Experiences of Testdriven Development

Eva Holmquist

Presentation

F7

International Conference On

Software Testing, Analysis & Review

November 19 - 23 Stockholm, Sweden

Friday 23rd November, 2001

Learning Objectives

The listeners shall understand the significance of:

- An independent Test Team.
- A structured Test Process, which is continuously improved.
- Configuration Management in test.

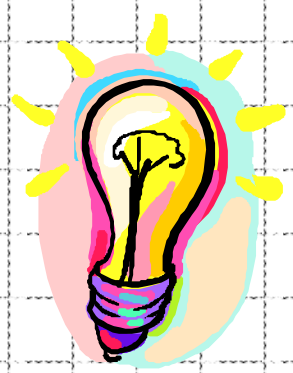
Independent Test Team

Maintenance Project

Current release.

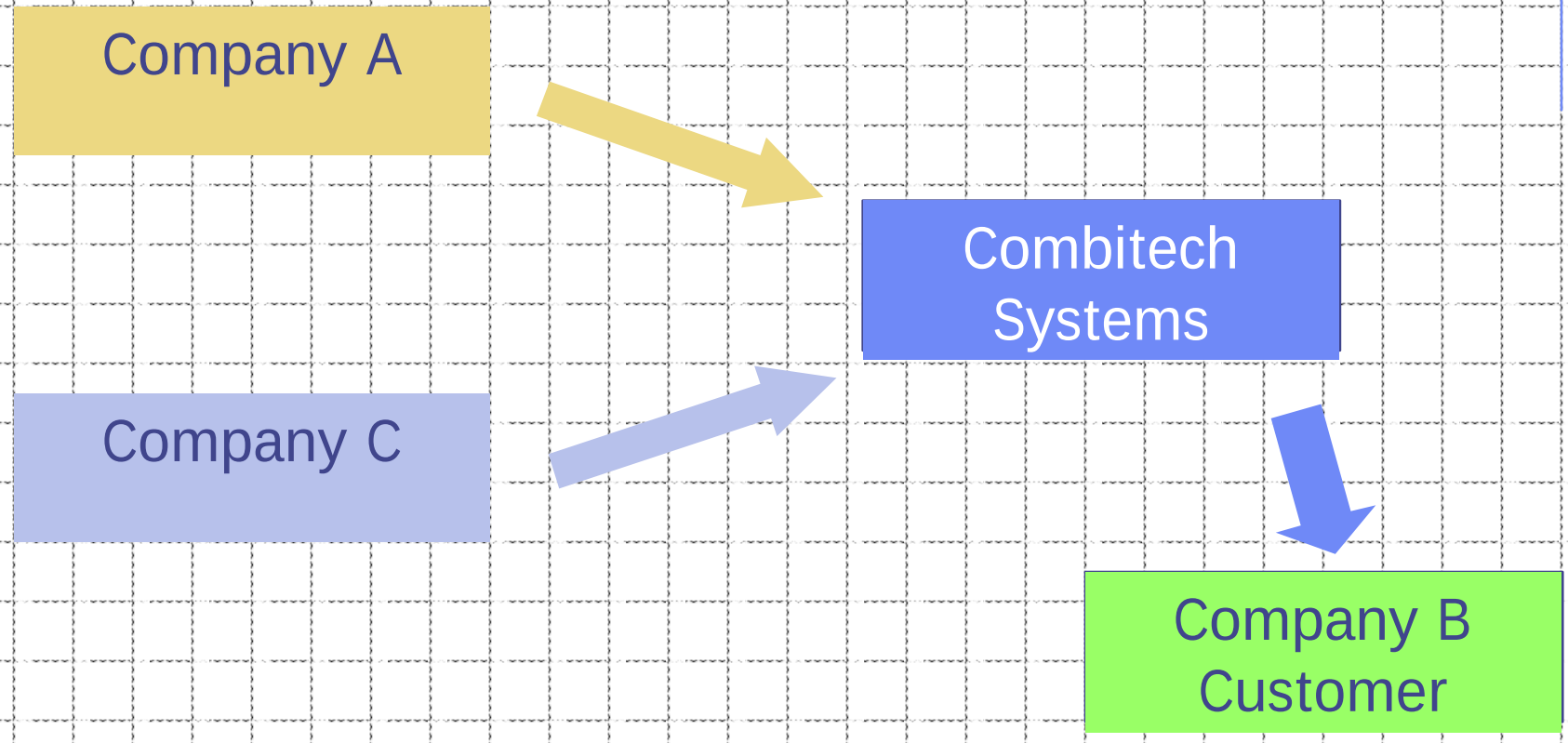
Change Request (CR)

New functionality



New release based on previous and with some corrections or new functionality.

Technical Project Interfaces



Device Definition

Project definition of a device:
"Methods or functions related to a specific group of services".
Example: Video, audio, modem and scart.

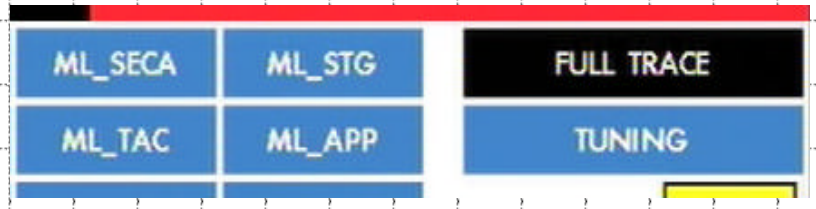
Devices in the Architecture

Company A

Combitech Systems

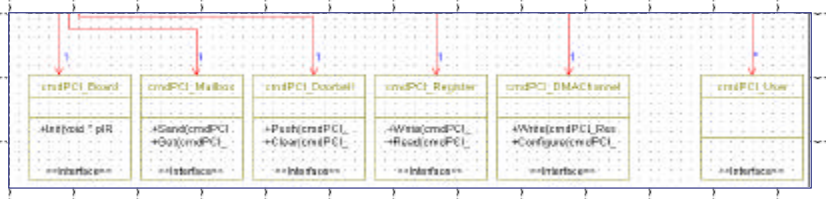
Company C

Test Architecture



Device Test

Company A



Device Layer Interface

Combitech Systems

Hardware

Company C

Test Levels

Normally, several test levels are defined in a software development process:

- Module Test (Device Test)
- Integration Test
- System Test
- Acceptance Test

Project Test Levels

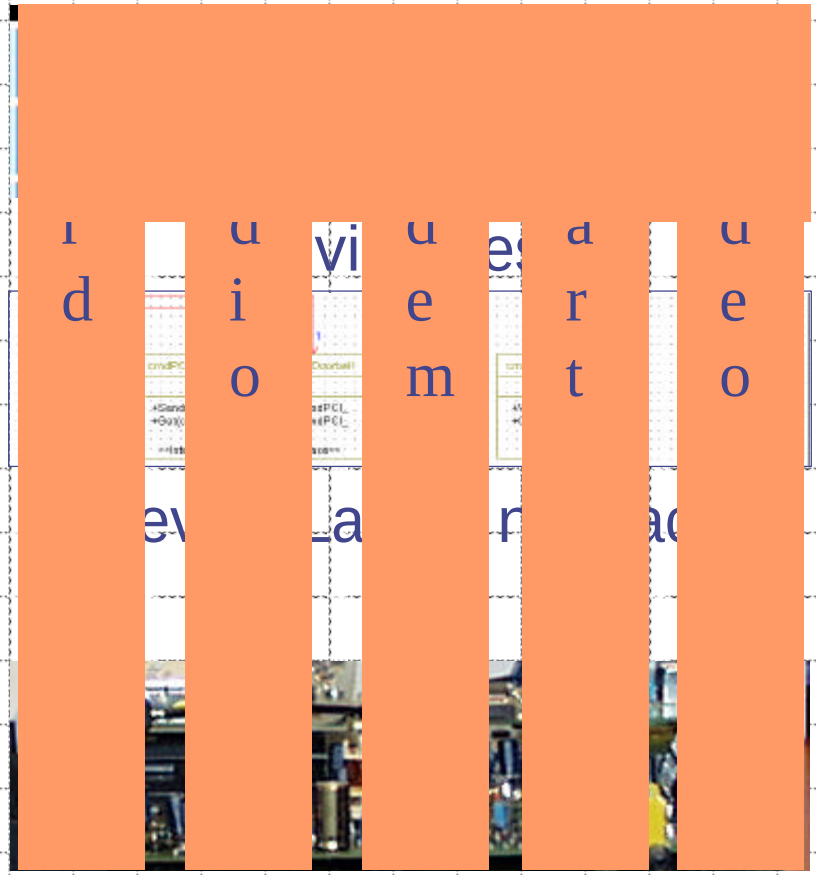
In this project the following test levels are used:

- Device Test, Combitech Systems
- Device Test, Company A
- Reduced Integration Test, Company A
- Reduced System Test, Company B (includes the acceptance tests)

Device Test Definition

- All methods or functions related to a specific service are tested by an application called Device Test.
- Methods or functions from other devices are used in order to support current test. Example: Video needs Tuner functionality.
- Device Test are performed on real hardware.

Device Test



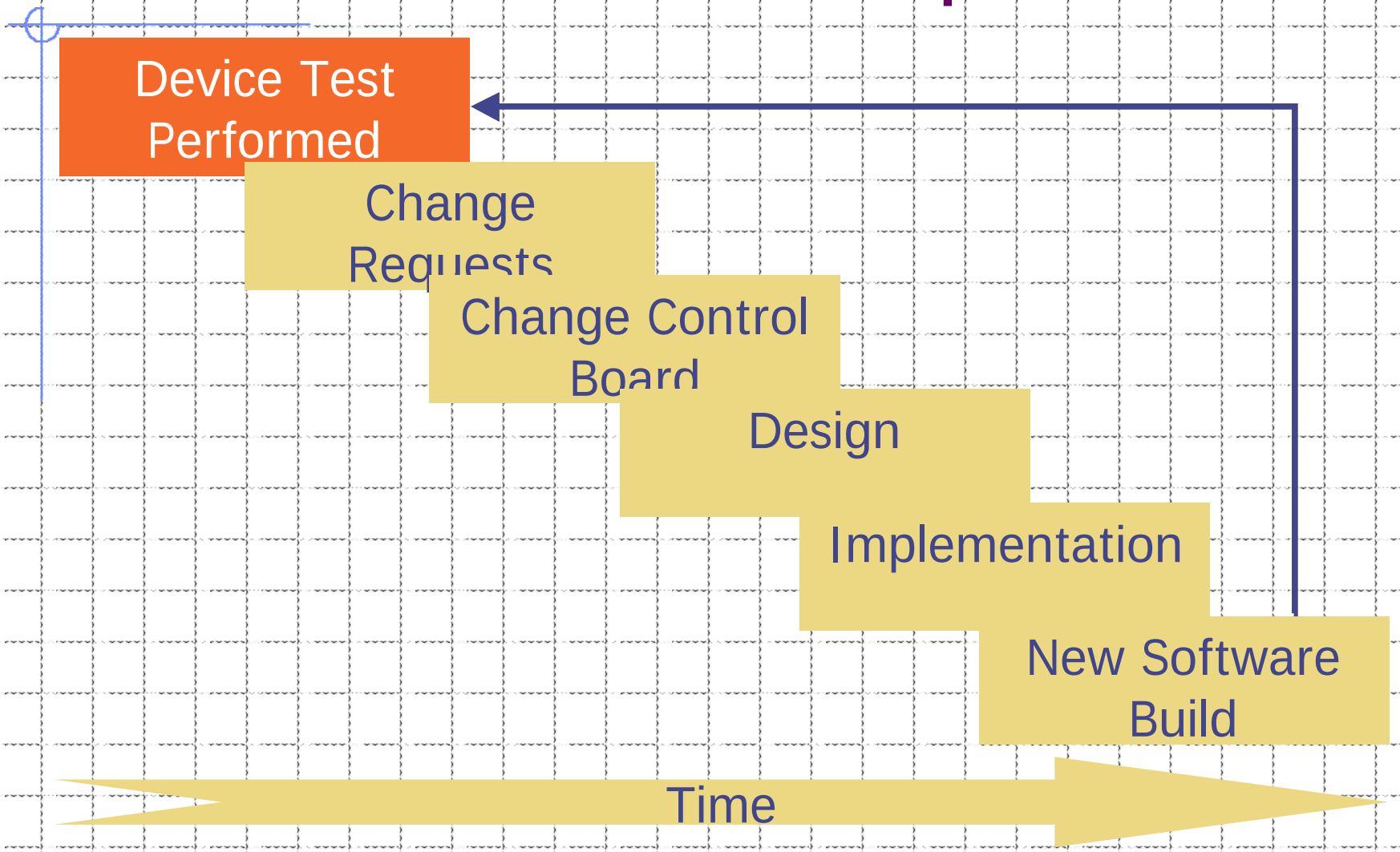
Hardware

Company A

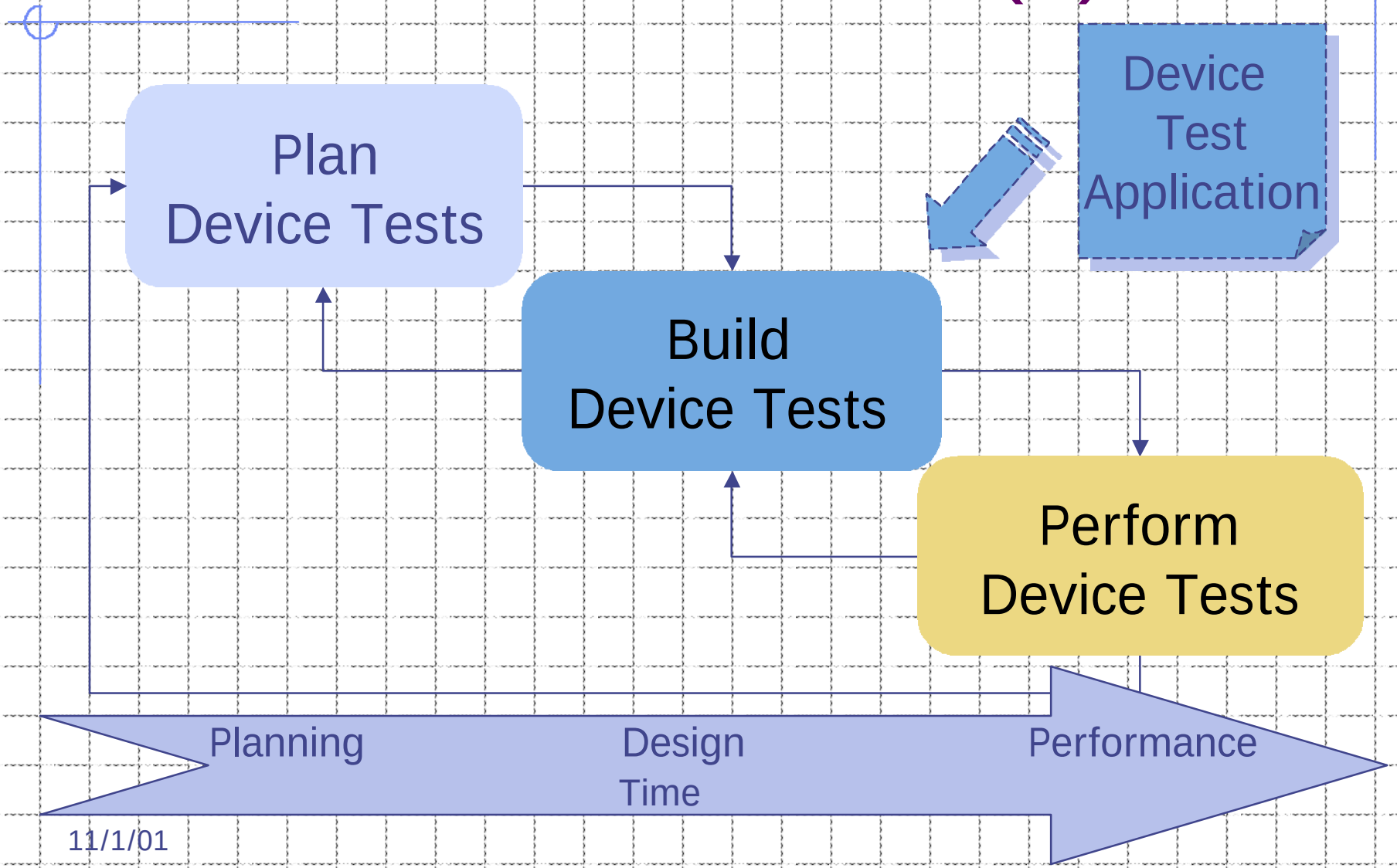
Combitech Systems

Company C

Testdriven Development



The Test Process 1(4)



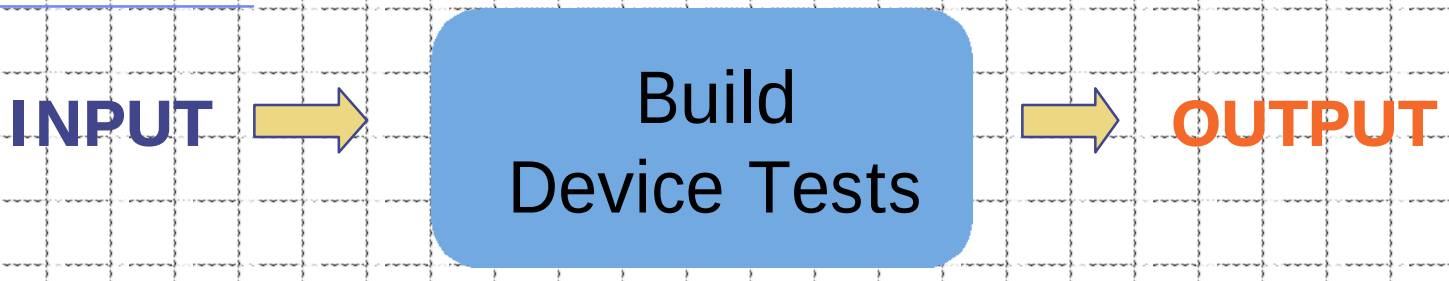
The Test Process 2(4)

INPUT → Plan Device Tests → **OUTPUT**

- ‘Build Meeting’
  - Suggested and approved changes
  - Which tests to build
- ‘DWP’*
  - Protocol over error corrected and/or new functionality implemented

* Development Work Package

The Test Process 3(4)



- 'DWP' with related CR
- New or existing Device Test Applications

- Device Test Package

The Test Process 4(4)

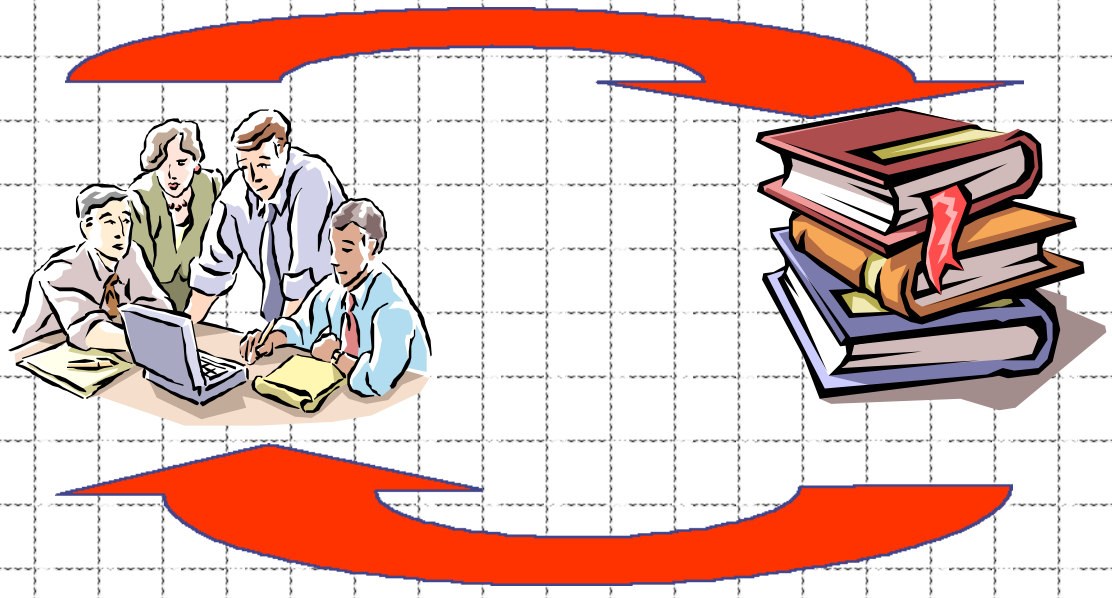
- Device Test Package
- CS Internally Test Descriptions
- Company A Test Descriptions

- CR with status *Verified* or *Failed*
- Device Trace Files
- Device Test Report for each device

Test Process Improvement 1(3)

The Test Process is improved continuously by the Test Team.

Team Members



Improved Process

Test Process Improvement 2(3)

- A problem is noticed by a test engineer.
- Test Manager calls to a meeting.
- The team discusses the problem and suggest a solution.

Team members

Test Process Improvement 3(3)

- The team members try the solution in reality.
- The Test Process is updated if the solution works properly.

Team members

Lessons Learned: Test Process Improvement

- Team members shall participate in process improvements.
- Continuous process improvement leads to an efficient Test Process.
- Try improvements in reality before updating the process.

The purpose of Configuration Management

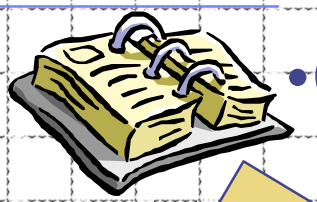
- Control and manage projects.
- Manage multi-site development.
- Distribute status (lifecycle) and defect information.
- Help developers and test engineers to synchronize their work.

Project use of Configuration Management

CM make it possible to keep track of status like:

- Correct version of a software release is delivered to the test team.
- New functionality.
- CR during corrections.
- If corrected defects re-occurs.
- Results from the Device Test

Example: Project Use of CM



•CR

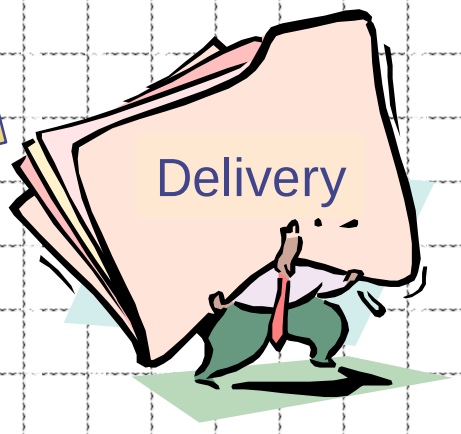
•DWP

Test Team

CM Tool

•DTR
•Status on DWP

•DTR



Configuration Management from a test perspective

- Handles that:
  - The right version of the product, included components, are built (to avoid regression!).
  - In which version a defect in the software was found.
  - Test results are reported with found defects and actual software version.
  - Different customers gets the right release of the product.

Example: Test Perspective of CM

Trace File

Defect found in Test Device Trace File

Created

Analyze Defect

Defect Analyzed

Ready

Closed

Test Team

Developer Team

CM Tool

By Test Manager

CR

Lessons Learned: Configuration Management

- All information saved shall serve a specific purpose.
- The work to produce the delivery documentation can be facilitated.
- Enables a complete traceability of all components included in a release.
- Enables full reproduction of customer anomalies.

Summary

The significance of:

- An independent Test Team.
- A structured Test Process, which is continuously improved.
- Configuration Management in test.

Experience of Testdriven Development

Speaker:
Eva Holmquist
Combitech Systems AB
Box 1017
SE-551 11 Jönköping
Sweden

Phone: +46 36 19 48 42
Mobile: +46 708 895 186

E-mail: eva.holmquist@combitechsystems.com

Co-speakers:
Malin Jernrup malin.jernrup@combitechsystems.com
Susanne Lieberg susanne.lieberg@combitechsystems.com
Tore Qvist tore.qvist@combitechsystems.com

Abstract
As our first delivery to the customer was a real disaster, it was clear that we had to do something. We formed an independent test team and created a development process with a test part. Configuration Management was also introduced to support the development and the test efforts. Our process includes a close co-operation between the developers and the test engineers. We think that this is one of the major success factors that helped us making an excellent delivery.

Definitions & Acronyms

Project Definition:	**Explanation:**
Acceptance Test	The process of comparing the end product to the current needs of its end users (according to Edward Kit).
Company A	Owns a standard for set top boxes.
Company B	Develops and sells set top boxes.
Company C	Delivers the hardware for the set top boxes.
CR	Change Request.
CS	Combitech Systems AB
Device	Methods or functions related to a specific group of services Example: Video, audio, modem and scart.
Device Test	All methods or functions related to a specific service are tested by an application. Methods or functions from other devices could be used in order to support current test. Example: Video needs Tuner functionality. Device test are: • Performed on real hardware. • Performed at level Module Test.
DTR	Device Test Report. CS Test Team reports the result of each test in the DTR. Contains a summary of defects.
DWP	Development Work Package. Defines the contents of a release, including all CR that are to be incorporated in the product delivery.
Field Test	A reduced System Test performed at customer site.
Integration Test	The process of combining and testing multiple components together. The primary objective is to discover errors in the interfaces between the components (according to Edward Kit).
KO	Knock-out
Module Test (or Unit Test)	The process of testing the individual components of a program (according to Edward Kit).
Set-top box	Digital TV Box
System Test	The process of attempting to demonstrate that a program or system does not meet its original requirements and objectives, as stated in the requirements specification (according to Edward Kit).
Test Levels	• Acceptance Test • System Test • Integration Test • Module Test

Learning Objectives

The learning objectives of this paper is that the reader shall understand the significance of:

1. An independent test team.
2. A structured Test Process, which is continuously improved.
3. Configuration Management in test.

Introduction

We (Combitech Systems AB) are responsible for development of the software in *Company B*'s set top box. The story started in 1998 when consultants from CS were engaged as operating system experts. Due to the complex software and lack of resources *Company B* realized that they needed a partner in 1999. As a consequence CS was asked to take responsibility for the maintenance of the software in the set top boxes. There were only a few anomalies left and they were included in the maintenance commitment. The project was planned to be finished in 6 months, but the project is still continuing with increased responsibilities. This was in the summer of 1999.
Company A, which is the customer of *Company B*, developed and performed the device tests. Several serious anomalies were raised. We had to take care of these and after a while, the real big problem appeared. No development process or test process had been defined. The source code was a real mess. A file system with several directories was used for configuration management. The device tests were the developer's responsibility and we did not have any personnel dedicated to testing activities. The developers performed both the development and the tests. When the first delivery was approaching it was evident that this approach did not work. We did not have time to run all device tests and we did not have time to correct the found defects.

Why couldn't we run all device tests? It was a personal conflict for the developer between the role as developer and as a test engineer. Every time a defect was found in a device, the developer had to prioritize. Should he interrupt the testing and start to locate the defect or should he continue testing? It was hard to prioritize testing when you knew you had a defect. Often the developer prioritized fixing the defects. Therefore most defects delayed the device tests.

At this time we saw the problem as a lack of resources. The developers did not have time to both perform the device tests and to correct the defects. The solution was to add a test engineer to the project to help out with the device tests.

The first delivery included an unwieldy pack of code, but practically no delivery documentation. There was for instance no list of found defects. The customer did not accept this! We had to do something and that very fast.

We realized that the problem was not just a matter of resources.

1. We had to give the personnel in the project the possibility to focus on dedicated tasks.
2. We had to have a structured way of working with device test.
3. We had to know exactly what was delivered and the known defects in the delivery.

Independent Test Team

Problem 1:
The personnel in the project had to have an opportunity to focus on dedicated tasks.

Our solution:
We formed an independent test team, which dedicated task was to test the software and keep track of all defects, which was reported. The developers had now the possibility to focus on fixing the defects and implementing changes. Accordingly they do not have to perform the device tests and find the defects. An independent test team does not mean that the test engineers belong to a different organization. The developers and the test engineers work in the same project and with the same project goals. In fact they work together in the process of locating and sometimes analyzing the defects. One major success factor is in fact the close co-operation between the developers and the test engineers. The test team is independent of the development activities and the developer team is independent of the test activities. They have the possibility to focus on the testing activities.

You could ask yourself why this change in the project had such major effect. In some books, e.g. “Software Testing in the Real World” by Edward Kit, it is proposed that the developer should do device testing. In this case we tried that approach, but failed. Why? We have described the difficult prioritizing that occurred as a result of the developer as the test engineer approach. How was this situation different when using an independent test team?

Example:
The test engineer run the device tests and finds a defect. He/she reports the defect when it is established that it is a defect. After that the test engineer can continue to run the device tests and report the defects. In this way we can run all the tests earlier, which means that defects are found earlier. This makes it possible to prioritize the defects and handle the major defects first. We could also keep track of defects left at delivery. There may be defects left at delivery, but we know about them. Our delivery documentation includes the remaining defects in the delivery and the solved defects. Company A appreciates this.

Lessons Learned

- A team in the project shall have the dedicated purpose of testing.
- An independent test team enables a known quality of the product, which makes it possible to prioritize defect removal.
- Close co-operation between test engineers and developers are crucial to the success of the project and independent test team removes the personal conflict between the role of developer and test engineer.

Organize your project with a team with the dedicated purpose of testing, and the quality of your deliverance will be higher.

Maintenance project

Working in a maintenance project means that every development of the software is triggered by either a Change Request (CR) from the customers, internally reported defects or by a new functionality.

In our project new functionality are introduced as new extended tests derived from our customer (*Company A*). Along with these tests, test descriptions are delivered. They describe the test in more detail and enable the test engineer to understand more about the tests. There are test descriptions for each device. These test descriptions are not complete and easy to misinterpret. Therefore we have created test descriptions of our own.

Project Description
CS is responsible for development of the software in set top boxes manufactured by *Company B.* Software for set top boxes are advanced, real-time applications providing basic functionality for Digital TV and Internet. Several international standards exists, all of them aiming to enable smooth application operation on any manufacturer's hardware. Currently most of the applications are aimed at providing the TV services ranging from channel switching to electronic program guides. Integration of Internet related services, e.g. e-mail, surfing, e-shopping etc. in these real time systems is rapidly evolving.

CS is also responsible for device test. This kind of testing aims at verifying the correctness of each modification with respect to integration and implementation. Any remaining issues are included in a test report, a so-called Device Test Report (DTR).

Technical Project Interfaces

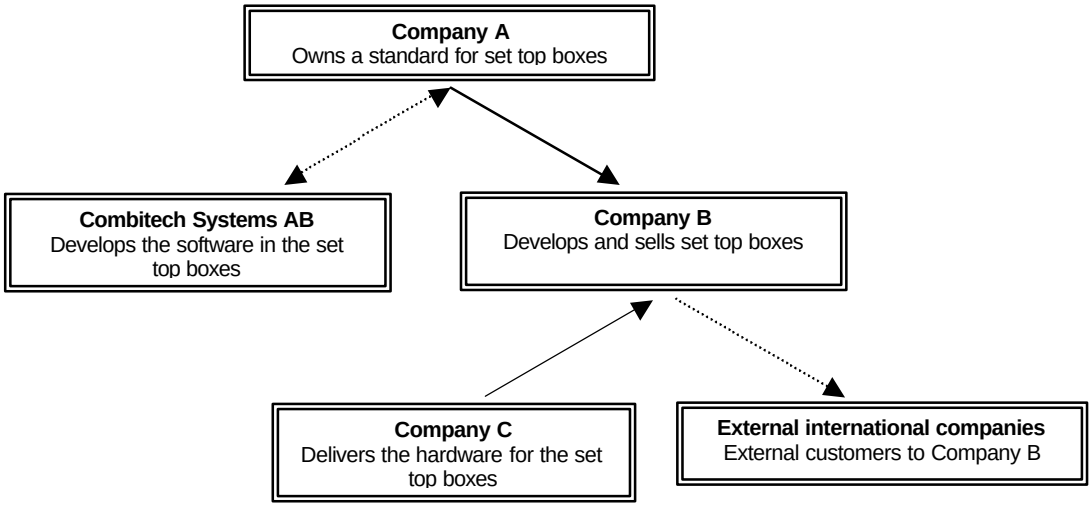


Figure 1. Overview of Technical Project Interfaces

- *Company A* own a standard for set top boxes and thus it owns the brand name, the specifications and the exclusive rights to it. The company is also responsible for the device tests and the integration tests. On top of that *Company A*, together with *Company B*, is responsible for system level testing.
- *Company B* develops and sells set top boxes according to the standards owned by *Company A* and is also responsible for field tests.
- *Company C* delivers the hardware for said set top box.

CS has had the main responsibility of providing the software for *Company B*'s set top boxes since 1999. Software is tested according to the following table.

- Field Test, i.e. reduced system test, including acceptance testing, *Company B*.
- Integration test, *Company A*.
- Device test, *Company A*.
- Device test, CS.

Architecture

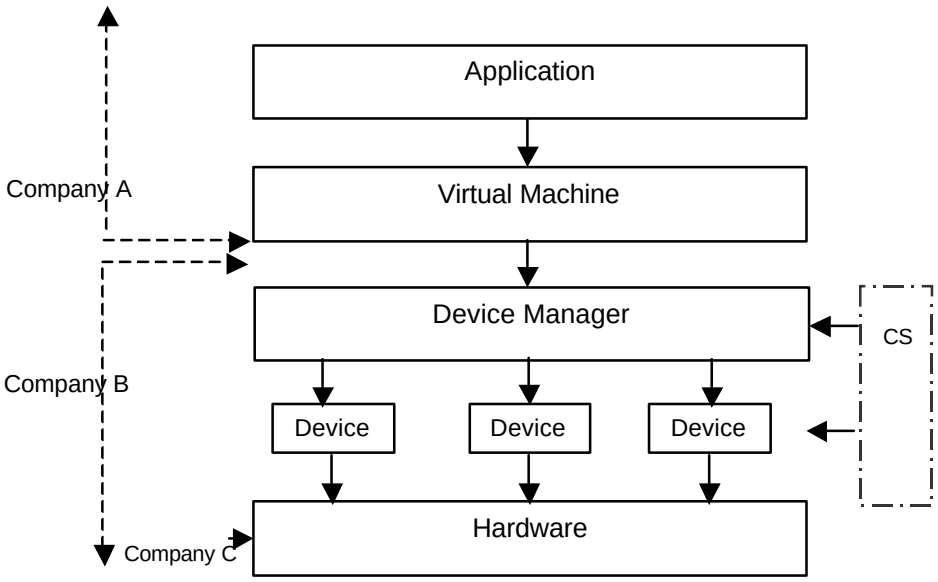


Figure 2. Architecture

- *Company A* – develops the application that shall be run against *Company B* set top boxes.
- *Company B* – produce the set top box with hardware from *Company C*.
- CS – is responsible for development of the software in *Company B*'s set top box, i.e. to implement the interface between the application and the hardware. This implementation is divided in different devices. Each device serves a specific purpose which can be methods or functions related to a specific group of services Example: Video, audio, modem and scart. The tests performed by CS are performed on these devices and on boxes produced by *Company B* and test applications developed by *Company A*.

Test Architecture

When CS test engineers start to test the different devices in the software, the application from *Company A* is replaced by a test application developed by *Company A*.

Device Test

Some of the device tests are extended because some test demands functionality from another device. For example to run Video test you must have a working tuner. There are test environment limitations, which results in incomplete device tests. For instance we do not have access to the satellites at the customer site.

Testdriven Development

A maintenance project is in a sense always testdriven. This is because most development occurs as a result of defects found in test or reported by the customer. This project is even more testdriven, because new functionalities arrive as updated Device Tests. Consequently the test engineer and the developer have to work in close co-operation when reproducing and explaining anomalies. Anomalies are unexpected results from Device Tests. The unexpected results can be defects, new functionalities or erroneous test applications from *Company A*. Anomalies occurring from new functionalities have to be fully understood. We see the test team as a support function.

We call this project for a testdriven development because:

1. Development occurs as a result of defects found in test.
2. New functionalities arrive as updated Device Tests.
3. The test team supports the development.

Test Process

Problem 2:
We had to have a structured way of working with device test.

Our solution:
For device testing, CS receives test applications specifically developed by a test team from *Company A*. This application shall verify the requirements placed on every software device. All tests have to be performed and approved before the software is delivered to *Company A*. *Company A* performs all device tests after delivery and they will get more confidence in our ability if we find at least all defects that they find. Up to this day we have succeeded in finding more defects than *Company A*. We have had less than five re-occurring defects. The test process has evolved since 1999 and today it bears little resemblance to the original one. Testing proceeds roughly according to the following steps:

1. Plan.
2. Design.
3. Perform.
4. Change request / handling.

1. Plan Device Test

Initially a building meeting is called where the project manager, the test manager, the integrator as well as any other concerned party are present. The project specification, including change requests on fixing the defects found in the previous delivery as well as all planned changes, serves as the foundation for the meeting. Here any of the following decisions will be made:

- Decide which changes to implement and link the devices to the test application from *Company A*.
- Decline a new build and include more changes.

Whenever the former decision is taken, which is most of the time, it results in a DWP, which serves as a foundation for the test engineers thus ensuring that they are aware of which changes were made and how to verify them. A dialogue with the developer is a handy way of sorting out difficult verification procedures and result interpretation.

2. Design Device Test

Beside the DWP the test engineer has access to a test description from *Company A* to further facilitate verification of the made change. There is one of these for each device. These descriptions are subjected to version control and are continuously adapted as the devices evolve. Whenever there are ambiguities in the source code or the test specification, a technical issue is sent to *Company A*. Such queries along with any clarifications are documented. The test engineers at CS maintain separate test descriptions as it greatly simplifies the testing process and achieves a more detailed level of testing. Tests must be repeatable so we are able to quickly verify that the defects are fixed. Therefore many parameters must be duly noted in the test descriptions to make sure that we can be confident in the verification of the fixes. Some examples include:

- Hardware and software settings.
- Order of sub tests wherever applicable.
- What to look for in which segment of the test.

This also implies rigorous version control of both test applications and device software, which is done by a dedicated tool, a so-called CM Tool. Testing demand some basic tools among which we find such items as:

- TV
- Set top box.
- Test stream, antenna signal.

In the device tests performed at CS an already written test application is obtained from *Company A* that owns the standard. A test group at *Company A* writes the test application. The test application shall verify all requirements that are allocated to each device in the software. All test must be performed and all defects explained before the software is delivered to *Company A*.

New test application comes once a month. We are able to rule out the influences of the test application if we know all defects found in the test application in the previous test round.

3. Perform Device Test

Test streams are sent from designated generators to each test engineers' work place through cables formatted either as antenna signals or as Cable TV signals. The test application is shown on screen and the test engineer acknowledges each result by remote control. The test engineer presses OK if the result is as expected and KO, as in knock out, if the test fails. Simultaneously the results are assembled in a trace file. Tests may also be automated in which case it is marked as such in the trace file. Test engineers are encouraged to keep separate notes of the test run so that no queer results are left out. These notes are very important since it may be difficult to express in source code how to test behavior or it may be difficult to recreate the exact condition to excite the defect. Each test run is made up of new versions of software and test applications. The first order of business is to link the new software to an older test application in order to rule out any influences from the test application if any defect is found. When the test run has been completed a DTR is compiled where all failures, defects, are noted and numbered.
A trace file shows the outcome of the test run. The execution of tests concerns judgment test and automated tests.

The automated tests are run by the test application and the test result is evident in the trace files.

The experiences among the test engineers are very important and valuable when it comes to the judgments tests. How can you tell for example that there is a sequence missing in the test from previous to current if there has been a switch of test engineers in the team? This is a quality question.

There are some difficulties connected to when another company writes a test application. Every time an anomaly, i.e. an unexpected result, occurs test engineers and developers must sort out the source of the problem, i.e. is it in the test environment, the test application or in the new software. By knowing which configuration you had last time the software where tested you can build a new test with the new software and the old test application to see whether to problem still is present (probably defect introduced in new software) or if it disappeared (probably defect in new test application).
Perform Device test take the main part of the time during the test process.
In maintenance project, test has to verify corrected defects. To avoid loss of CRs to be verified by device test, a DWP is created. The DWP can be seen as a list of all CRs that should be verified in this test round. The test engineers need only to know the DWP to find all corrected CRs.
A different problem the test engineer has to handle is the knowledge about each device test. *Company A* has developed them and without test descriptions. To understand the test, reverse engineering has to be performed. This is a waste of time for us. This has to be done more or less after each new release of device tests.
Appearance of unexpected behavior is difficult to investigate because the development environment is unknown for device test engineers.

Test Process Improvement

We have during the project improved the test process continuously. This has given us an opportunity to react very quickly and adjust these changes as they come up. All improvements have been suggested, tested and implemented by the members of the test team. This has made the team members more committed.

The work with the test process is performed in the following way:

1. A test engineer notices a problem when working.
2. The Test Manager calls to a meeting.
3. The problem is thoroughly discussed by the team members.
4. The team comes up with a solution to the problem.
5. The test members try the solution in reality.
6. If the trial period was successful the test process is updated.
7. Back to step1.

For instance:
Bob runs the Device_A Test. He has to verify a couple of corrected CRs. Unfortunately the test fails. After spending a couple of hours trying to figure out what is going wrong Bob discovers that he has performed the test on the wrong hardware (box). To avoid this situation again the problem is discussed at the next meeting and a solution is suggested. Since the boxes were not marked when they arrive from *Company B*, the team decides on marking the boxes. In this way all test engineers immediately can see what kind of hardware is in the box.

If the proposed solution works in reality the improvement is made permanently. If the proposed solution fails the test engineer tries to figure out a new and better solution.
Later on we have handled this problem by organizing the way we keep track of the boxes. Therefore fewer mistakes concerning the boxes are made. This makes the testing more efficient. By improving our process step by step we have in this way been able to learn along the way. Thus enabling more efficient solutions to problems.

We have discovered the following key success factors of process improvement:

- The team members initiate most process improvements.
- The team members can influence, participate in and feels committed to the process improvements.
- The change is tried in reality before the change is made permanent.

We have learned that:

- Fast feedback on changes makes it easy to evaluate and adapt to the changes.
- A continuous process improvement is a free education of the team members.
- The Test Process works in reality and not only in theory if the test team try changes before they are made permanent.

<u>Lessons Learned</u>

- Team members shall participate in process improvements.
- Continuous process improvement leads to an efficient Test Process.
- Try improvements in reality before updating the process.

Improve a process with the team who shall use it and the team will be committed.

Introduction to Configuration Management

Configuration Management (CM) is the management of system change. When a system is maintained, it is important that changes are incorporated in a controlled way. CM is a discipline within software engineering with the aim to control and manage projects and to help developer and test engineers synchronize their work with each other. This is obtained by defining methods and processes to obey, making plans to follow and by using a CM Tool. This is a controlled way to avoid the most frustrating software problems in reality. These problems take time to fix, they often happen at the worst time, and they are totally unnecessary.

Example: A difficult defect that was fixed at great expense suddenly reappears, a developed and tested feature is mysteriously missing or a fully tested program suddenly doesn't work.

Configuration management helps to reduce these problems by coordinating the work products of the many different people who work on a common project. Without such control, their work will often conflict. From a CM perspective, it is important that the all members of the different teams in a project could receive information about what others in the team are doing, how the project is developing, its status, which changes have been done etc. It is important to support the distribution of files and concurrent, simultaneous changes. Common frequently asked question (FAQ) answered by the use of CM are

- What is my current software configuration?
- What is its status?
- How do I control changes to my configuration?
- How do I inform everyone else of my changes?
- What changes have been made to my software?
- Do anyone else's changes affect my software?

The CM concept is a simple concept, but is often complex in its detailed practice. It is applicable to and critically important for programs as well as test and data, and for code as well as all preceding life-cycle documents in the derivation path of the code. A CM tool can help developers, test engineers and project leaders with their daily work. The CM tool makes it possible to organize, manage and protect software assets, supporting effective software configuration management across the entire enterprise.

Project use of Configuration Management

Problem 3:
We had to know exactly what was delivered and the known defects in the delivery.

Our solution:
We handle these factors according to:

- *Simultaneous update*; to make sure that the last one to make changes does not destroy the others work when two or more developers work separately.
- *Shared code*; to make sure every developer is notified when a defect is corrected in a shared code.
- *Common code*s; to make sure that all the users are notified when common program functions are modified.
- *Versions*; to make sure that we can handle evolutionary releases i.e. with one release in customer use, another in test, and a third in development; defect fixes must be

propagated between them. If found by the customer, for example, a defect should be fixed in all the later versions. Similarly, if a defect is found in a development release, it should be fixed in those prior versions that contained it. In larger systems with several simultaneous active releases and many programmers working on fixing defects and enhancement, conflicts confusion are likely.

These problems stem for confusion and lack of control, and they can waste an enormous amount of time. The key is to keep track:

- On that correct version of software release is delivered to the test team.
- Of Change Request (CR) during correction and also make sure that no corrected defect re-occurs.
- On new functionality.

Example: Project use of Configuration Management
The Configuration Management Tool is a center part of all developing and testing activities. A CR (Change Request) is created for all internal and external anomalies. The Configuration Management Tool handles all CRs. The DWP (Development Work Package) can be seen as a list of all CRs that should be verified in this test round. Based on the information in the DWP the Test Team performs the Device Tests. Each Device Test is documented in a DTR (Device Test Report). After the test run the Test Team updates the status of the CR (verified or failed). Report Generator from the Configuration Management Tool Database generates a draft to the delivery documentation. The delivery documentation includes all DTRs and selected parts from the Change Requests

Configuration Management from a test perspective

From our testing perspective CM is important since it handles that the right version of the product, included components, are built and also in which version a defect in the software was found. The found defects from the test of the actual software version are reported in a test report i.e. DTR. Beside this we use CM to handle that different customer gets the right release of the product.

These are some of the key success factors that we have used in our project. The CM support us in the management of:

- Configuration control
- Change Request
- Revisions
- Versions
- Relations

Example: Configuration Management from a test perspective
The Test Team finds defects during Device Tests. This triggers a continued revision control process there the DTR (Device Test Report) summarize all defects in the Trace File. Initially the DTR has the status of “Created” but once all the defects has be entered along with explanations and linked to a condition it receives the status “Analyze Defect”. The appropriate developer now owns the DTR, i.e. the developer is responsible to explain all defects in the DTR. When the responsible developer has analyzed the defects, old and new, a CR is created for each new defect by the developer. The developer changes the status to “Defect Analyzed” when the analyze is completed. It is the responsibility of the test engineer to understand new CRs made by the developer and that the DTR is complete. Each defect

must be traced to the identity of the anomaly report received from the customer in a traceability matrix before changing the status to “Ready”. This will be used when the software is delivered.
On the eve of delivery the DTRs and trace files are included in a baseline and the Test Manager updates the status to “Closed”.

By using CM we have up to this day only had less than five re-occurring defects. As a result of this *Company A*’s confidence in our ability has increased.

Lessons Learned

- All information saved shall serve a specific purpose.
- The work to produce the delivery documentation can be facilitated.
- Configuration Management enables a complete traceability of all components included in a release.
- Configuration Management enables full reproduction of customer anomalies.

Use Configuration Management to support the development and test activities and you will be able to reproduce all defects.

Our conclusions

During this project we have learned that different factors are crucial to the success of a project. We have learned this by in the beginning of the project we realized that the following problem where important and in strongly need of solution to achieve a successful project.

Problem in the beginning of the project:		Our solution to the problems
1	We had to give the personnel in the project the possibility to focus on dedicated tasks.	An independent test team.
2	We had to have a structured way of working with test.	A structured Test Process, which is continuously improved.
3	We had to know exactly what was delivered and the known defects in the delivery.	Using Configuration Management in test.

As our first delivery to the customer was a real disaster, the faith in our ability to handle the project was at zero. It felt like we slipped into the project on a banana skin. Soon we gained our balance and bit-by-bit we have regained the confidence of our customer and our customer's customer and now they have complete faith in our ability. In this paper we have tried to show you some of the lessons learned when turning a disaster to a success. We hope you will make good use of our experiences.

Experiences of Test Driven Development

Eva Holmquist

Eva Holmquist is a senior System Engineering Consultant at Combitech Systems AB with experience of system development ranging from software development to project management. She is specializing in consulting, mentoring, and training software development teams in inspection and test. Eva is one of the founders of Combitech Systems AB, which was founded in 1992.

Tore Qvist, Malin Jernrup and Susanne Lieberg are consultants at Combitech Systems AB. They are working in the project described in the case study as Test Manager, Test Team Leader and Test Engineer.

Combitech Systems AB is a knowledge-oriented development company with leading-edge competence in technical real-time systems. Within this field the company provides services for systems and software development, electronic and mechanical design, management support and education.