******

**Heilbronn University**

**Software Engineering**

Prof. Dr. -Ing. Thomas Benz

**Bachelor Thesis**

**„High Availability” Computing Infrastructures**

**Stress Test Framework for Validation of SAP System**

Editor: Duncan Muema Muasya

Matriculation Number: 180087

Supervisor: Prof. Dr. - Ing. Thomas Benz

Supervisor @ IBM: Eric Kass +49-(0) 6227-760834

[KASSERIC@de.ibm.com](mailto:KASSERIC@de.ibm.com)

Begin: 02.12.2014

Copyright ©

Duncan Muema Muasya

2014/2015

*All rights reserved*

STATEMENT BY THE AUTHOR

I hereby declare ….

Heidelberg, den 02.12.2014

Place, Date Signature

# DEDICATION

I dedicate this document to Agnes Mutio (the fond Munene), my grandmother. I owe you and I know you can’t wait to see me move on to another stage in my life.

# Abstract

This thesis focuses on writing a software framework that tests SAP High Availability

(HA) structure for its stability.

The method of establishing a comprehensive HA test is to validate that redundant resource can switch between nodes in a cluster without causing any loss of data.

The software test framework will force HA events (like failover, switch over) to occur, validate data as well as the application consistency without any single point of Failure.

The sequence of testing is as follows: End SAP Processing, update SAP code on nodes in the cluster, start SAP processing, initiate switchover and failover of resources in the cluster, and finally validate data consistency.

The results of HA test framework will be integrated into SAP standardized Output Quality (OQ) nightly test infrastructure so that nighty results of the HA tests will be automatically reported to OQ quality management.

# Keywords

# Acknowledgements

Table of Contents

[Abstract i](#_Toc404851249)

[Keywords i](#_Toc404851250)

[Acknowledgements ii](#_Toc404851251)

[Table of Figures iv](#_Toc404851252)

[List of abbreviations v](#_Toc404851253)

[Preface vi](#_Toc404851254)

[1 Introduction 1](#_Toc404851255)

[1.1 Background 1](#_Toc404851256)

[1.2 PowerHA SystemMirror 2](#_Toc404851257)

[1.3 Aim and Contribution 4](#_Toc404851258)

[1.4 Limitation of the thesis 4](#_Toc404851259)

[1.5 Thesis Structure 4](#_Toc404851260)

[2 Existing work 5](#_Toc404851261)

[3 Literature Review 6](#_Toc404851262)

[4 Methodology 6](#_Toc404851263)

[4.1 Output Quality Driven w/ Java glue code Sub item 6](#_Toc404851264)

[4.2 Comprehensive Java based Test Framework 6](#_Toc404851265)

[5 Presentation and Analysis of Results 6](#_Toc404851266)

[5.1 Flow Charts of Test Scenarios 6](#_Toc404851267)

[5.2 GUI output of Tests 6](#_Toc404851268)

[6 Conclusions and Future Work 6](#_Toc404851269)

[Bibliography 7](#_Toc404851270)

[Appendices i](#_Toc404851271)

[References ii](#_Toc404851272)

# Table of Figures

[Figure 1: typical sap landscape 11](#_Toc404334410)

[Figure 2: as/400 menu driven console 1](#_Toc404334411)

[Figure 3: disaster recovery with HA 2](#_Toc404334412)

[Figure 4: cluster nodes 3](#_Toc404334413)

[Figure 5: sap HA concept 4](#_Toc404334414)

[Figure 6: replicated SAP-ENQ with PowerHA 1 5](#_Toc404334415)

[Figure 7: replicated SAP-ENQ with PowerHA 2 6](#_Toc404334416)

[Figure 8: replicated SAP-ENQ with PowerHA 3 7](#_Toc404334417)

[Figure 9: replicated SAP-ENQ with PowerHA 4 8](#_Toc404334418)

[Figure 10: sap output quality tests 9](#_Toc404334419)

[Figure 11: sap output quality test (statistics corner) 10](#_Toc404334420)

# List of abbreviations

ASCS ABAP SAP Central Services

ERS Enqueue Replication Service

CRG Cluster Resource Group

APP Application Cluster Resource Group

ISAP Independent Auxiliary Storage Pool

UDFS User Defined File System

HA High Availability

IBM i IBM Integration

TCP/IP Transmission Control Protocol/ Internet Protocol

# Preface

# Introduction

Most companies businesses want their services to be available to their customers seamlessly as may be needed. To avoid outages and reduce downtimes, companies have to provide continuously available services for their customers. IT support system manufacturers must, therefore, offer highly dependable solutions to address these gaps in order to guarantee service continuity and to forestall loss of business.

IBM’s High Availability architecture (HA) is a good solution for this problem. The idea is to plan for the worst case scenario, be it hardware or maintenance failures, so that should one system fail, another backup machine is immediately activated and simultaneously kicks in.

Businesses running SAP for their operations need an operating system that has availability functions built into it. This creates a more robust environment for SAP having no Single Point of Failure (SPoF) and making SAP highly available.

This chapter offers a brief background of the themes in this landscape.

## Background

IBM i is an operating system which runs on IBM Power Systems. It was first developed as AS/400 and over the years it underwent further developments and changes to the current IBM i.

It is menu driven….

## PowerHA SystemMirror

PowerHA SystemMirror is designed to provide replications possibility for IBM i.

IASPS is an independent auxiliary storage pool for which PowerHA SystemMirror for i is built upon.

The data to be replicated is located in the IASP, which is created from specific disk drives. There is always asynchronous replication between the source and the IASP.

SAP and PowerHA team work closely together to provide availability for the database and the application servers for the entire SAP.

The high availability options for SAP initially had its entire component contained on one host: the SAP application server and all the components of the *central instance,* the database and the file system (see Figure 1: typical sap landscape). If one component failed, the e­ntire SAP application went down. The Central Instance performs server processing as well as lock management (Enqueue) as well as message handling (Message Server)

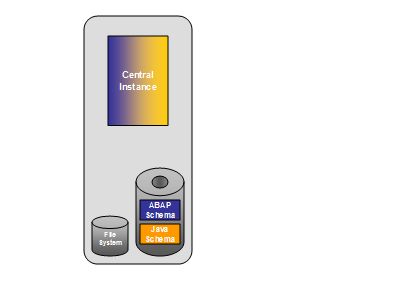


Figure 1: typical sap landscape

A Node is a standalone server or logical partition (LPAR). Different nodes of a high availability configuration are referred to as a cluster as shown in the Figure 4: cluster nodes.

By setting up two up two or more of the environments with equivalent configurations on different hosts or nodes and mirroring the database and the file system between the nodes as shown in Figure 4: cluster nodes makes it possible to recover most of this components from a disaster.

SAP Central Services for Java (SCS) and ABAP SAP Central Services (ASCS) are primarily the combination of the SAP Message Server and the SAP Enqueue Server; both formerly residing within the SAP Central instance. The message Server is the communication hub for distributed components of an SAP System. The Enqueue Server is the central application lock manager providing flow control and contention management between SAP applications running in parallel within multiple work-processes on multiple nodes.

The motive for separating message and Enqueue services into a separate instance was specifically to facilitate high availability by; firstly, providing the architecture to support a replica of the Enqueue server state; and secondly, the new continuous availability of Enqueue and massage services by way of backup SCS and ASCS instances. The Enqueue replication server (ERS) running on a backup node actively records the state of the active Enqueue Server.

In Figure 5: sap HA concept SAP GUI clients communicate to one of many application servers (middle) directly or perhaps by first load-balancing via the Central Services message server. Application servers maintain persistent connections to the clustered database server, clustered file system, and the clustered SAP Central Services.

If any of the clustered components switch or fail-over, the application servers automatically re-establish connections to the failing component.

## Aim and Contribution

The primary focus of this thesis is to explore the High Availability (HA) architecture to test SAP System. Two IBM i server machines each installed with SAP System, File-System and database will be used for the HA test.

The idea is write a stress test framework that loads HA drivers, invoke methods and verify results. The stress test framework should start and stop the SAP components on one machine and call the HA to start the switch over and then display the results.

Additionally setup an auto mode, which dynamically configures HA cluster, invoke methods and verify results in the web browser. See Figure 10: sap output quality tests and Figure 11: sap output quality test (statistics corner).

## Limitation of the thesis

## Thesis Structure

Chapter 1 covers the Introduction, giving a brief detailed account of High Availability landscape, file systems, Database, clusters, switch over and fail over.

Chapter 2 covers related work that formed the research for this thesis.

Chapter 3 Literature Review

Chapter 4 methodology

Chapter 5 Presentation and Analysis of Results

# Existing work

How SAP works – 3 tier – GUI – App Server – Database

Clustered Resources (CRG): Mirrored-File-System, Journal DB, Application State

TCP/IP Virtual Host naming

Detail the processing involved in a switchover

1. Files are simply duplicated on both sides
2. Two different types of Database Replication
   1. Logical
      1. Duplicates DB events on both sides (by duplicating Journal Entries=
   2. Physical
      1. Disk sector copies
      2. Switchover consistency by rollback recovery.
3. ASCS – How the Replicated ENQ server works.

How Bernd Franz test tool works

Client – Server

Java based “users” imitate work in the SAP system (ABAP): generic:

Pre-defined test scripts (ABAP) for

* ENQ testing (No lost Locks – Application State is retained over switchover.
* DB testing (No inconstant Database – only complete transaction loss)

? Describe how site-wide OQ tests work: Screenshots: Management reports?

Maybe describe the SAP code development process – code changes, patches, releases, and software logistics.

# Literature Review

# Methodology

## Output Quality Driven w/ Java glue code Sub item

## Comprehensive Java based Test Framework

# Presentation and Analysis of Results

## Flow Charts of Test Scenarios

## GUI output of Tests

# Conclusions and Future Work

# Bibliography

**There are no sources in the current document.**

# Appendices

# 

# References

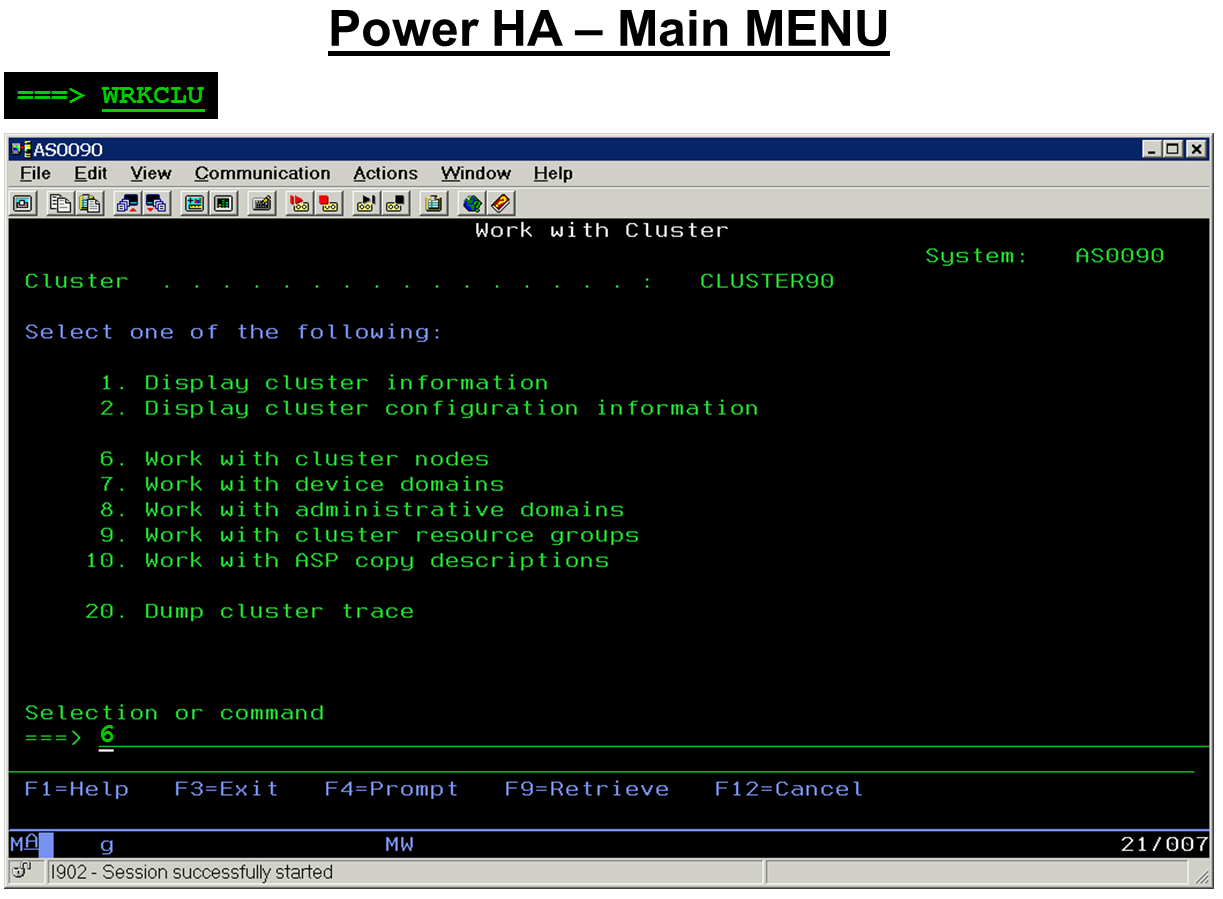


Figure 2: as/400 menu driven console

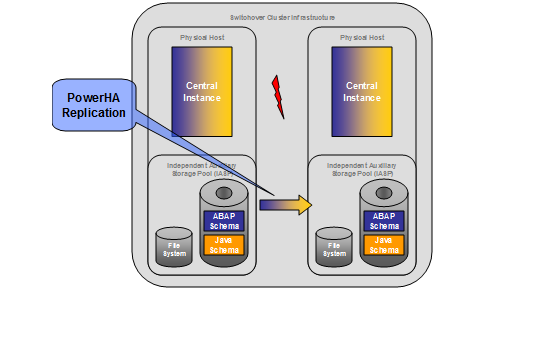


Figure 3: disaster recovery with HA

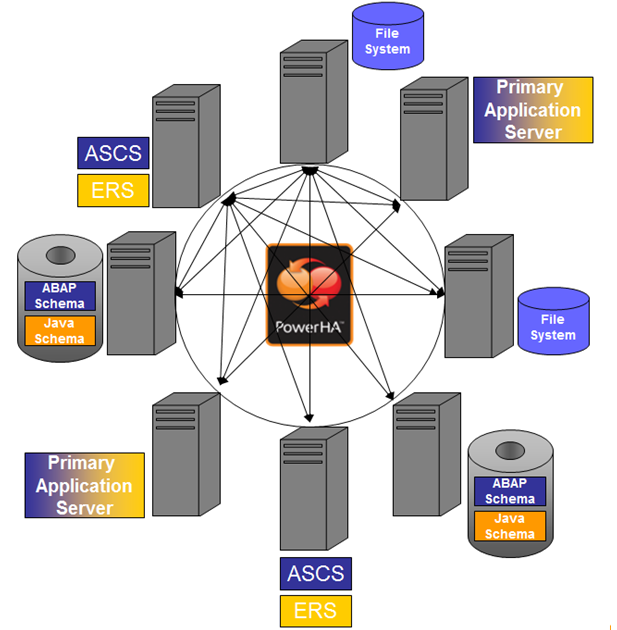


Figure 4: cluster nodes

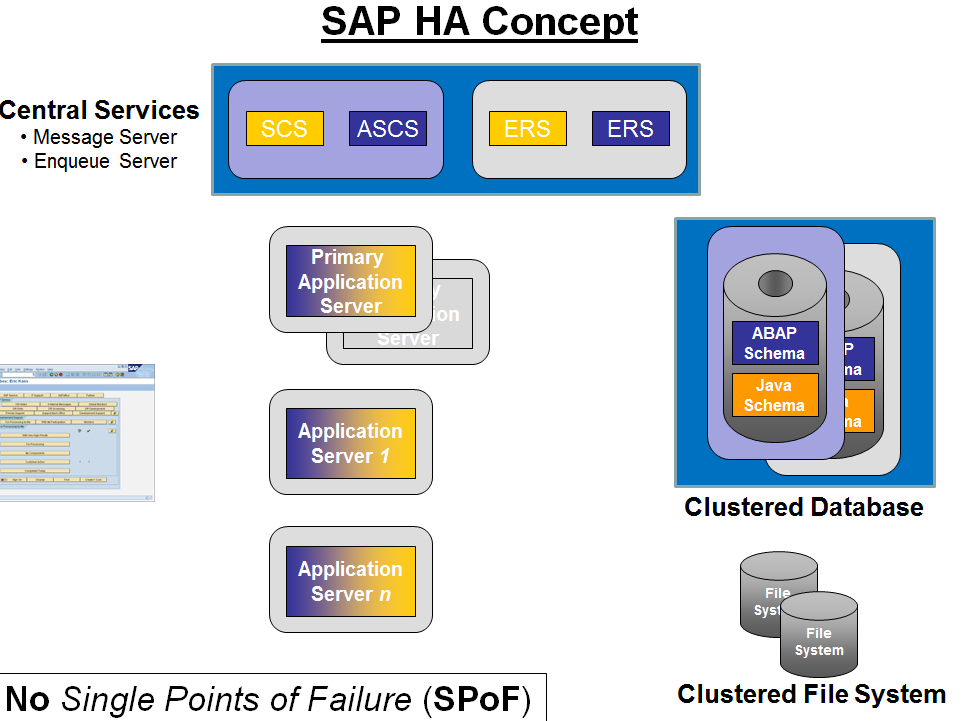


Figure 5: sap HA concept

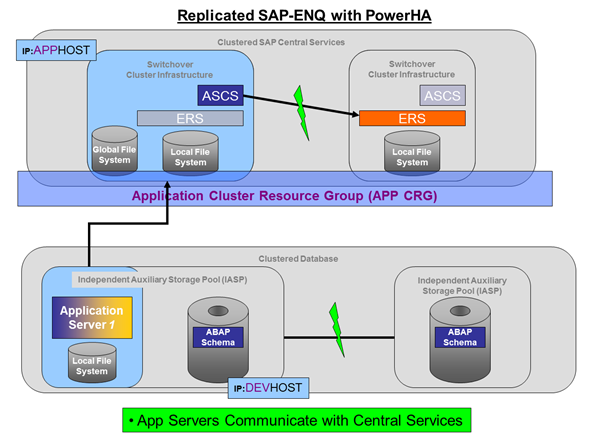


Figure 6: replicated SAP-ENQ with PowerHA 1

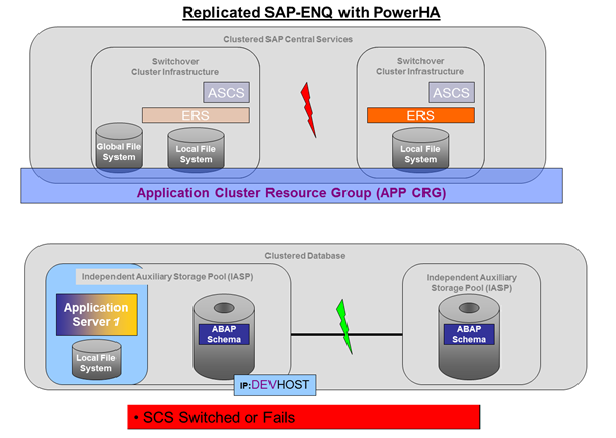


Figure 7: replicated SAP-ENQ with PowerHA 2

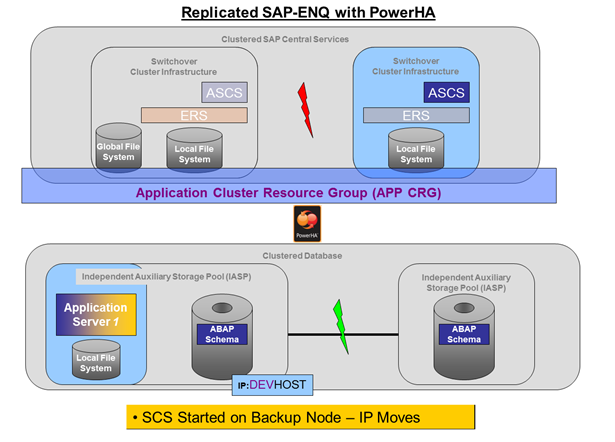


Figure 8: replicated SAP-ENQ with PowerHA 3

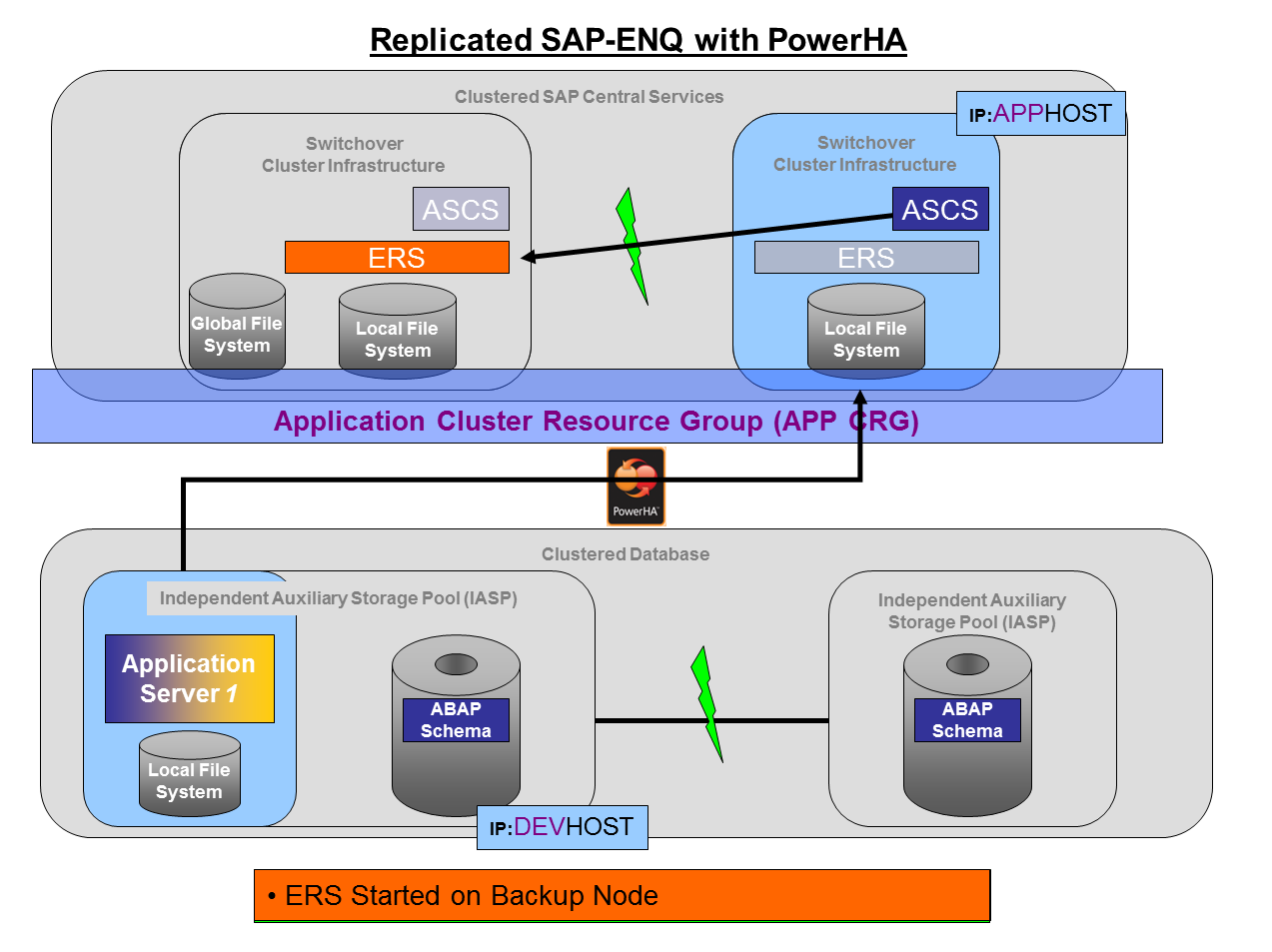


Figure 9: replicated SAP-ENQ with PowerHA 4

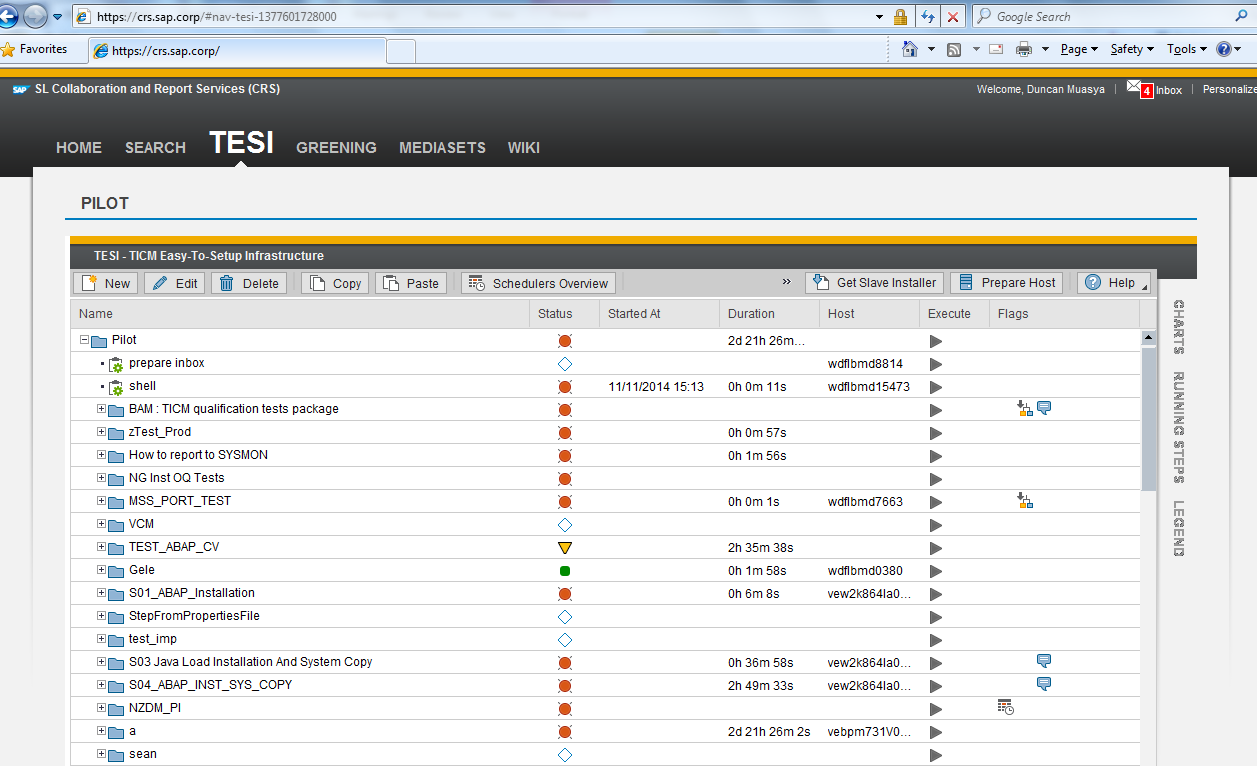


Figure 10: sap output quality tests

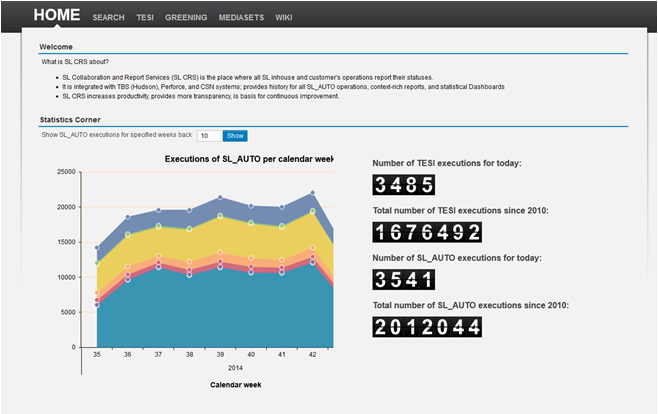


Figure 11: sap output quality test (statistics corner)

