

Metrics as an Early-Warning System for Software Development Projects

2005-11-24 W. Druyts



Agenda

1. Problem statement
2. An early-warning system for process monitoring
3. Implementation choices
4. Social aspects
5. An implementation example
6. Conclusion : Lessons learned

Section 1 - Problem statement

1. Discover 'bad smells' in software development
2. Suitable information to detect and remedy

Problem statement – Discover bad smells

- Traditional project checkpoints
 - Project status reviews (task completion, milestones, ...)
 - Document reviews
 - Verification / validation
- ‘Recent’ trends perform slightly better
 - Agility (XP, FDD, ...)
 - Iterative development
 - Unit testing

Problem statement – Discover bad smells

- Common shortcomings
 - Relation with the ‘failing’ process is weak
 - Not truly early
 - Often only isolated snap-shots
 - Resource-intensive
- Example
 - Stability problems discovered during verification, Should have been resolved in an architecture revision
 - Project status report shows significant slip of 1st milestone Drill down reveals late finalization of requirements

Problem statement – Discover bad smells

- In both examples
 - The ‘revealing’ process is not the root cause
 - Significant amount of time has passed since initial bad smells
- Conclusion
 - More accurate status monitoring is required
 - Deviations must be detected earlier

Problem statement – Suitable information

- Target :
 - A process monitoring system
 - Allows to correct deviations early (read : ‘cheaper’)
- How :
 - Monitor the behavior of development processes
 - Compare with expected behavior
 - Analyze and report deviations
- Proposal : A metrics-based early-warning system

Section 2 – An early-warning system

1. Define the information needs
2. Define the context
3. Define the customer

Early-warning– Information needs

Drivers of information needs can be diverse

- Success criteria of your project or application domain :
 - Mission critical systems : quality
 - Consumer goods : early market entrance, cost
- Risk analysis of the project
- Audit results
 - Known process weaknesses
 - Process improvement objectives

Early-warning – The Context

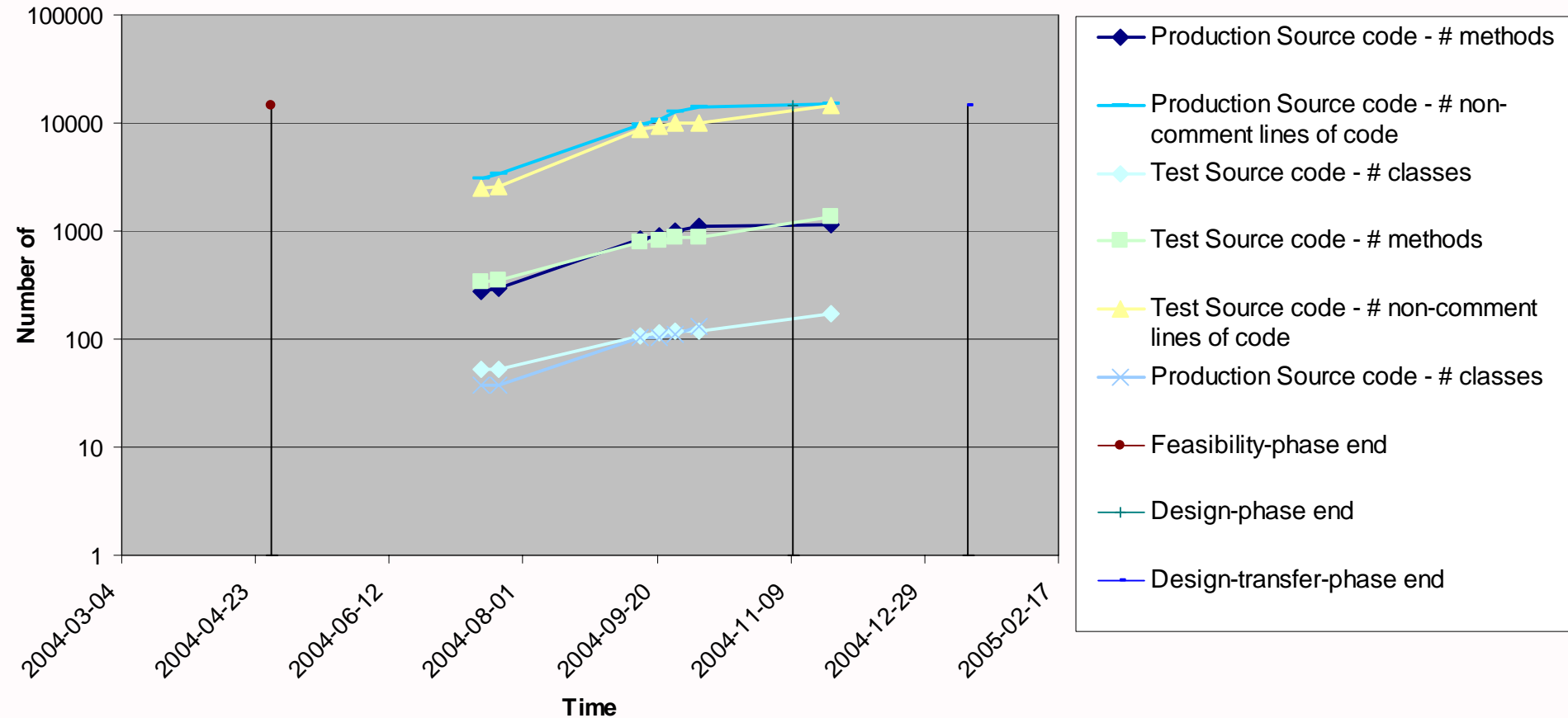
- Methodology
 - Know the ‘big picture’ : waterfall, iterative, agile, ...
- Process
 - Define the purpose, gear metrics to the purpose
 - Define time-correlation and interaction with other processes
- Best practices and lessons learned
 - E.g. agile project should proceed in short iterations
 - Consider ‘anti-patterns’ too
- Deliverables
 - Tangible subject for collecting metrics

Process monitoring – The Customer

- People working in the project
E.g. the developers, the architect, testers
- People managing the discipline
E.g. team leads, QA team members
- People managing the project
E.g. the project manager, ...
- People controlling the project
E.g. sponsors, a steering committee, portfolio managers, ...

Example – Unit testing

Project X - Production versus Test Code



Section 3 – Implementation Choices

1. Out-of-the box features of SW development tools
2. Custom development using the tool's APIs
3. Business process management tools

Implementation – Out-of-the-box features

Potential

- Analysis/reporting features for the tool's domain model
 - IDE (coding environment) : knows how to count 'SLOC's'
 - TestDirector (test mgm) : understands what a 'Test Set' is
- Familiar for the software professional's every-day work,

Drawbacks

- Information extraction is geared for the tool, not the process
- Information format is not suited to be combined with other sources
- Example
 - Cost and timing information is rare in development tools
 - Historical information – to build trends – is not available
 - Mixing processes is usually not possible

Implementation – Custom development

Potential

- Information collection can be tuned to the needs
 - Information can be matched to the process
 - Information can be collected because it's required, not because it happens to be available
- Re-use of existing analysis and reporting tools

Drawbacks

- Needs to be custom-built
 - Unaware of existing tools that cover these needs
 - Unaware of a standard domain model for SW engineering (follow up on OMG's SPEM though)
- Building this type of tools is not core-business

Implementation – Business Process Mgm.

Potential

- This technology seems to come closest to the needs:
 - Collection, Storage, Analysis, Reporting
 - Of multi-source, time-correlated data
- A mapping is possible between
 - the actual use of the tool
 - the desired domain model

Drawbacks

- The use of this technology is uncommon in SW engineering, hence costly to implement
- Balance the acceptance level of metrics with the cost of the tool

Section 4 – Social Aspects

1. People and metrics
2. Organizational support for metrics

Social Aspects – People and Metrics

- Little or no successful precedents in application of metrics :
 - Metrics still play a secondary role in SW projects
 - Process performance management is uncommon
- Metrics are easily perceived as a judgment tool, and it's so tempting to do so...

Social Aspects – Organizational Support

- Transparency,
Common questions are
 - What is being collected ?
 - What is it used for ?
- Confidentiality
 - For whom is the information available
- Management support
 - Use the information
 - Demand availability and quality

Section 4 – An Implementation Example

Examples

1. The technology
2. Examples of available reports

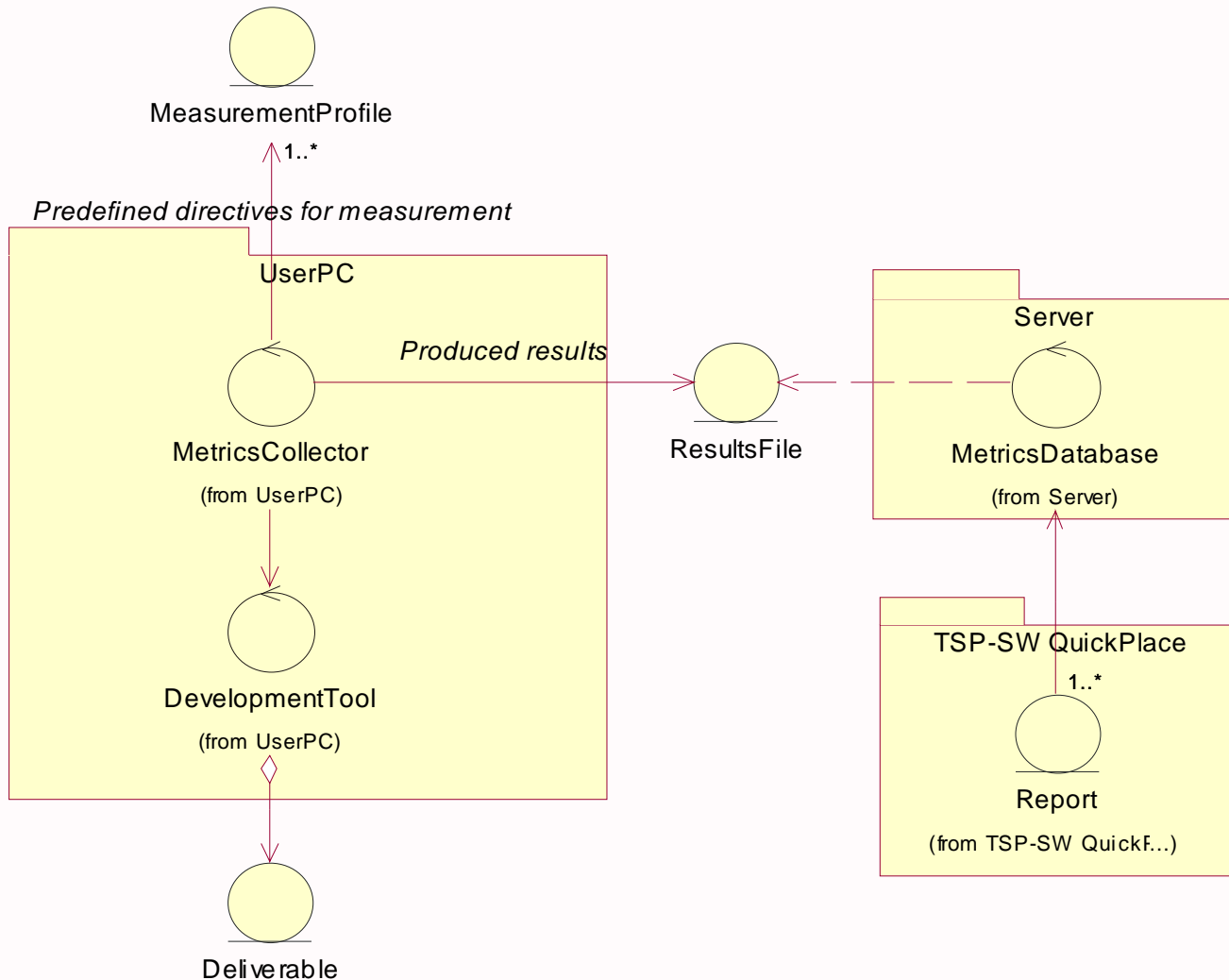
Notice :

most examples show authentic data, some do not...

Background:

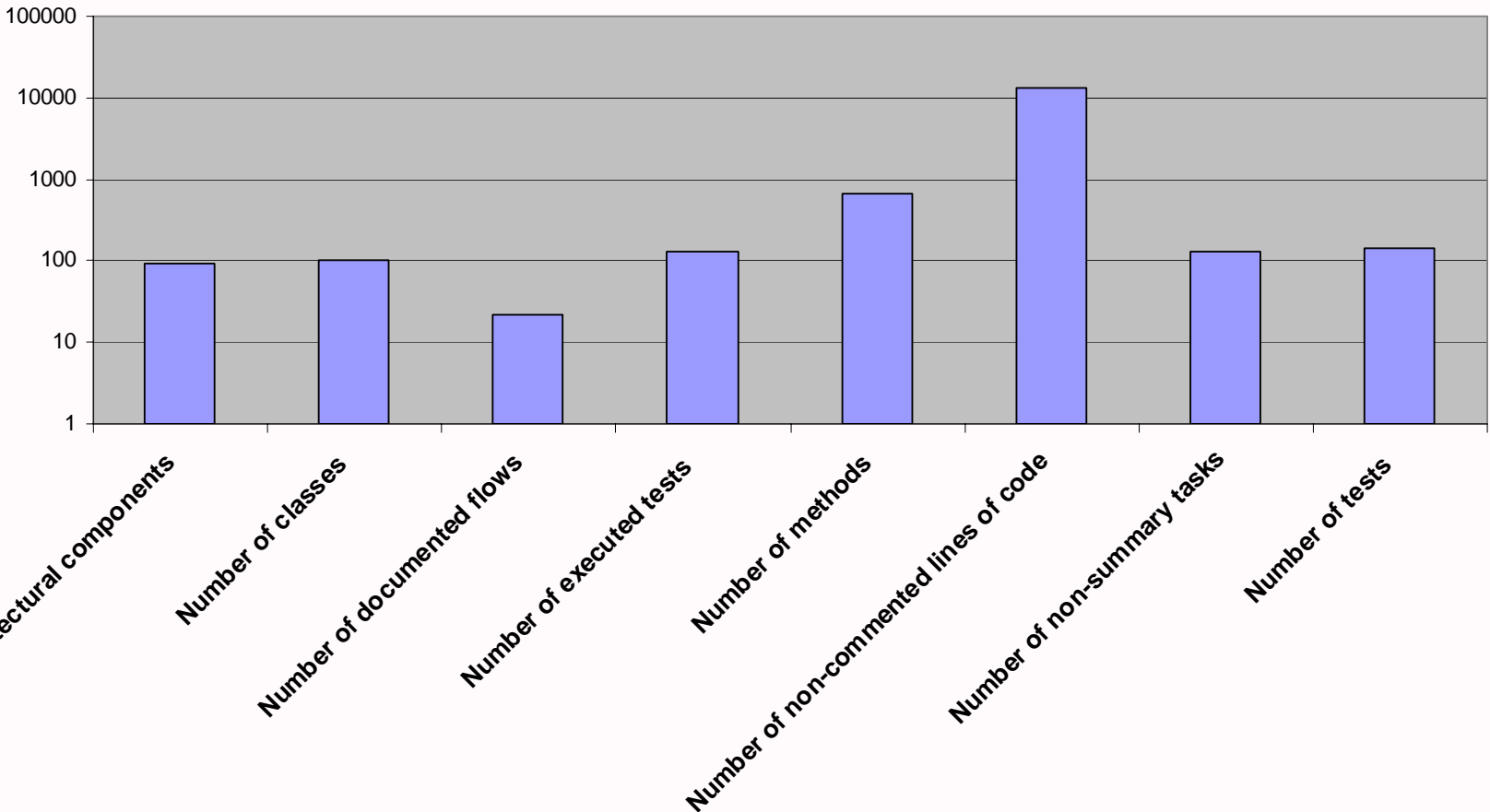
- A process improvement project
- Targeted at improving project predictability

The Technology



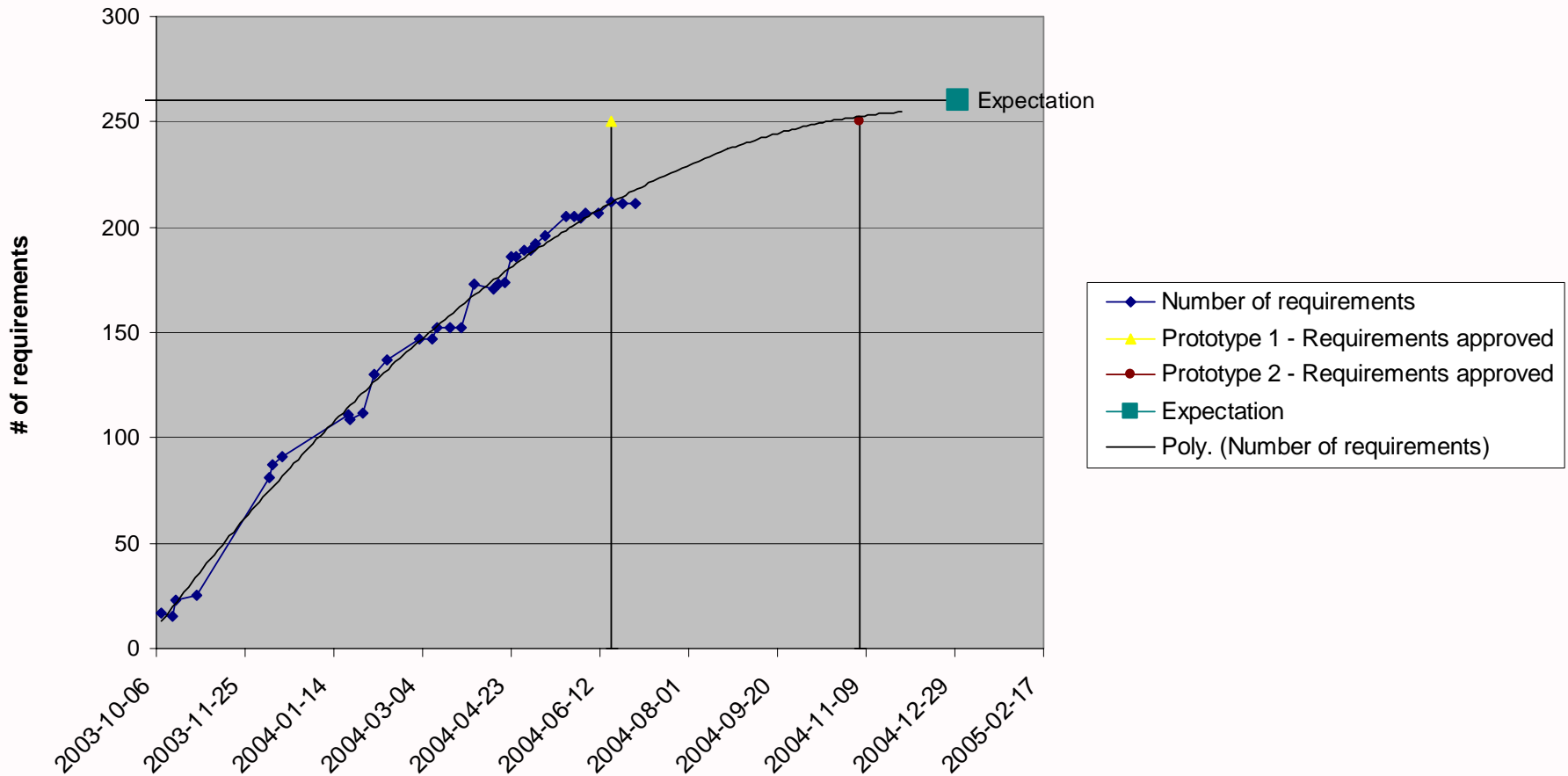
Example – Project size indicators

Project Size Indicators



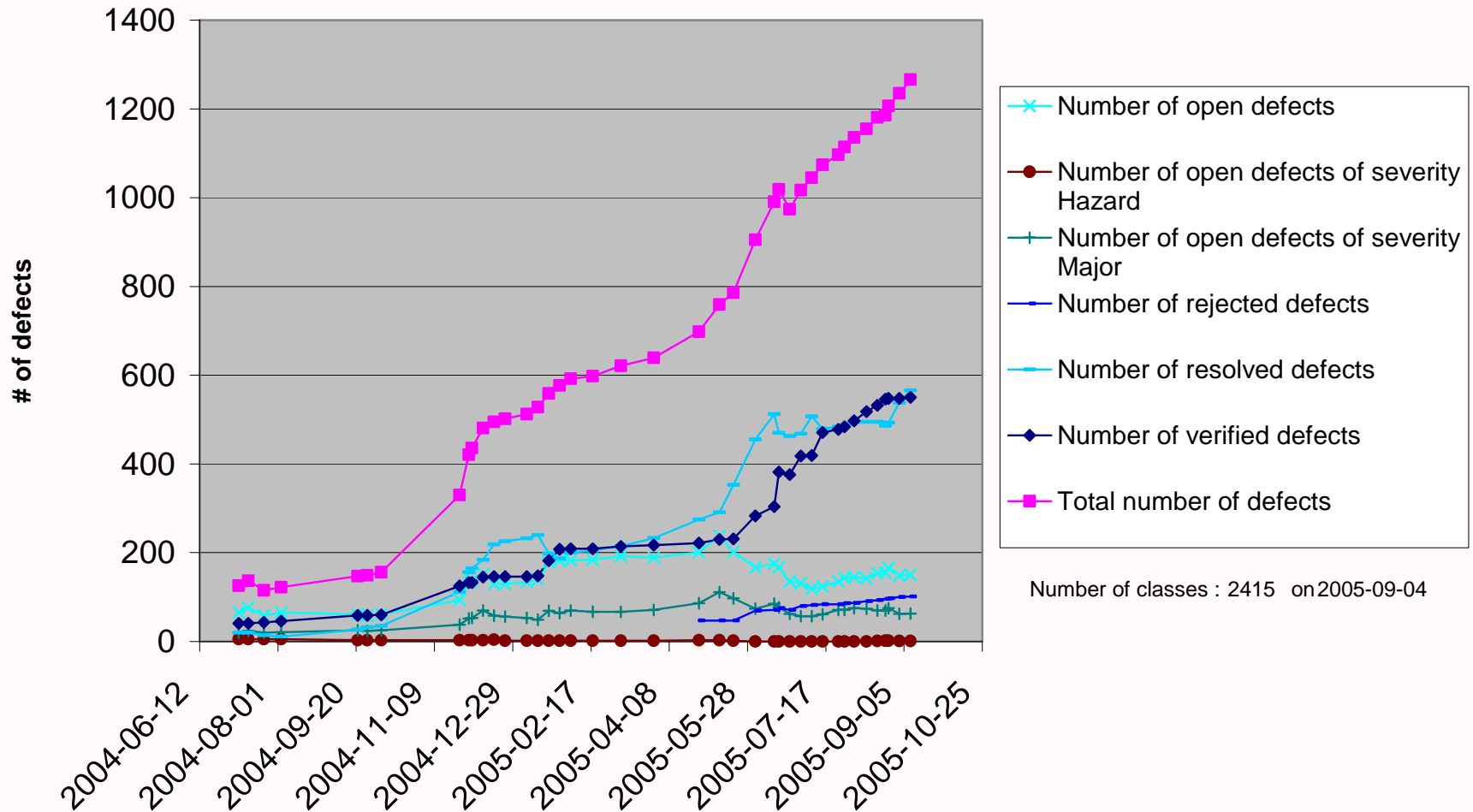
Example – Growth rates

Requirements growth with milestones



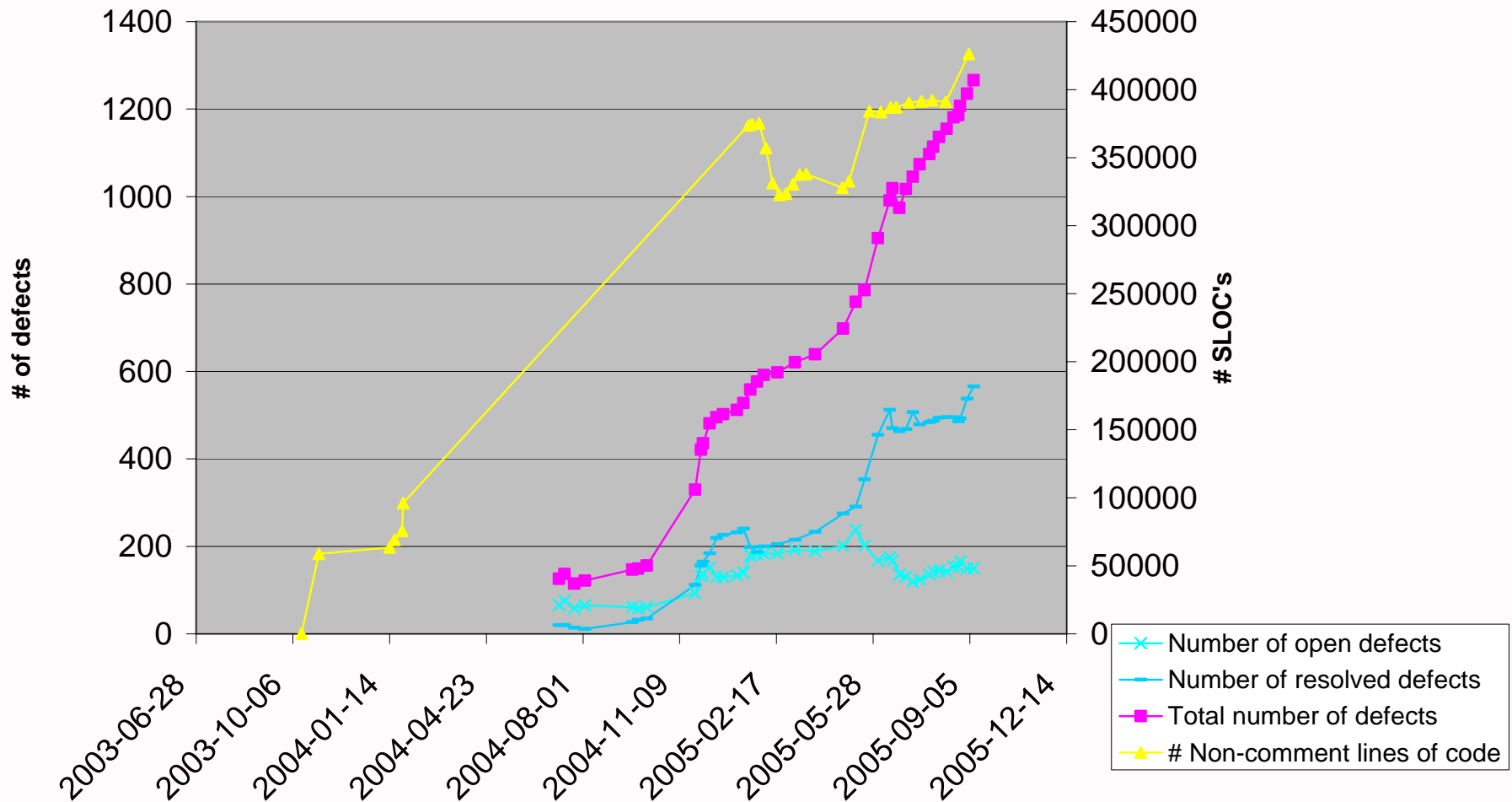
Example – Defect Trends

Defect Evolution per Category



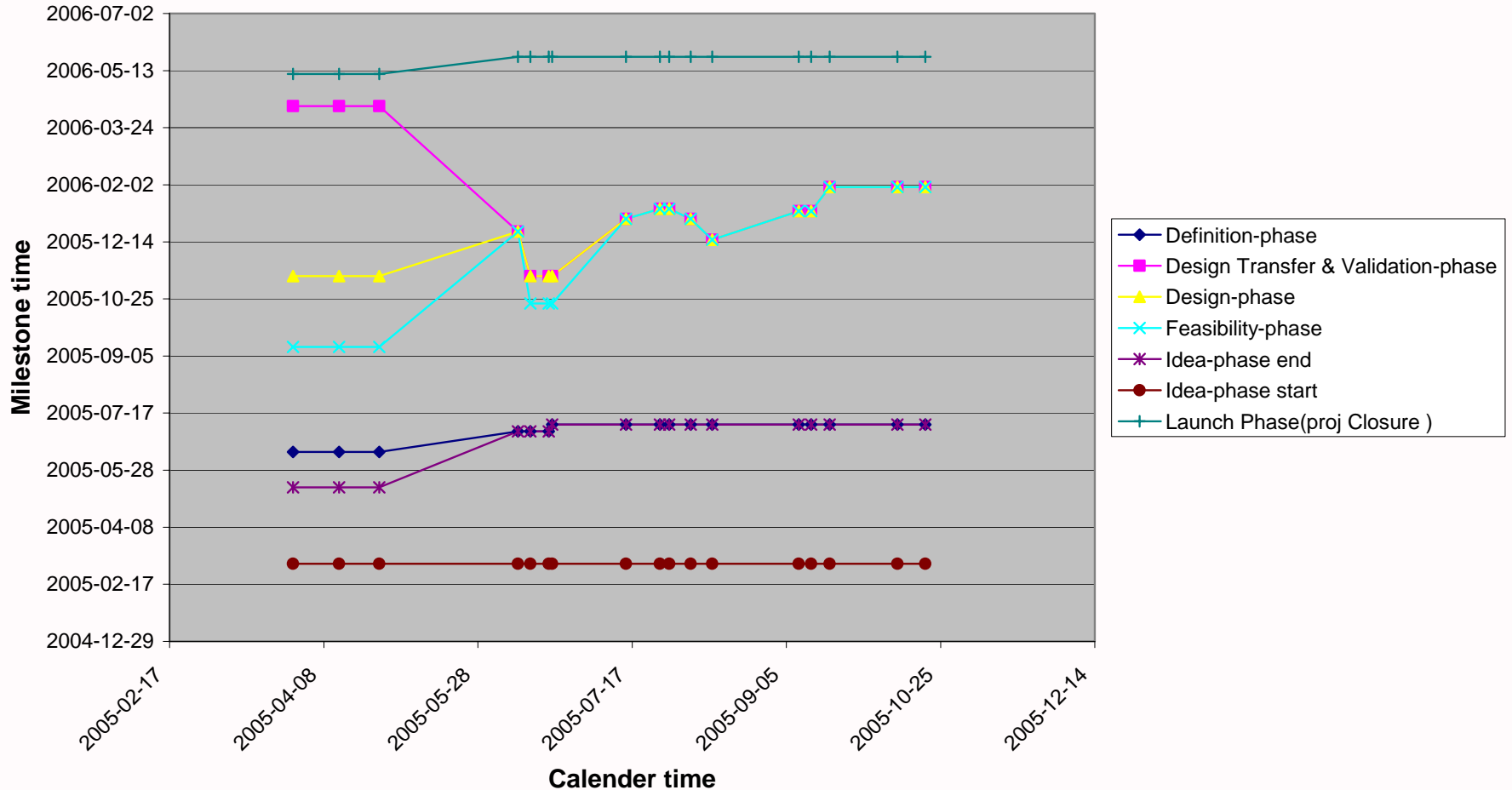
Example – Source Growth and Defect Trends

Defect and Code Size Evolution



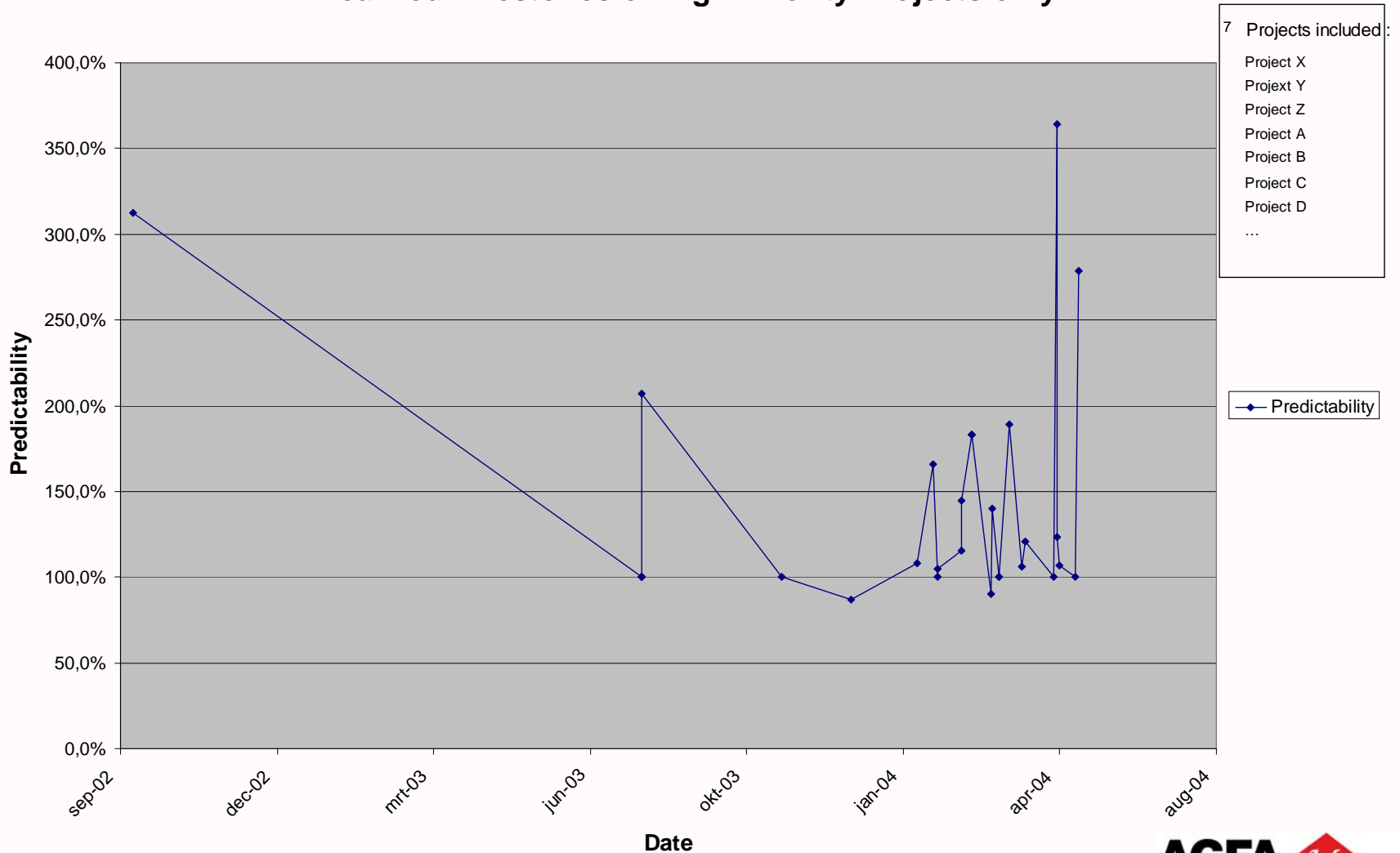
Example – Milestone follow-up

Milestone evolution



Example - Milestone predictability

Milestone Predictability per Month Realized Milestones of High-Priority Projects only



Section 5 – Lessons Learned

- The intermediate solution is feasible:
 - Technology is not too complex
 - Cost can be acceptable if usage is broad
- Own development is not ideal :
 - Ongoing maintenance is required, e.g. when tools are upgraded
 - Is not core-business, should be purchasable
- Responsibilities have to be clearly defined
 - Collection is everyone's responsibility
 - Keeping the system up, both technically and operational, must be a clearly defined responsibility
- Key-risk is the 'human and organizational aspects'