Artur SOSNÓWKA
WEST POMERANIAN UNIVERSITY OF TECHNOLOGY,
Żołnierska 49, 71-210 Szczecin

Visualisation metaphor as help in test case analysis and selection

Mgr inż. Artur SOSNÓWKA

Artur Sosnówka actualy PhD student at West Pomeranian University of Technology in Szczecin. Since more than 10 years involved in software engineering process in the industry. Actually working as a consultant – Test & Project Manager. Involved in the research for test process optimization and decreasing the cost of software testing.

e-mail: arsosnowka@wi.zut.edu.pl

Abstract

Detection of problems within a test base can save much effort and associated cost as time progresses. One method of performing routine assessment for tests is to collect software metrics associated with functionality and complexity. One way of potentially increasing empirical analysis activity is to contemplate visualisation as a means to readily analyse different software artefacts. In this paper several visualization techniques and metaphors as well as usable metrics for test analyses will be summarized. Those criteria allow evaluating the existing metaphors and searching for adequate metaphors to design new specialized systems.

Keywords: test archaeology, data mining, test case visualization.

Analiza i selekcja testów za pomocą przenośni wizualizacyjnej

Streszczenie

Wykrywanie błędów w bazie testowej może oznaczać znaczące oszczędności kosztów i koniecznego do ich przeprowadzenia czasu. Zarówno koszta zgodności jak i braku zgodności odgrywają ważną rolę w doborze koniecznych testów. Jedną z metod pozwalających na rutynową kontrolę testów jest zbiór metryk sprzężonych z funkcjonalnością oraz kompleksowością oprogramowania. Jednym z ostatnio coraz częściej używanych empirycznych podejść do analizy zarówno kodu jak i dużej ilości wzajemnie sprzężonych informacji jest wizualizacja, za pomocą której w łatwy i czytelny sposób mogą zostać przedstawione różne artefakty dostępnych danych. W artykule przedstawiono zestawienie różnych przenośni wizualizacyjnych oraz proponowanych metryk, które mogą zostać wykorzystane w celu optymalizacji, reorganizacji i zarządzania testami. Przedstawione kryteria mogą służyć jako wytyczne kierunku badań i być pomocne do oceny i doboru odpowiedniej metafory wizualizacyjnej nowego systemu analitycznego oraz najlepiej do tego pasujących metryk.

Słowa kluczowe: Archeologia testów, Data Mining, Wizualizacja testów, Application Lifecycle Management.

1. Introduction

Today's software is going to increase its complexity, which grows exponential and their development and maintenance involves big amount of different groups of technicians. The required quality is, however, not decreasing with the product complexity, the quality needed is increasing! This makes the tasks of programming, understanding and maintaining the software code and its test cases more and more difficult. This is even more difficult when working on the code or test cases written by other organisations or supporters.

During the whole project and software lifecycle, available requirements change and new one come on top. New requirements/functionality become the main focus and the old ones are very often not that important as before. The management of those stays and keeps going to be not any more affordable or getting tendency to be forgotten on purpose. The maintenance costs increase and very often reach a limit when it is easier to develop a new product than to create new functionality for a current system.

2. Testing in software development lifecycle

Software development life cycle is a structure imposed on the development of a software product. There are several models describing different approaches to variety of tasks or activities that take place during the whole process. This process is also described in ISO/IEC 12207:2008 for "Systems and software engineering – Software life cycle processes" standard [1]. The standard has the main objective of supplying a common structure so that the buyers, suppliers, developers, maintainers, operators, managers and technicians involved in the software development use a common language [2].

The primary lifecycle process is divided into five different main processes involved in software product creation:
- Acquisition
- Supply
- Development
- Operation
- Maintenance

This primary lifecycle covers a very large area, therefore it is necessary to define a scope in order to differentiate the needed details. For the detailed definition of the scope please follow ISO/IEC 12207:2008 [1].

Based on the definition for the primary lifecycle we can treat the software/programs as an independent instances within which testing is a part of each Development, Operation and Maintenance process.

Understanding of Software Testing mostly consists in executing test cases for the software under test and giving the feedback about not working part of the system. This is just a part of the whole testing activity within the software lifecycle process, which includes:
- Planning and controlling.
- Documentation review and static analysis (known as static testing).
- Design and test execution (known as dynamic testing).
- Result checking.
- Evaluation of exit criteria.
- Reporting on test process and system under test.

All the above mentioned activities can be used to improve a system being tested, development or testing process itself.

Testing can have several objectives:
- Finding defects (Error detection).
- Acquire trusting about the quality level.
- Support process of decision-making.
- Avoiding defects [3].
- Validate that what was specified was actually what a user wanted.
- Verify that software behaves as specified.

In other words, validation checks to see if we build what the customer wants/needs, and verification checks if we build that system correctly. Both verification and validation are necessary, but different components testing activity. This is a dynamic process driven by the system adaptation necessity and human needs.

The definition of testing according to the (IEEE, 1059-1993 - IEEE Guide for Software Verification and Validation Plans [4]) standard is that testing is the process of analysing a software item to detect the differences between the existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

3. The costs of software testing

Each part of the software development process involves resources with different budgeting scope, starting from man power and time, finalising on hardware, software and license costs. One of the parts of the process is the software testing. The title of Phil Crosby book says: Quality Is Free [5]. Jim Campenella illustrates a technique for analysing the costs of quality in Principles of Quality Costs [6]. Campenella breaks down those costs as follows:

Cost of Quality = Cost of conformance + Cost of non-conformance

Conformance Costs include Prevention Costs and Appraisal Costs.

Prevention costs include money spent on quality assurance tasks like training, requirements and code reviews, and other activities that promote good software. Appraisal costs include money spent on planning test activities, developing high and low level test cases and executing those test cases once.

Non-conformance costs come in two flavours:
- Internal Failures and
- External Failures.

The costs of internal failure include all expenses that arise when test cases fail the first time they are run, as they often do. A programmer incurs a cost of internal failure while debugging problems found during own unit and component testing.

According to the software development lifecycle, the last, longest and most costly part of it, is the maintenance. The prevailing notion is that software is easy and cheap to change but this is seldom the case. Software maintenance can account for 60 to 80 per cent of the total life cycle cost of a computer program. Most of the expenditures, as much as three fourths of the total maintenance costs, are for enhancements to the code rather than correction of defects.

Developers and managers often believe that a required change is minor and attempt to accomplish it as a quick fix. Insufficient planning, design, impact analysis and testing may lead to increased costs in the future. Over time successive quick fixes may degrade or obscure the original design, making modifications more difficult [7] and finishing in not acceptable, low quality of the system.

Two observations lay the foundation for the enlightened view of testing as an investment. Firstly, like any cost equation in business, we will want to minimize the cost of quality. Secondly, since it is often cheaper to prevent problems than to repair them, if we must repair problems, the internal failures cost less than external ones, especially during the maintenance.

There are, however, projects which cannot afford to create completely new software and gain the problems to get satisfying quality within specific budget. The number of available low and high level test cases is much bigger than the available execution time. The projects come into the problem to select proper test cases.

4. Test case selection

Test case selection is the problem of selecting a small set of tests from a large test suite such that the most defects are revealed when this subset is executed. Test case prioritization is the problem of finding an optimal scheduling of the tests in a test suite so that the number of defects found earlier during testing is maximized.

In the real life test case selection is performed manually by a test analyst or even often by a tester or test manager. Selection itself is based on several decision criteria which are experience based. Selection of the correct test subset is, however, very difficult in case of very large and long lifetime projects where thousands of low and high level tests exist.

There are researchers which try to address this problem using profiling, which is looking at the amount of the executed code by the used test subset. The other ones (e.g. Dickinson – [8]) propose distribution of the profiles with the profile space by using the cluster analysis on the profiles [9].

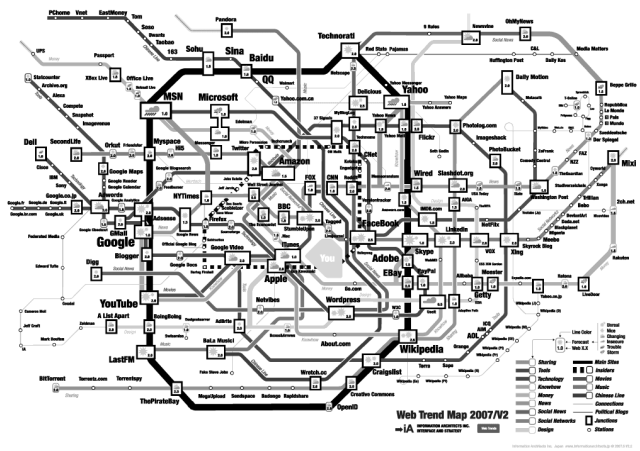
One of the possible ways to perform correct test subset selection could be usage of a visualisation metaphor for available low and high level test cases.

5. Visualisation metaphor

A visualization metaphor is defined as a map establishing the correspondence between concepts and objects of the application under test and a system of some similarities and analogies. This map generates a set of views and a set of methods for communication with visual objects in our case - test cases.

Lev Manovich has said, an important innovation of computers is that they can transform any media into another. This gives us possibility to create a new world of data art that a viewer will find as interesting. It does not matter if the detail is important to the author; the translation of raw data into visual form gives the viewer possibility to get an info which is the most important just for him. Hence, any type of visualization has specific connotations, which may become metaphoric when seen in context of a specific data source. A metaphor in visualization works at the level of structure - it compares the composition of a dataset to a particular conceptual construct, and the choice of any visualization is always a matter of interpretation.

There are many different visualisation metaphors used in the software engineering:
- Mindmaps ([10] presents the 200 most successful websites on the web, ordered by category, proximity, success, popularity and perspective in a mindmap. Apparently, web-sites are connected as they have never been before)

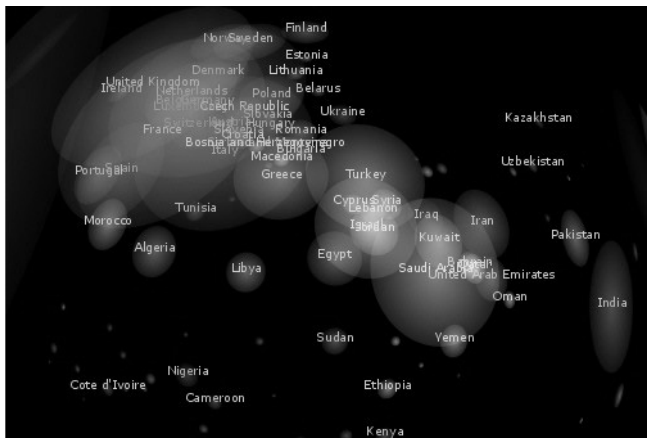


Rys. 1. Mindmap – najlepsze strony web
Fig. 1. Mindmap - Most successful websites on the web

- Voyage ([11] an RSS-feader which displays the latest news in the “gravity area”. News can be zoomed in and out. The navigation is possible with a timeline.

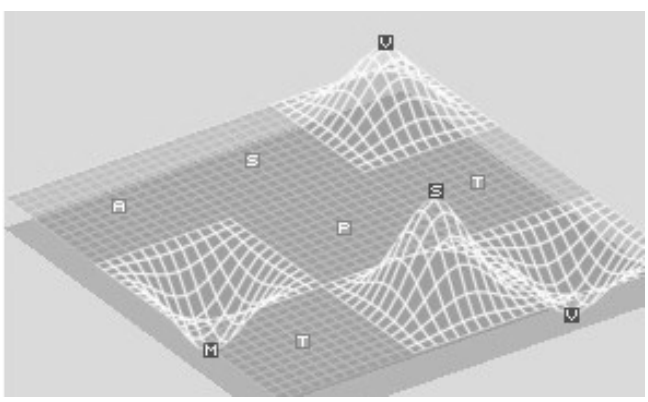
Rys. 2. Wyswietlanie wiadomości RSS Feed
Fig. 2. News displaying RSS Feed

- TED Talk (is a legendary talk of the Swedish professor Hans Rosling, in which he explains a new way of presenting statistical data. His Trendalyzer software, recently acquired by Google, turns complex global trends into lively animations, making decades of data pop. Asian countries, as colourful bubbles, float across the grid — toward better national health and wealth. Animated bell curves representing national income distribution squish and flatten. In Rosling's hands, global trends - life expectancy, child mortality, poverty rates – become clear, intuitive and even playful [12].)



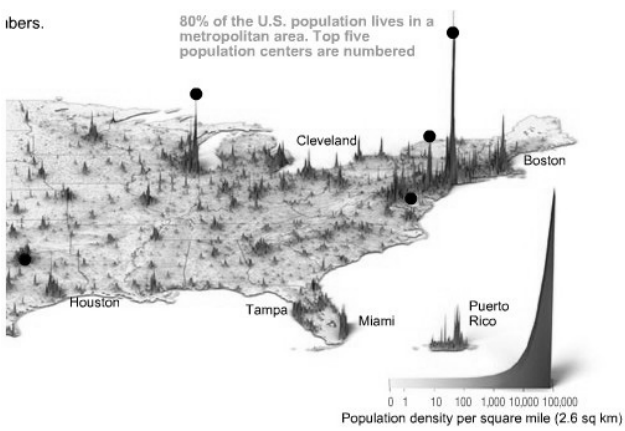
Rys. 3. TED Talk
Fig. 3. TEDTalk

- 2D and 3D Graphs



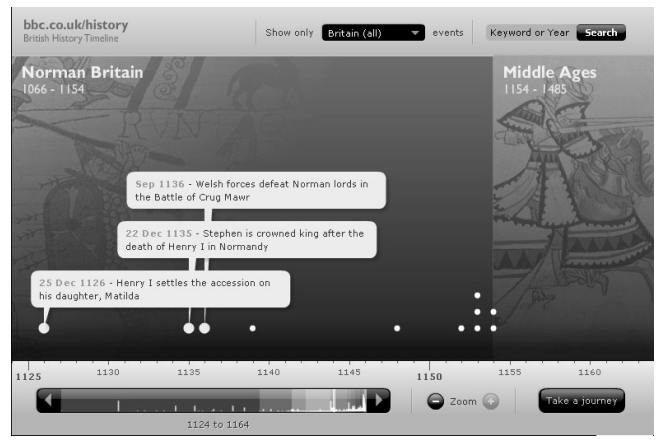
Rys. 4. Grafy 3D
Fig. 4. 3D Graphs

- The Map - (Time Magazine uses visual hills (spikes) to emphasize the density of American population in its map [13].)



Rys. 5. Zagęszczenie populacji USA
Fig. 5. Density of American population

- Interactive history lifetime (presents the history of Great Britain, divided into interactive data blocks. The density of events is displayed on the map [14].)



Rys. 6. Interaktywna historia Anglii
Fig. 6. Interactive history lifetime

- Shape of Song (The Shape of Song is an attempt to answer this seemingly paradoxical question. The custom software in this work draws musical patterns in the form of translucent arches, allowing viewers to see – literally – the shape of any composition available on the Web [15].)
- The Solar System Metaphor (The solar system is an attractive metaphor due to its ability to effectively represent size and inheritance (through orbits and/or rings) [16].)

For all metaphors the used content is presented in a very subjective way. This is due to the necessity to show most important information in a certain data domain. Therefore the choice for a correct metaphor is very dependent on the data domain and its metrics.

6. Test metrics

To be able to perform data visualisation, a certain set of the static and dynamic data will be prepared. Based on the available information for the low and high level test cases we can subtract following metrics for later mapping:

- Amount of low and high level test cases
- Amount of duplicated low and high level test cases
- Amount of obsolete low and high level test cases
- Amount of executed vs. available low and high level test cases
- Percentage test coverage
- Amount of low and high level test cases per Unit/Functionality
- Test Cost (in %) - Cost of testing / total cost *100
- Test Execution Productivity - No of Test cycles executed / Actual Effort for testing
- Acceptance criteria tested - Acceptance criteria tested / total acceptance criteria
- Quality of Testing - No of defects found during Testing/(No of defects found during testing +
- No of acceptance defects found after delivery) *100
- Effectiveness of testing to business - Loss due to problems / total resources processed by the system
- System complaints - Number of third party complaints / number of transactions processed.
- Costs of reviews, inspections and preventive measures
- Costs of test planning and preparation
- Costs of test execution, defect tracking, version and change control
- Costs of diagnostics, debugging and fixing
- Costs of tools and tool support
- Costs of test case library maintenance
- Costs of testing & QA education associated with the product

- Costs of monitoring and oversight by the QA organization (if separate from the development and test organizations).

Dependent on the metric, the metrics are to be taken as a data export through the available API from the test, defect management tool or statistical data taken from the support or test organisation.

The available metrics can be mapped into the chosen visualisation metaphor as:

- Data physical properties (colour, geometry, height mapping, abstract shapes)
- Data granularity (unit cubes, building border or urban block related)
- Effect of Z axis mappings on the image of the city
- Abstraction of data and LOD are key issues
- Resulting "data compatible" urban models are much larger than the original VR urban models

7. Conclusions and future work

In this paper there has been presented a set of visualisation metaphors and test metrics which can be combined to help to perform test archaeology and extended test selection to provide the same test quality with a smaller resources effort. Providing a visual mapping of correspondence data can be a very useful method for analysis within big and long life projects. The result of the paper is, however, very subjective and will be a part of future work in which the comparison between the related metrics and possible usage in the visualisation metaphor will be presented. The selection of proper metrics and its relevance for test selections and related quality of testing would be very useful and can be included.

8. References

[1] ISO, ISO/IEC 12207:2008 - Systems and software engineering - Software life cycle processes - http://www.iso.org/iso/catalogue_detail?csnumber=43447

[2] Wikipedia, Software development process, 2011 - http://en.wikipedia.org/wiki/Software_development_process

[3] ISTQB, Syllabus, 2010 - http://istqb.org/download/attachments/2326555/Foundation+Level+Syllabus+%282010%29.pdf

[4] IEEE, 1059-1993 - IEEE Guide for Software Verification and Validation Plans, 1993 - http://standards.ieee.org/findstds/standard/1059-1993.htm

[5] Phil Crosby, Quality is Free, 1980, ISBN-13: 978-0451621290.

[6] Jack Campenella, Principles of Quality Costs, 1999 , ISBN 978-0-87389-443-2.

[7] Don Patton, eHow.com , 2010 - http://www.ehow.com/about_6460450_software-maintenance-costs.html

[8] Dickinson, W. The Application of Cluster Filtering to operational testing of Software. Doctoral dissertation. Case Western Reserve University, 2001.

[9] David Zaen Leon Cesin, Profile analysis techniques for observation-based software testing, Doctoral dissertation, Case Western Reserve University, 2005.

[10] Informationarchitects.jp, Information Architects, 2011, http://store.informationarchitects.jp/category/web-trend-map

[11] RSSvoyage.com, http://rssvoyage.com/, 2011.

[12] Hans Rosling, TED Talk, http://www.ted.com/talks/, 2011.

[13] Time Magazin, Cover, 30.10.2006, http://www.time.com/time/covers/20061030/where_we_live/

[14] BBC Online, Interactive British History Timeline, 2011, http://www.bbc.co.uk/history/interactive/timelines/british/index.shtml

[15] Shape of Songs, 2011, http://www.turbulence.org/Works/song/gallery/gallery.html

[16] Hamish Graham, Hong Yul Yang, Rebecca Berrigan: A Solar System Metaphor for 3D Visualisation of Object Oriented Software Metrics, Australasian Symposium on Information Visualisation, Christchurch, 2004.

otrzymano / received: 03.11.2011
przyjęto do druku / accepted: 03.01.2012 *artykuł recenzowany / revised paper*

INFORMACJE

Newsletter PAK

Wydawnictwo PAK wysyła drogą e-mailową do osób zainteresowanych Newsletter PAK, w którym są zamieszczane:

- spis treści aktualnego numeru miesięcznika PAK,
- kalendarz imprez branżowych,
- ważniejsze informacje o działalności Wydawnictwa PAK.

Newsletter jest wysyłany co miesiąc do osób, które w jakikolwiek sposób współpracują z Wydawnictwem PAK (autorzy prac opublikowanych w miesięczniku PAK, recenzenci, członkowie Rady Programowej, osoby które zgłosiły chęć otrzymywania Newslettera).

Celem inicjatywy jest umocnienie w środowisku pozycji miesięcznika PAK jako ważnego i aktualnego źródła informacji naukowo-technicznej.

Do newslettera można zapisać się za pośrednictwem:

- strony internetowej: www.pak.info.pl, po dodaniu swojego adresu mailowego do subskrypcji,
- adresu mailowego: wydawnictwo@pak.info.pl, wysyłając swoje zgłoszenie.

Otrzymywanie Newslettera nie powoduje żadnych zobowiązań ze strony adresatów. W każdej chwili można zrezygnować z otrzymywania Newslettera.

Tadeusz SKUBIS
Redaktor naczelny Wydawnictwa PAK