Comments on paper: *Improving Software Development Management through Software Project Telemetry*, PM Johnson et al 2005.

Juan José Amor Iglesias
Universidad Rey Juan Carlos

jjamor _at_ gsyc.escet.urjc.es

March 22th 2006

Presentation based on [Johnson et al., 2005]

Summary

- Introduction. Software Project Telemetry.
- Why it is useful?
- What is Hackystat
- Telemetry in practice
- Conclussions

Introduction

- Metrics can make project management more efective
- Software metrics: from internal attributes (size, complexity, etc) to external proccess attributes (effort, productivity)
- Practitioners face various barriers in applying metrics
- The agile community generally argues against model-based metric applications: they prefer softer metrics for decision making.

Software Project Telemetry

Telemetry: *"Highly automated communication process by which measurements are made and other data collected at remote or inaccesible points and transmitted to receiving equipment for monitoring, display and recording"*

As defined in Encyclopædia Britannica.

Software Project Telemetry

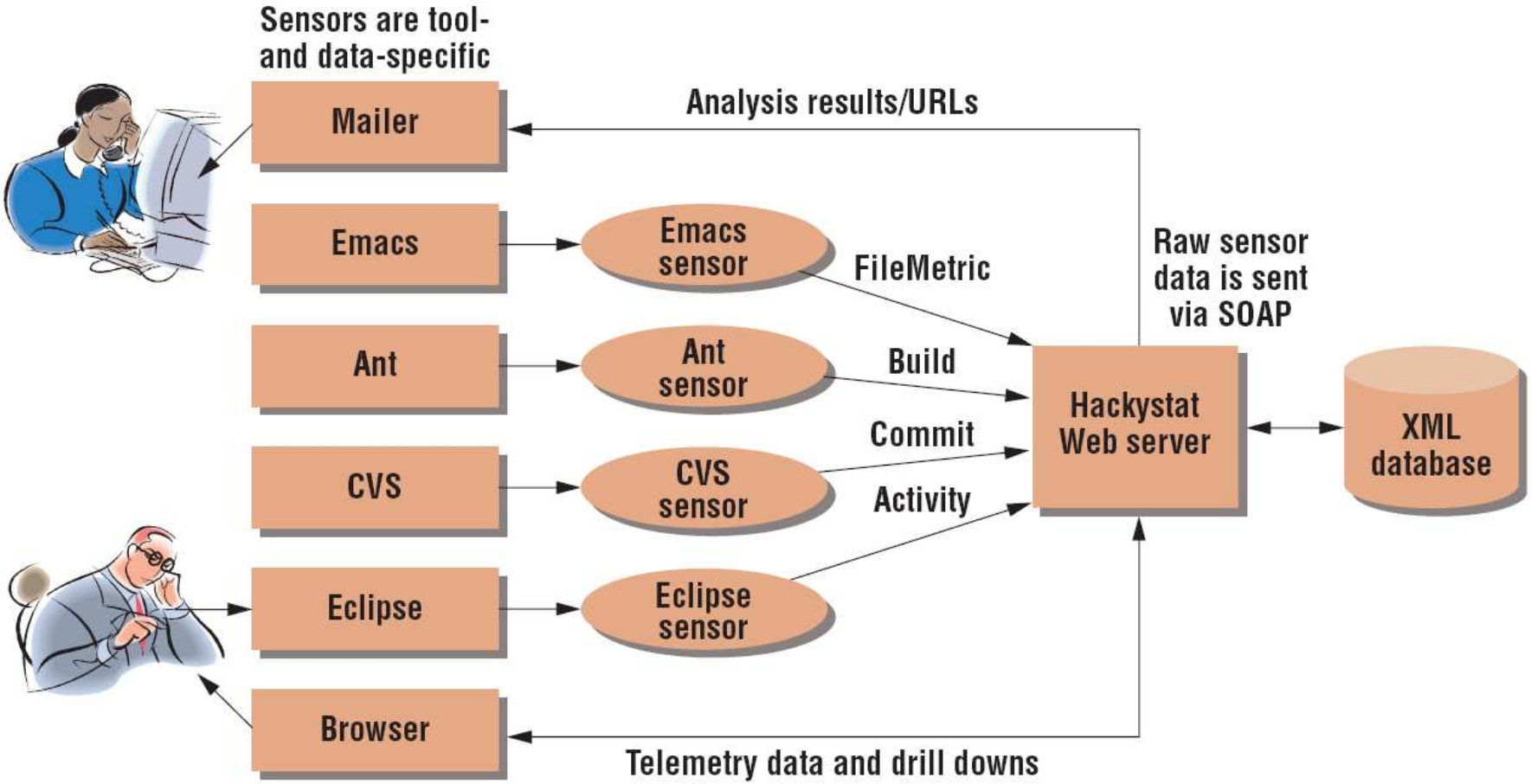
Software Project Telemetry:

- Data collected automatically by tools that regularly measure various characteristics of development environment.
- Data consists of a stream of timestamped events. Timestamp is significant for analysis.
- Developers and managers can immediately access the data.
- Telemetry analyses exhibit graceful degradation.
- Analysis includes in-process monitoring, control, and short-term prediction.

Why it is useful?

- SP Telemetry enables an incremental, distributed, visible, experiential approach to project decision making.
- Example: we see both complexity and defect-density telemetry are increasing.
  - Solution: We can take corrective actions (by simplifying complex modules, etc)
- We can test hyphotesis: "If I reduce module complexity, will defect density decrease?"
- We can do impact analysis: "Do the process changes required to reduce module complexity produce unintended side effects?"
- Decentralized project management
- . . . etc . . .

Hackystat's basic structure



SP Telemetry: Supported classes

- Development telemetry. Data related with usage of development tools: files edited, time spent using tools. Through sensors over editors (emacs), CMS (CVS) or IDEs (Eclipse)
- Build telemetry. Data related with results of tools invoked to compile, link and test system. Sensors over Ant, make, etc.
- Execution telemetry. Data gathered from system execution, through sensors that test system load or stress, etc.
- Usage telemetry. Data related with user's interaction with built system.

Telemetry in practice

Hackystat:

- 95 KLOC
- 30 modules
- 10 developers
- Maintained through CVS
- Nightly builds

The facts: At the end of 2004, integration-build failure rate significant: of 300 yearly attempts, failed 88 times with 95 distinct build errors.

Telemetry in practice

An analysis involving several Hackystat sensor streams revealed:

- Six categories for the 95 build errors.
- Substantial differences between experienced and new developers, with respect to integration-build failures.
- The most dependend two modules on other modules, also had the two highest number of build failures.
- The days with build failure, had a statistically greater number of distinct module commits.
- No relationship between failures and number of lines committed or amount of active time before commit.

Telemetry in practice

How to reduce integration-build failures? (Hypotheses)

- Increased support for new developers (pair programming)
- Refactoring modules to reduce coupling and frequent multimodule commits.
- 82% of of integration failures can be prevented if developers had run a full system compile and test before committing their changes.

Conclusions

- SP telemetry can support project management decision making.
- Hackystat, an open source reference framework for SP telemetry.
- Social issues: SP telemetry is not complete with respect in effort measurement: many legitimate and productive activities not measured, such as meetings, emails, informal conversations...

`http://www.hackystat.org/`

References

[Johnson et al., 2005] Johnson, P. M., Kou, H., Paulding, M., Zhang, Q., Kagawa, A., and Yamashita, T. (2005). Improving software development management through software project telemetry. *Software, IEEE*, 22(4):76–85.