UNIVERSITY OF RIJEKA
Faculty of Engineering

Tihana Galinac Grbac
Software Engineering and Information Processing Laboratory, SEIP Lab
http://www.seiplab.riteh.uniri.hr

Reliable Software Networks

1st EIT ICTLabs Future Networking Solutions Outreach Workshop
5th December, 2014, Budapest, Hungary

UNIVERSITY OF RIJEKA
Faculty of Engineering

Department of Computing

- Software engineering in large scale and complex software systems,
- Distributed computing, concurrent systems, network softwerfication
- Communication systems, middleware platforms and sensor networks
- Intelligent Computing Systems and Autonomy Oriented Computing
- Human-Computer interaction, usability of mobile technologies
- Artifical inteligence, smart syste and big data

Requirements that change traditional software engineering practices

- We need adaptable software networks that enables
  - **real time** and **live** adaptation to user needs
  - **end user** development
- Our system need to be **live.**
  - it must be possible for (globally distributed) end-users and software-developers to adapt and evolve the software systems **while the systems are in use**.
  - Problems of interoperability, reliability, scalability

Our research focus
Live system evolution and adaptation

- Software engineering for software netowrks
  - How to develop, verify and maintain reliable software networks at runtime enable evolution and adaptiation
  - How to enable runtime reliable reconfiguration, evolution, scale
  - Programming languaues, tools, processes for engineering such systems
  - Cost, measurement models that enables us to efficiently handle this systems and their evolution, adaptation
  - Human, technology interaction in global development and system use
- Industrial case studies, in global companies, SMEs, involving massive use; several hundreds of developers, users

Projects

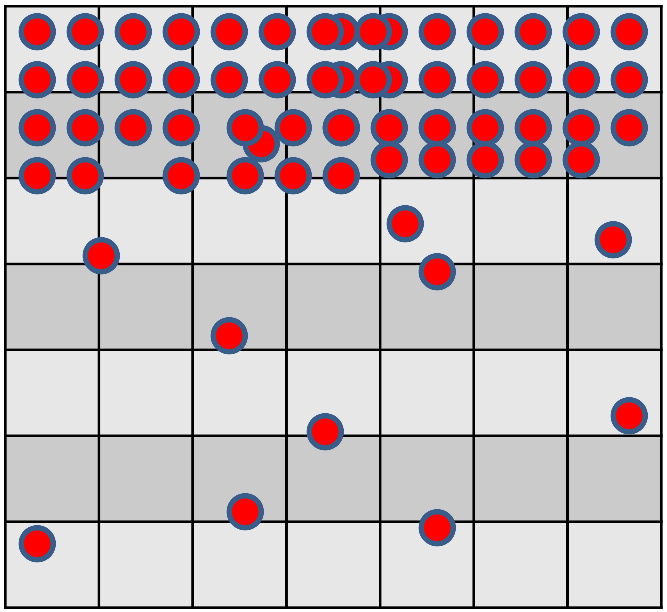
- *Analysis and inovative approaches to software development, management and application of complex software systems*
  - 2013 – 2016: *funded by Croatian Ministry of Science*
- *Autonomous Control for a Reliable Internet of Services*' funded by *European Cooperation in Science and Technology* – COST.
  - 2013- 2017: EU Project COST Action 1304 ACROSS
- *Behavioural Types for large-scale reliable systems*' funded by *European Cooperation in Science and Technology* – COST.
  - 2012- 2016: EU Project COST Action 1201 BETTY
- *Laboratory and educational support in ICT technologies*
  - 2010 – 2016: funded by Ericsson Nikola Tesla
- *New architectures and protocols in converged telecommunication networks*
  - 2007 - 2014: Colaboration with Ericsson funded jointly by Ericsson and Croatian Ministry of Science

Analysis and inovative approaches to software development, management and application of complex software systems

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

MZOS

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

MZOS

Software Fault and failure

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

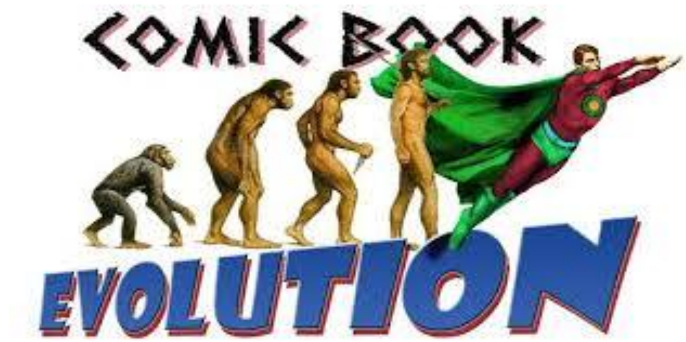
Fault execution leads to system failure

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

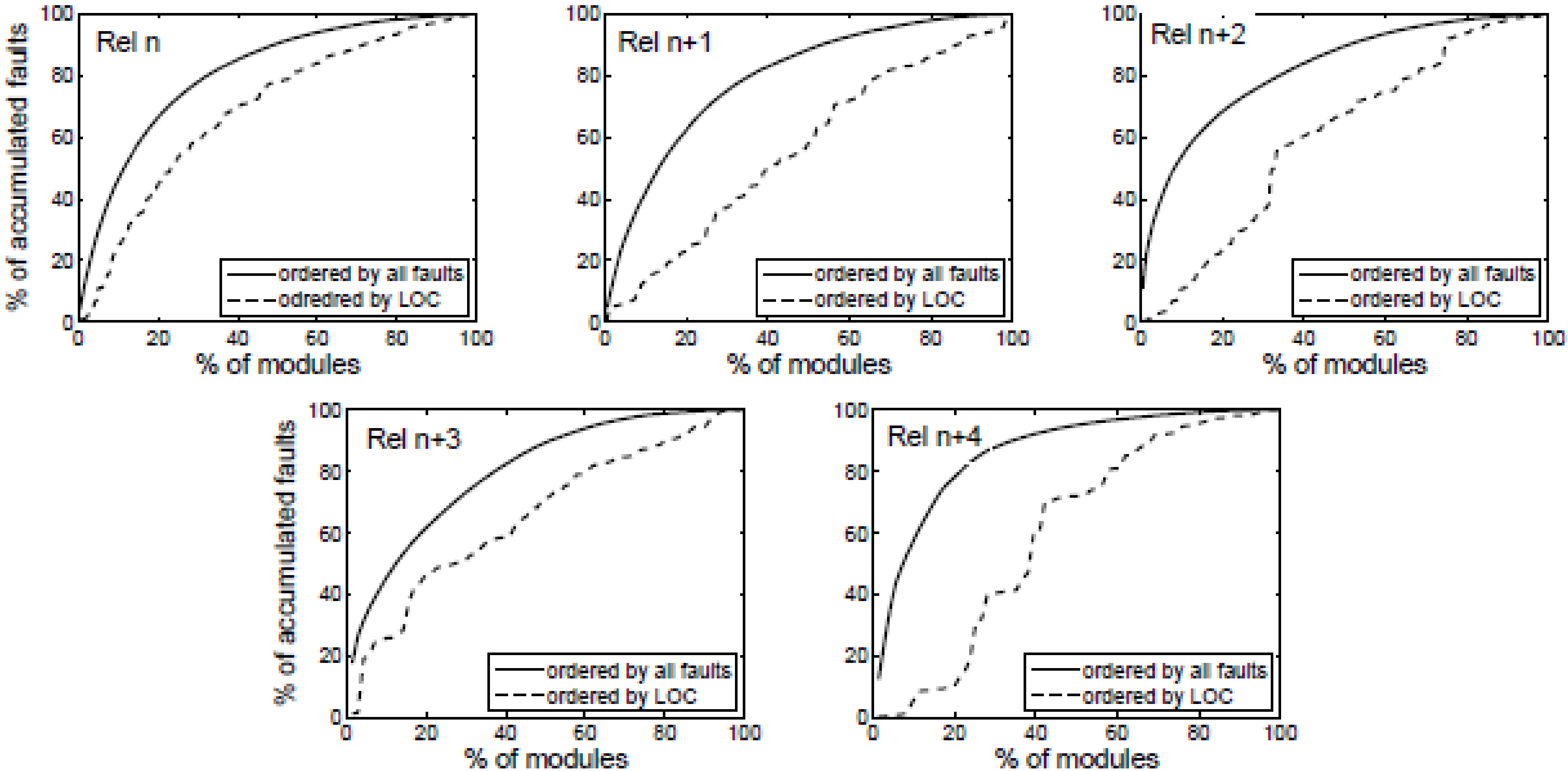
- Evolving system demands reusability
- Usually impacts thousands of end users
- Number of system versions may coexist at the same time
- Consequences of faults are impossible to predict
- Problem is not only effect of one fault, but effect of its repairment on the system as a whole



Fault distributions

- 2000: First systematic study on fault distributions by N.E. Fenton and N. Ohlsson in [ORIG] based on Basili and Pericone [BASPER]
- 2007: 1st systematic replication by C. Andersson and P. Runeson in [1REPL],
- 2013: 2nd systematic replication by T. Galinac Grbac, P. Runeson and D. Huljenić in [2REPL]

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

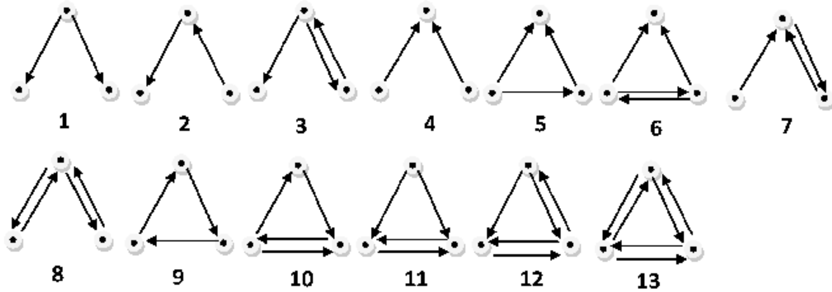
2009. Les Hatton. Power-Law Distributions of Component Size in General Software Systems. IEEE Trans. Software Eng. 35(4): 566-572

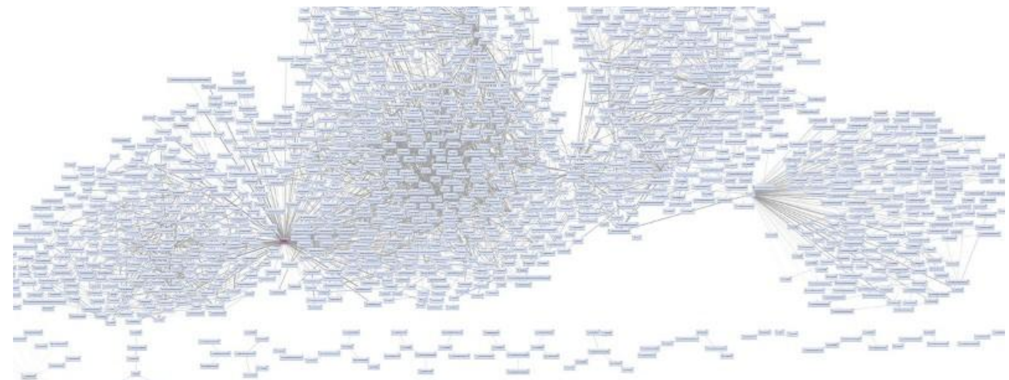
2014 Tihana Galinac Grbac, Darko Huljenić. On the Probability Distribution of Faults in Complex Software Systems, Information and Software Technology, published online first.

MZOS

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.*

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

2) Support vector machine
   - Support Vector Machine
   - Lagrangian SVM
   - Least Squares SVM
   - Linear Programming
   - Voted Perceptron

3) Nearest neighbor methods
   - k-Nearest Neighbor
   - K-Star

4) Decision trees
   - C 4.5 Decision Tree
   - Classification and Regression Tree
   - Alternating Decision Tree

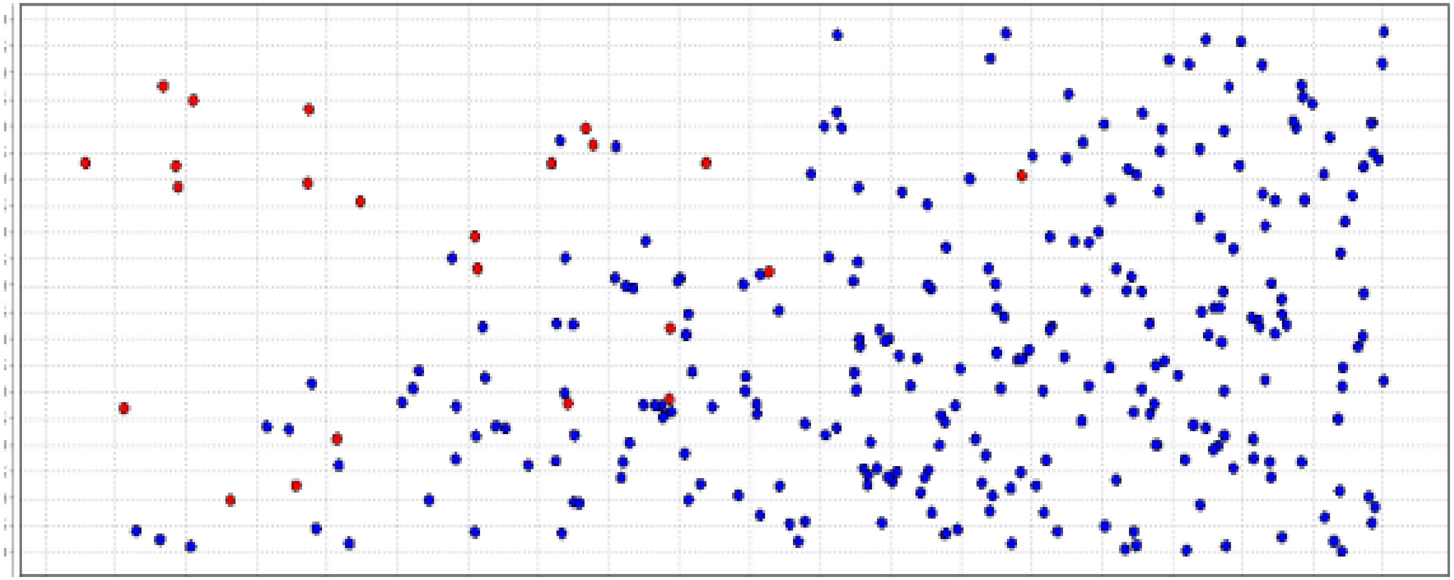
5) Neural networks
   - Multi-Layer Perceptron
   - Radial Basis Function Network

6) Ensemble methods
   - Random Forest
   - Rotation Forest
   - Logistic Model Tree

MZOS

Genetic approach

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



Predicting and simulating complex software systems behaviour

- Software become central part of the modern network, allow reconfiguration, flexibiliy
- 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

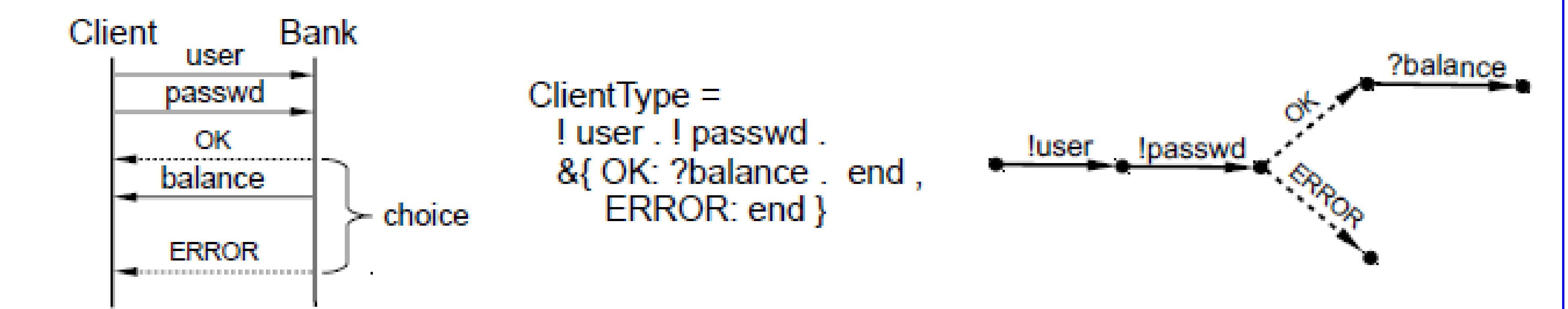
- More and more software systems tend to evolve towards complex software systems (e.g. IoS)
- Interconnection of peripheral systems over distributed network into system of systems (IoT)
- Key problems become:
  - Can we develop foundations on software behavior?
  - How can we measure software behaviour 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 its overall behaviour?

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



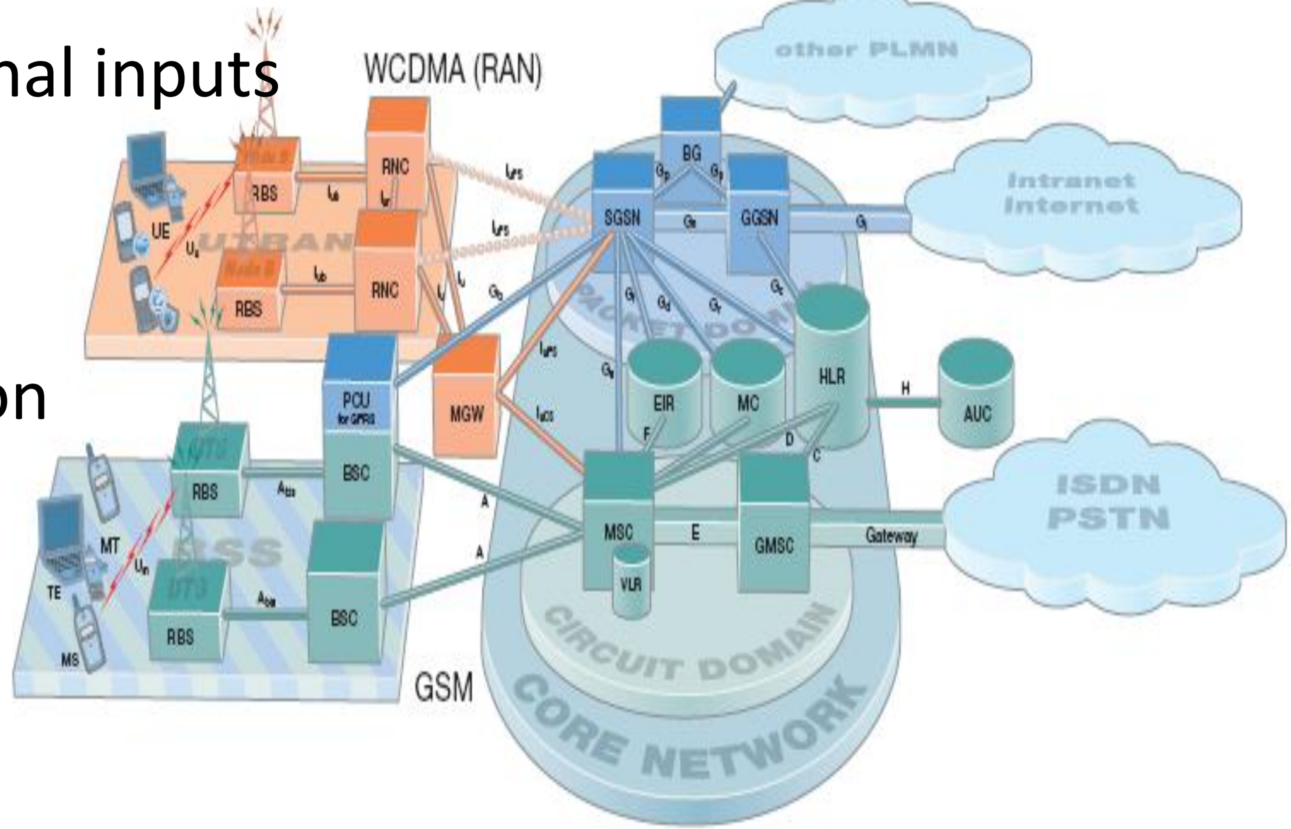
(a) sequence diagram (b) behavioural type (c) transition system

- Developed language: e.g. Scribble

Industry collaboration

Complex software system – an example: Ericsson NGN solution

- Properties of complex software systems:
  - Large scale > 3 millions Lines of code
  - Open to external inputs
  - Distributed
  - Concurrent
  - High interaction between parts
  - Evolutionary developed



Example of complex software system is telecommunication software

RIJEKA Competence center for smart CITIES

Ericsson cooperation with municipalities

Rijeka Smart City

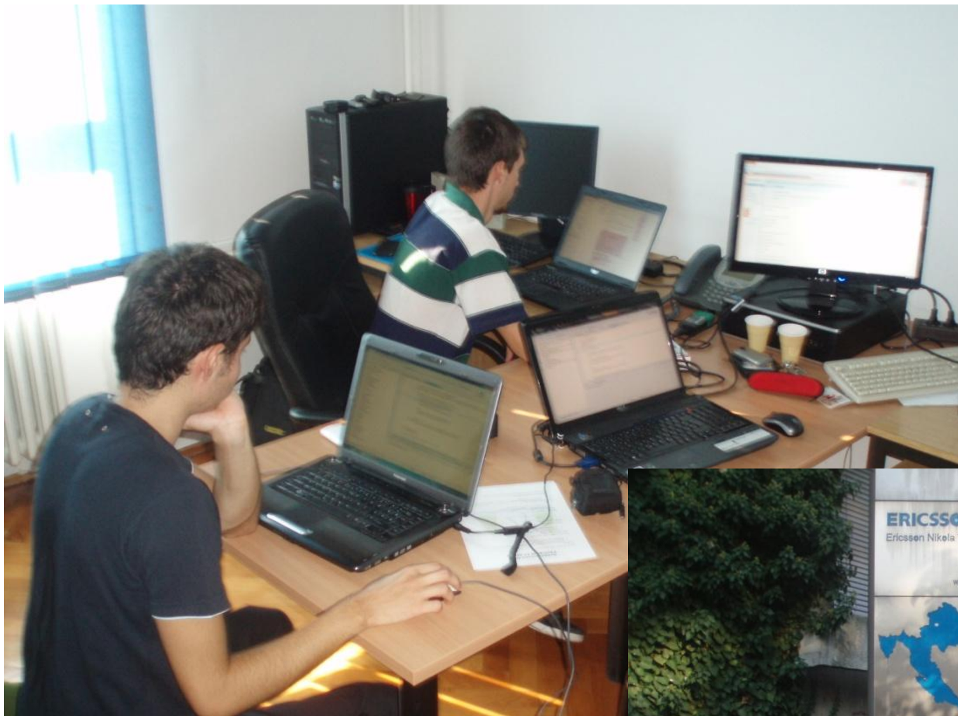
Urban and Traffic Revitalisation of the Rijeka Downtown Area
The pan European corridor - The Rijeka Traffic Corridor as part of the Pan-European Traffic Corridor Vb (Rijeka-Zagreb-Budapest)

iURBAN

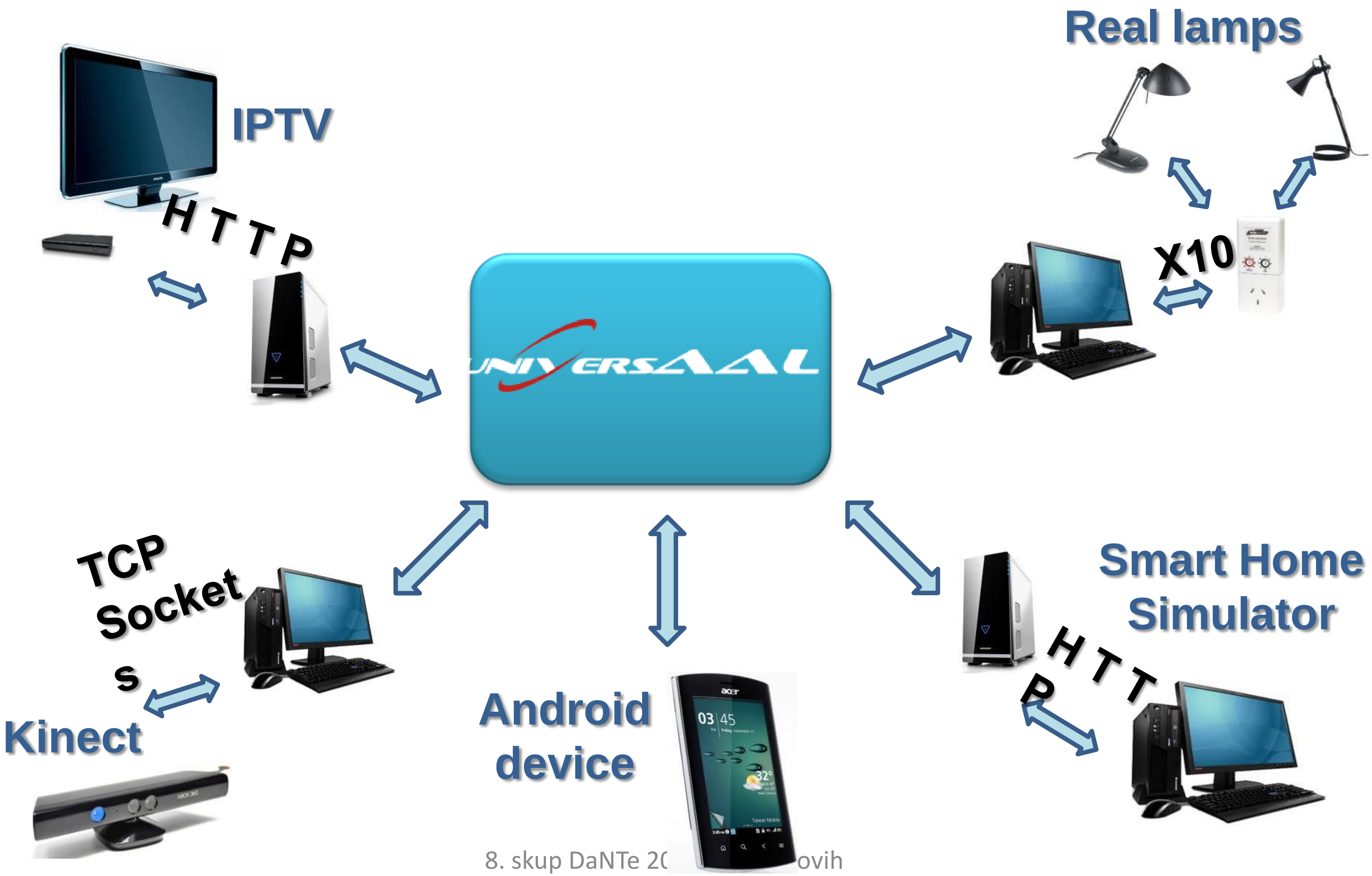
EGOV4U
connecting everyone

2020
Rijeka Healthy City

Summer school - industry collaboration in SEIP Lab



E-Health example: UniversAAL

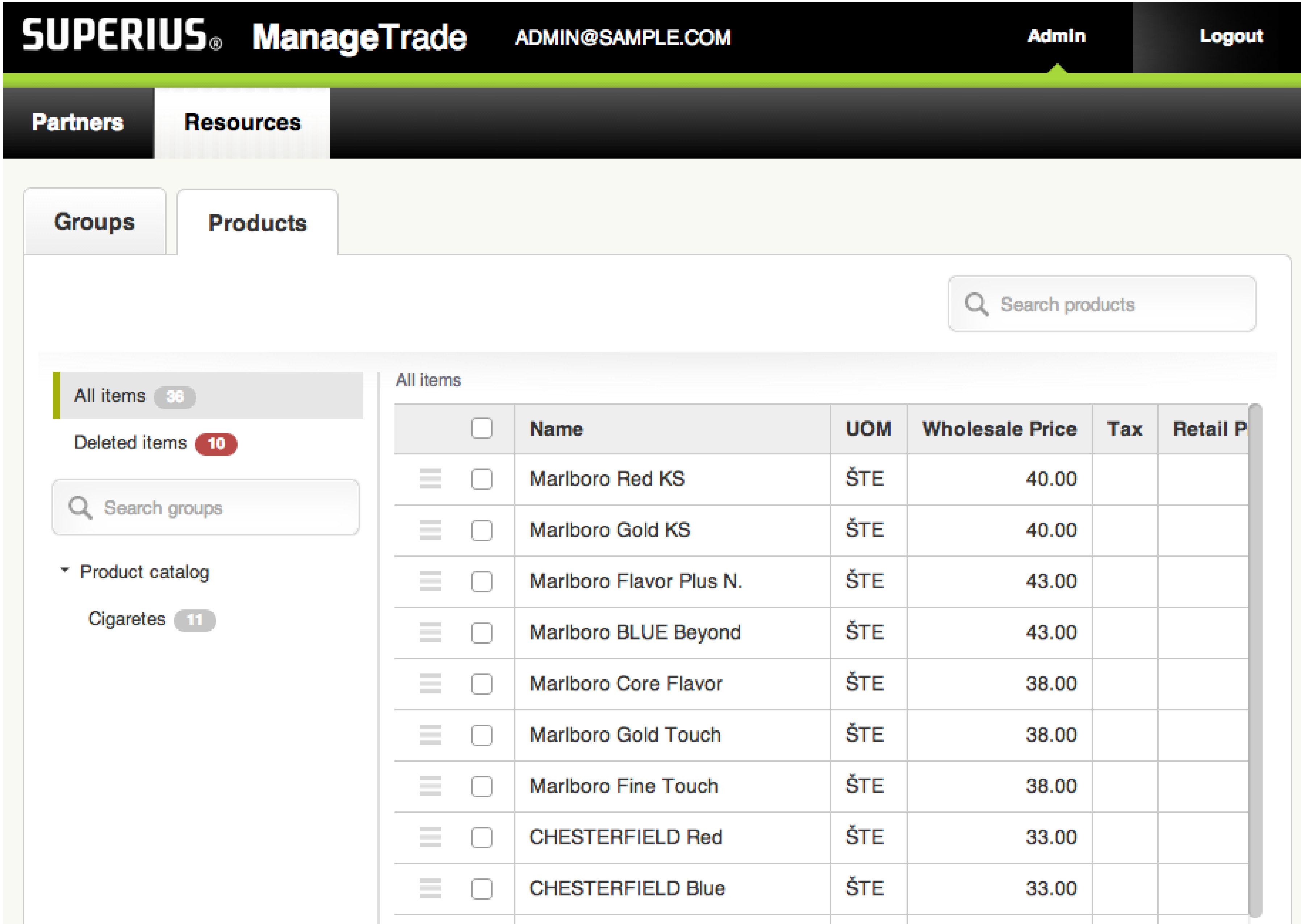


e-Finance Example: Distributed system with massive users

- Supply Chain Management (SCM), and Cash Register
- Producer -> Retailer -> Consumer
- A trading contract is made between Producer and Retailer (product positioning in stores, minimum stock and exposition, …)
- Producers dispatch Agents to ensure contracts are being met
- Agents use our “Manage Trade” software on their smartphone


MOTOROLA
2:55 PM
Manage Trade
Logout
SUPERIUS®MobileOffice
Tasks
Customer Visit
Reports
Data Synchronization
Help
2.0.14
verizon


MOTOROLA
2:55 PM
Cancel
Price Monitoring
Save
General
Items
Karlovačko crno 0.50
OOS
Akcija
Cijena
1.20
Karlovačko 0.50 boca
1.20
Karlovačko 0.50 limenka
1.70
Radler limenka 0.50L 1/24
1.80
Heineken 0.33.boca
1.60
Heineken 0.33 boca 1/6
Akcija false
9.60
MB Pils 2L PET boc 1/6
2.50
verizon


SUPERIUS ManageTrade
ADMIN@SAMPLE.COM
Admin
Logout
Partners
Resources
Groups
Products
Search products
All items 36
Deleted items 10
Search groups
Product catalog
Cigaretes 11
All items
Name
UOM
Wholesale Price
Tax
Retail P
Marlboro Red KS ŠTE 40.00
Marlboro Gold KS ŠTE 40.00
Marlboro Flavor Plus N. ŠTE 43.00
Marlboro BLUE Beyond ŠTE 43.00
Marlboro Core Flavor ŠTE 38.00
Marlboro Gold Touch ŠTE 38.00
Marlboro Fine Touch ŠTE 38.00
CHESTERFIELD Red ŠTE 33.00
CHESTERFIELD Blue ŠTE 33.00

- Questions?