
- **In the example:**
  - 3 possible transitions: State-sig1, State-sig2 & State-sig3
  - Only 2 of them are covered (State-sig1 and State-sig2)
  - So, the transition coverage is 66.7%
- **Practically the 80% coverage is considered good**

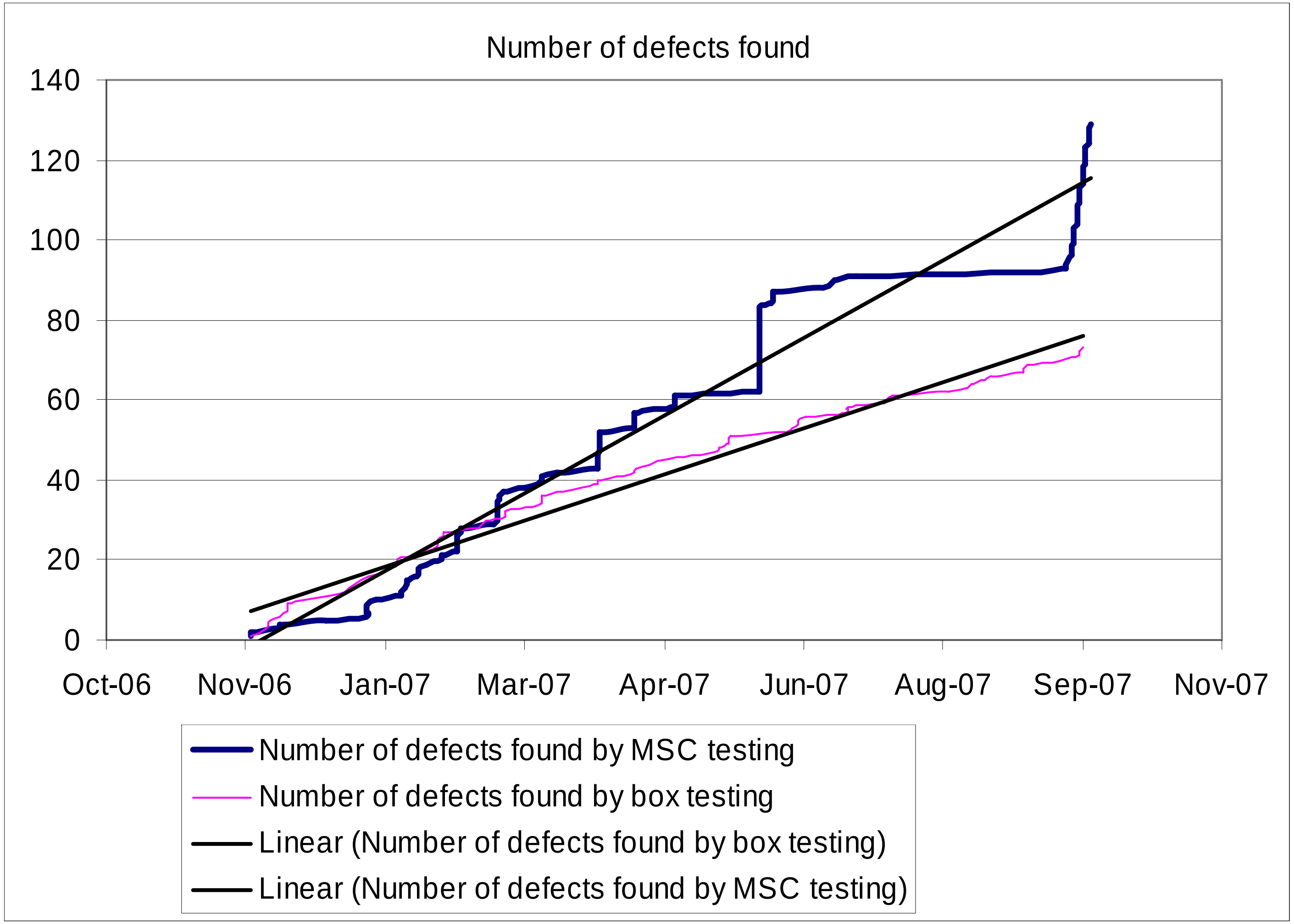
Coverage Analysis: Constraints

- **Even 100% coverage can’t grant the model is error free**
  - The most critical and non-obvious errors are caught by complicated scenarios
  - Coverage analysis tracks only if the symbol or transition is reached and takes no scenario pre-history in consideration
- **Coverage collection can’t show that the testing is sufficient, however:**
  - It can’t show if the testing is sufficient
  - It can show only if the testing is insufficient
  - It can show the exact areas, which should be tested

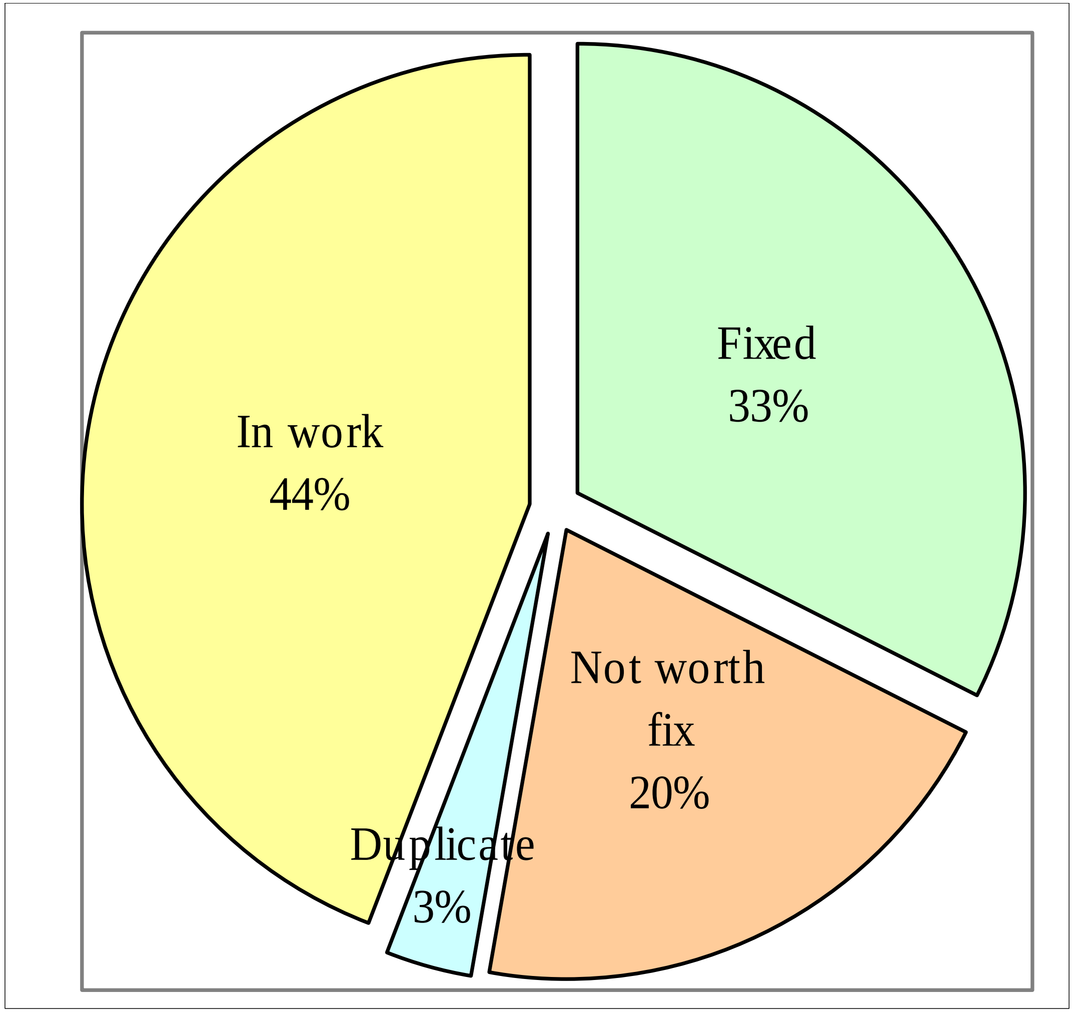
MSC Coding Style

- **Functional areas**
  - Test suite should be divided by functional areas
  - Groups of MSC tests that uses the same external test should can be grouped to the one executable module
  - Each functional area should have unique area identification code
- **MSC test case style**
  - Each MSC test case should include revision history area for future revision history control
- **Usage of macroses**
  - Common message headers, important constants and some parts of signals should be placed to common file of macroses
- **Usage of common references**
  - Some parts of MSC test should be placed in common references with input parameters (if needed)
  - Every input parameter should be defined as macros

MSC Testing Usage Statistics (1/2)



MSC Testing Usage Statistics (2/2)



FMEA-based Methods (1/2)

- **FMEA was developed by the US Military in the 1950’s and is used in the aerospace industry.**
- **FMEA is not specific to software engineering and can be used in any discipline where inputs to a system can fail**
- **This is a mature analysis tool that has been integrated into the development lifecycle of many engineering industries**
- **FMEA focuses on prevention. The objective is to look at all of the ways a product or process can fail, analyze risks, and take action where warranted.**

FMEA-based Methods (2/2)

- **Special number is calculated for every defect**
- **It is influenced by following factors:**
  - Severity Rating (critical functions affected – higher number)
  - Occurrence Rating (higher repeatability – higher number)
  - Detectability Rating (harder detection - higher number)
  - Complexity of Fix (more complex fix - lower number)
  - Testability of Fix (less testable fix - lower number)
  - Potential Field Impact (higher impact – higher number)
  - Field Recovery Difficulty
- **If the number exceeds certain value, the defect should be fixed**

Summary

- **In some areas box testing is the best choice, while some areas can be tested by MSC only.**
- **Coverage analysis can be the method to track if the test set is insufficient**
  - However, it has some serious constraints and does not mention complicated scenarios
- **MSC testing may detect the defects that are not really worth fix due to low probability and low impact**
  - FMEA methods can help to decide, what defects are worth fix and what defects are not.
- **Practically implementation of MSC testing shows good results.**
- **MSC testing combine with box testing can provide additional reliability to the system.**