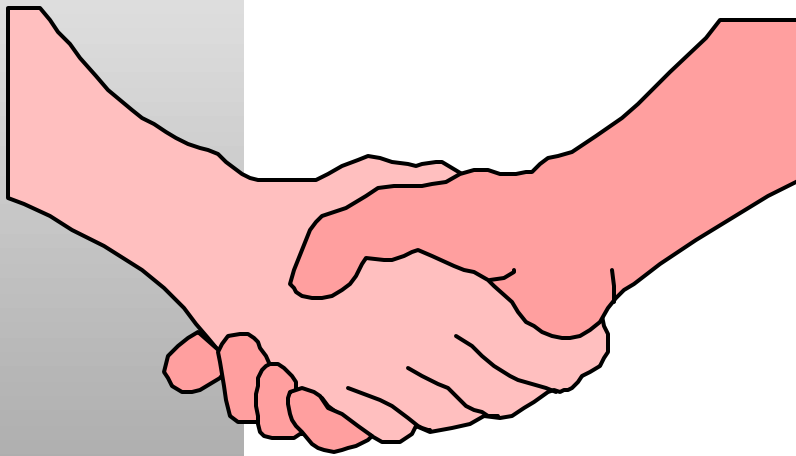
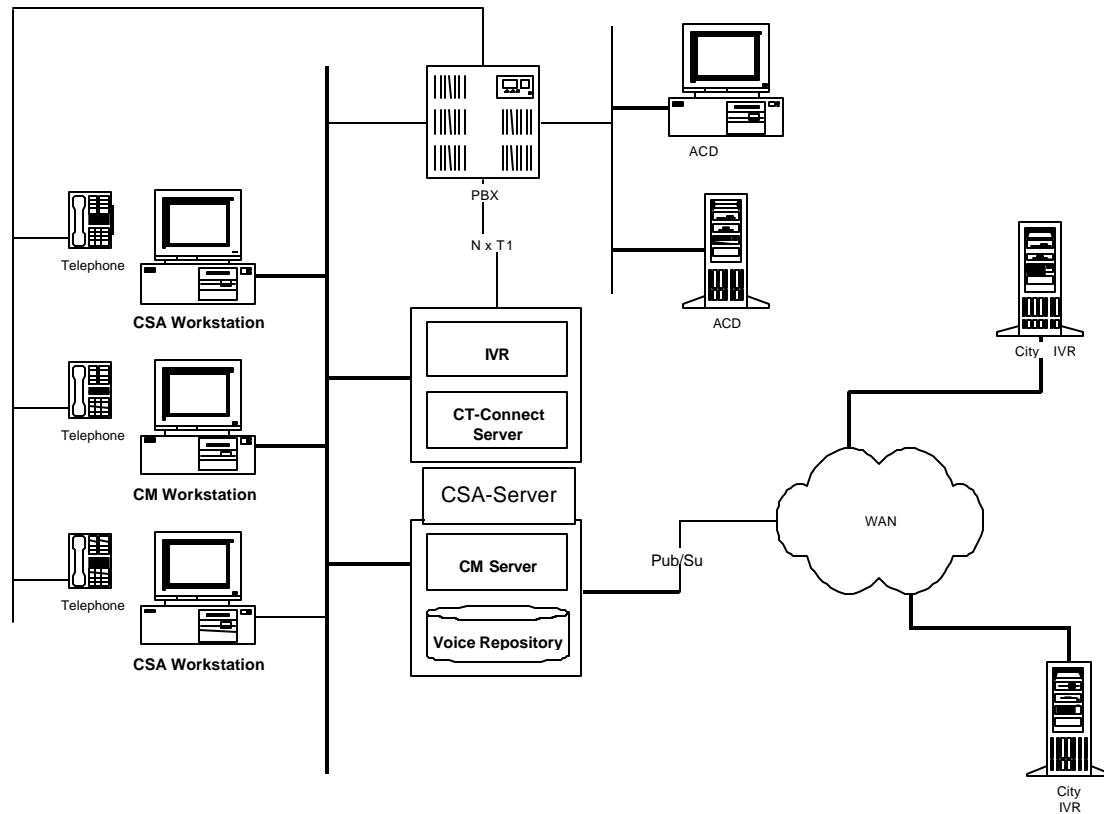


Integrated Testing of Telephony and Call Center applications



- **Torsten Baumann**
- **Rex Black**
- **Serban Teodorescu**
- **Gordon Page**

SUT Architecture



Discussion Topics

- **Client Server GUI (CSA)**
- **CTI Client Server (CM)**
- **IVR applications using multi-tools**
- **Product Wide Integration**
- **Management consideration**

Product Wide Integration Testing Using Multi-tools

- Create a Test Software and Hardware Infrastructure That Allows Automated Testing of a Heterogeneous Enterprise Wide System

Problems to overcome:

- **The subsystems involved in the project run each on different platform using different operating systems. Each subsystem has a different kind of interface based on a different set of paradigms, and different type of test tools for test purposes.**
- **Testing must be non-intrusive, therefore the target applications are running in production mode (no debug and/or test configurations) as much as possible.**
- **The subsystems under test interact widely and continuously; therefore we must be able to test the system as a whole.**

Solutions

- **We must find a way to make the test drivers for each subsystem talk to each other outside the (sub)system(s) under test.**
- **The test feedback loop must be able to follow all the data transformations as the information flows across the system under test.**

Implementation I

- **For each test thread, we establish a pool of data containing information about the state of each test driver involved as well as the associated test data. Each test driver is aware of the states of all the other test drivers involved, and is responsible to make correct test decisions based on this information.**
- **The integrated test applications use a state driven architecture:**
 - The first test driver launches the tests and prepares the state information pool. It then drives its target subsystem through the programmed test flow, and posts the associated state transition info and test data to the state information pool. It pauses and waits for the next test driver to execute.

Implementation II

- The second test driver was monitoring the state information pool. It senses the state transitions, drives its target subsystem through the programmed test flow, and posts the associated associated state transition info and test data to the state information pool. It then waits for the next test driver to execute.
- For the purposes of this presentation, we assume there are only three subsystems under test involved. The first test driver senses the state transition, wakes up and drives its subsystem into a “verification of the results” operation. It posts the info and exits.
- The second test driver senses the state transition, wakes up, and executes a similar operation for its target subsystem and exits.

Other Considerations

- **Each integrated test thread is independent. Any number of them, subject to resource availability, can execute simultaneously, enabling us to execute realistic test scenarios for the system under test.**
- **The implementation is very flexible. We use an .ini file structure with a general section, and a particular section for each subsystem (and its test driver) under test. Any type of test tool that can access text files over a TCP/IP network can be part of this test harness.**
- **The implementation is highly scalable. We can add any number of subsystems to our integrated test infrastructure by extending the state info pool with a corresponding section.**

Test System Fake Pipe (TSFP)

[Data]

Gender=

PostalZip=

BirthDate=

MemberUAN=

UAN=

PPC=

PreTestIVRBal=

PreTestCSABal=

PostTestIVRBal=

PostTestCSABal=

CCType=

CCNum=

CCExpMonth=

CCExpYear=

Test System Fake Pipe

[Registration]
RegistrationMeth=
Product=
Region=
UANGeneration=
IVRSuccess=
CSASuccess=
CSAStartTime=
IVRStartTime=
CMStartTime=
CSAEndTime=
IVREndTime=
CMEndTime=
CMWarn=
CSAWarn=
IVRWarn=
IVRError=
CSAError=
CMErrror=

Post Test Run Results File

[] Fake Pipe	TestCase	Status	Error	Warning
[] 4900006675_11313_TSFP.ini	Registration	PASS	(CSA)	NA
[]		PASS	(IVR)	NA
[]				
[]	Login	PASS	(IVR)	No Error
[]				
--				
[]	UpdateMember	FAIL	(CSA)	CSA Error
[]		FAIL	(IVR)	IVR Error
[]				
--				
[]	InitialPurchase	PASS	(IVR)	Manual Authorize

Logistics

- **Distributed test teams**
 - **Two locations about 15 kilometers apart**
 - Downtown Toronto
 - Airport area
- **Manual and automated testing approaches**
- **Complex test environment simulating an even more complex field environment**
 - **Hardware (dozens of servers, about 100 clients)**
 - **Software**
 - Unix (2 variants), Windows, call center, and telephony
 - Custom, COTS, and customized COTS
 - **Network (100BT E'net, PRI ISDN, WAN, PSTN)**

Test Team Size and Composition

- **IMG employees and RBCS consultants**
- **Two Test Managers**
 - One consultant
 - One employee
- **Five Test Engineers**
 - Three contractors for IVR, CM, and Integration
 - Two employees for IVR and Integration
- **Eight Test Technicians**
 - All contract IMG employees
 - About two each for IVR, CM, CSA, and Integration
- **About 12 temps for manual testing**
- **About 30 CSRs, other IMG employees ran manual tests**

Bug and Test Tracking

- **Integrated test management system from RBCS**
- **Quantitative management of risks to quality through FMEA failure and test coverage analysis**
- **Test tracking**
 - Customized Excel spreadsheets
 - Automatic test case and suite summarization
 - Test case details included in subordinate worksheets
- **Bug tracking**
 - Customized GUI, VBA-driven, Access-based database
 - Distributed to each tester's desktop w/ shared tables
 - State-based, with ownership for management to closure