UNIVERSITY OF BIRMINGHAM

Evaluating Software Products

Dr. Rami Bahsoon
School of Computer Science
The University Of Birmingham
r.bahsoon@cs.bham.ac.uk
www.cs.bham.ac.uk/~rzb

Office 112 Computer Science

Software Products Evaluation

UNIVERSITY OF BIRMINGHAM

- Aimed at establishing measurable effects of using a method/tool;
- Aimed at establishing method/tool appropriateness i.e. how well a method/tool fits the needs and culture of an organisation?
- Confidence that both the product and the process are sound
- Evaluation in our context?
  - Quality of the product?
  - Quality of the process?
  - How systematic?
  - For many BSc/MSc conversion Projects with product results -- Feature Analysis Method

Quantitative and Qualitative Evaluations

UNIVERSITY OF BIRMINGHAM

- Evaluation can be quantitative, qualitative or hybrid
  - There is another dimension to an evaluation: the way in which the evaluation is organized
- Quantitative evaluations assume that you can identify some measurable property (or properties) of your software product [or process that you expect to change as a result of using the methods/tools] you want to evaluate.
  - Ways to quantitative evaluations: case studies, experiments, and surveys
- Qualitative evaluations: identifying the requirements that users have for a particular task/activity and mapping those requirements to features that a method/tool aimed at supporting that task/activity should possess.
  - Devising the evaluation criteria and the method of analyzing results using the standard Feature Analysis

Evaluation Approaches in Software Eng

UNIVERSITY OF BIRMINGHAM

[Zelkowitz and Wallace, 1997] classify experimental validation methods for software engineering

- Observational Methods
  - Project Monitoring
  - Case Studies
  - Field study
- Historical Methods
  - Literature Search
  - Legacy Data
  - Lessons-learned
  - Static Analysis
- Controlled Methods
  - Experiments(Replicated and/or Synthetic Environment)
  - Dynamic Analysis
  - Simulation
  - Feature Analysis

We will now briefly review each of these methods …

Project Monitoring

UNIVERSITY OF BIRMINGHAM

- Observational method
- Collection and storage of data produced during software development
- Passive experimentation technique
- Project is not influenced or directed in choice of methods and tools
- Data to be used for immediate analysis and to provide baseline for future improvement techniques
- Examples:
  - Collection of data on defects
  - Collection of performance measures

Case Studies

UNIVERSITY OF BIRMINGHAM

- Observational Method
- Focus is on a specific goal while with Project Monitoring data was collected without particular goal in mind
- Project that is undertaken anyway
- Data collection of a few specific attributes
- If they are performed on real projects, they are already "scaled-up" to life size
- With little or no replication, they may give inaccurate results
- There is no guarantee that similar results will be found on other projects
- There are few agreed standards/procedures for undertaking case studies

Field Study

UNIVERSITY OF BIRMINGHAM

- Observational Method
- Combination between Project Monitoring and Case Study
- Goal: Compare several projects simultaneously
- Outside group monitors subject groups in projects and collect relevant information
- Process is not modified
- Example:
  - [Curtis et al, 1988]: Study of impact of limited application domain knowledge of designers in 17 large projects

Literature Search

UNIVERSITY OF BIRMINGHAM

- Historical Method
- Confirm existing hypothesis by analysing previously published results
- Pitfalls:
  - Selection bias
  - Tendency to report on positive results only

Legacy Data

UNIVERSITY OF BIRMINGHAM

- Historical Method
- Completed projects leave legacy data
  - Source code
  - Specifications
  - Design
  - Test plans and test data sets
  - Further data collated during the development
- Open-source projects are accessible
- Those data can be analysed quantitatively (as opposed to qualitative analysis in Lessons-learned experiments)

Dynamic Analysis

UNIVERSITY OF BIRMINGHAM

- Controlled Method
- Controlled experiment for product rather than process
- Execute product and measure its behaviour
  - Memory footprint
  - Latency
  - Scalability
- Requires benchmarking: Define a synthetic load that is representative for real loads.
- Benchmark definition effort can be shared by several labs that work on different solutions for the same problem

Simulation

UNIVERSITY OF BIRMINGHAM

- Controlled Method
- Execute the product in a simulated environment
- Use to predict how the real environment will interact with the product
- Use to select promising alternatives for real experimentation

Feature Analysis

UNIVERSITY OF BIRMINGHAM

- Suitable for evaluating products
- A specific feature may be further decomposed into sub-features…
  - "Usability" could be expanded into two features: "Learnability" and "Graphical User Interface Features".
  - Learnability might be further expanded into the following features: Quality of documentation; Learning curve; Training effort; Training quality; On-line help.
  - Tips: Use your functional and no-functional requirements document as way to analyze for the features

UG	MSc	Mark Band	Quality of Product
			The 'product' may or may not be a fully complete piece of software: for example, in a more research-oriented project or a project in which systems/business analysis forms the main part, it may be sufficient to produce a prototype as 'proof of concept'. In such cases the 'product' is a combination of the outcome of the research and analysis (e.g. the solution to the problem) and the prototype. The descriptors below need to be adapted appropriately.
1st	Distinction	85-100	The product is exceptional, with novel and original features. In the case of a research project, it makes a significant contribution to the state of knowledge. Where appropriate, the user interface and interaction are superior for their purpose, the product has excellent performance, and is stable and robust.
		78-84	The product is outstanding. In the case of a research project, it goes beyond the required objectives by incorporating new insights. Where appropriate, the user interface and interaction are superior for their purpose, the product has excellent performance, and is stable and robust.
		70-77	The product has all the required, or expected, features. Where appropriate, the user interface and interaction are excellent for their purpose, the product has excellent performance, and is stable and robust.
2:1	Merit	60-69	The product has almost all the required or expected features. Where appropriate, the user interface and interaction are good for their purpose, the product has good performance, and is mostly stable and robust.
2:2	Pass	50-59	The product has acceptable features. Where appropriate, the user interface and interaction are adequate for their purpose, the product has acceptable performance and is reasonably stable but not necessarily fool-proof or robust.
3rd		40-49	The product has minimal to adequate features. Where appropriate, the user interface and interaction are poor or inconsistent, the performance is possibly poor, but not entirely unacceptable, and the product tends to instability and lacks robustness.
Fail		30-39	The product has inadequate features. Where appropriate, the user interface is very poor or inappropriate, the performance is poor, and the product is unstable.
		15-29	The product has considerably fewer features than could be considered adequate. Where appropriate, the user interface is nonexistent or incoherent, and the product is highly unstable.
		0-14	The product achieves almost nothing and has almost no appropriate features, if any.

UG	MSc	Mark Band	Quality of Process
			Projects can be very varied in their nature. In each case, two extreme descriptors are presented here: one for projects which focus on software development where the end product is intended to be a fully functional computer system; and one for projects which focus on research and analysis, where the end product may be a requirements definition, a software specification or a design, coupled with sufficient implementation (e.g. a prototype) to achieve 'proof of concept'. Many if not most projects will fall between these extremes and the descriptors must be adapted accordingly.
1st	Distinction	85-100	• Novel or innovative software development process with exceptionally good results in terms of quality. Thorough analysis of requirements and well-defined system architecture. Outstanding application of software engineering techniques at every stage , including testing, verification and validation. An outstanding analysis of the product's suitability for its purpose. Well-defined project plan and evidence of its use. • Exceptionally clear and full statement of initial problem. Exceptionally methodical survey of previous work, tending to comprehensiveness in large part and being intellectually exceptionally well presented. Investigation or development of solution carried out exceptionally systematically, with outstanding application of software engineering techniques where appropriate. Evaluation of process of investigation or of development of solution carried out exceptionally well (e.g. to the level expected of a first year research student) and clearly related to the initial problem statement. [Work at this level could be considered for presentation at an international conference.]
		78-84	• Superior adherence to a systematic software development process. Thorough analysis of requirements and well-defined system architecture. Superior application of software engineering techniques at every stage, including testing, verification and validation. An excellent analysis of the product's suitability for its purpose. Well-defined project plan and evidence of its use. • Very clear statement of initial problem. Very methodical survey of previous work either comprehensive in important and relevant areas (e.g. in solving specific technical subproblems) or being exceptionally well (intellectually) presented. Investigation or development of solution carried out unusually systematically, with superior application of software engineering techniques where appropriate. Evaluation of process of investigation or development of solution carried out very well, clearly related to initial problem

Feature Analysis

UNIVERSITY OF BIRMINGHAM

- Assess a balanced set of features, not only looking at the *technical* aspects, but also the *economics*, *cultural* and *quality* aspects
  - Ease of introduction in terms of cultural, social and technical problems;
  - Reliability/Dependability of the tool software;
  - Robustness against erroneous use;
  - Effectiveness in current working environment;
  - Efficiency in terms of resource usage;
  - Elegance in the way certain problem areas are handled;
  - Usability from the viewpoint of all the target users in terms of learning requirements and "user-friendliness";
  - Maintainability of the tool;
  - Compatibility of the method and tool with existing or proposed methods/tools/standards;
  - Maturity of method or tool;
  - Economic issues in terms of purchase, technology transfer costs and cost of ownership;

Feature Analysis

UNIVERSITY OF BIRMINGHAM

- A list of all the required features, along with their definitions.
  - For each compound feature, a judgment scale with appropriate levels of importance.
- For each user group:
  - an assessment of the importance of each feature
  - for each compound feature, the minimum acceptable level of support required (i.e. the acceptance threshold)

Feature Analysis – Basis of Evaluation

UNIVERSITY OF BIRMINGHAM

- Organized demonstrations or trade fairs;
- Study/survey of published evaluations in the technical literature;
- Interviews of current users;
- Detailed evaluation of technical documentation plus "hands-on" usage of demonstration or evaluation versions of software (if appropriate);
- Practical experience of using the candidate method or tool in a wide range of situations.

Feature Scoring

UNIVERSITY OF BIRMINGHAM

Generic scale point	Definition of Scale point	Scale Point Mapping
Makes things worse	Cause Confusion. The way the feature is implemented makes it difficult to use and/or encouraged incorrect use of the feature	-1
No support	Fails to recognise it. The feature is not supported nor referred to in the user manual	0
Little support	The feature is supported indirectly, for example by the use of other tool features in non-standard combinations.	1
Some support	The feature appears explicitly in the feature list of the tools and user manual. However, some aspects of feature use are not catered for.	2
Strong support	The feature appears explicitly in the feature list of the tools and user manual. All aspects of the feature are covered but use of the feature depends on the expertise of the user	3
Very strong support	The feature appears explicitly in the feature list of the tools and user manual. All aspects of the feature are covered and the tool provides tailored dialogue boxes to assist the user.	4

References

UNIVERSITY OF BIRMINGHAM

Kitchenham, B. et al., DESMET: A Methodology for evaluating software engineering methods and tools. IEE Computing and Control Journal 8(3): 120-126. 1997

Other useful references/selected reading:

Basili, V.R., The Role of Experimentation in Software Engineering: Past, Current, and Future. *Proc. 18th Int. Conf. Software Eng.*, IEEE Computer Soc. Press, Los Alamitos, Calif., March 1996.

Curtis et al., A Field Study of the Software Design Process for large systems. *CACM* 31(11): 1268-1287. 1988.

Mockus et al., A Case Study of Open Source Software Development: The Apache Server. *Proc. 22nd Int. Conf. Software Eng.*, ACM Press, March 2000.

Tichy, W., Should computer scientists experiment more? *IEEE Computer* 31(5): 32-40.1998.

Zelkowitz, M. and Wallace, D. Experimental validation in software engineering. *IST* 39:735-743. 1997.

Zelkowitz, M. and Wallace, D. Experimental models for validating technology. *IEEE Computer 31*(5):23-31. 1998.

[...And as informed by Wolfgang Emmerich...]