Impact of MDE Approaches on the Maintainability of Web Apps

Yulkeidi Martínez, Cristina Cachero, Maristella Matera,
Silvia Abrahao & Sergio Luján

Universitat d'Alacant
Universidad de Alicante

POLITECNICO
DI MILANO

UNIVERSIDAD
POLITECNICA
DE VALENCIA

Speaker:
Cristina Cachero

"We (practitioners) need your help. We need some better advice on how and when to use methodologies"

[Glass 2004]

Outline

Introduction

SOME FACTS

- Which are the claims of MDD?
  - Short and long term productivity gains (process)
  - Improved project communication (process)
  - **Defect and rework reduction (better quality of the resulting product) (product)**

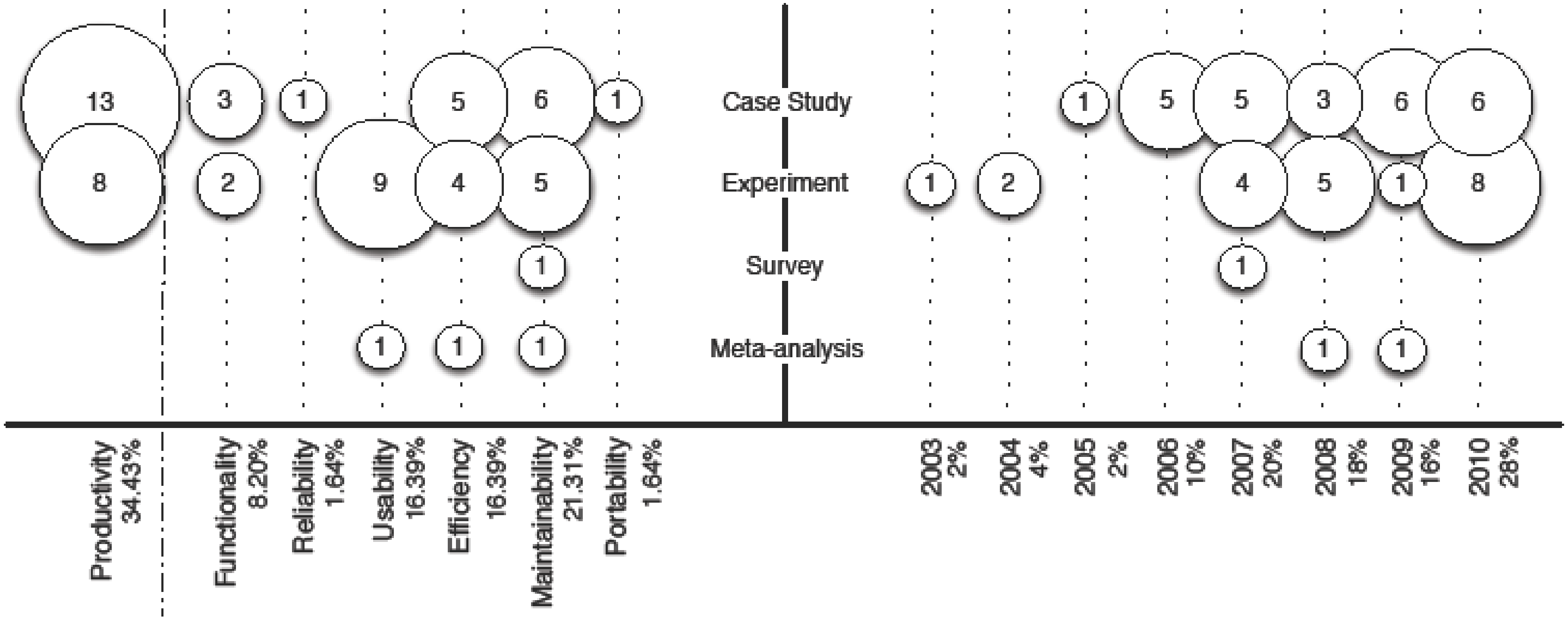
- In particular: better maintainability
  - From a developers' perspective (as a surrogate of the manager perspective)
  - Maintainability makes for 45-60% of development costs [Ruiz and Polo 2007]

Outline

Empirical evidence

SOME FACTS

- A systematic mapping carried out in Dec 2010 on 600 papers that claimed to provide empirical evidence sustaining in any way MDD assertions about impact on **product quality and process productivity** produced the following results:



Empirical evidence

SOME FACTS: MDD vs Code

- Reported impact of MDD on efficiency and maintainability (with respect to traditional, code-centric approaches) includes lower time to evaluate the impact of a change (37%) [Mellegard and Staron 2010]

Outline

Our experiment

OBJECTIVE

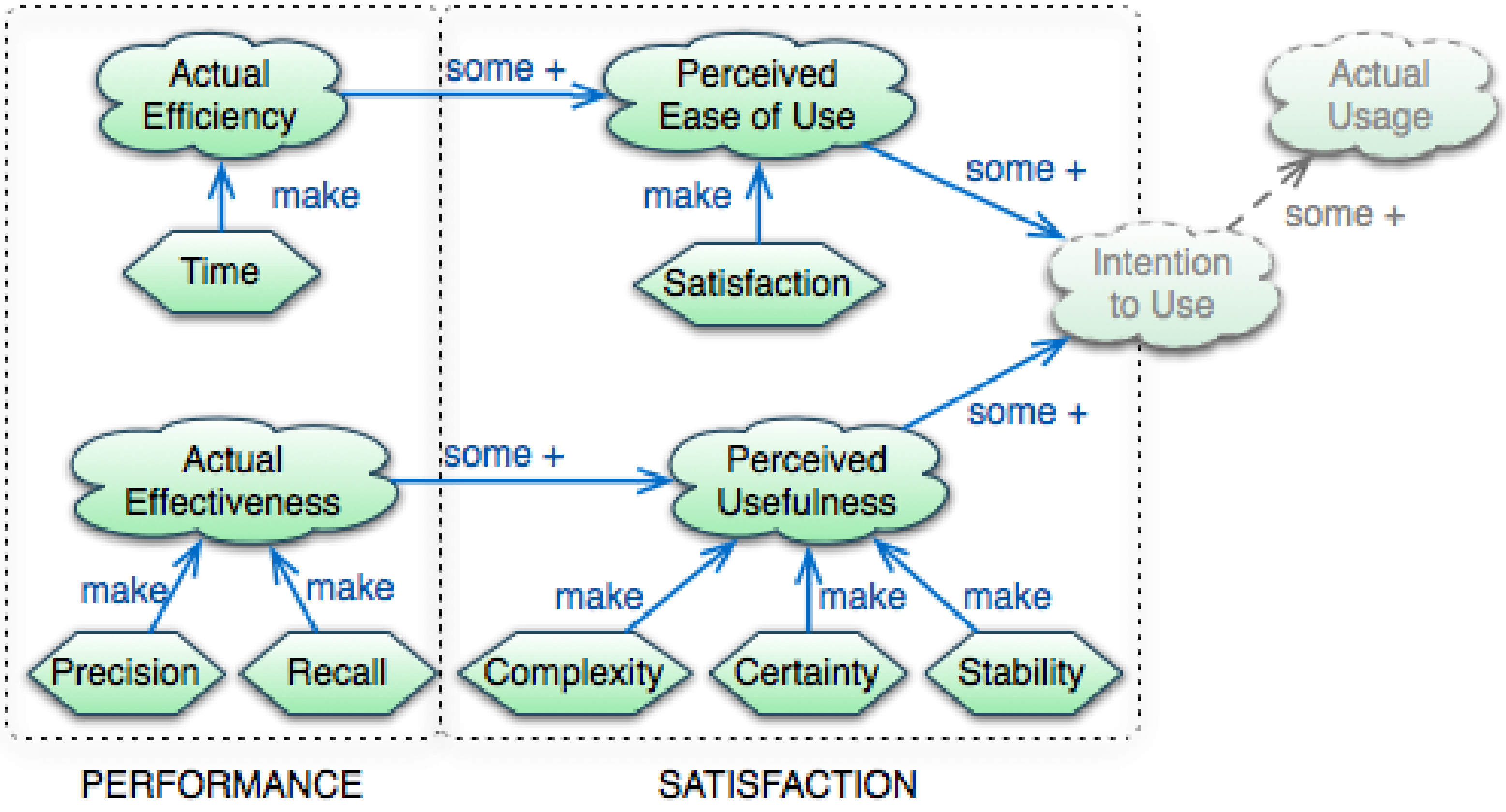
- GQM: *Analyzing* WebML and PHP for the purpose of *comparing* model-driven against code-based maintenance practices *with respect to* their performance and satisfaction *from the point of view of* young developers and prospective adopters.
  - Context: Academy
  - Size of the project: Small-sized applications
  - Developer experience: more experienced in PHP than in WebML
  - Type of application: Information systems

Our experiment

- Maintainability components [ISO 9126]:
  - Analysability: Capability of the software product to be diagnosed for deficiencies or causes of failure in the software, or for the parts to be modified to be identified.
  - Changeability: Capability of the software product to enable the application of a specified modification.
    - Corrections: Corrective maintainability/changeability refers to the capability to detect errors, diagnose the problems and fix them.
    - Improvements: Perfective maintainability refers to the capability to extend the software according to new requirements or enhancements.
    - Adaptations: Adaptive maintainability/changeability refers to the capability to modify the software in order to cope with the effects of environmental changes.
    - Preventions: Preventive maintainability refers to the software capability to support internal reengineering processes without adverse impact.
  - Stability: Capability of the software product to avoid unexpected effects from modifications of the software.
  - Testability and compliance not applicable

THEORETICAL FRAMEWORK

P: correct/total reported R: correct/total correct



Adapted from [Moody 2001]

DESIGN

Our experiment

- Between-subjects quasi-experiment (not random selection of subjects)
  - Same task complexity for both domains.
  - Same tasks for PHP and WebML methods
  - 44 Subjects randomly assigned, each one performing an analysability, a corrective maintainability and a perfective maintainability task with a given approach in a given domain

	Concert App	Bookstore App
WebML	16	16
PHP	6	6

- Mann-Whitney U non parametric test (conservative approach)
- Satisfaction scale reliability: Cronbach's α=0,861

Our experiment

HYPOTHESES

- HAA (Actual Analysability): Analyzing maintainability issues over WebML models allows for a better actual performance than analyzing them over PHP code.
  - Mann-Whitney U non parametric test
    - Precision(WebML)>Precision(PHP)
    - Time(WebML)<<Time(PHP)
    - Recall (WebML) = Recall (PHP)

Performing analysability tasks on WebML seems to speed up the identification of errors, and to avoid misclassifying a feature in the application as an error. However, it does not seem to significantly help to detect the errors in the application.

HYPOTHESES

Our experiment

- HACC (Actual Corrective Changeability): Correcting errors over WebML models allows for a better actual performance than correcting them over PHP code.
  - Mann-Whitney U non parametric test
    - Precision(WebML)>Precision(PHP)
    - Time(WebML)<Time(PHP)
    - Recall (WebML) = Recall (PHP)

Performing corrective changeability tasks on WebML seems to speed up the correction of errors, and to avoid proposing corrections that do not actually correct anything. However, it does not seem to significantly help to correct the actual errors in the application.

Our experiment

HYPOTHESES

- HAPC (Actual Perfective Changeability): Improving a Web application over WebML models allows for a better actual performance than improving it over PHP code.
  - Mann-Whitney U non parametric test
    - Precision(WebML)=Precision(PHP)
    - Time(WebML)<Time(PHP)
    - Recall (WebML) = Recall (PHP)

Performing perfective changeability tasks on WebML seems to speed up the inclusion of new requirements. However, it does not seem to significantly help to come up with useful perfective changes nor to avoid useless perfective changes.

Our experiment

- HPA (Perceived Analysability): Subjects feel that finding errors over WebML models is less complex; they also feel more certain about the results.
  - Mann-Whitney U non parametric test
    - Complexity(WebML)<Complexity(PHP)
    - Certainty(WebML)=Certainty(PHP)
    - *Stability: Not applicable*

The subjects regard performing maintenance tasks on models as simpler than performing them on code, but they feel equally (un)secure about the errors they identify.

Our experiment

HYPOTHESES

- HPCC (Perceived Corrective Changeability): Subjects feel that correcting errors over WebML models is less complex and more stable than doing so over PHP code; they also feel more certain about the results.
  - Mann-Whitney U non parametric test
    - Complexity(WebML)=Complexity(PHP) (PHP less complex)
    - Certainty(WebML)=Certainty(PHP) (WebML more certain)
    - Stability (WebML)<Stability (PHP)

The subjects seem to feel more sure about what needs to be done to correct a mistake when using WebML, but they seem to feel more 'in control' of the changes when they directly work over the code

HYPOTHESES

- HPPC (Perceived Perfective Changeability): Subjects feel that correcting errors over WebML models is less complex and more stable than doing so over PHP code; they also feel more certain about the results.
  - Mann-Whitney U non parametric test
    - Complexity(WebML)=Complexity(PHP) (WebML less complex)
    - Certainty(WebML)=Certainty(PHP) (PHP more certain)
    - Stability (WebML)=Stability (PHP)

The subjects seem to find equally complex to evolve the application by working on models or over code. They also feel similarly certain about the correctness of their work, and equally certain about the stability of the evolved application.

Our experiment

HYPOTHESES

- HS (Satisfaction): Generally speaking, subjects feel more satisfied when performing maintainability tasks with WebML than with PHP.
  - T-Test
    - S(WebML)=S(PHP) (PHP>WebML)

On average, evaluations were moderately positive for both methodologies, with a mean difference of .06 slightly favoring the PHP group.

Satisfaction=Actual Experience – Expectations

Are we arising unrealistic expectations?

THREATS TO VALIDITY

- Threats to internal validity (hidden factors that may be affecting the results)
  - Students belonging to different cultures, although similar in age and training;
  - Different facilitators, although well-structured experiment;
  - Aleatory selection of TASKS!!!

THREATS TO VALIDITY

- Threats to external validity: generalizability of the results
  - Limited environment: students, WebML and PHP, just on paper
  - Limited time (2 hours)
- Threats to construct validity: we have associated well-known measures to each theoretical construct dimension.
- Threats to conclusion validity: conservative statistical tests, but still
  - Manual evaluation of test results, although clear taxonomy;
  - Not validated subjective measuring scales, except for Satisfaction

Outline

Conclusions

We are conscious that…

- Objectively measuring impact of MDD on product and process improvements is DIFFICULT
  - Generalization of the results
  - Selection of appropriate sample
  - Experimentation COSTS
  - … you name it!

Conclusions

This notwithstanding…

- **We need to increase the body of evidence concerning impact of modelling activities on objective/subjective parameters (productivity, internal, external or quality in use)**

Conclusions

And now… YOUR TURN!

- We would be very grateful if we had your insight…
  - Refinement of theoretical framework?
  - Refinement of maintainability framework (tasks)?
  - More elaborated measures?
  - New or refined hypotheses?
  - **Variables** which should be taken into account when making claims (and sometimes aren't):
    - Context: Academy vs Industry
    - Size of the project
    - Developer experience with methodology (learning curve?)
    - Type of application
    - ???
  - *Satisfaction=Actual Experience – Expectations*

    Are we arising unrealistic expectations?

Impact of MDE Approaches on the Maintainability of Web Apps

Yulkeidi Martínez, Cristina Cachero, Maristella Matera,
Silvia Abrahao & Sergio Luján

Universitat d'Alacant
Universidad de Alicante

POLITECNICO
DI MILANO

UNIVERSIDAD
POLITECNICA
DE VALENCIA

Speaker:
Cristina Cachero

"We (practitioners) need your help. We need some better advice on how and when to use methodologies"

[Glass 2004]