Software Testing and/or Software Monitoring: Differences and Commonalities

Antonia Bertolino
ISTI-CNR, Pisa, Italy

antonia.bertolino@isti.cnr.it

Keynote at Jornadas Sistedes 2014 - Cádiz, 18 September 2014

In the third millennium society software is increasingly pervasive

Unfortunately software may be faulty …

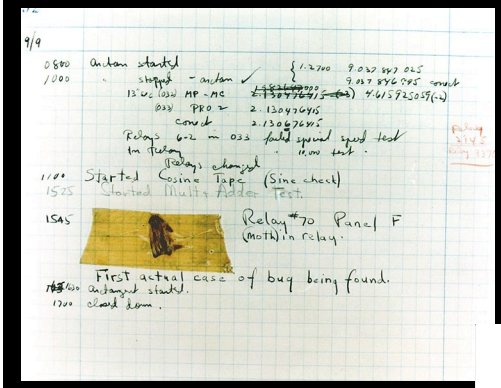
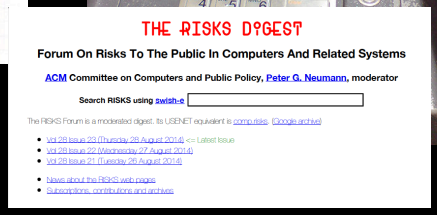


Photo #: NH 96566-KN : The First "Computer Bug", Courtesy of the Naval Surface Warfare Center, Dahlgren, VA., 1988.



© 2014 A. Bertolino: Sw testing and/or sw monitoring

Errare humanum est…

Awareness that introducing faults in software is somehow unavoidable has been acquired very soon …

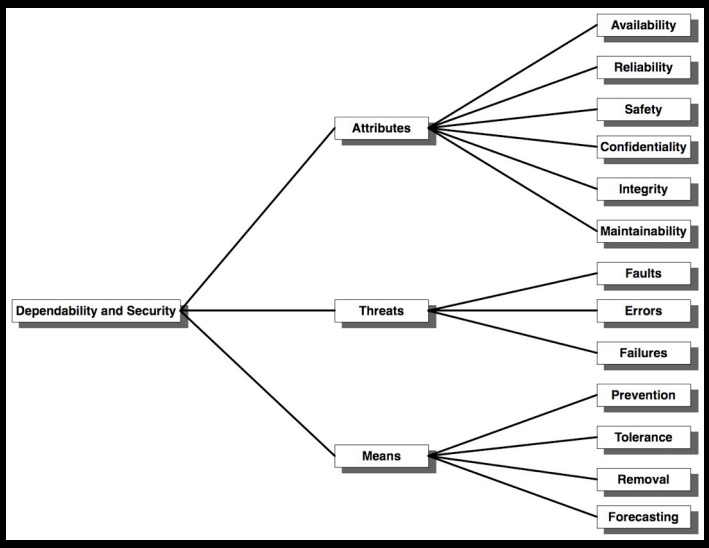
© 2014 A. Bertolino: Sw testing and/or sw monitoring

… perseverare autem diabolicum

… this is why, as with any other engineering artifact, along the production of software we apply approaches and tools for assuring and evaluating its quality (fitness for purpose)

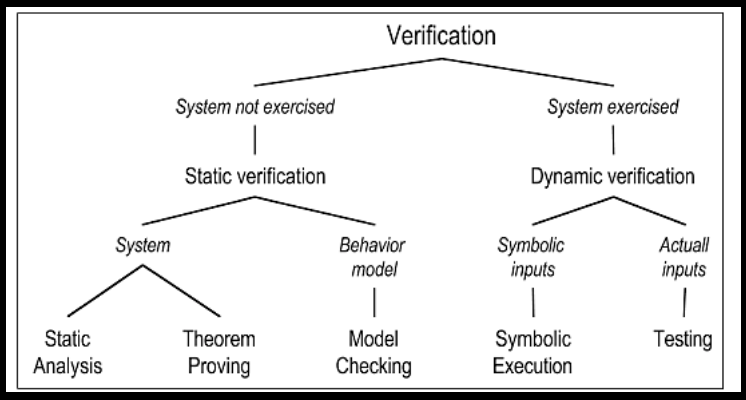
© 2014 A. Bertolino: Sw testing and/or sw monitoring

Potential threats to software dependability and means to deal with them are intensely studied



Avizienis, A; Laprie, J.-C.; Randell, B.; Landwehr, C., "Basic concepts and taxonomy of dependable and secure computing,", *IEEE TDCS,* vol.1, no.1, pp.11-33, 2004.

Many approaches to sw verification



© 2014 A. Bertolino: Sw testing and/or sw monitoring

Mixing different approaches

- Much research effort since the early days of SE has been spent to assess and compare the effectiveness of V&V approaches

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Programming Techniques
S. L. Graham, R. L. Rivest Editors

A Controlled Experiment in Program Testing and Code Walkthroughs/ Inspections

Glenford J. Myers
IBM Systems Research Institute

This paper describes an experiment in program testing, employing 59 highly experienced data processing professionals using seven methods to test a small PL/I program. The results show that the popular code walkthrough/inspection method was as effective as other computer-based methods in finding errors and that the most effective methods (in terms of errors found and cost) employed pairs of subjects who tested the program independently and then pooled their findings. The study also shows that there is a tremendous amount of variability among subjects and that the ability to detect certain types of errors varies from method to method.

Key Words and Phrases: software reliability, program verification, debugging, testing, code walkthroughs, code inspections, personnel selection

gies and tools over the last few years has greatly outpaced our efforts to experiment with alternatives in order to analyze their differences and assess their benefits. One frequently finds statements in the literature such as "Method *A* was found to increase productivity by 39 percent over method *B*," but when one attempts to analyze the underlying evidence, one normally sees that methods *A* and *B* were used on two different projects with different people, different objectives, and literally hundreds of other differing characteristics. In other words, the statement is derived in many instances from a completely uncontrolled environment, rendering it at best misleading.

Unfortunately, controlled experiments in the area of software development tend to be costly and time consuming. Researchers have attempted to circumvent this by performing experiments using "cheap" labor (e.g. the captive audience in an introductory computer science course). However, although much of this work has provided useful insights, there exists the question of whether one can extrapolate, to a typical industrial environment, experimental results obtained from trainee programmers or programmers with only a few years of experience.

A third problem is the lack of data on program-testing methods, which is particularly alarming in light of the fact that program testing consumes approximately half of most organizations' development budgets. For instance, the idea of code walkthroughs or inspections [1] (a semiformal, noncomputer-based method of program testing performed by a team of individuals) has become popular, but no controlled data are available establishing that this method is more effective than traditional computer-based testing. As another example, there are differences of opinion as to whether a program

Commun. ACM 21, 9 (September 1978)

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Comparing the Effectiveness of Software Testing Strategies

VICTOR R. BASILI, SENIOR MEMBER, IEEE, AND RICHARD W. SELBY, MEMBER, IEEE

Abstract—**This study applies an experimentation methodology to compare three state-of-the-practice software testing techniques: a) code reading by stepwise abstraction, b) functional testing using equivalence partitioning and boundary value analysis, and c) structural testing using 100 percent statement coverage criteria. The study compares the strategies in three aspects of software testing: fault detection effectiveness, fault detection cost, and classes of faults detected. Thirty-two professional programmers and 42 advanced students applied the three techniques to four unit-sized programs in a fractional factorial experimental design. The major results of this study are the following. 1) With the professional programmers, code reading detected more software faults and had a higher fault detection rate than did functional or structural testing, while functional testing detected more faults than did structural testing, but functional and structural testing were not different in fault detection rate. 2) In one advanced student subject group, code reading and functional testing were not different in faults found, but were both superior to structural testing, while in the other advanced student subject group there was no difference among the techniques. 3) With the advanced student subjects, the three techniques were not different in fault detection rate. 4) Number of faults observed, fault detection rate, and total effort in detection depended on the type of software tested. 5) Code reading detected more interface faults than did the other methods. 6) Functional testing detected more control faults than did the other methods. 7) When asked to estimate the percentage of faults detected, code readers gave the most accurate estimates while functional testers gave the least accurate estimates.**

subjects that had a wide range of professional experience. This controlled study is intended to evaluate different testing methods that are actually used by software developers, "state-of-the-practice" methods, as opposed to state-of-the-art techniques.

This work is intended to characterize how testing effectiveness relates to several factors: testing technique, software type, fault type, tester experience, and any interactions among these factors. The study presented extends previous work by incorporating different testing techniques and a greater number of persons and programs, while broadening the scope of issues examined and adding statistical significance to the conclusions.

There are multiple perspectives from which to view empirical studies of software development techniques, including the study presented in this paper.

- *Experimenter*—An experimenter may view the study as a demonstration of how a software development technique (or methodology, tool, etc.) can be empirically evaluated. Experimenters may examine the work as an example application of a particular experimentation methodology that may be reused in future studies.

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A systematic mapping study on the combination of static and dynamic quality assurance techniques

Frank Elberzhager [a,*], Jürgen Münch [b,a], Vi Tran Ngoc Nha [c]

[a] Fraunhofer Institute for Experimental Software Engineering (IESE), Fraunhofer Platz 1, 67663 Kaiserslautern, Germany
[b] University of Helsinki, P.O. Box 68, Gustaf Hällströmin katu 2b, 00014 Helsinki, Finland
[c] University of Kaiserslautern, Gottlieb-Daimler Str., 67663 Kaiserslautern, Germany

ARTICLE INFO

Article history:
Received 7 January 2011
Received in revised form 31 May 2011
Accepted 3 June 2011
Available online 16 June 2011

Keywords:
Combination
Dynamic quality assurance
Inspection
Static quality assurance
Systematic mapping study
Testing

ABSTRACT

Context: A lot of different quality assurance techniques exist to ensure high quality products. However, most often they are applied in isolation. A systematic combination of different static and dynamic quality assurance techniques promises to exploit synergy effects, such as higher defect detection rates or reduced quality assurance costs. However, a systematic overview of such combinations and reported evidence about achieving synergy effects with such kinds of combinations is missing.
Objective: The main goal of this article is the classification and thematic analysis of existing approaches that combine different static and dynamic quality assurance technique, including reported effects, characteristics, and constraints. The result is an overview of existing approaches and a suitable basis for identifying future research directions.
Method: A systematic mapping study was performed by two researchers, focusing on four databases with an initial result set of 2498 articles, covering articles published between 1985 and 2010.
Results: In total, 54 articles were selected and classified according to multiple criteria. The two main dimensions of a combination are integration (i.e., the output of one quality assurance technique is used for the second one) and compilation (i.e., different quality assurance techniques are applied to ensure a common goal, but in isolation). The combination of static and dynamic analyses is one of the most common approaches and usually conducted in an integrated manner. With respect to the combination of inspection and testing techniques, this is done more often in a compiled way than in an integrated way.
Conclusion: The results show an increased interest in this topic in recent years, especially with respect to the integration of static and dynamic analyses. Inspection and testing techniques are currently mostly performed in an isolated manner. The integration of inspection and testing techniques is a promising research direction for the exploitation of additional synergy effects.

© 2011 Elsevier B.V. All rights reserved.

Mixing different approaches

- Much research effort since the early days of SE has been spent to assess and compare the effectiveness of V&V approaches
- It is now widely acknowledged that the different V&V approaches yield differing effectiveness and target different types of faults, and to get the most benefits they should be used in combination

© 2014 A. Bertolino: Sw testing and/or sw monitoring

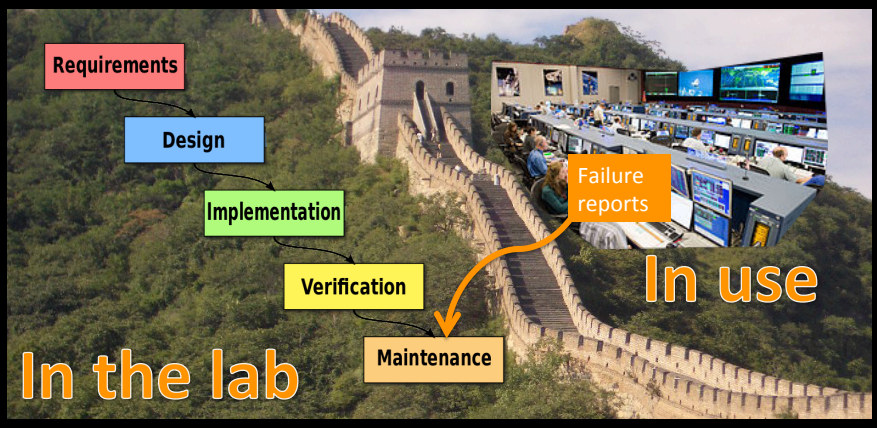
Mixing different approaches

- Much research effort since the early days of SE has been spent to assess and compare the effectiveness of V&V approaches
- It is now widely acknowledged that the different V&V approaches yield differing benefits and target different types of faults. To reap most benefits they should be used in combination

However, so far this has mostly addressed the combination of V&V approaches within the software production

© 2014 A. Bertolino: Sw testing and/or sw monitoring

The traditional software life cycle



In early SE a neat separation between product development and usage was considered.

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Breaking the separation between development (lab) and operation (real world)

e.g., see Luciano Baresi and Carlo Ghezzi. 2010. The disappearing boundary between development-time and run-time. In *Proceedings of the FSE/SDP workshop on Future of software engineering research* (FoSER '10)

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Emerging trends

Software intensive systems have become the foundation of modern technology

- Software content in virtually every product and service will continue to grow—in some cases dramatically
- Software must be demonstrably safe, secure, and reliable
- Requirements continuously emerge as systems dynamically evolve

[R. Pressman, "Emerging trends in software engineering", JASST09]

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Emerging trends

- Unprecedented pace of change:
  - Requirements change continuously
  - Blurring boundaries between the system and the external world
  - Both components and the overall structure can change
  - No single organization is in control of the whole system
  - Self-healing and self-adaptation

"We must shift from the prevailing synchronous approach to distributed programming to a fundamentally more delay-tolerant and failure resilient asynchronous programming approach. Global behaviors emerge by asynchronous combinations of individual behaviors, and bindings and compositions change dynamically."

[L. Baresi, E. Di Nitto, C. Ghezzi, "Toward Open-World Software: Issue and Challenges," Computer , vol.39, no.10, 2006]

QoS contracts

- More and more systems are formed by the cooperation among autonomous software services, and their global behaviour emerges as a result.
- For such kind of systems we need to verify that each involved entity complies with its "contract", which is impossible prior to execution.

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Challenges

- System assembled dynamically
- Reference specification of expected/correct operation not a-priori available
- Specifications are learnt/inferred, thus they can be incomplete, unstable, uncertain
- Special emphasis on run-time assessment
- Assessment activities must accommodate change (and must be adaptable themself)

© 2014 A. Bertolino: Sw testing and/or sw monitoring

What about V&V?

- It is our thesis that with the blurring boundary between production and use (pre and post deployment) we should try to get the most mutual benefits from combining pre and post deployment V&V

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Validation @ runtime

- All such trends go towards the direction of moving V&V to the field after deployment
- Relies on sensing what is happening and on timely collecting relevant information

→ To do this, we need to monitor systems behaviour

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Two generic approaches to sw verification

© 2014 A. Bertolino: Sw testing and/or sw monitoring

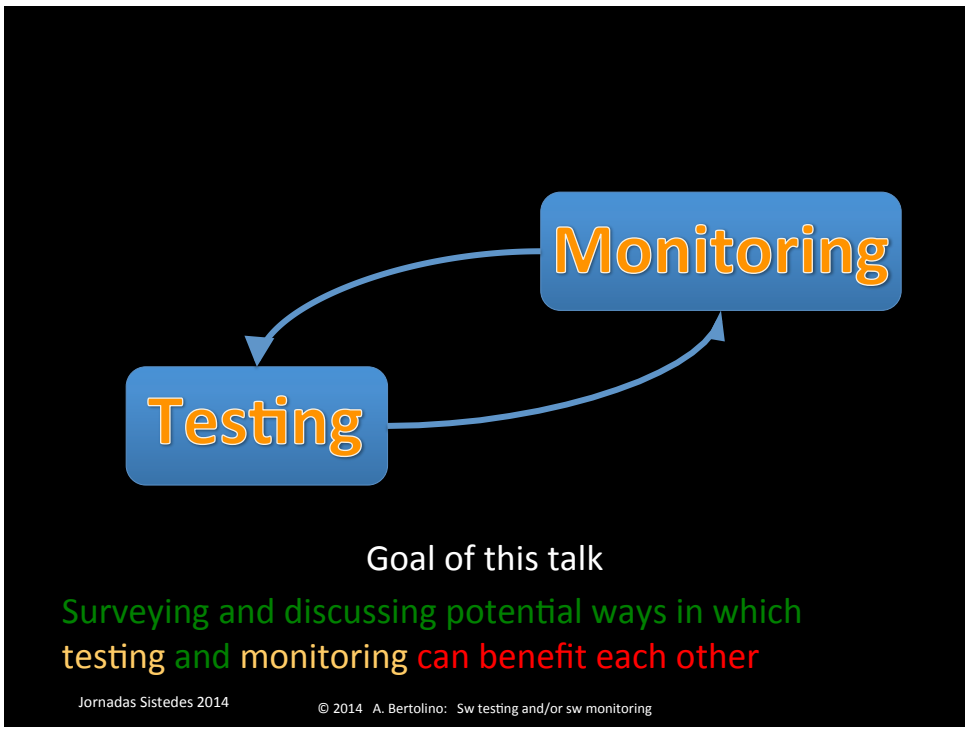
Two generic approaches to sw verification

Monitoring

Testing

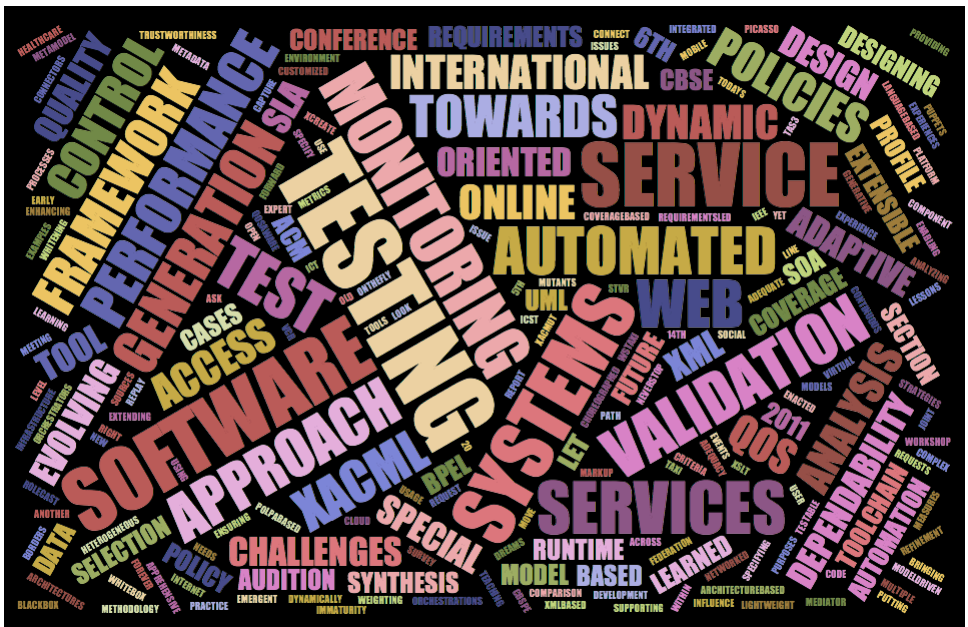
Research and solutions for testing (in the lab) and monitoring (in real usage) have taken quite separate tracks

© 2014 A. Bertolino: Sw testing and/or sw monitoring



Why testing and monitoring?

© 2014 A. Bertolino: Sw testing and/or sw monitoring



A word cloud from titles of most recent papers of mine

© 2014 A. Bertolino: Sw testing and/or sw monitoring

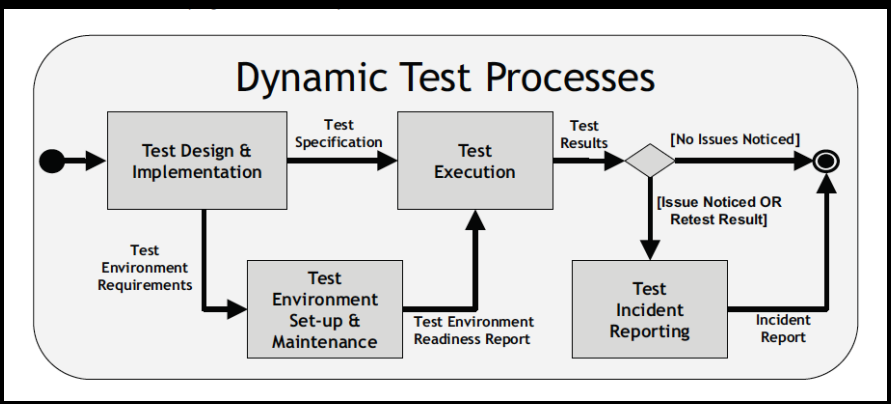
Agenda

- In the rest of the talk I will:
  i. Make a shot recap of main concepts and research challenges in software testing and software monitoring
  ii. Overview a few proposals in the literature for combining methods or mutually benefit from each other, while…
  iii. … attempting a taxonomy of testing+monitoring research

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Software Testing

Software testing is a major activity within a product Verification and Validation. Testing is a complex process including more than "just" running test items, it also requires several preparation and follow-up activities.



ISO/IEC/IEEE 29119 Standard for Software and Systems Engineering--Software Testing--Part 1: Concepts and Definitions (2013)

All-inclusive definition

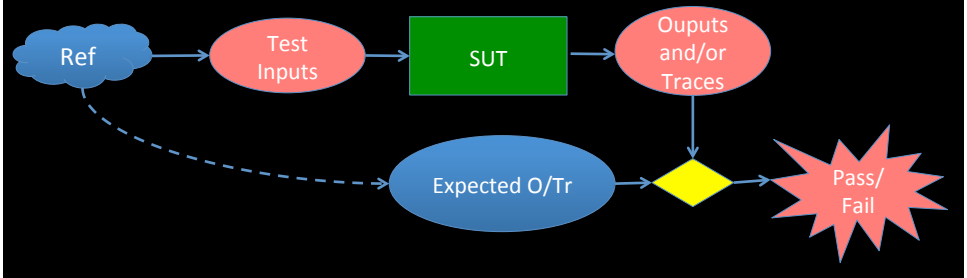
Software testing consists of

- the dynamic verification of the behavior of a program
- on a finite set of test cases
- suitably selected from the (in practice infinite) input domain
- against the expected behavior

A. Bertolino, "Software Testing", Chapter 5, in
IEEE Guide to the SWEBOK (2001, stoneman v.), www.swebok.org

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Software testing conceptual flow



© 2014 A. Bertolino: Sw testing and/or sw monitoring

Software Monitoring

- the process of dynamic collection, interpretation, and presentation of information concerning objects or software processes under scrutiny

J. Joyce, G. Lomow, K. Slind, and B. Unger. Monitoring distributed systems. ACM Trans. Comput. Syst., 5(2):121–150, 1987

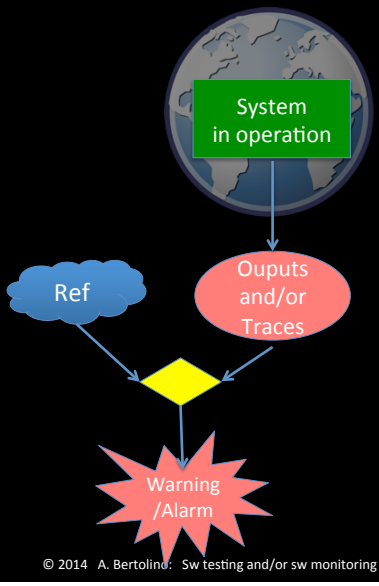
© 2014 A. Bertolino: Sw testing and/or sw monitoring

In other words…

- A monitor is a system that observes the spontaneous behavior of a system and, given a specification of desired properties, it checks that such properties hold for the given execution

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Software Monitor conceptual flow



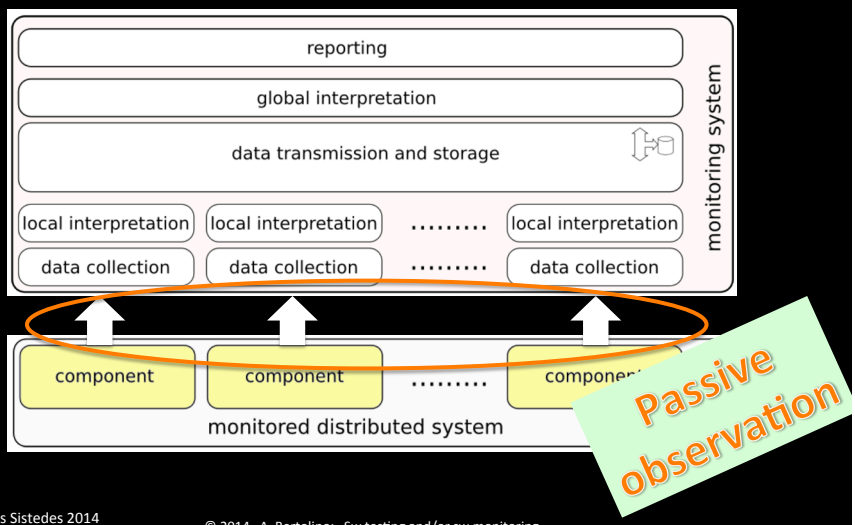
© 2014 A. Bertolino: Sw testing and/or sw monitoring

An over-loaded term

- Large (but fractioned) body of research, carried out over decades
- Different authors use the term “monitoring" to indicate different things/aspects
- A monitoring system is in fact an assembly of different pieces dealing with different concerns

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Components of a monitoring systems



© 2014 A. Bertolino: Sw testing and/or sw monitoring

A useful reference

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 30, NO. 12, DECEMBER 2004

A Taxonomy and Catalog of Runtime Software-Fault Monitoring Tools

Nelly Delgado, *Student Member, IEEE*, Ann Quiroz Gates, *Member, IEEE Computer Society*, and Steve Roach, *Member, IEEE Computer Society*

Abstract—A goal of runtime software-fault monitoring is to observe software behavior to determine whether it complies with its intended behavior. Monitoring allows one to analyze and recover from detected faults, providing additional defense against catastrophic failure. Although runtime monitoring has been in use for over 30 years, there is renewed interest in its application to fault detection and recovery, largely because of the increasing complexity and ubiquitous nature of software systems. This paper presents a taxonomy that developers and researchers can use to analyze and differentiate recent developments in runtime software fault-monitoring approaches. The taxonomy categorizes the various runtime monitoring research by classifying the elements that are considered essential for building a monitoring system, i.e., the specification language used to define properties; the monitoring mechanism that oversees the program's execution; and the event handler that captures and communicates monitoring results. After describing the taxonomy, the paper presents the classification of the software-fault monitoring systems described in the literature.

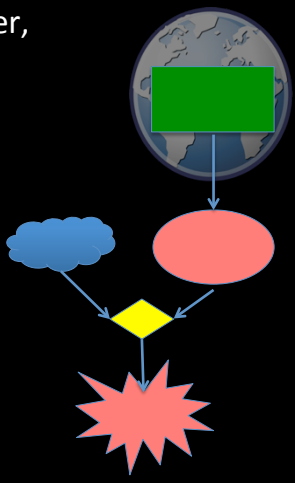
Index Terms—Assertion checkers, runtime monitors, specification, specification language, survey, software/program verification.

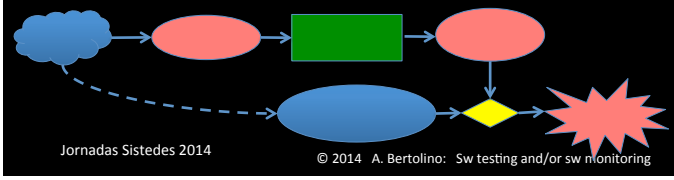
© 2014 A. Bertolino: Sw testing and/or sw monitoring

Testing vs. Monitoring

In a nutshell, both activities observe some executions in order to detect issues or assess sw quality (wrt some specified property). However, major differences/complementarities can be noticed concerning:

- What executions are observed
- What properties are checked
- The context, i.e., when/where/how the observation is performed

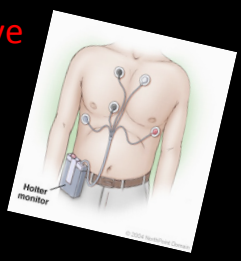




© 2014 A. Bertolino: Sw testing and/or sw monitoring

Testing vs. Monitoring: what executions?

Active vs. Passive



- In ST, "generally" the test cases are selected based on various criteria, including strategy, testing stage, fault model, available tools and resources
- In SM, system executions are observed ideally without interference: the aim is to keep under control the system spontaneous behaviour

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Pros and Cons of

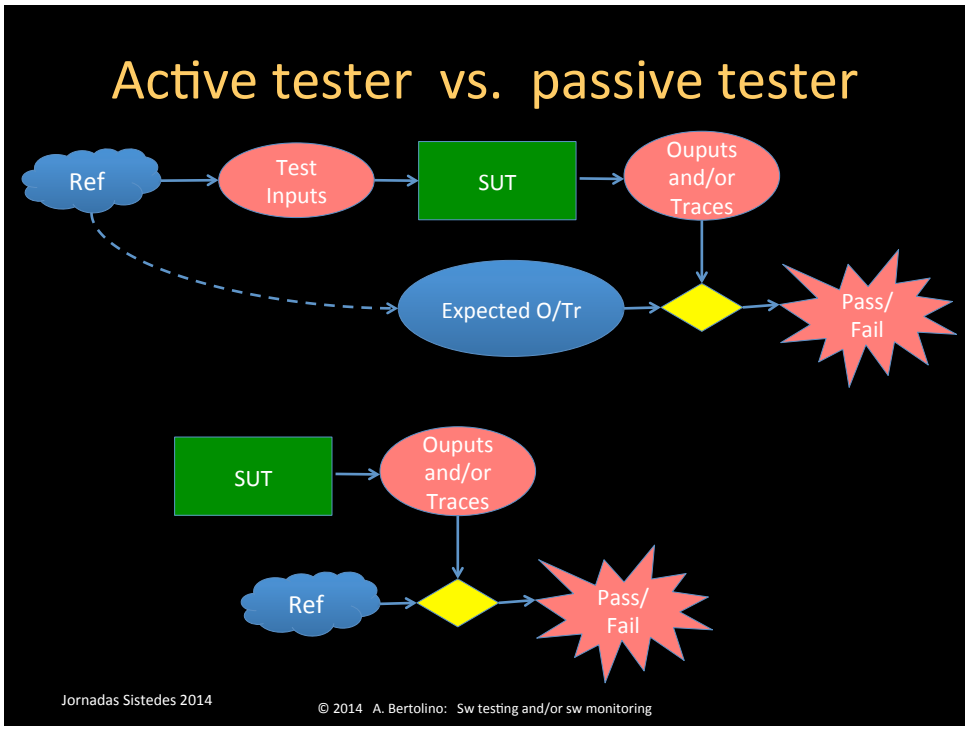
Actively selecting test cases	Passively observing what happens
+ can detect possible issues in advance	- a problem is found only after it happened
- but the identification of "effective" test cases can be quite complex and costly	+ no cost involved in selection
+ negative tests	- no robustness or stress check
- for some test approaches, determining the test oracle can be very difficult	+ oracle somewhat implicit in the monitored property

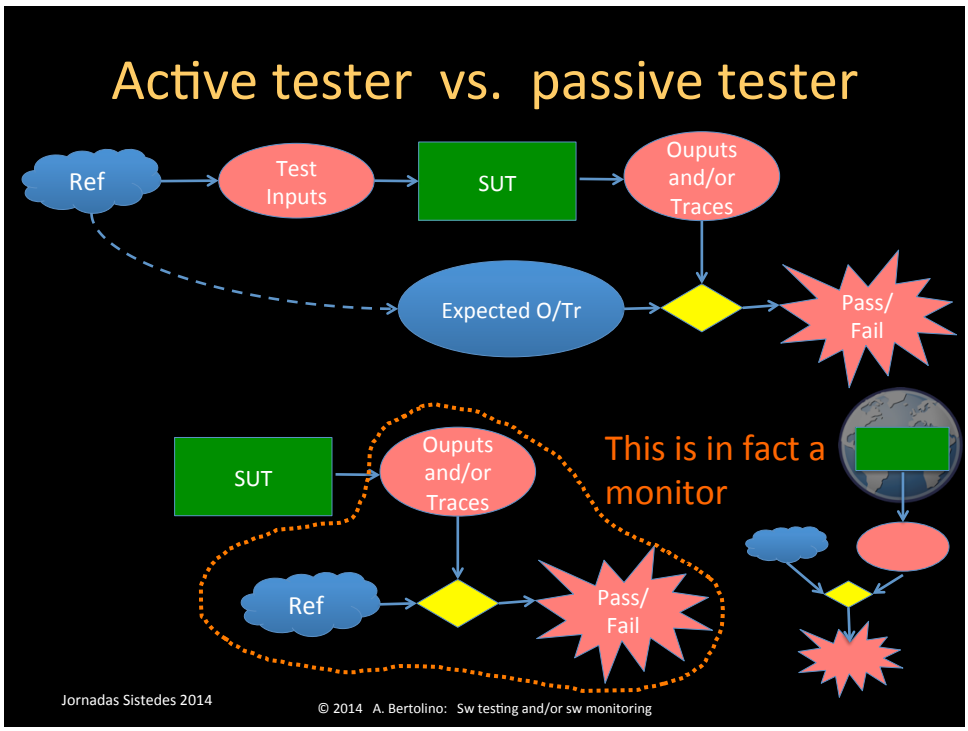
© 2014 A. Bertolino: Sw testing and/or sw monitoring

Passive testing

- In contrast to more common "active testing", passive testing consists of passively observing a system in operation without interfering with it in order to detect faults
- The approach has been mostly investigated in the field of networks for conformance testing of network protocols

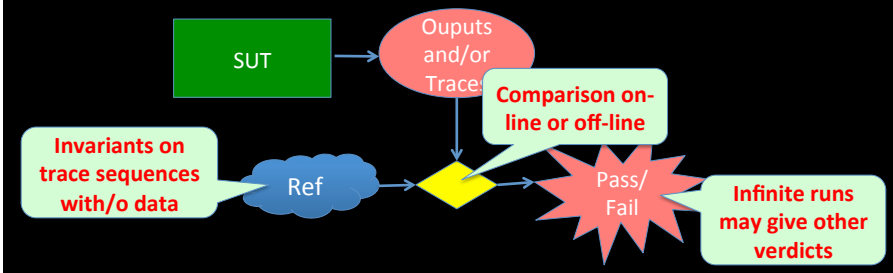
© 2014 A. Bertolino: Sw testing and/or sw monitoring


Active tester vs. passive tester
Ref
Test Inputs
SUT
Ouputs and/or Traces
Expected O/Tr
Pass/ Fail
SUT
Ouputs and/or Traces
Ref
Pass/ Fail
Jornadas Sistedes 2014
© 2014 A. Bertolino: Sw testing and/or sw monitoring


Active tester vs. passive tester
Ref
Test Inputs
SUT
Ouputs and/or Traces
Expected O/Tr
Pass/ Fail
SUT
Ouputs and/or Traces
This is in fact a monitor
Ref
Pass/ Fail
Jornadas Sistedes 2014
© 2014 A. Bertolino: Sw testing and/or sw monitoring

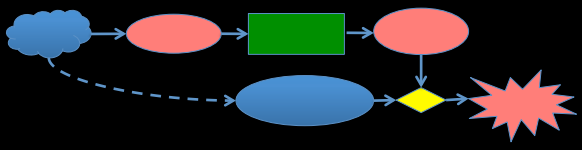
Active tester, passive tester, monitor

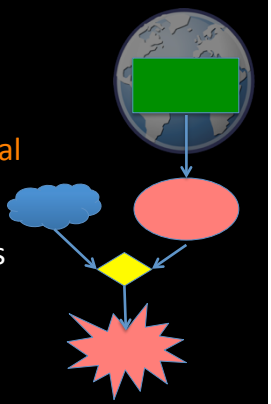
- Emphasis of research in passive testing is in formal approaches to express the properties to be checked and in making the monitor more effective in detecting failures



© 2014 A. Bertolino: Sw testing and/or sw monitoring

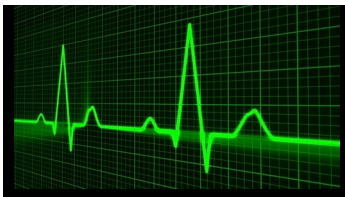
Testing vs. Monitoring: what properties?





- In both ST and SM functional and non-functional properties can be checked
- In ST functional somehow prevails, while research in testing of non-functional properties stays behind
- In SM non-functional is more common domain
  - This descends naturally from the difficulty of assessing performance, reliability, and security properties in a context different from the actual one

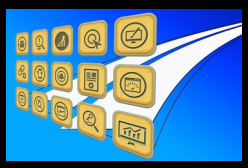
© 2014 A. Bertolino: Sw testing and/or sw monitoring



Complex event processing

The concept of event processing is older than computing itself: reacting to events is a natural part of our society and economy.

Event processing has recently emerged as a substantial new field of software engineering.



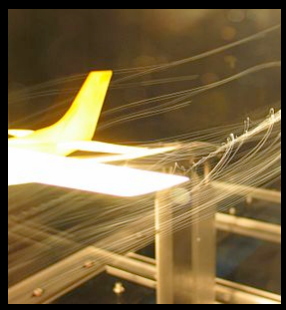
Complex event processors (CEP) combine information from many data points to generate new insights, and use algorithms and rules to process event data received from one or more sources during some time period. Complex events are produced as summary level facts that are put in context to identify threats and opportunity situations.

Ref: The event processing manifesto
Dagstuhl Seminar on Event Processing 2010

Testing vs. Monitoring: the context

Reproduction vs. genuine, off-line vs. "in-process"

- In ST, the test execution requires a proper test environment, i.e. all *facilities, hardware, software, firmware, procedures, and documentation intended for or used* to perform the tests
  - Setting up the test environment may be costly and complex



© 2014 A. Bertolino: Sw testing and/or sw monitoring

Testing vs. Monitoring: the context

Reproduction vs. genuine, off-line vs. "in-process"

- In ST, the test execution requires a proper test environment, i.e. all *facilities, hardware, software, firmware, procedures, and documentation intended for or used* to perform the tests
  - Setting up the test environment may be costly and complex
- On the other hand, lab tests are safe, i.e. no effect in real world

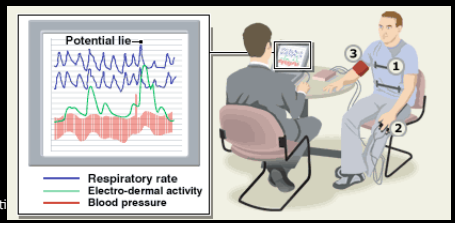
© 2014 A. Bertolino: Sw testing and/or sw monitoring

Testing vs. Monitoring: the context

Reproduction vs. genuine, off-line vs. "in-process"

- In ST, the test execution requires a proper test environment, i.e. all *facilities, hardware, software, firmware, procedures, and documentation intended for or used* to perform the tests
  - Setting up the test environment may be costly and complex
- On the other hand, lab tests are safe, i.e. no effect in real world

- In SM, the system is verified in operation: no need to reproduce environment, however costs are still implied for instrumenting or annotating the system so to allow observation
- Probe effect: The monitor could interfere with system behaviour



© 2014 A. Bertolino: Sw testi

How can ST and SM benefit each other?

- Traditionally separate research domains, but several recent proposals for positively contaminating the two activities

Let's see a sample from literature

© 2014 A. Bertolino: Sw testing and/or sw monitoring

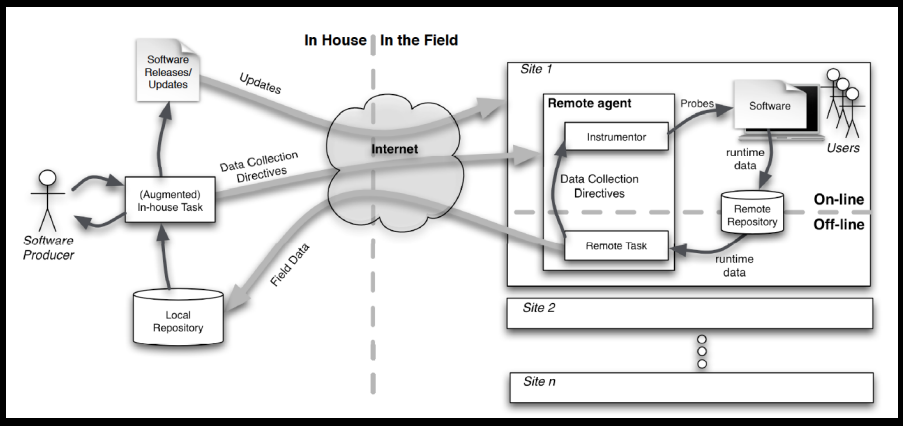
Collecting and mining field data for improving ST process

- An obvious synergy between ST and SM is collecting data about the runtime behavior of deployed programs while they are in use, and use such data for improving the ST process, including:
  - Bug statistics
  - Fault localization and repair
  - Augmenting a test suite
  - Reproducing real failures in the lab

Alessandro Orso, Monitoring, analysis, and testing of deployed software. In Proceedings *of* FoSER '10. ACM, 2010, pp. 263-268.

© 2014 A. Bertolino: Sw testing and/or sw monitoring

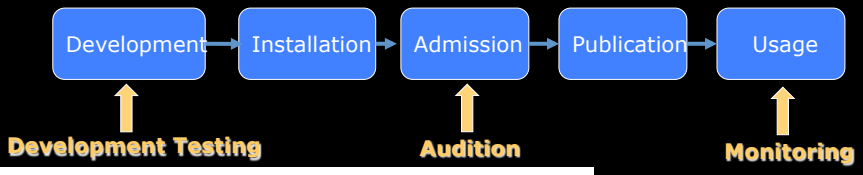
Collecting and mining field data for improving ST process

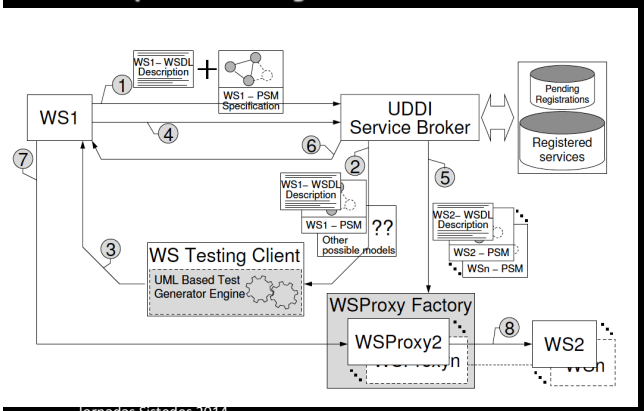


Alessandro Orso, Monitoring, analysis, and testing of deployed software. In Proceedings *of* FoSER '10. ACM, 2010, pp. 263-268.

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Moving testing to the field



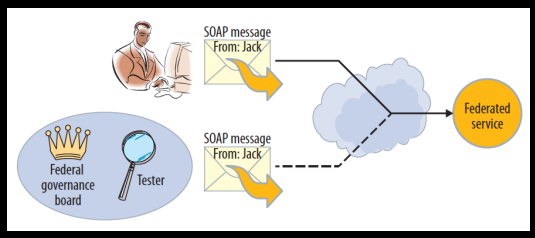


A. Bertolino, A. Polini: The Audition Framework for Testing Web Services Interoperability. EUROMICRO-SEAA 2005

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Testing in the field

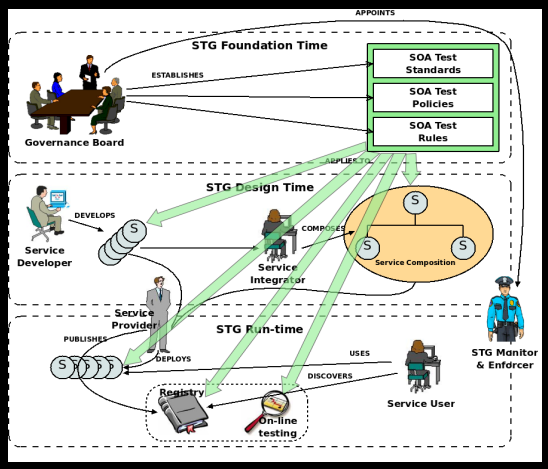
- For dynamic systems whose behaviour, changes and user profile are hard to predict and simulate
- Differently from monitoring, online testing is proactive:
  - Big challenges in terms of costs, acceptance, side effects…



Antonia Bertolino, Guglielmo De Angelis, Andrea Polini: (role)CAST: A Framework for On-line Service Testing. WEBIST 2011

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Context for testing in the field



- M. Ali, F. De Angelis, D. Fanì, A. Bertolino, G. De Angelis, A. Polini: An Extensible Framework for Online Testing of Choreographed Services. IEEE Computer 2014
- A. Bertolino, G. De Angelis, S. Kellomaki, A. Polini: Enhancing Service Federation Trustworthiness through Online Testing. IEEE Computer 2012
- A. Bertolino, A. Polini: SOA Test Governance: Enabling Service Integration Testing across Organization and Technology Borders. ICST Workshops 2009: 277-286

© 2014 A. Bertolino: Sw testing and/or sw monitoring

And viceversa, moving monitoring in the lab

- Runtime verification is a type of monitor, with an emphasis of the research on the specification of properties to be monitored.
- Runtime verification monitors programs for testing purposes, whereby the emphasis is on collecting the traces rather than on the actual usage context
- For the purpose of this talk RV can be seen as monitoring performed in the lab

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Runtime verification

- A relatively new discipline (RV series started in 2001)
- Introduced as a lightweight verification approach, to complement heavyweight ones such as theorem proving and model checking.

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Runtime verification

- Runtime verification is the discipline of computer science that deals with the study, development, and application of those verification techniques that allow checking whether a run of a system under scrutiny satisfies or violates a given correctness property.

Leucker, Martin, and Christian Schallhart. "A brief account of runtime verification." The Journal of Logic and Algebraic Programming 78.5 (2009): 293-303.

© 2014 A. Bertolino: Sw testing and/or sw monitoring

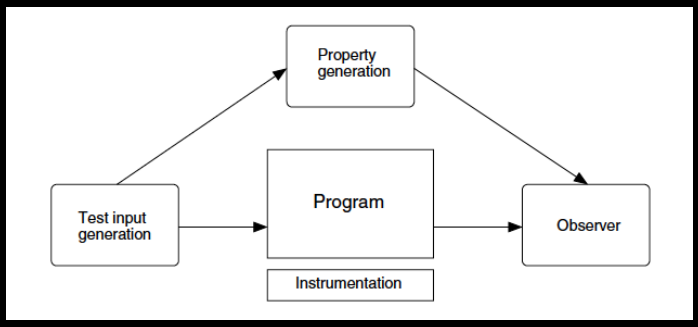
Monitoring as seen by RV

- A monitor is a device that reads a finite trace and yields a certain verdict
- For a monitor to be ideally suited for RV, it should adhere to the two maxims impartiality and anticipation.
  - Impartiality requires that a finite trace is not evaluated to true or, respectively false, if there still exists an (infinite) continuation leading to another verdict.
  - Anticipation requires that once every (infinite) continuation of a finite trace leads to the same verdict, then the finite trace evaluates to this very same verdict.

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Test process automation

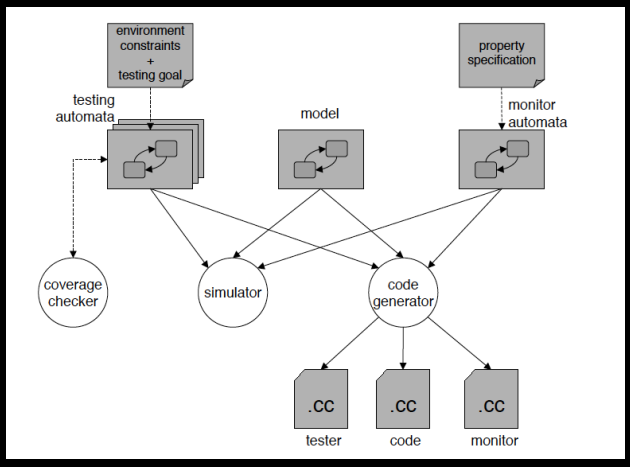
- By exploiting the instrumenter and the trace analyzer of a monitor a test session could be fully automated



Cyrille Artho et al. Experiments with test case generation and runtime analysis. ASM'03.

© 2014 A. Bertolino: Sw testing and/or sw monitoring

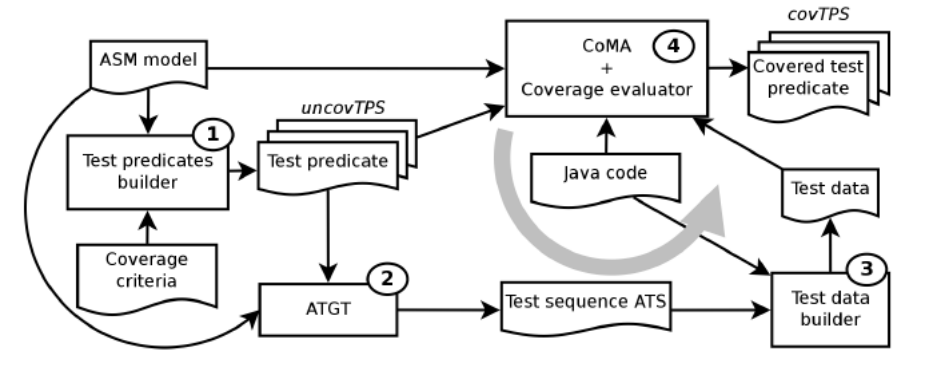
Example of a complete framework for MBT and RV of model simulator and/or code



Li Tan, Jesung Kim, Insup Lee, Testing and Monitoring Model-based Generated Program, ENTCS, 89 (2), 2003, 128-148

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Monitoring can be exploited as a test oracle



Paolo Arcaini, Angelo Gargantini and Elvinia Riccobene: "Combining Model-Based Testing and Runtime Monitoring for Program Testing in the Presence of Nondeterminism, ICST workshop A-MOST 2013

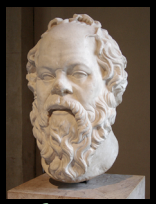
© 2014 A. Bertolino: Sw testing and/or sw monitoring

Using test coverage criteria for improving monitor observation capability

Even using different approaches, they all share one limitation: they can only passively observe and elaborate what happens

Sometime it can be important also to know what does NOT happen, e.g. …

- some service has not been invoked for long time, or
- some usage profile is different from expectations …



"Wisdom is knowing how little we know"

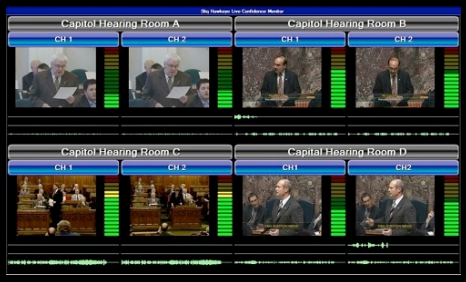
[Socrates, Athens, 469–399 BC]

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Using test coverage criteria for improving monitor observation capability

Empower the monitor with the knowledge to decide if what is being observed is adequate

We introduce the notion of a monitoring coverage criterion



© 2014 A. Bertolino: Sw testing and/or sw monitoring

Monitoring coverage

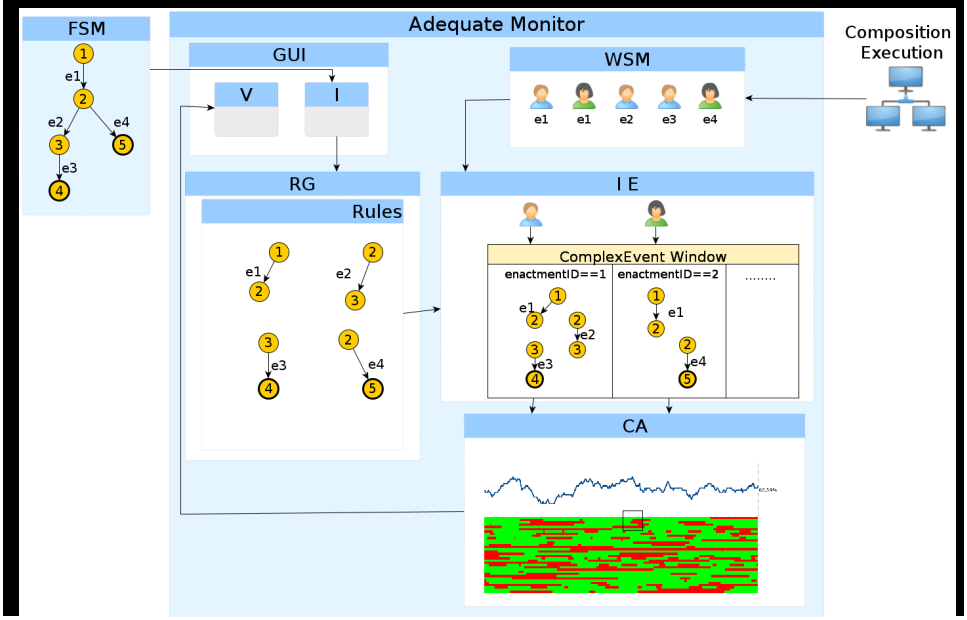
✧ Similarly to test coverage criteria, we define a set of entities that a monitor should cover

✧ Differently from test coverage:

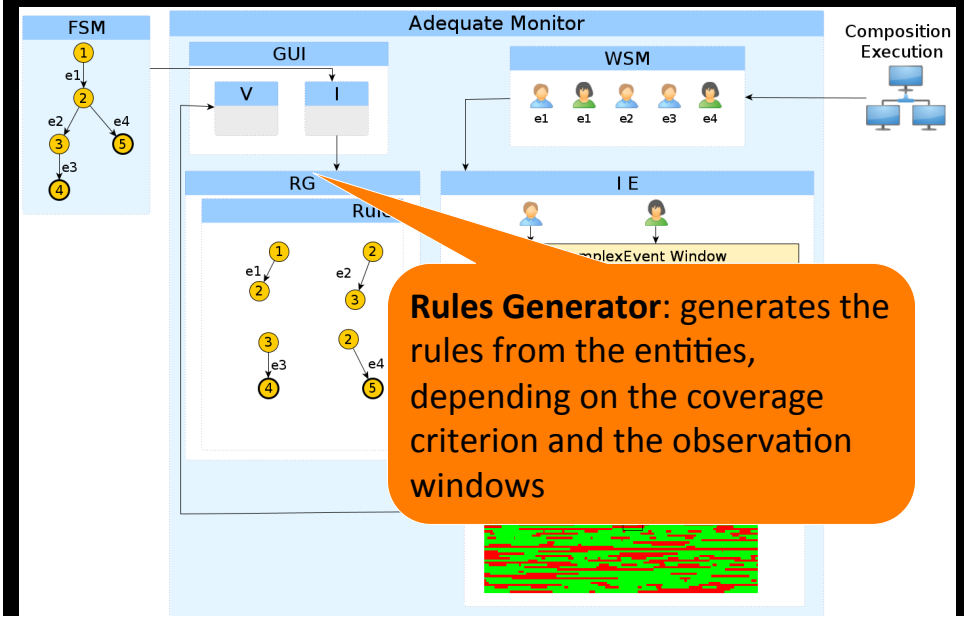
- we need to set how long to wait for observing an entity
  - ⇒ Observation window δ
- monitor adequacy can increase or decrease along time

© 2014 A. Bertolino: Sw testing and/or sw monitoring

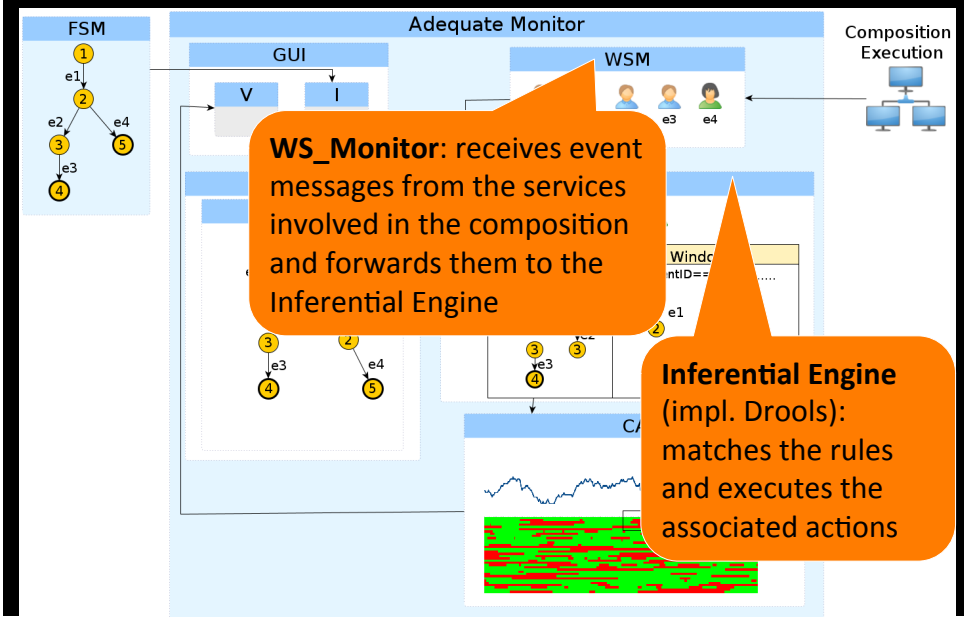
Adequate monitor architecture



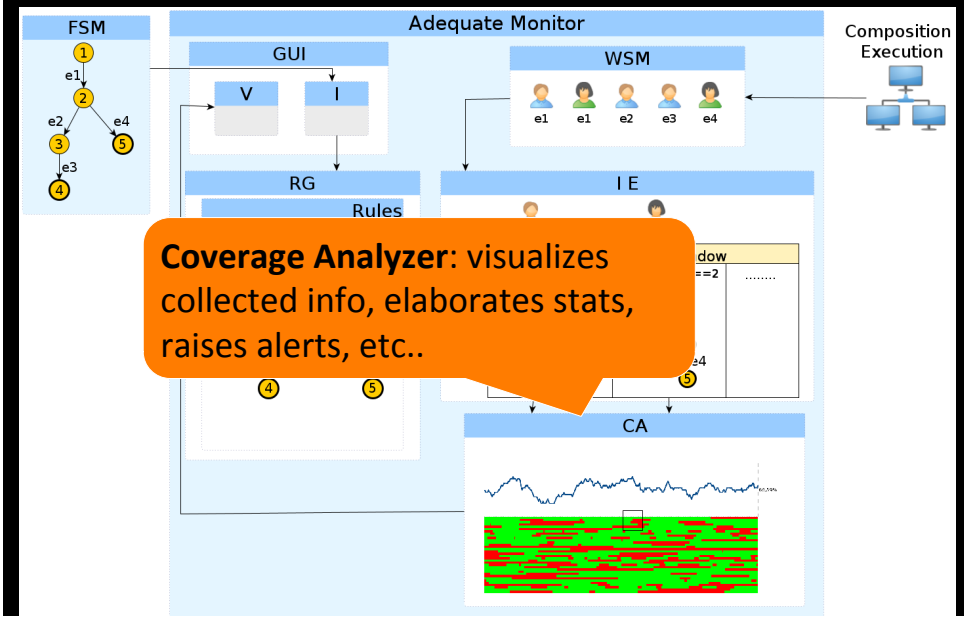
Adequate monitor architecture

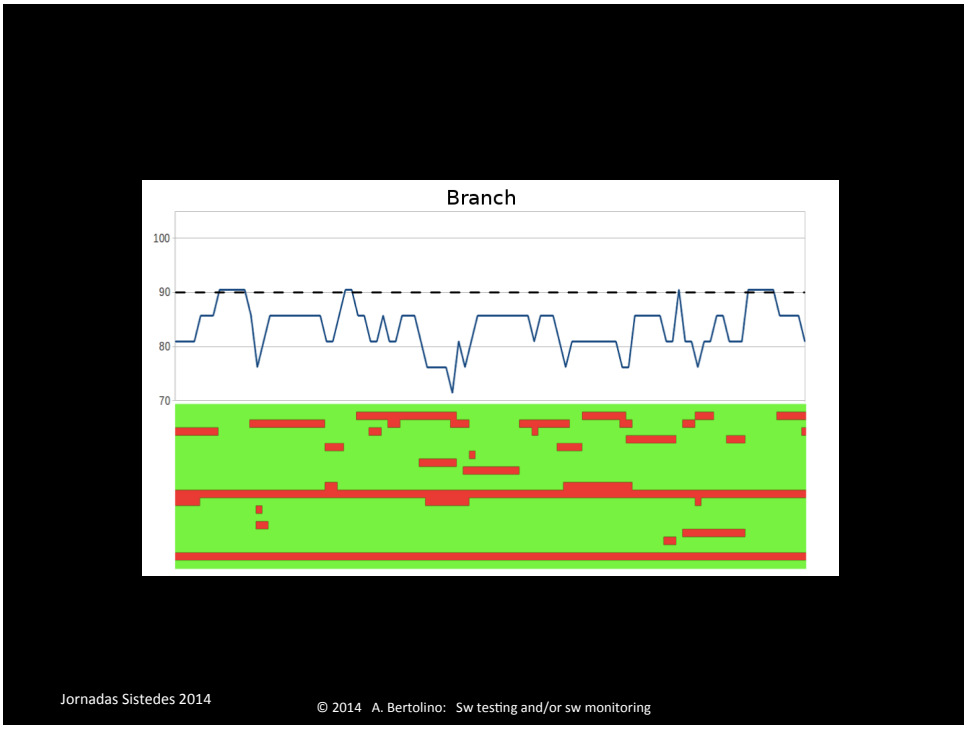


Adequate monitor architecture

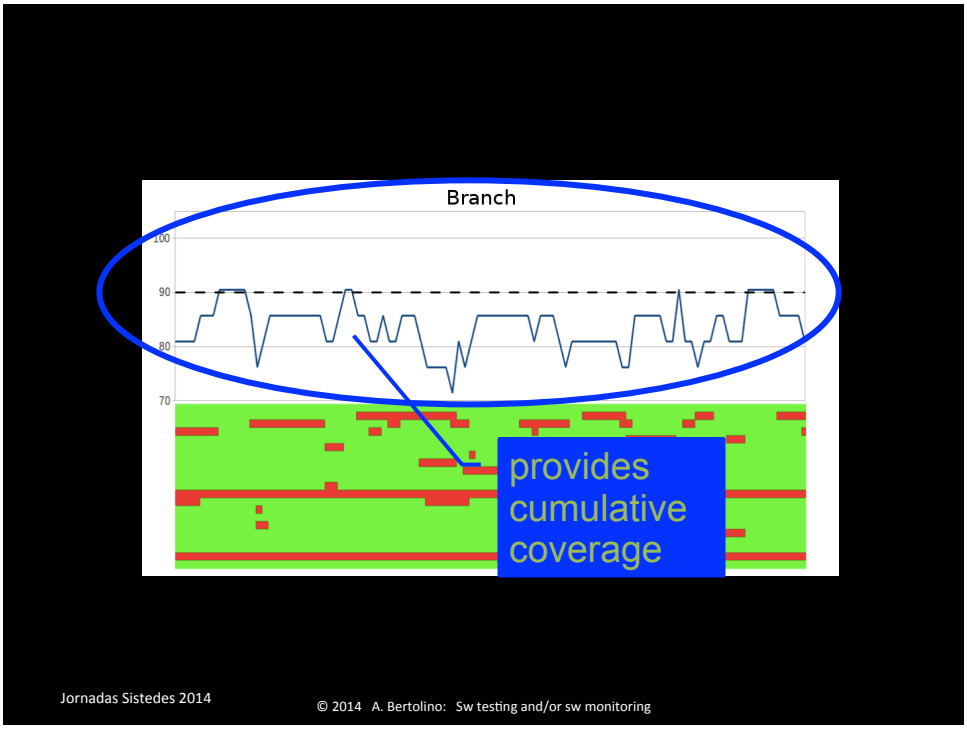


Adequate monitor architecture

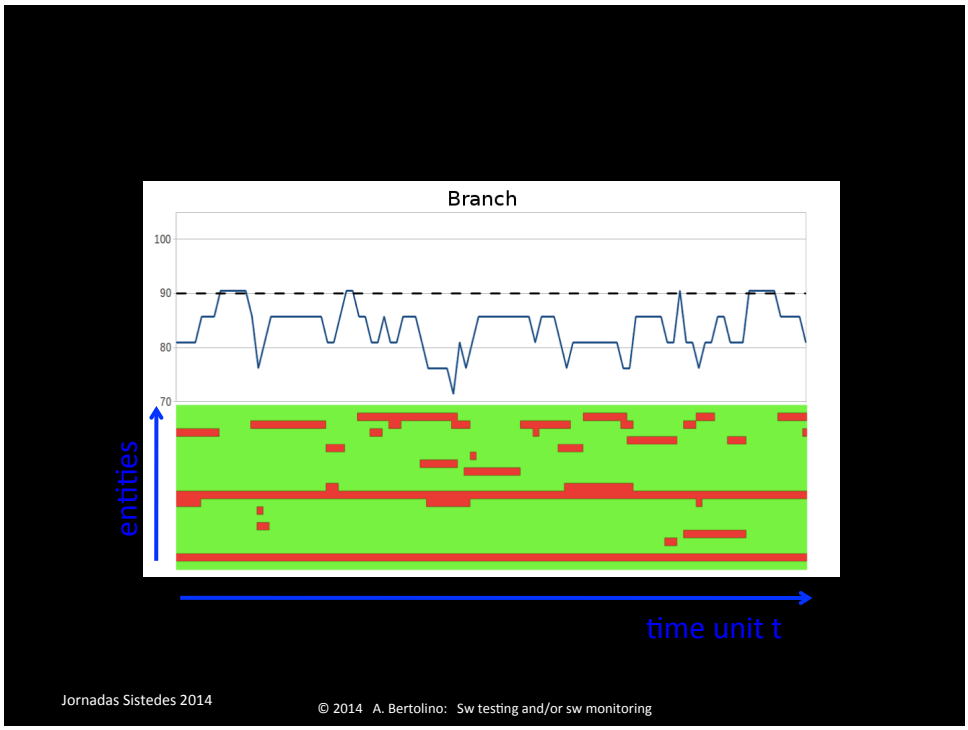




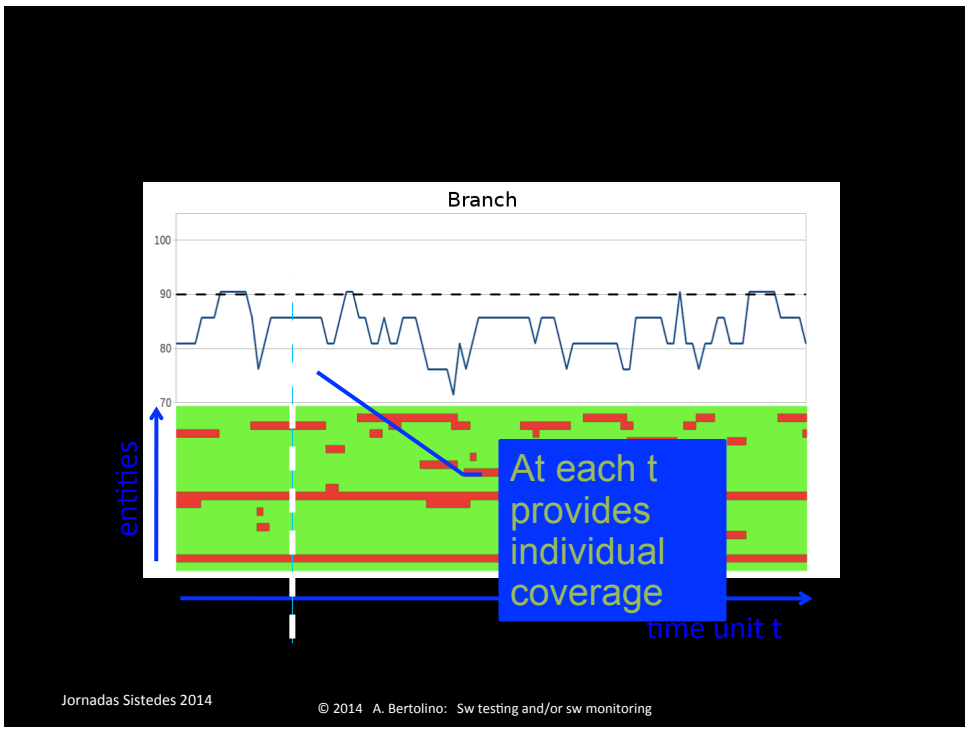
© 2014 A. Bertolino: Sw testing and/or sw monitoring



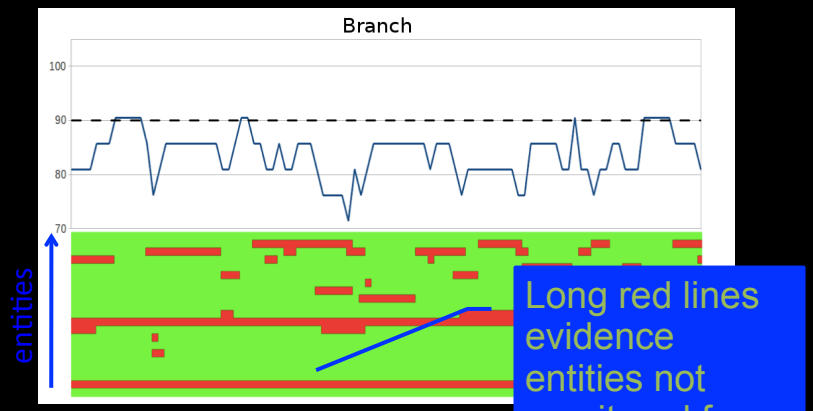
© 2014 A. Bertolino: Sw testing and/or sw monitoring



© 2014 A. Bertolino: Sw testing and/or sw monitoring



© 2014 A. Bertolino: Sw testing and/or sw monitoring



© 2014 A. Bertolino: Sw testing and/or sw monitoring

Summarizing:

- There are techniques from testing discipline that could be used in operation to make monitoring more powerful, e.g.
  - ❖ proactive launch of test cases in the deployed system (how to deal with effects)
  - ❖ measuring coverage within defined observation windows

© 2014 A. Bertolino: Sw testing and/or sw monitoring

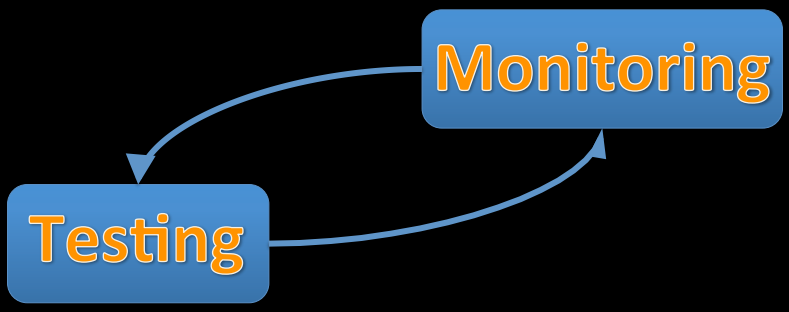
Summarizing:

- There are techniques from monitoring discipline that could be used in laboratory testing to improve automation and reduce costs, e.g.
  - ❖ resolve non determinism (oracle)
  - ❖test selection

© 2014 A. Bertolino: Sw testing and/or sw monitoring

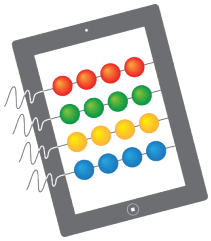
Summarizing:

- Each activity could mutually benefit from each other in a perpetual verification approach



© 2014 A. Bertolino: Sw testing and/or sw monitoring

Adv time …



LEARN PAD

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Not only system test and monitoring

- E.g., adequate monitor can be used to assess learners performance

Learn PAd Project
Model-Based Social Learning for Public Administrations
http://www.learnpad.eu

© 2014 A. Bertolino: Sw testing and/or sw monitoring

Learn PAd Motivation

Training and knowledge sharing within PA are characterized by (state of the practice):

- Informative and off-line approach
- Difficulties in providing "on-the-job" support
- Difficulties in knowledge sharing
- Difficulties in identifying knowledge sources and competences
- Users perspective is often ignored

...

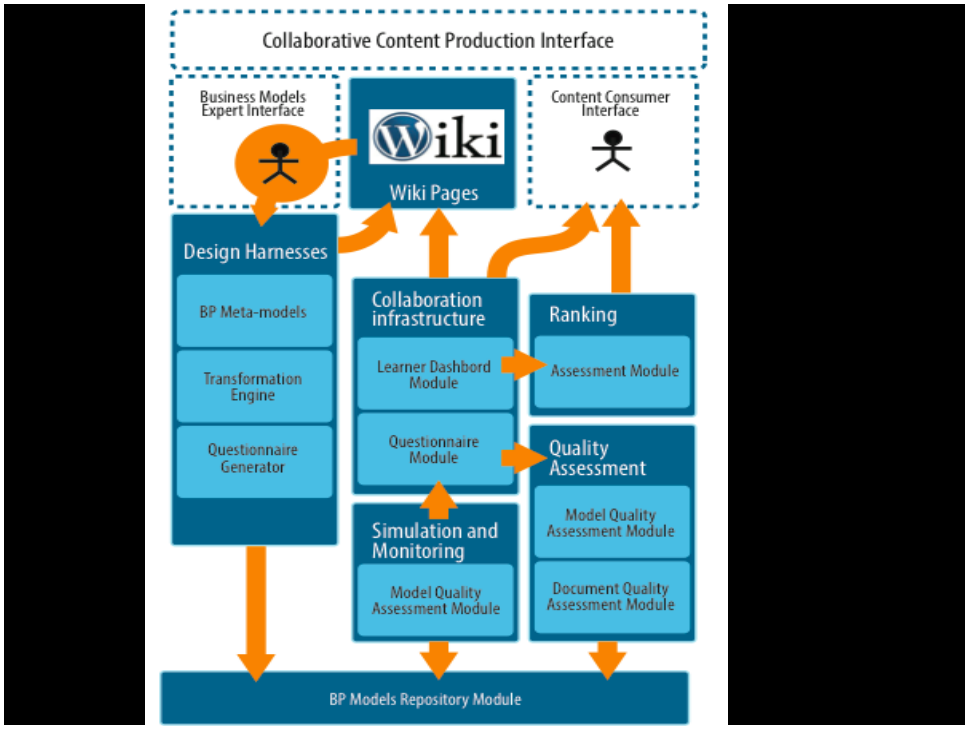
© 2014 A. Bertolino: Sw testing and/or sw monitoring

Learn PAd objectives

To provide a collaborative content production and learning platform for:

- Supporting different approaches to learning:
  - performative, informative
  - self-assessment
- Knowledge sharing
- Competences highlighting
- Fostering cooperation
  - collaborative contents production
  - collaborative learning

© 2014 A. Bertolino: Sw testing and/or sw monitoring



Learn PAd innovation

- Model based collaborative learning
- BP driven content production
- Automated verification of local/global process coherence
- Automated evaluation of natural language defined contents
- Automated model-based derivation of self-assessment tasks
- Model based simulation for learning purpose
- Ontology based and OSS inspired KPIs

© 2014 A. Bertolino: Sw testing and/or sw monitoring