Modeling fault behavior in evolving complex open software environments

ACROSS MC meeting, October 13, 2016, Bilbao, Spain

Tihana Galinac Grbac

UIP-2014-09-7945
EVOSOFT
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.)
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?
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 awareness based on system structure and measurements on software abstract levels
Previous studies on fault distributions

- Empirical studies on fault distributions
- Analytical studies on fault distributions
- Industrial versus open source
System verification and reliability

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

Source: Complex software systems: Formalization and Applications - Work done in EU project GENNETTEC: GENetic NeTworks: Emergence and Complexity
A small number of modules contain most of the faults.

System and system components
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*
• Small occurrences are extremely common and large occurrences are extremely rare

Fault distributions
Empirical studies on fault distributions

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 distributions like double Pareto, Weibull, lognormal, Pareto, and Yule-Simon with power-law in the tail are confirmed

Research questions

• RQ1: How faults are distributed across the software units
• RQ2: Does fault distribution depends on development environment
• RQ3: Is the fault distribution persistent over the system evolution
Results of analytical distributions fit

Nonlinear regression fit for Pareto, double Pareto, Weibull and Lognormal distribution

COST ACROSS 2nd Summer School, May 27-31, Opatija, Croatia
Autonomous Control for Reliable Future Networks and Services
Data selection

- complex enough for network analysis
- evolve over a number of system releases
- Industrial and open access software
- Access to source code and defect repository
- We selected one industrial telecom core network product from 4G network and two Eclipse plugins: PDE, JDT
Data collection

Source code repository
- collection of modules, Classes, software units

Failure report repository
Repository
- collection of modules, Classes, software units
Distribution fit across the studies

<table>
<thead>
<tr>
<th>Rank</th>
<th>This study</th>
<th>Galinac Grbac et al. [2]</th>
<th>Concas et al. [1]</th>
<th>Zhang [3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yule–Simon</td>
<td>Double Pareto</td>
<td>Yule–Simon</td>
<td>Weibull</td>
</tr>
<tr>
<td>2</td>
<td>Pareto</td>
<td>Lognormal</td>
<td>Double Pareto</td>
<td>Pareto</td>
</tr>
<tr>
<td>3</td>
<td>Double Pareto</td>
<td>Yule–Simon</td>
<td>Lognormal</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>Lognormal</td>
<td>Weibull</td>
<td>Weibull</td>
<td>—</td>
</tr>
<tr>
<td>5</td>
<td>Weibull</td>
<td>Pareto</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>


Distributions fit – $R^2$
Evolution of Yule-Simon distribution
Conclusion

• Yule Simon gives the best fit for all analysed projects
• Yule Simon is similar for projects in system evolution
• But, there are differences between parameters in different environments (JDT and PDE projects – Open source Eclipse projects)
• We can reuse Y-S but only between releases in system evolution
• Fitting Y-S with p0 (number of modules with no faults) from the data and p0 not from the data gives almost the same parameters (similarity up to 2 decimal places)
• In environment where there is a lot of software units with no faults Pareto distribution is almost as good as Yule Simon because the tail starts close to ‘0’.
• Simulations aiming to find underlying distributions for generative models and finding simulation model of software fault-behaviour in network over time
• Next step we want to find a model based on Yule process that can explain evolution of faults and other system properties of large complex systems