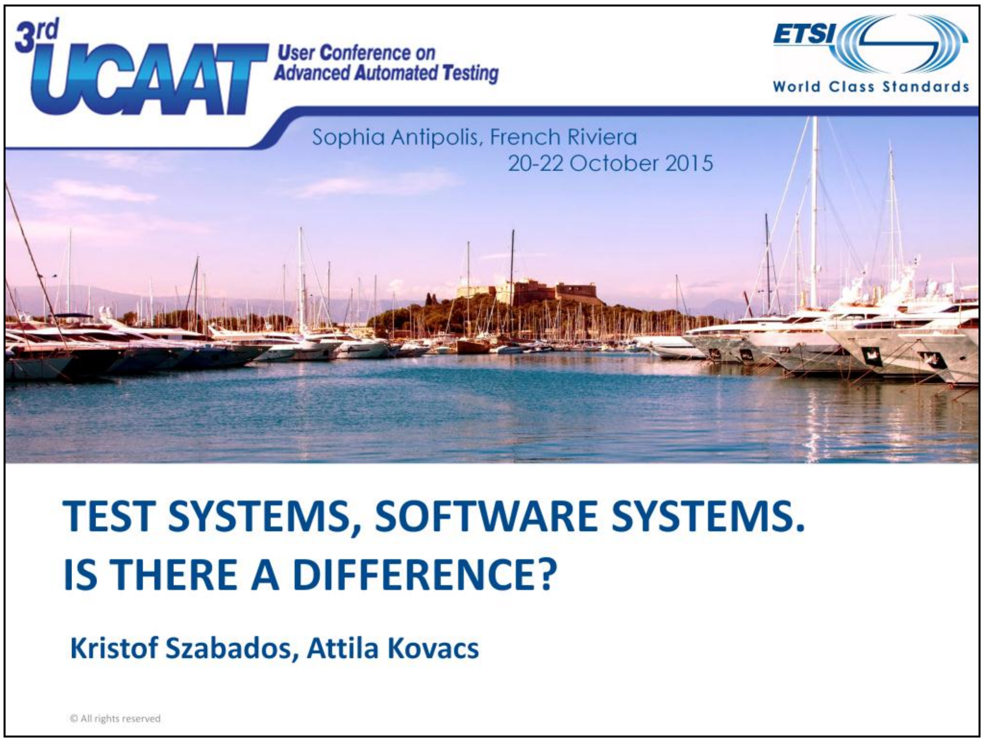


3rd UCAAT User Conference on Advanced Automated Testing

ETSI World Class Standards

Sophia Antipolis, French Riviera
20-22 October 2015

TEST SYSTEMS, SOFTWARE SYSTEMS. IS THERE A DIFFERENCE?

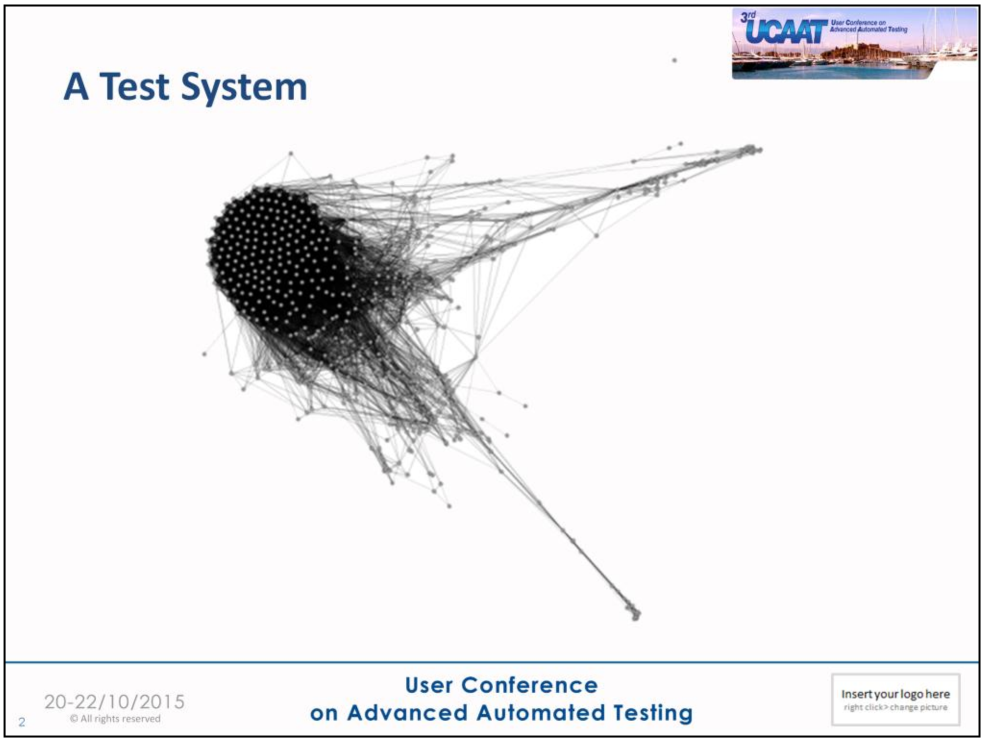
Kristof Szabados, Attila Kovacs

I believe every one has already heard the joke: what is the difference between testers and developers? Developers could learn programming, testers do not.

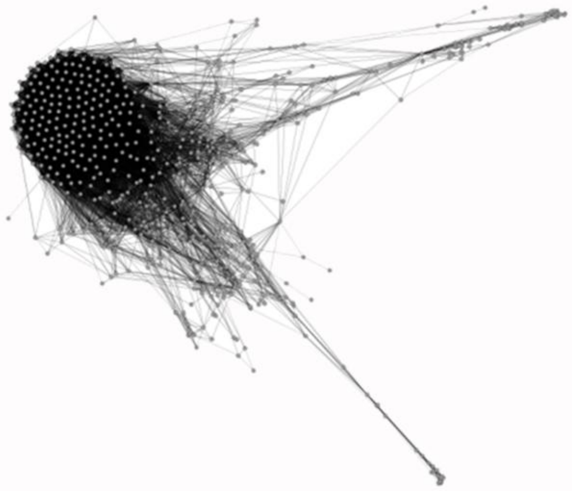
I have to admit. I was thinking in a similar way until 9 years ago we received a request from a user of our tool.

She did not know how to design the architecture of her test system.

When we look at it …



A Test System



20-22/10/2015
© All rights reserved
2

User Conference
on Advanced Automated Testing

Insert your logo here
right click> change picture

This is the architecture of a test system.

The points are TTCN-3 and ASN.1 modules.

The lines are import statements connecting the modules.

3rd UCAAT User Conference on Advanced Automated Testing
20-22/10/2015
3
© All rights reserved
User Conference
on Advanced Automated Testing

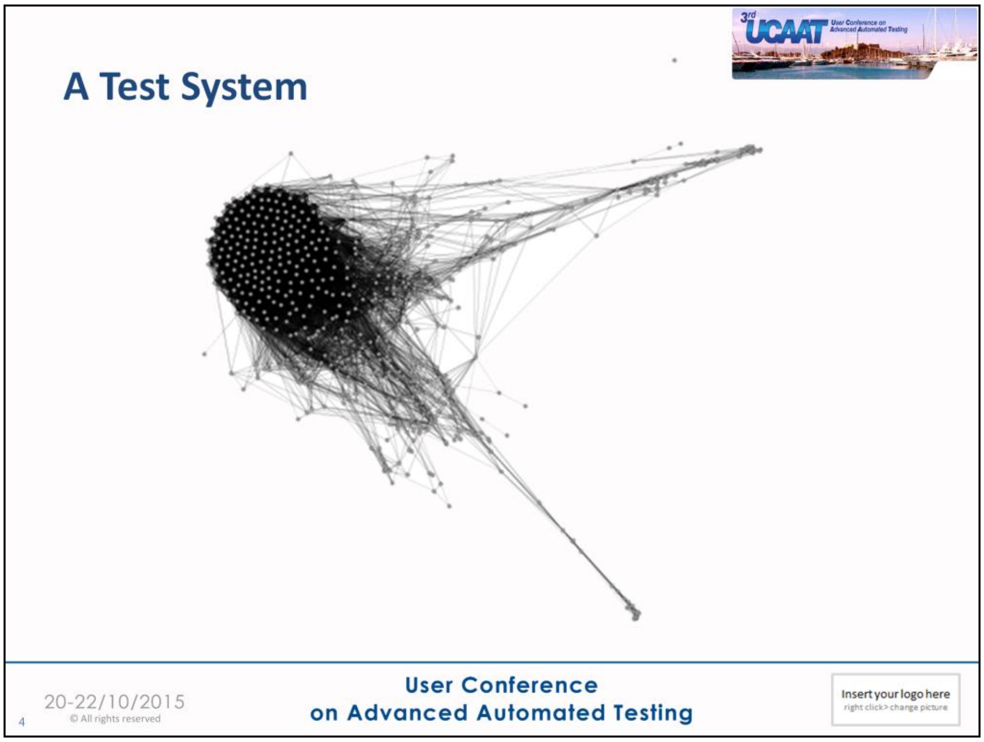
Our project is part of a cooperation between Ericsson and the Eötvös Loránd University, Budapest.

Personal pages of the authors:

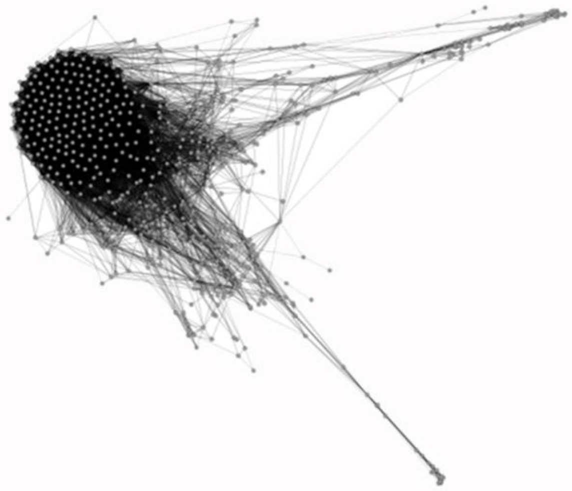
http://compalg.inf.elte.hu/~attila/

https://www.researchgate.net/profile/Kristof_Szabados

The home page of the project:
http://compalg.inf.elte.hu/~attila/TestingAtScale.html



A Test System



20-22/10/2015
© All rights reserved
4

User Conference
on Advanced Automated Testing

Insert your logo here
right click> change picture

This is the architecture of a test system.

The points are TTCN-3 and ASN.1 modules.

The lines are import statements connecting the modules.

This network contains small world and scale-free properties.

If we select any 2 nodes, the shortest path between them will contain at most 5 other nodes.

The distribution of the connection follows the power law. Just like the Internet, the human brain and software systems.

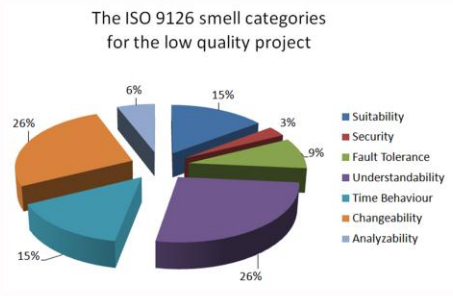
Related publication:

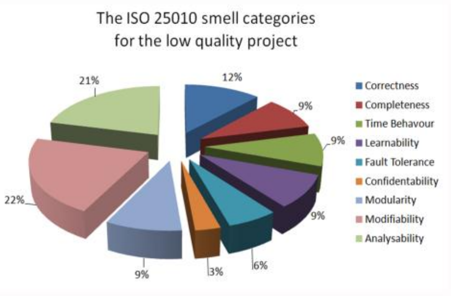
K. Szabados, Structural Analysis of Large TTCN-3 Projects in proceedings of: Testing of Software and Communication Systems, 21st IFIP WG 6.1 International Conference, TESTCOM 2009 and 9th International Workshop, FATES 2009, Eindhoven, The Netherlands, November 2-4,

2009.

3rd UCAAT User Conference on Advanced Automated Testing

From the ground up





20-22/10/2015
5 © All rights reserved

User Conference
on Advanced Automated Testing

We look at test system as software systems (that happen to be used for testing).

We are interested in the properties of test systems as software systems: maintainability, reliability, performance, etc…

For this reason we have translated ISO 9126 and ISO 25010 software requirements for tests written in TTCN-3.

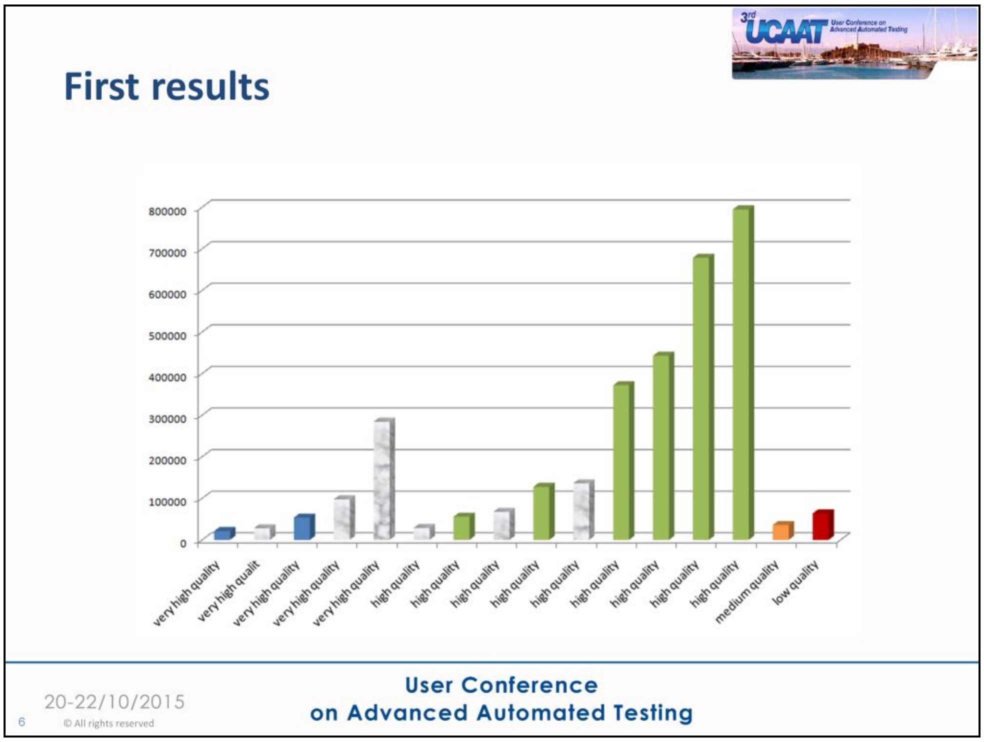
We looked at several tools analyzing software systems: FindBugs, FxCop, PMD, CheckStyle, etc…

We have defined and implemented 40+ code smells, covering all main categories of the ISO 9126 and ISO 25010 software quality standards.

Related publication:

A. Kovács and K. Szabados, Test software quality issues and connections to international standards in Acta Universitatis Sapientiae, Informatica,

5/2013. pages 77-102.



First results

20-22/10/2015

6 © All rights reserved

User Conference on Advanced Automated Testing

We assumed that standard tests are of good quality.

We measured and categorized the internal quality of 16 applications in the telecom area.

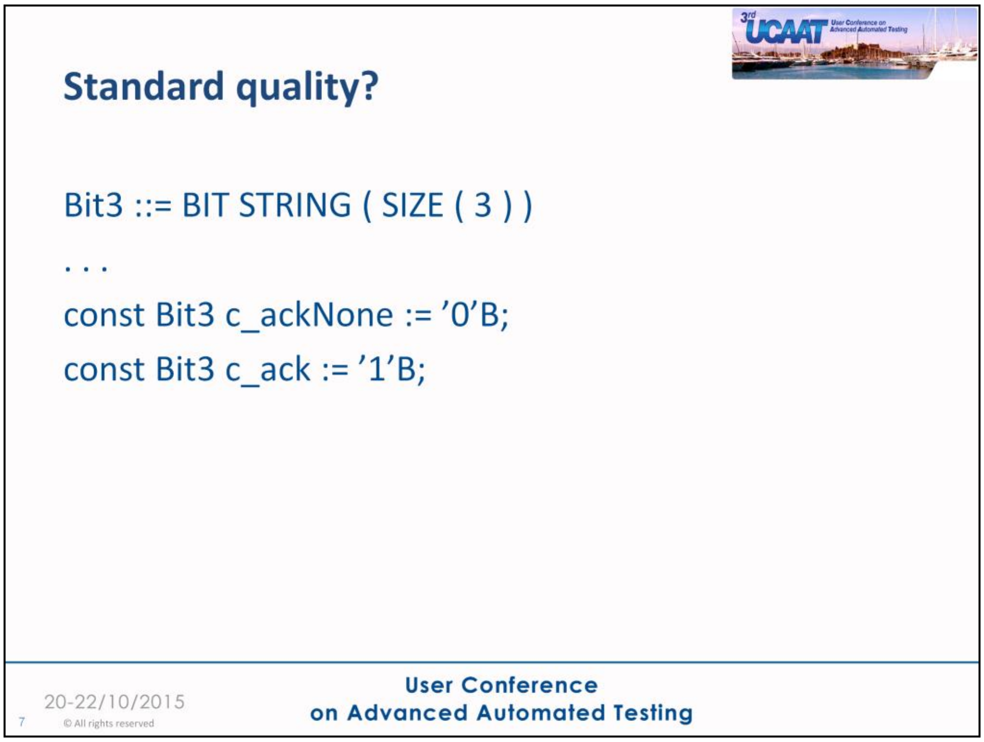
On the above charts the column with marble texture denotes standardized TTCN-3 test systems.

As it can be seen we were already able to find some bad quality test systems.

Related publication:

A. Kovács and K. Szabados, Test software quality issues and connections to international standards in Acta Universitatis Sapientiae, Informatica,

5/2013. pages 77-102.



Standard quality?

```
Bit3 ::= BIT STRING ( SIZE ( 3 ) )
. . .
const Bit3 c_ackNone := ’0’B;
const Bit3 c_ack := ’1’B;
```

20-22/10/2015
7 © All rights reserved
User Conference on Advanced Automated Testing

Assuming that standardized test systems are of good quality might not be a good idea:

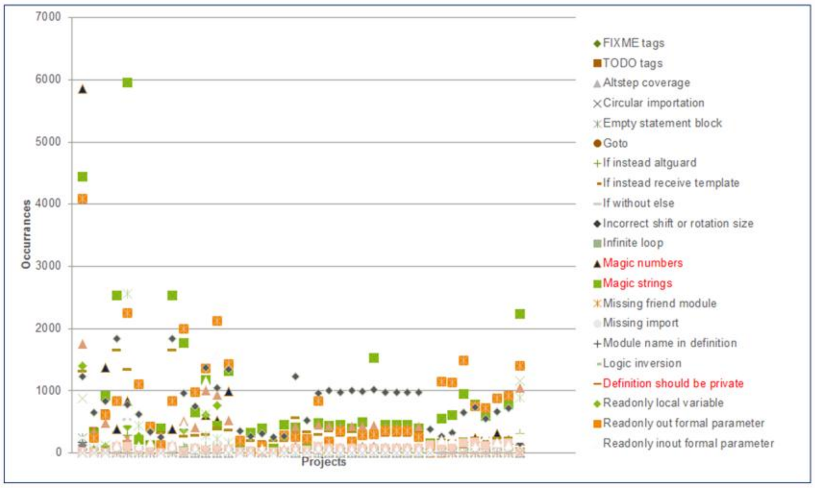
- We have found syntactical issues, where the TTCN-3 code did not follow the BNF of the TTCN-3 language.
- We have found semantical issues (the chart shows a bit string of 3 elements long getting twice a single character long value)

Related publication:

A. Kovács and K. Szabados, Advanced TTCN-3 Test Suite validation with Titan, 2014, In Proceedings of the 9th International Conference on Applied Informatics, Vol. 2, pages 273-281.

3rd UCAAT User Conference on Advanced Automated Testing

Standard quality?



20-22/10/2015

8 © All rights reserved

User Conference
on Advanced Automated Testing

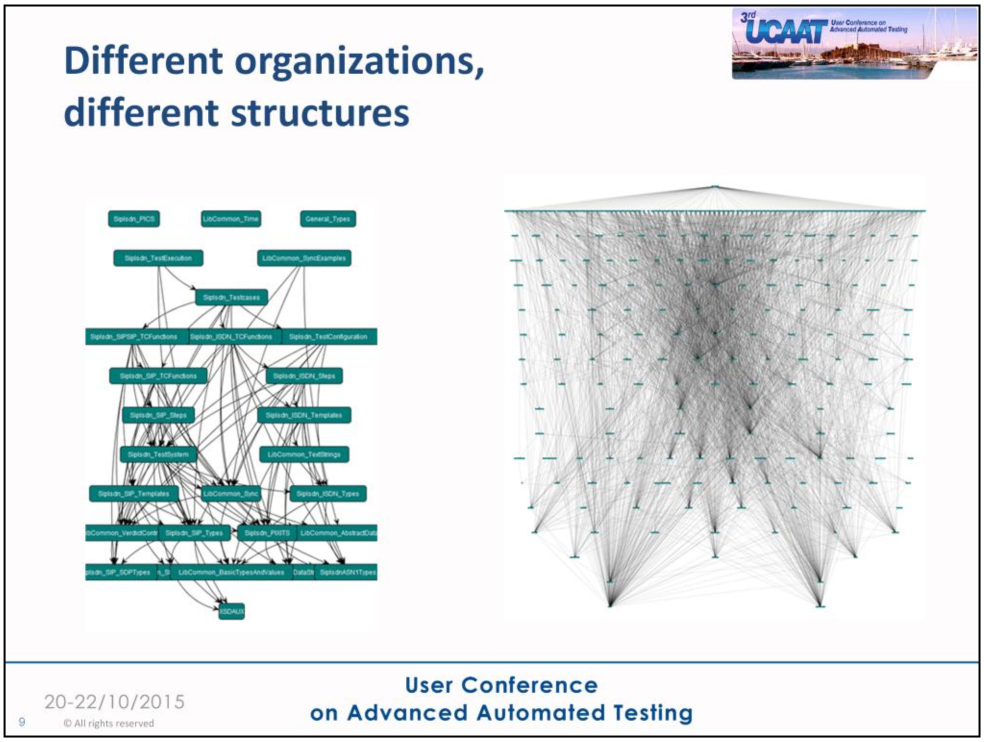
We also found hundreds, in some cases thousands of code smell instances.

The most popular being:

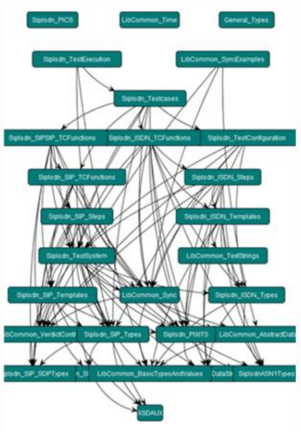
- Magic strings and numbers
- Un-initialized local variables
- Unused global definitions
- Definitions that should be private, but are not set so

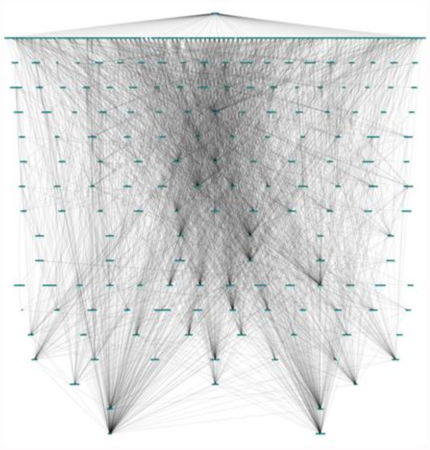
Related publication:

A. Kovács and K. Szabados, Advanced TTCN-3 Test Suite validation with Titan, 2014, In Proceedings of the 9th International Conference on Applied Informatics, Vol. 2, pages 273-281.



Different organizations, different structures





20-22/10/2015

9 © All rights reserved

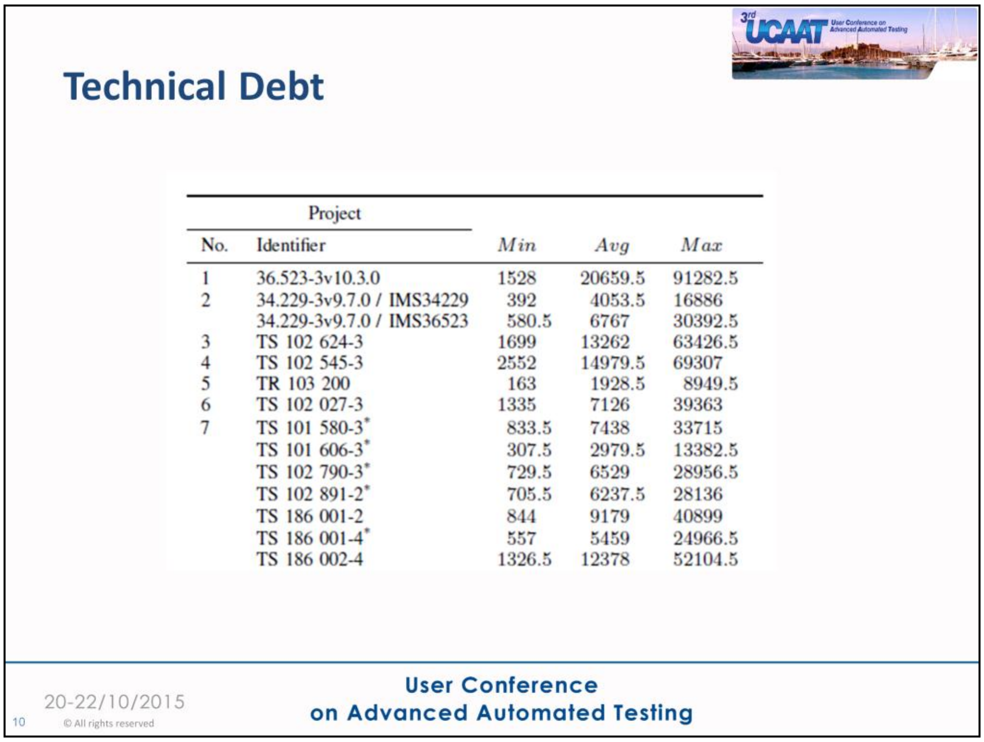
User Conference on Advanced Automated Testing

We also had some interesting observations on architectural level:

- Systems developed by ETSI look alike, systems developed by 3GPP look alike … but ETSI and 3GPP systems differ visibly
- ETSI developed test systems tend to contain modules that are not connected to the rest of the test system. While we have not seen this in any 3GPP test system.

Related publication:

A. Kovács and K. Szabados, Advanced TTCN-3 Test Suite validation with Titan, 2014, In Proceedings of the 9th International Conference on Applied Informatics, Vol. 2, pages 273-281.



Technical Debt

Project				
No.	Identifier	Min	Avg	Max
1	36.523-3v10.3.0	1528	20659.5	91282.5
2	34.229-3v9.7.0 / IMS34229	392	4053.5	16886
	34.229-3v9.7.0 / IMS36523	580.5	6767	30392.5
3	TS 102 624-3	1699	13262	63426.5
4	TS 102 545-3	2552	14979.5	69307
5	TR 103 200	163	1928.5	8949.5
6	TS 102 027-3	1335	7126	39363
7	TS 101 580-3*	833.5	7438	33715
	TS 101 606-3*	307.5	2979.5	13382.5
	TS 102 790-3*	729.5	6529	28956.5
	TS 102 891-2*	705.5	6237.5	28136
	TS 186 001-2	844	9179	40899
	TS 186 001-4*	557	5459	24966.5
	TS 186 002-4	1326.5	12378	52104.5

20-22/10/2015 User Conference on Advanced Automated Testing

10 © All rights reserved

To better understand the situation we translated the technical debt of the code smells, into Man-Hours.

Using the Delphi method, with the help of test developers and test system architects we have estimated the cost of fixing code smell types.

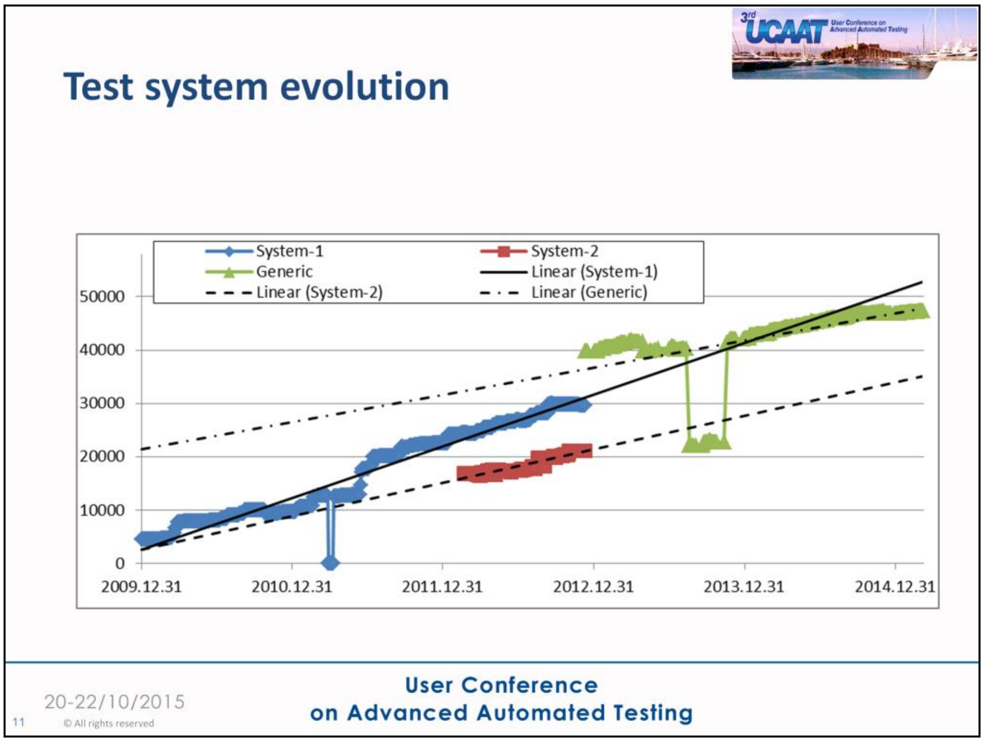
The slide shows a part of our measurements for all publicly available test systems from www.ttcn-3.org.

It can be seen that even if all code smell instances are the easiest to correct, the total cost of correction would be month in most of the test systems.

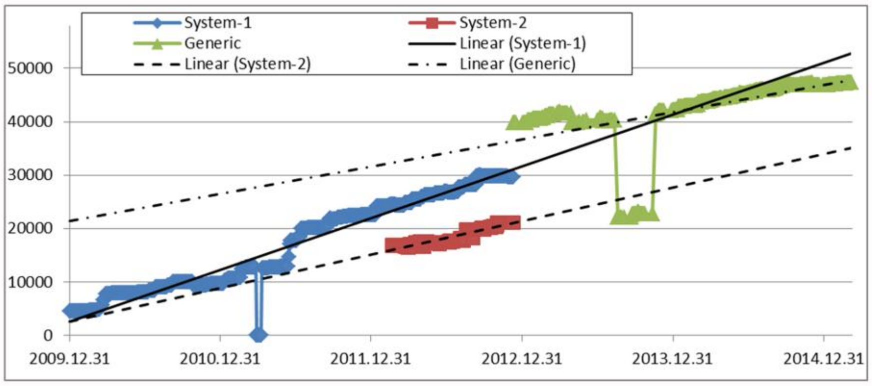
If they would the hardest to solve, the total technical debt would be years to solve.

Related publication:

A. Kovács and K. Szabados, Technical Debt of Standardized Test Software, MTD 2015



Test system evolution



20-22/10/2015

11 © All rights reserved

User Conference
on Advanced Automated Testing

We observed a 5 years period of the development of a large test system.

On the slide we show our measurement on the number of one of the code smells (others followed similar trends).

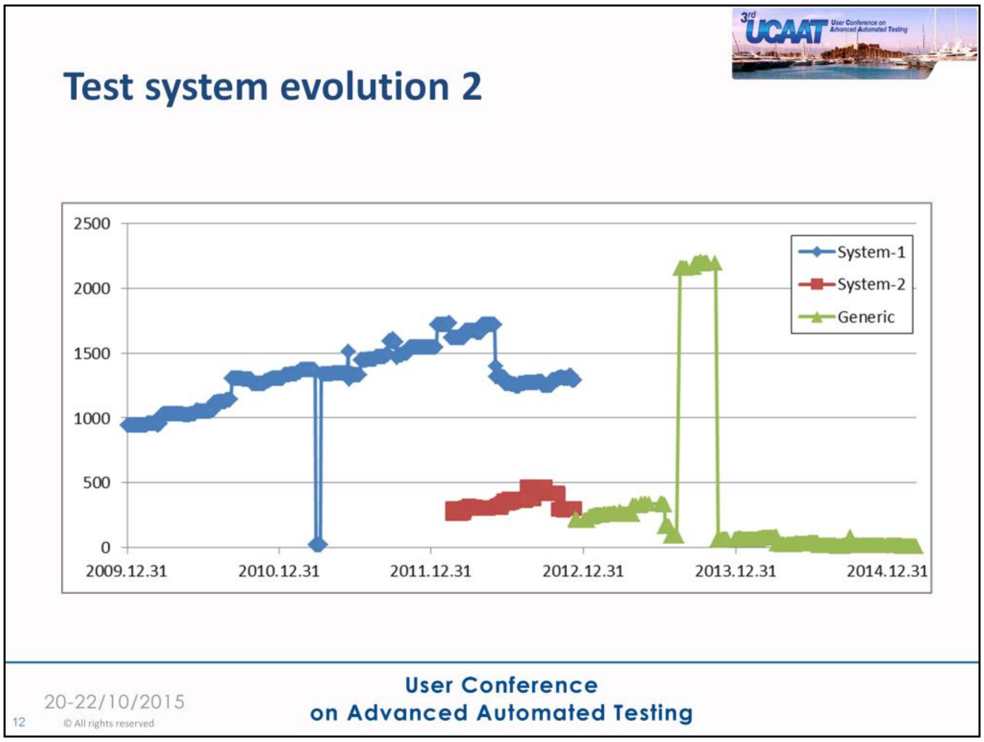
System-1 was developed for 3 years, System-2 for 1 year. At the end of 2012 they were merged, creating a new system (Generic).

It is visible that the merge increased the size of the system, and the number of issues in it. After the merge the development slowed down for a year.
The growth followed an almost linear trend.

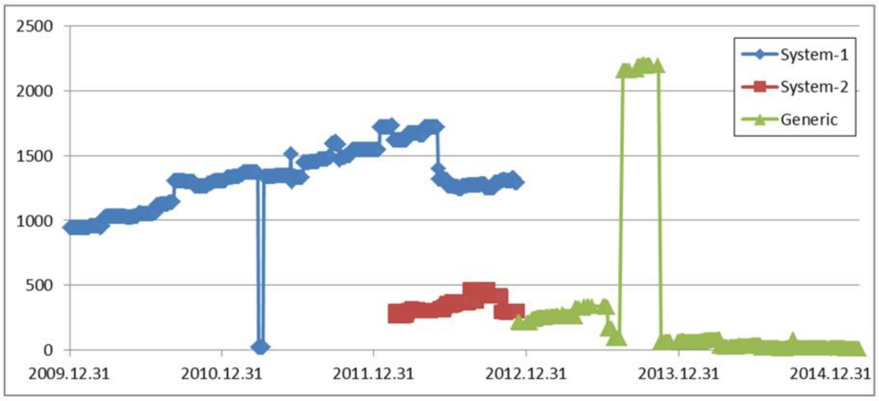
Several changes that had no effect on the quality of the system:

- Introduction of Agile, Scrum teams, Kanban teams.
- System architect leaving, creation of a system architect group, assigning a head system architect.
- Introduction of continuous integration
- The teams doubling the starting size.

Submitted publication



Test system evolution 2



20-22/10/2015 User Conference on Advanced Automated Testing

12 © All rights reserved

In some cases were we able to see an effect on the quality.

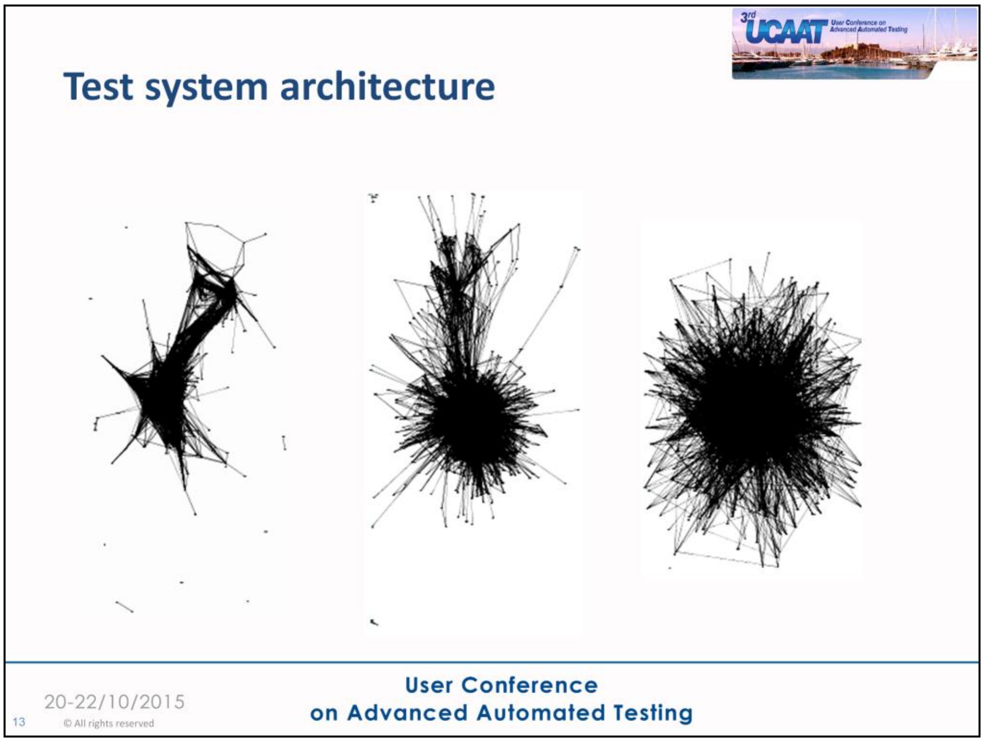
On the above chart we can see the effect of one tester taking one day to correct issues, and 2 hackathons/colorful days.

(One day events organized by developers with the only aim to correct code smells)

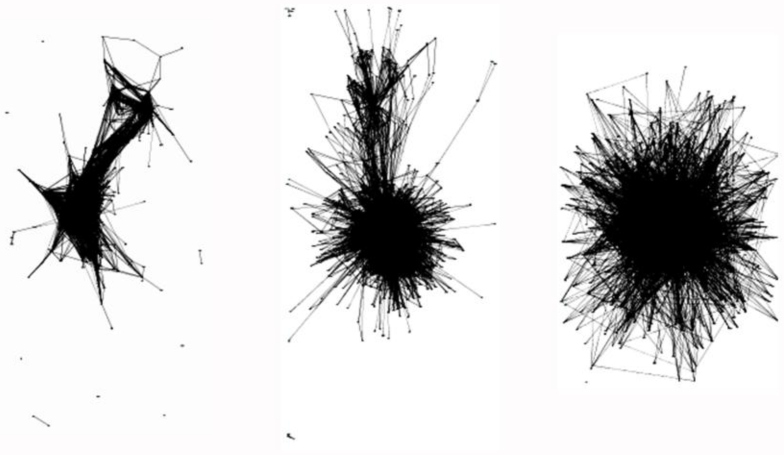
This is the first observation of the Lehman's law on test systems as far as we know it:

- Law 2: "As an E-type system is evolved its complexity increases unless work is done to maintain or reduce it"
- Law 4: "Unless feedback mechanisms are appropriately adjusted, average effective global activity rate in an evolving E-type system tends to remain constant over product lifetime"

Submitted publication



Test system architecture

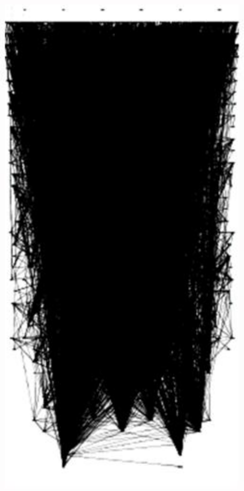


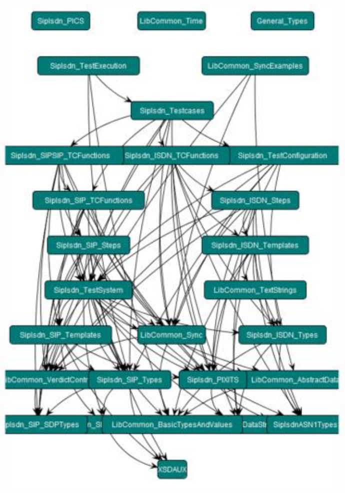
We have implemented several layouts to visualize the architecture of test systems.

Unfortunately, the usual layouts generate visualizations are too hard to understand.

Publication under construction

Layered architecture





20-22/10/2015

14 © All rights reserved

User Conference
on Advanced Automated Testing

We have found that in the case of test systems the most useful visualization could be a levelized one.

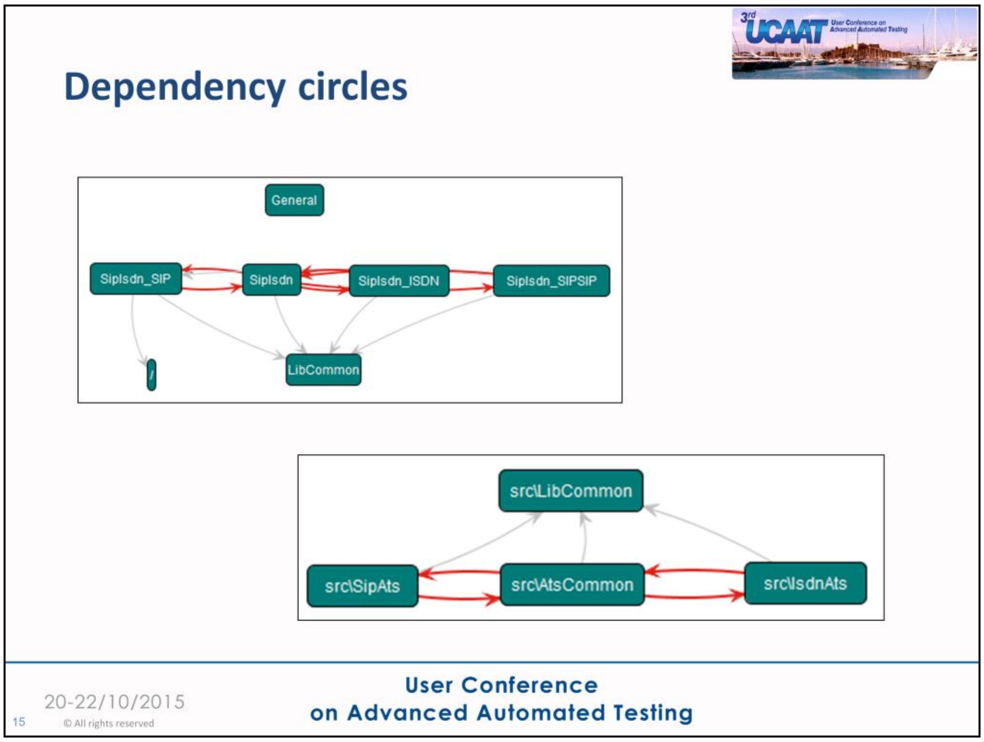
In this visualization nodes with no incoming or outgoing connections are extracted to the 0th level.

Node with only outgoing edges are listed on the 1th level.

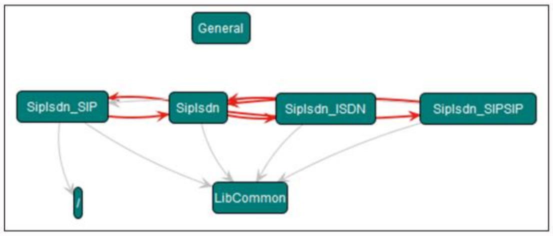
On each further level we locate nodes that have incoming edges only from nodes on lower levels.

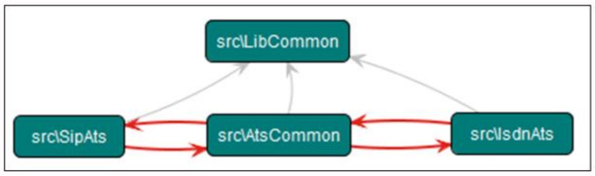
This visualization enables test systems architects to work efficiently, by answering the question: what should ne checked next ?

Publication under construction



Dependency circles





20-22/10/2015

15 © All rights reserved

User Conference on Advanced Automated Testing

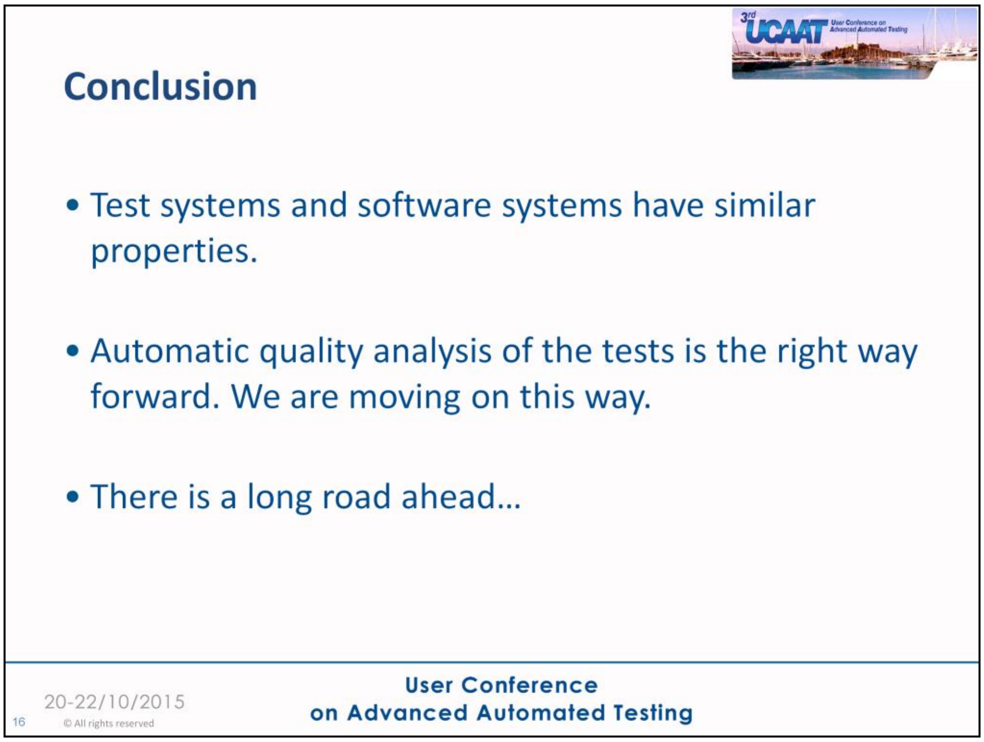
We have observed dependency circles in the architecture of most of the test systems.

These dependency circles usually point to architectural confusion in software systems.

We wished to see how the architecture of test systems looks like on higher abstraction levels, so we defined:

- A layout where modules are aggregated and represented by the folder they are located in, keeping the dependency of the modules.
- A layout where a package hierarchy is created based on the names of the modules (where _ and small/capital letter change).

On both abstraction levels we were able to observe circular dependency, in most of the test systems available from www.ttcn3.org



Conclusion

- Test systems and software systems have similar properties.
- Automatic quality analysis of the tests is the right way forward. We are moving on this way.
- There is a long road ahead…

20-22/10/2015 © All rights reserved

User Conference on Advanced Automated Testing

16

So far we have seen that test systems are built like software systems, into architectures seen in software systems and governed by laws applying to software systems.

We believe that test automation is the right way forward and we are on the right track now.

Although the current systems have several issues, this is because we know very little of them right now.

To be able to build better test systems in the future we must understand this aspect of testing in more detail.

More research is needed in this field, more test systems being studied.