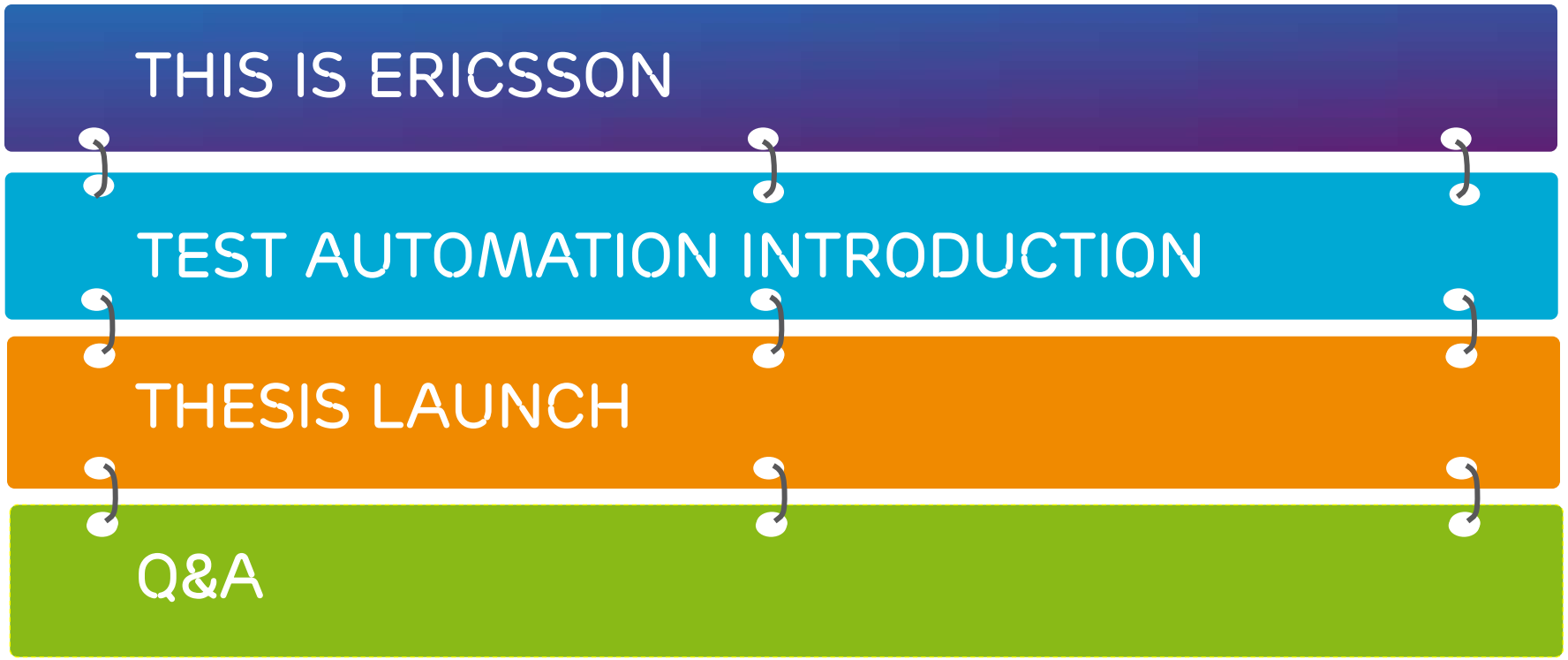
ERICSSON

TEST AUTOMATION IN E///

PDU OPTICAL&METRO

GENOVA, 9/12/2013

AGENDA


THIS IS ERICSSON
TEST AUTOMATION INTRODUCTION
THESIS LAUNCH
Q&A

THIS IS ERICSSON

- Ericsson is a world-leading provider of telecommunications equipment and services to mobile and fixed network operators
- We provide:
  - Communication networks
  - Services to network operators
  - Enablers to service providers
- We have customers in more than 180 countries
- More than 40% of the world's mobile traffic pass through our networks
- The networks we support for operators serve more than 2.5 billion subscribers
- We have 33,000 granted patents, comprising one of the industry's strongest portfolios.
- We are the fifth largest software supplier in the world
- More than 110000 employees
- We have been in the telecoms market for 136 years

SPO - SMART PACKET OPTICAL
E2E METRO NETWORKING

IPT NMS

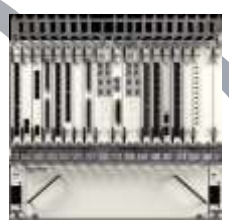
3G to LTE
SME to Data Center

TDM to Packet

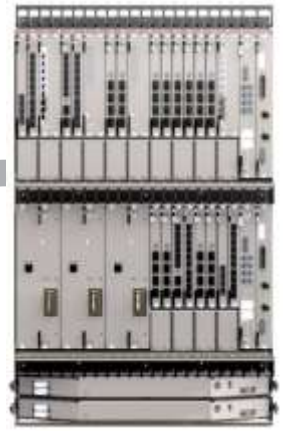
10G to 100G+

SPO 1100
40G packet
20G TDM

SPO 1410
80G packet
20G TDM



SPO 1460
800G OTN
320G packet
80G TDM



SPO 1485
Multiservice DWDM Platform

ROADMs + Amplifiers

HS ‘TDM-like’ Ethernet

Ultrahigh Capacity & Flexibility

ACCESS | AGGREGATION | IP EDGE | CORE

SW DEVELOPMENT: CUSTOMER REQUIREMENTS

- Tough competition – High expectations:
  - Functionality
  - Reliability
  - User Experience
  - Fast time to market

SW DEVELOPMENT: AGILE METHODOLOGIES

- Individuals and interactions
- Working software
- Customer collaboration
- Responding to change

TEST CHALLENGES:

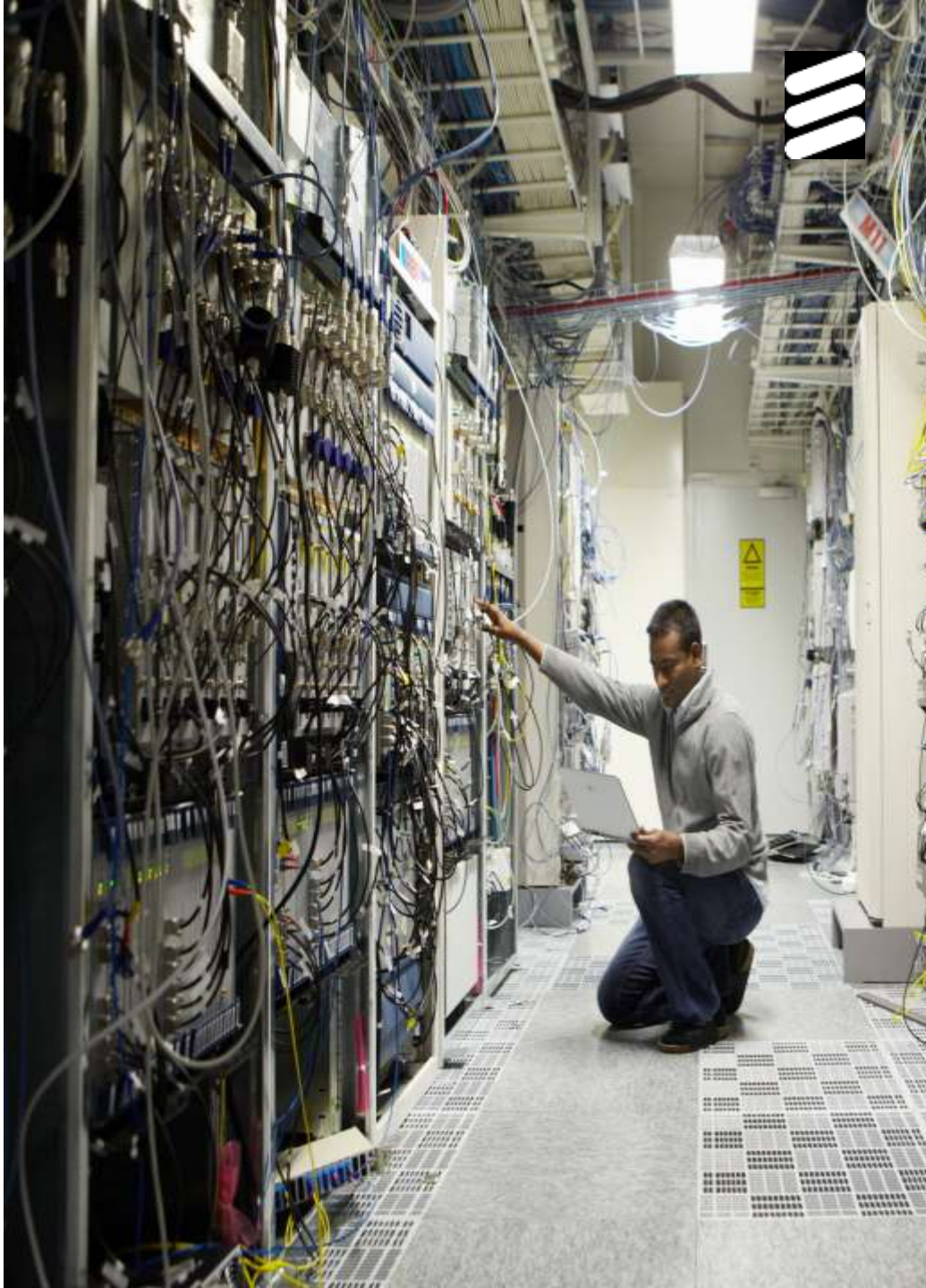
- Functionality:
  - High complexity system
- Reliability:
  - Wide tests list
- User Experience:
  - Complex interface test
- Fast time to market:
  - Reduced test period

TEST AUTOMATION

Test automation is a key concept in our organization and it is the enabler for Continuous Integration:

- Reduce test period (time to market)
- Balance the investment of hardware resources and test equipment
- Less repeating manual work
- SW upgraded and tests started also non office-hours
- Visibility and storage of results
- Ensure quality and find problems as early as possible

Once automated, regression tests can be efficient and effective.



TEST AUTOMATION FRAMEWORK

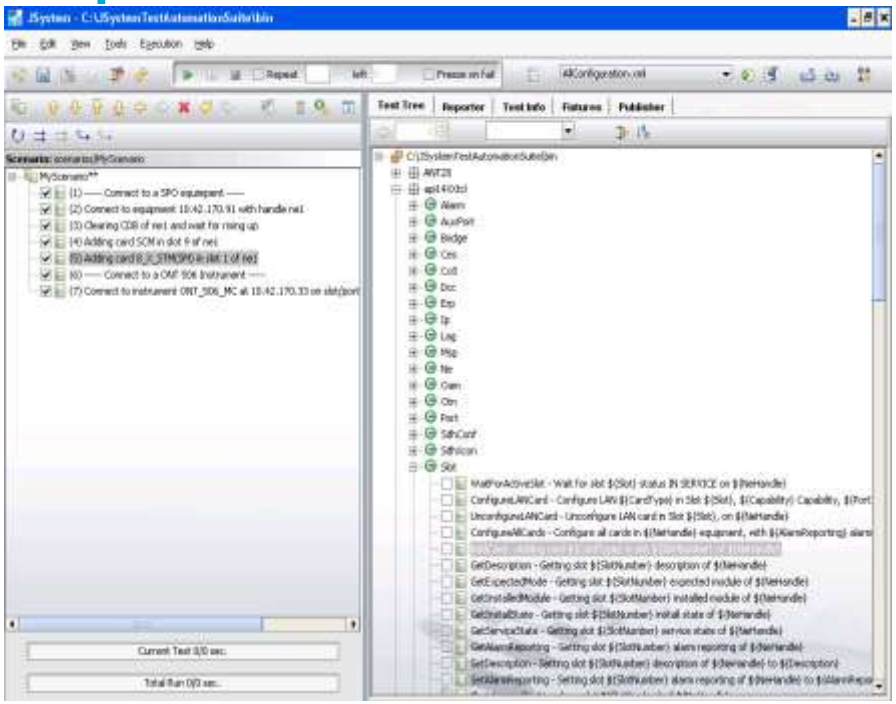
ERICSSON

› **JSystem** is a professional open-source framework for writing and managing automated system tests

› **Run your tests in a distributed setup**

Use the JSystem agent to load balance your tests between hosts or to run them concurrently on several hosts.

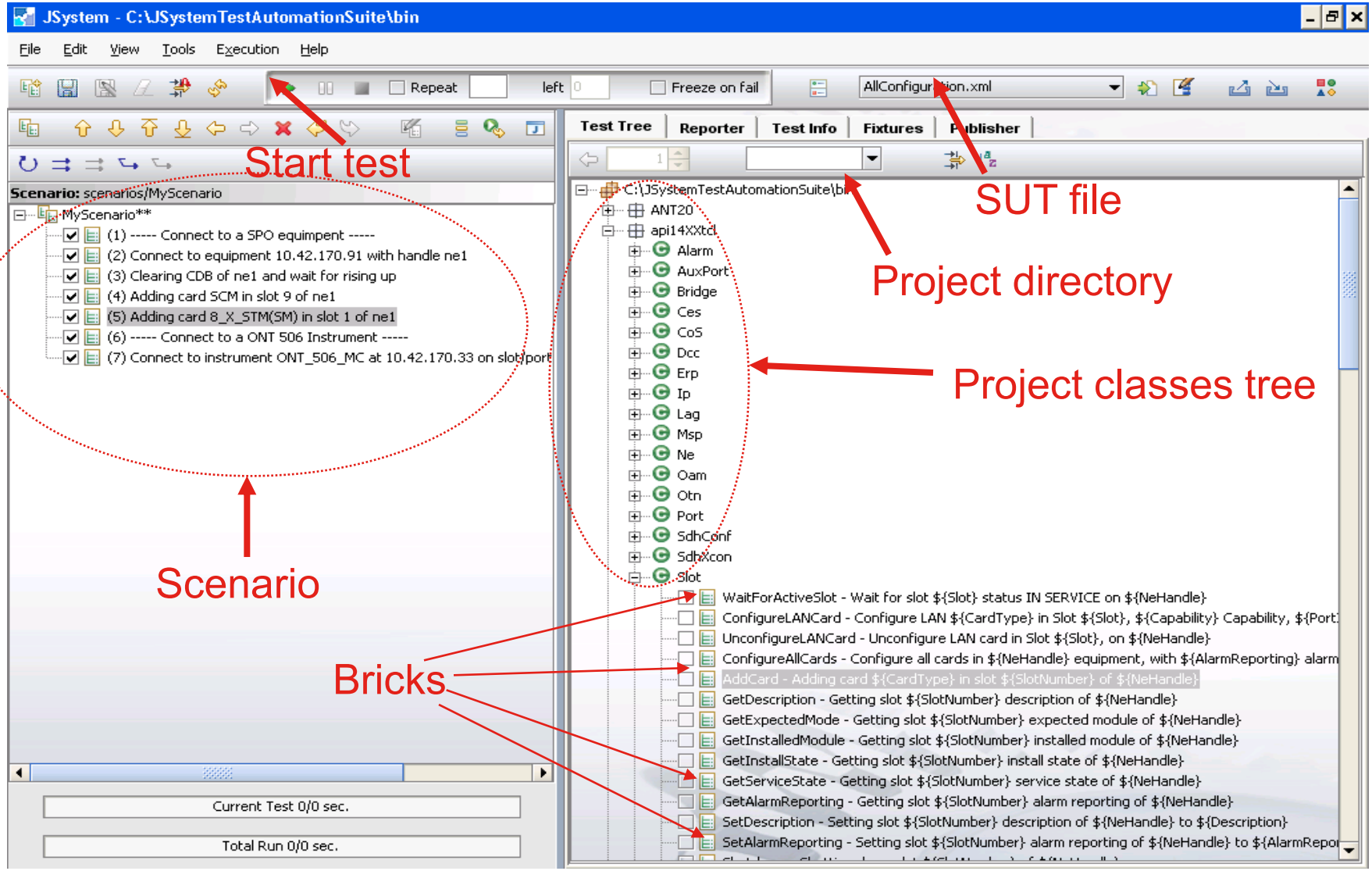
The JSystem distributed execution infrastructure enables customization and manipulation of distributed execution.



BASIC CONCEPTS – PART 1

- **TCL (Tool Command Language)**, is a versatile and powerful open-source programming language and development platform. It is renowned for its stability and utility, and its emphasis on providing a cross-platform programming API makes it an ideal choice for an enormous variety of programming jobs.
- **IM (Information Model**), is a representation of constraints, rules, and operations to specify data semantics for SPO SW. It can provide sharable, stable, and organized structure of information requirements. It is wriiten with Rational Rose IBM (transitioning to Architect)
- **XRPC (Xml-Rpc**), is a remote procedure call (RPC) protocol which uses XML to encode its calls and HTTP as a transport mechanism.
- **CDB (Configuration DataBase)** generated on node stores representation of physical entities and their configuration (multifile xml tree)

BASIC CONCEPTS – PART 2



THESIS LAUNCH

- What are we looking for?
  - Motivated and passionate students
  - Innovative ideas
  - People with a background in software, computer science and/or networking

1) DYNAMIC/ADAPTIVE TEST CASES.

- L'obiettivo finale di questo lavoro di tesi è rendere l'esecuzione dei test automatici flessibile rispetto una specifica configurazione di apparato.
- Questo si potrebbe realizzare ricavando le informazioni relative direttamente dal sistema sotto osservazione o presenti nel database dell'apparato.
  In questo modo si vuole velocizzare e semplificare l'esecuzione di test esistenti su nuovi/diversi banchi di test.

1) DYNAMIC/ADAPTIVE TEST CASES.

› FORMALIZZAZIONE DEL TEST BENCH.

- Studio dell'IM, Cdb, SUT e API per capire la relazione fra i file e come le informazioni sono trasformate da un formato all'altro.
- Formalizzazione di quanto rappresentato nel SUT, modellando in più anche i possibili strumenti e le relazioni/connessioni con i Nodi.

› SUT ADATTIVO

- Sviluppo di un tool in grado ricreare automaticamente il SUT a partire da un dato test bench (sfruttando i tools già a disposizione che automaticamente sono in grado di creare il cdb di un Nodo e di capire come/a quali strumenti è legato).

› FLEX AUTOTEST

- Sviluppo del tool in grado di "spostare" effettivamente un test automatico previsto per un test bench A su un altro test bench B.
- Inserire nel tool la verifica di coerenza dei banchi, analizzando i SUT file dei due test bench.
- Ricercare, dato un insieme di SUT già esistenti, quelli che potrebbero essere adatti alla sostituzione.

2) ADVANCED AUTOMATIC FIRST ANALISYS.

- L'obiettivo è avere un tool di supporto che tramite intelligenza artificiale identifichi le root-cause dei fails e/o relazioni complesse fra test (e fra fails).
- Il tool dovrebbe essere in grado quindi di modificare, a run time, l'esecuzione di un insieme di test, oppure di segnalare nel log riassuntivo questo tipo di problematiche.
- Questo tool dovrebbe essere in grado, a fronte di un fail di un test, di capire se esistono relazioni con i test precedenti che potrebbero averlo influenzato.
- Cercare di dedurre un problema ampio su un Nodo ed evitare di eseguire test successivi che quasi sicuramente darebbero risultato negativo.

2) ADVANCED AUTOMATIC FIRST ANALISYS.

› ANALISI E FORMALIZZAZIONE REGOLE.

- Alcune azioni intraprese dall'utente umano a seguito di un fail su un test possono essere formalizzate e automatizzate.
- Interagire con gli specialisti per capire/definire queste regole e formalizzarle (ad esempio in un linguaggio logico)
- Le relazioni fra test, dovute ad esempio a parti comuni nel SUT, sono nuovamente da evincere e formalizzare, così come l'azione da adottare per verificare l'errore.

› DEBUG/ROOT-CAUSE ANALYSIS .

- Analizzando il log di sistema si potrebbe ricercare la root-cause di un problema ampliando la ricerca nei log del Nodo, delle sue componenti e degli strumenti associati, così da migliorare la reportistica e specializzando le informazioni di debug a corredo.

› AUTOTEST ADATTIVO: DEFINIZIONE FORMALISMO.

- Esplicitare in maniera formale le entità su cui opera il test per inferire relazioni fra test che ad oggi sono difficilmente deducibili automaticamente.
- Questa formalizzazione dei test case potrebbe apportare altri benefici generali, ad esempio permettendo di salvare nel database delle esecuzioni informazioni più precise sui vari test case, per facilitare una successiva analisi dei dati.

3) AUTOMATIC TEST GENERATION.

- L'obiettivo finale è quello di rendere rapido lo sviluppo di test automatici delle nuove features, ad esempio generando automaticamente i test e i bricks utilizzando come input il modello informativo.
- Se si fosse già sviluppato il modello formale anche del test bench (tesi 1), si potrebbe usarlo come punto di partenza per lo sviluppo dei test automatici, creando quindi test automatici non solo a livello di Nodo ma anche a livello di configurazione.

3) AUTOMATIC TEST GENERATION.

› GENERAZIONE AUTOMATICA TEST CASE.

- Sfruttando le modelizzazioni del test bench e i documenti con i requisiti richiesti, sviluppare un tool di supporto per la stesura dei test automatici, generando «scheletri» di test auto da implementare.
- Verificare se è necessaria la modelizzazione dei requisiti di funzionalità e studio migliore formalismo.

› GENERAZIONE DEI TEST AUTOMATICI.

- Realizzare un tool che permetta di creare il test in modalita' automatica, selezionando le entità coinvolte dai modelli a disposizione.

AUTOMATIC TEST GENERATION.

› NEGATIVE TEST.

- Il test “negativo” (“Negative Test”) viene eseguito per determinare e verificare la risposta del sistema quando l’input non è quello atteso. Si possono valutare sia le risposte di errore già previste dal sistema sia cercare di generare input non previsti verificando così il comportamento dell’apparato in casi particolari.
- Creazione automatica di test negativi

› TOOL PIANIFICAZIONE TEST BENCHES E MATERIALE.

- Realizzare un tool che permetta, data la formalizzazione del test case e del SUT, di specificare/dedurre di che cosa fisicamente c’è bisogno (Nodi, schede, apparati).
- Sarebbe quindi anche possibile dedurre (almeno parzialmente) il test bench minimo necessario (e quindi il relativo SUT).

› VALUTAZIONE DEI TOOL DOCUMENTAZIONE AUTOMATICA PER JAVA .

- Valutare l’efficacia dei vari tool di generazione automatica di documentazione di test, considerando sia la documentazione di supporto per che si occupa dello sviluppo, sia la documentazione di supporto per chi si occupa del debug e della stesura di test suites.

ADVANCED TEST STRATEGY ANALISYS.

- Lo scopo è sfruttare l'intelligenza artificiale per identificare le aree su cui concentrare le attività di regressione considerando lo storico dei test effettuati e gli esiti, inseriti nel database delle esecuzioni di test.

- Si vorrebbero applicare algoritmi di analisi dei dati (ad esempio data Mining o ragionamenti logici) così da:
  - verificare la copertura di tutte le features/aree.
  - individuare aree che presentano problematiche ricorrenti
  - individuare eventuali relazioni fra errori

bruno.demartini@ericsson.com

pietro.policicchio@ericsson.com

THANKS

ERICSSON