ACROSS
Autonomous Control for a Reliable Internet of Services

ICT COST Action IC1304

Autonomous Control for a Reliable Internet of Services (ACROSS)

Predicting and simulating complex software systems behaviour

ACROSS MC meeting, October 23, 2014,Larnaca, Cyprus

Tihana Galinac Grbac
Software Engineering and Information Processing Laboratory
Faculty of Engineering

cost
EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY

UNIVERSITY OF RIJEKA
Faculty of Engineering

Department of Computing

- Large scale and complex systems, analysis and modelling
- Distributed computing, concurrent systems
- Intelligent Computing Systems and Autonomy Oriented Computing
- Artifical inteligence and big data
- Communication systems, middleware platforms and sensor networks
- Software Engineering and programming languages
- Human-Computer interaction







Modern networks and software

- Software become central part of the modern network
- It should run on any hardware, serve to many users, satisfy their complex communication needs and deliver proper ICT service, effectively and efficiently
- Modern software has to be flexible on network context, information context, communication context, ...
- Modern network should provide **reliable and robust** ICT services (resistant against system failures, cyber-attacks, high-load and overload situations, flash crowds, etc.)

Software in ‘Internet of Service’

- In service oriented architecture software is provided ‘**as a service’**
- In that concept ‘of service’ is referring to a technical understanding of software functions provided as Web service
- IoS combine that services and integrate functionalities that led to **complex service chains**
- Usually these service chains are developed by number of providers and offered to number of users
- Service chain composition is happening at layers above network layer
- Problem is how to secure quality of these service chains
- We need algorithms for autonomous control for a reliable IoS

Key problems with software evolution

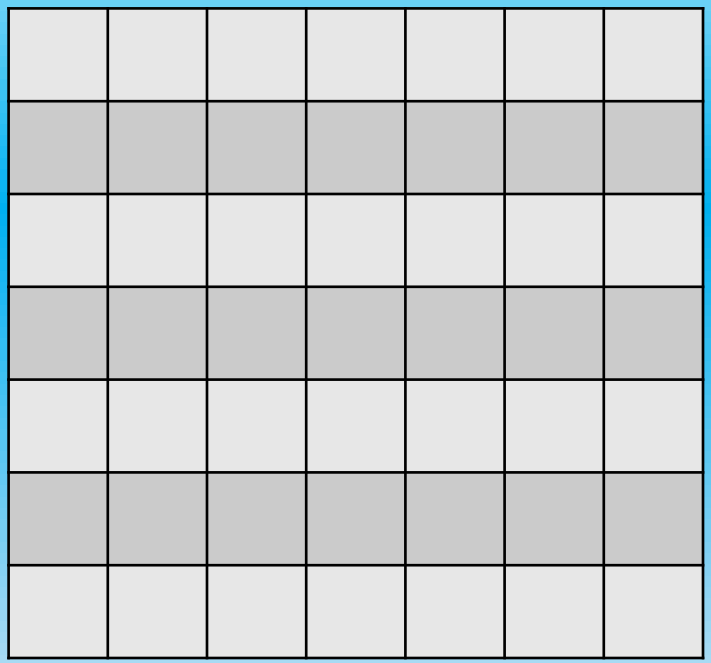
- More and more software systems tend to evolve towards complex software systems (e.g. IoS)
- Interconnection of peripheral systems over distributed network into system of systems (IoT)
- Key problems become:
  - Can we develop foundations on software behavior?
  - How can we measure software behaviour in network?
  - Can we predict and simulate software behaviour in network?
  - How to manage complex software system?
  - Are we able just by observing properties of system parts to predict and model its overall behaviour?

Focus of our research

Complex systems

- Number of levels of abstraction
- Global properties of system and local properties describing component behaviour
- Imposible to derive simple rules from local properties towards global properties

System and system components



Source: Complex software systems : Formalization and Applications -
Work done in EU project GENNETTEC: GENetic NeTworks: Emergence and Complexity

How to secure quality of complex software systems?

- Software Quality Assurance
  - a planned and systematic pattern of all actions necessary to provide adequate confidence that the software item conforms to its established requirements [IEEEStdGlos]
- Software testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item
- Number of possible test cases is infinite that is esspecially the case with complex systems
- One result of testing process is software failure

Software Fault and failure

```
public double getCurrentRate(String fromCurrency, Strin
    if (fromCurrency == null || toCurrency == null) {
        return Double.NaN;
    }
    double answer = 0;
    if (fromCurrency == null) {
        return answer;
    }
    Condition 'fromCurrency == null' is always 'false' more... (Ctrl+F1)
    if (toCurrency.equals("USD") && fromCurrency.equals
        answer = USD_CDN;
    }
    return answer;
}
```

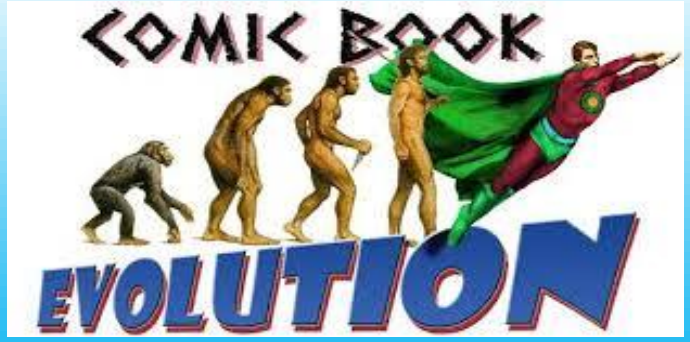
Fault execution leads to system failure

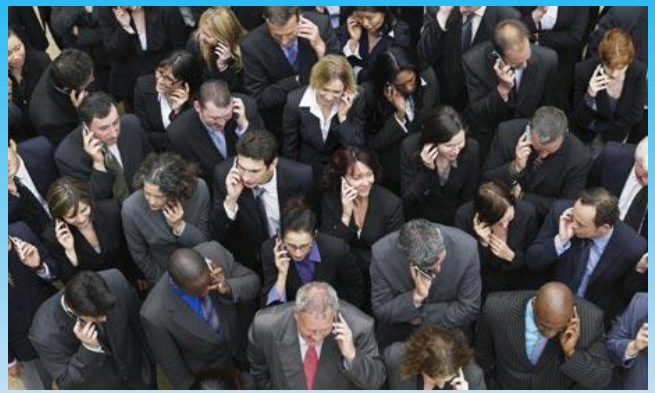
Source: http://vapresspass.com/2013/04/24/failure-is-not-fatal-by-marcia-zidle/

- One system failure may be result of several software faults
- One software fault may cause several system failures

Fault costs and complex systems

- Evolving system demands reusability
- Usually impacts thousands of end users
- Number of system versions may coexist at the same time
- Consequences of faults are impossible to predict
- Problem is not only effect of one fault, but effect of its repairment on the system as a whole





Fault distributions

Over software structure

Empirical studies on fault distributions

- *N. E. Fenton and N. Ohlsson, "Quantitative Analysis of Faults and Failures in a Complex Software System," IEEE Trans. Softw. Eng., vol. 26, no. 8 , pp. 797-814, Aug. 2000.*
- *C. Andersson and P. Runeson, "A Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems," IEEE Trans. Softw. Eng., vol. 33, no. 5, pp. 273-286, May 2007.*
- *T. Galinac Grbac, P. Runeson and D. Huljenic, "A Second Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems," IEEE Trans. Softw. Eng., vol. 39, no. 4, pp. 462-476, Apr. 2013.*

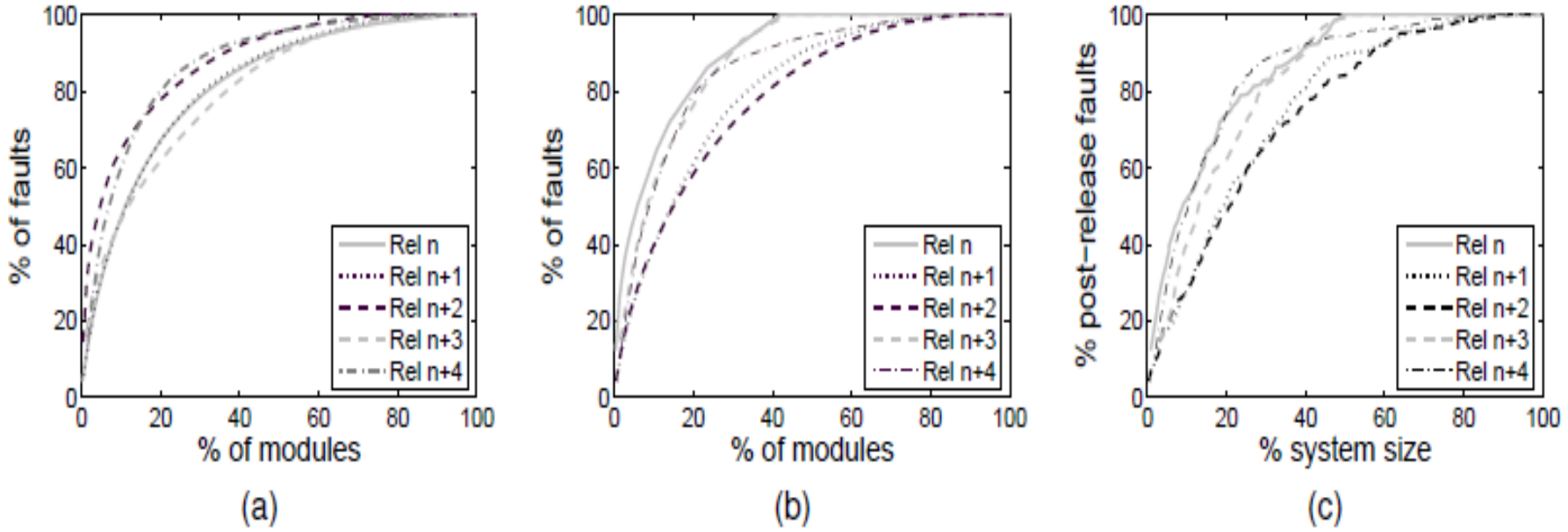
Pareto principle: 80 – 20 rule

Vilfredo Federico Damaso Pareto

- *1906: 80% of the land in Italy was owned by 20% of the population*
- *Income and wealth among the population follows a Pareto distribution, a power law probability distribution*

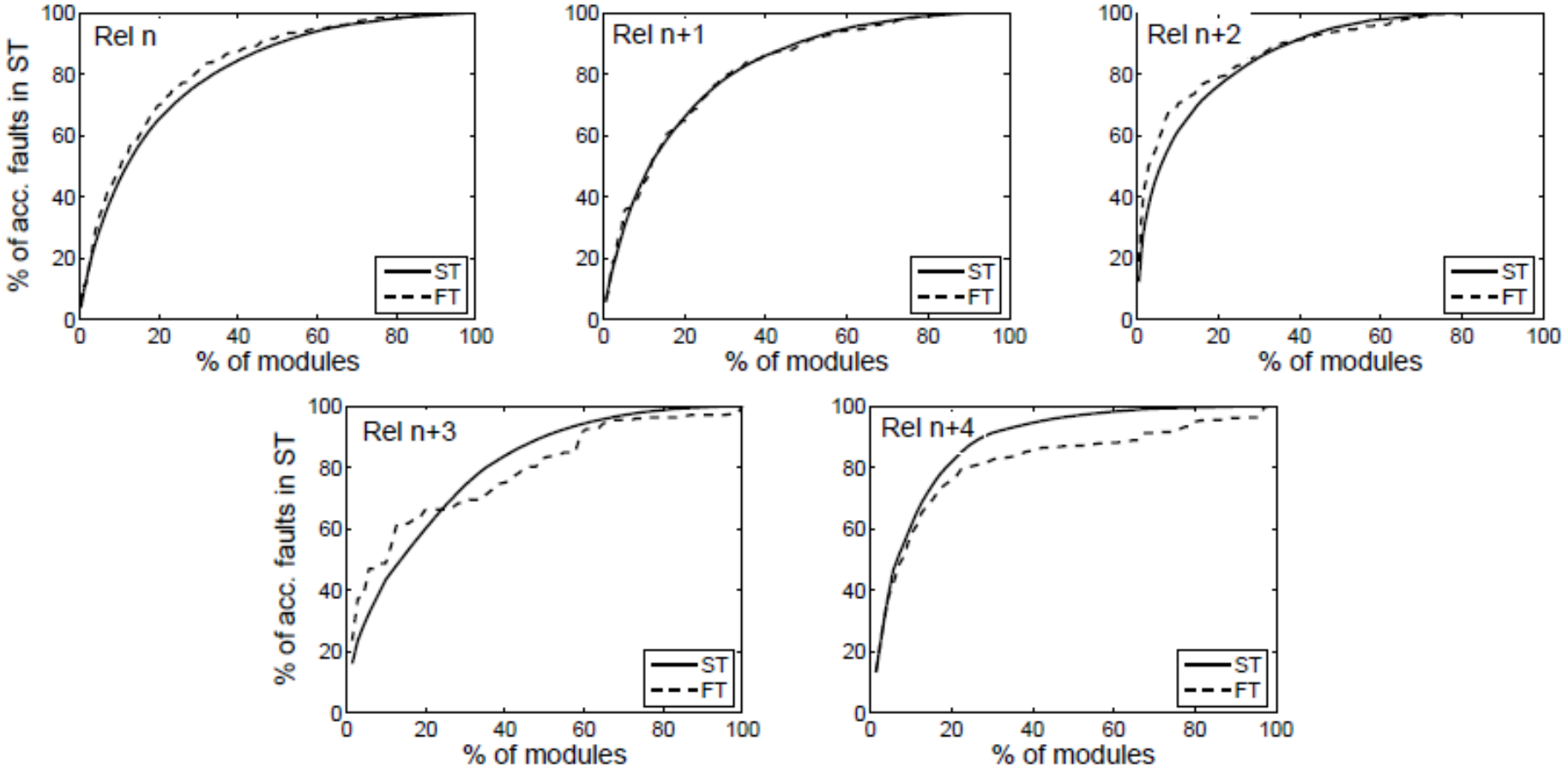
Source: http://en.wikipedia.org/wiki/Vilfredo_Pareto

Alberg diagrams: Pareto principle of fault distributions



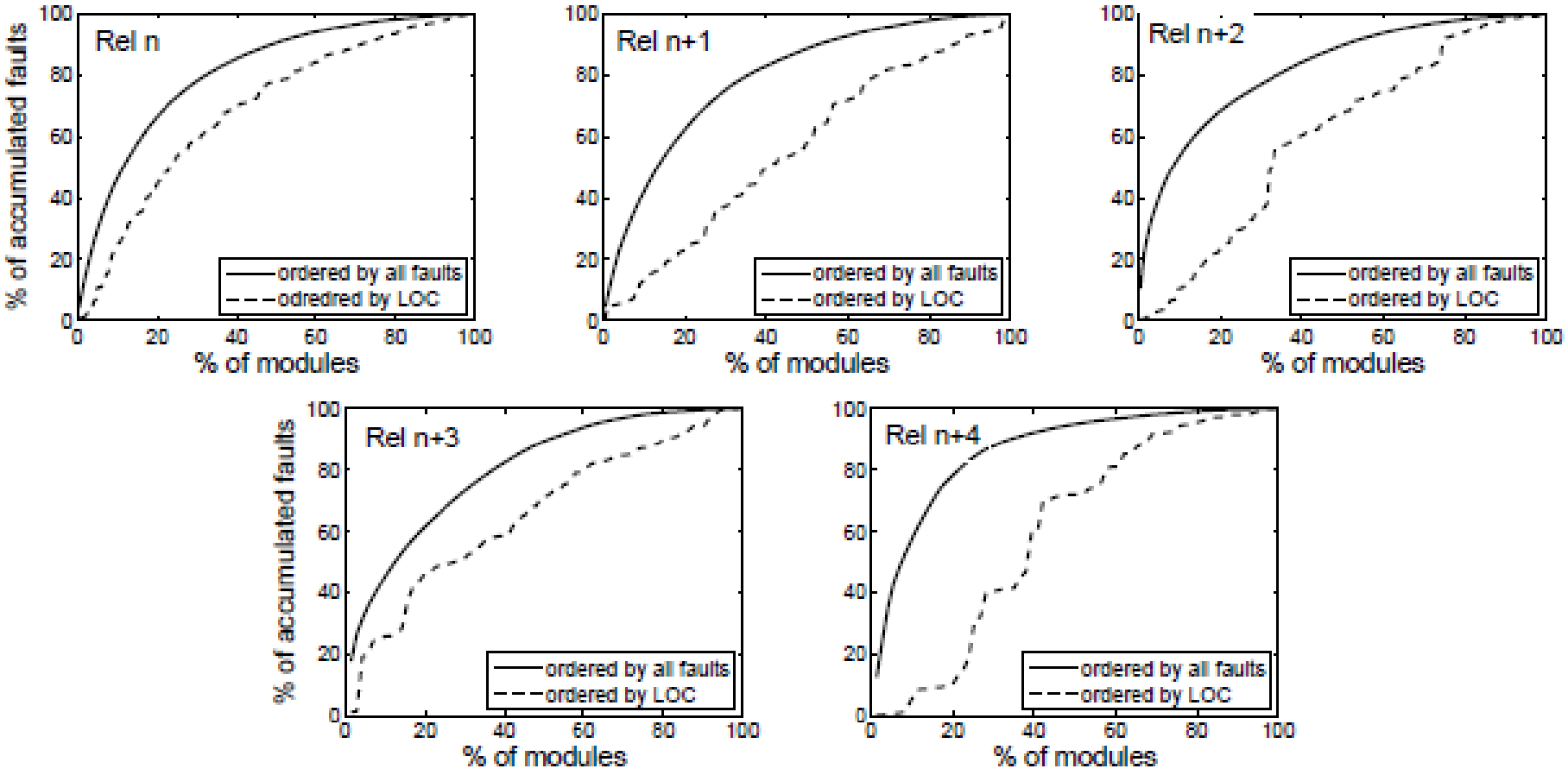
Alberg diagram showing the percentage of: (a) modules versus the percentage of pre-release faults, (b) modules versus the percentage of postrelease faults, and (c) system size versus the percentage of postrelease faults

Alberg diagrams: persistance of faults



Accumulated percentage of the number of faults in the system test when modules are ordered with respect to the number of faults in the system test and the function test.

Effects of module size and complexity on fault proneness



Accumulated percentage of number of faults when modules are ordered with respect to LOC

Analytical fault distributions

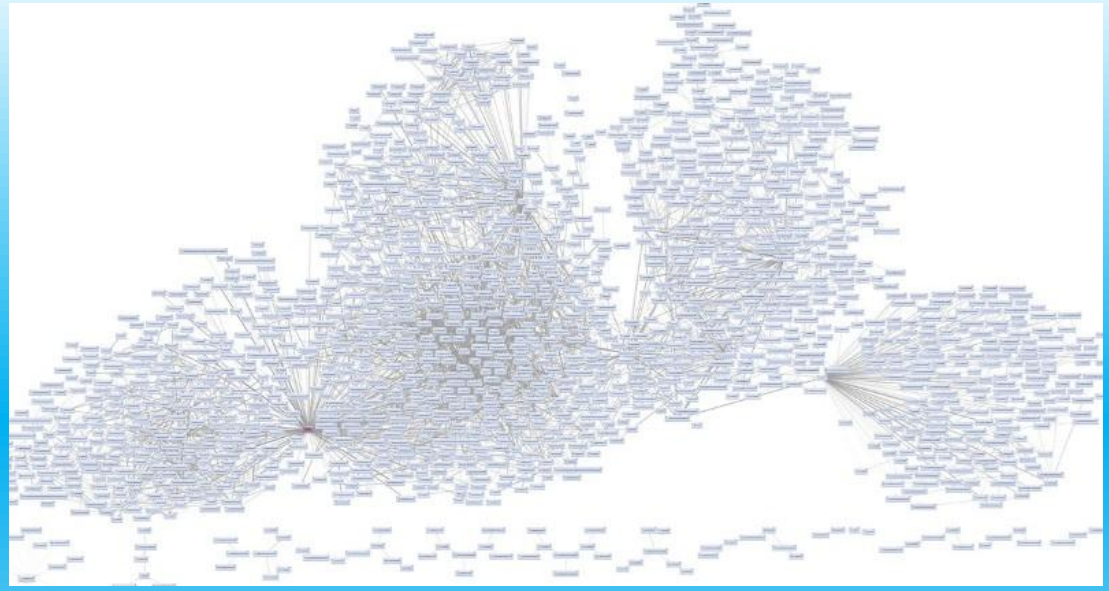
- All previous principles ultimately depend on the underlying probability distribution
- the fulfillment of a certain empirical principle does not determine the probability distribution uniquely
- The distibutions like double Pareto, Weibull, lognormal, Pareto, and Yule-Simon with power-law in the tail are confirmed

Les Hatton. Power-Law Distributions of Component Size in General Software Systems. IEEE Trans. Software Eng. 35(4): 566-572

Tihana Galinac Grbac, Darko Huljenić. On the Probability Distribution of Faults in Complex Software Systems, Information and Software Technology, published online first.

Software structure

- Software structure represented as dependency graph between structural components show promisable results in modeling fault behaviour
- Some subgraphs and motifs are dominant in faulty software structures



Petrić, Jean; Galinac Grbac, Tihana. ***Software structure evolution and relation to system defectiveness*** *// Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM New York, NY, USA : ACM, 2014.*

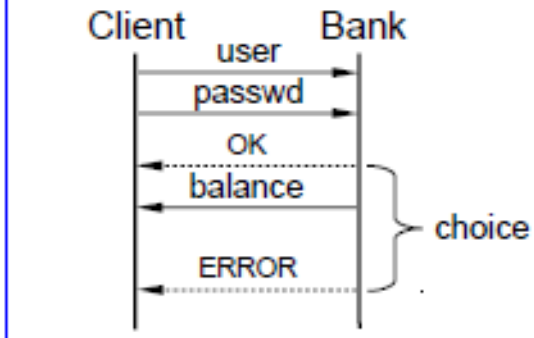
European Cooperation in Science and Technology

IC1201: Behavioural Types for Reliable Large-Scale Software Systems (BETTY)

- **Behavioural type theory** encompasses concepts such as interfaces, communication protocols, contracts, and choreography.
- As stuctural principle for building reliable software systems
- Idea:
  - to codify the structure of communication to support the development of reliable communication-oriented software.
  - to encode as types the communication structure of modern computer systems and statically verify behavioural properties about them

Example – Session types

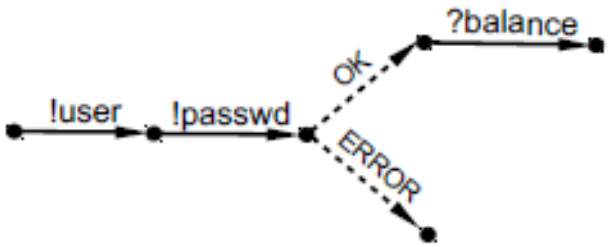
- Aim: to develop programming languages, tools for development of certified software solutions for global services



(a) sequence diagram

ClientType =
! user . ! passwd .
&{ OK: ?balance . end ,
ERROR: end }

(b) behavioural type



(c) transition system

- Developed language: e.g. Scribble

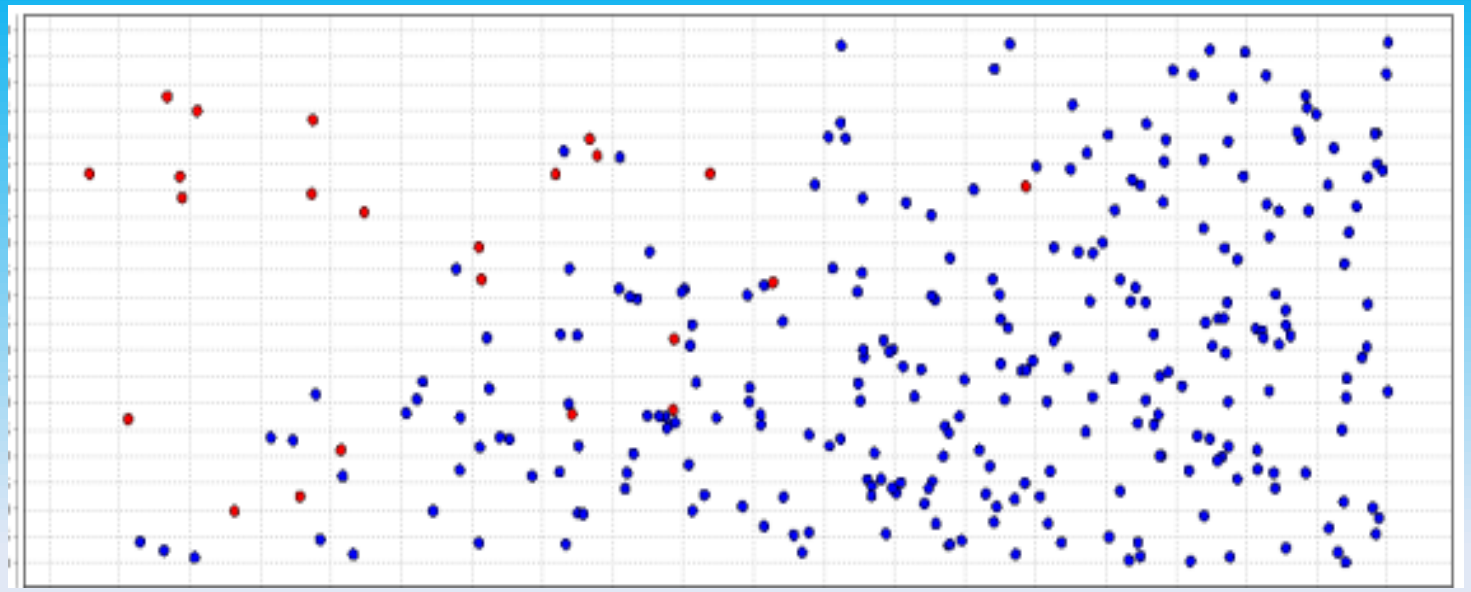
Fault Prediction using Classification modelling

1) Statistic classificators
   - Linear Discriminant Analysis
   - Quadratic Discriminant Analysis
   - **Logistic Regression**
   - Naive Bayes
   - Bayesian Networks
   - Least-Angle Regression
   - Relevance Vector Machine
3) Nearest neighbor methods
   - k-Nearest Neighbor
   - K-Star
5) Neural networks
   - Multi-Layer Perceptron
   - Radial Basis Function Network

2) Support vector machine
   - Support Vector Machine
   - Lagrangian SVM
   - Least Squares SVM
   - Linear Programming
   - Voted Perceptron
4) Decision trees
   - C 4.5 Decision Tree
   - Classification and Regression Tree
   - Alternating Decision Tree
6) Ensemble methods
   - Random Forest
   - Rotation Forest
   - Logistic Model Tree

Genetic approach

- Problems:
  - Unbalanced datasets
  - Soft computing approaches did not come to common solution
  - Data are very sensitive on linking bias



Relation to ACROSS

- Reliability and availability service chains will very much depend on their structure
- knowing the appropriate statistical fault distribution would enable more systematic approach for automated guidance for creation of reliable software chains
- Interesting is to model the underlying processes that generate distributions and how they influence the statistical fault distributions
- Context awarness based on system structure and measurements on software abstract levels

Future work

- Replications and knowledge systematization:
  - The experiments are performed on the data from software system of large scale telecommunication software and number of open source projects and previous work is confirmed
- Customizable data presentation tool for observing software structures and fault distributions over the structures
- Linking repositories problem – within software lifecycle number of repositories exists aiming to collect information for different information needs
- Simulations aiming to find underlying distributions for generative models and finding simulation model of software fault-behaviour in network over time

- Questions?