New look at software product documents

– from pRd to PrD!

Pekka Forselius, Senior Advisor, FiSMA

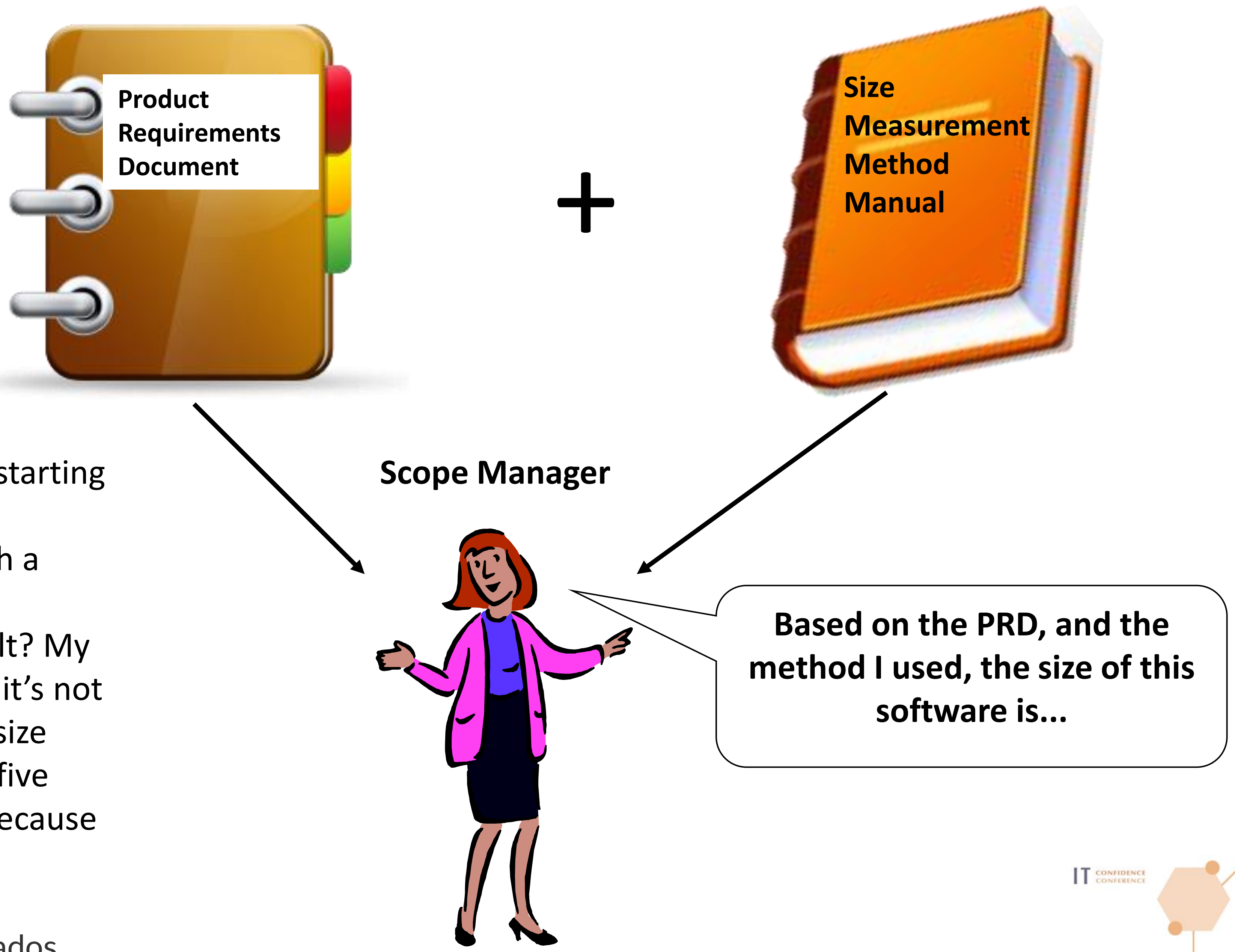
pekka.forselius@4sumpartners.com

Software

all or part of the programs, procedures, rules, and ***associated documentation*** *of an information processing system*.

Poor or missing specifications (= documentation) are the most common reason for wasted time and money in software development projects. Poor or missing documents make software maintenance slow, expensive and vulnerable. They make also project estimation and productivity measurement almost impossible.

Software size measurement:


Product Requirements Document
+
Size Measurement Method Manual
Scope Manager
Based on the PRD, and the method I used, the size of this software is...

As a Scpe Manager, this was my starting point to begin re-thinking the importance of a PRD. Why is such a simple task as software size measurement so often so difficult? My 30+ years of experience tell that it’s not because of poor or inapplicable size measurement methods (i.e. the five ISO/IEC standards). Usually it’s because of poor documents.

Audience: The users of Product Document?

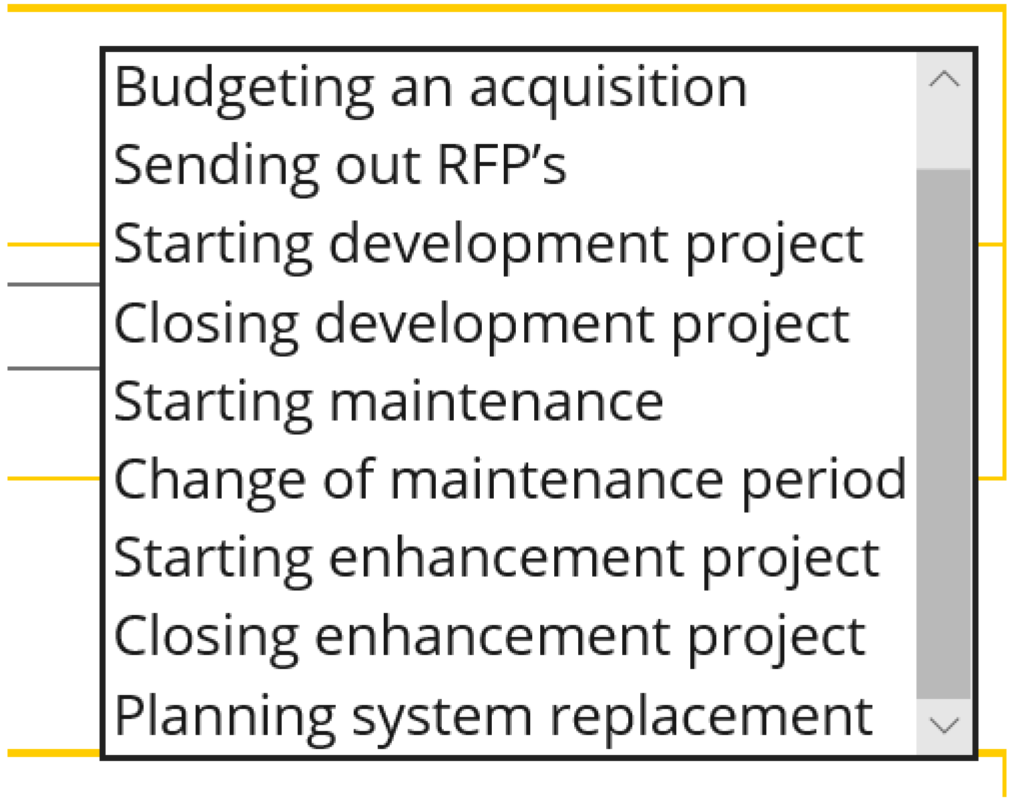
- Product owner –person or group making business decisions related to specifications
- Sponsor – person accepting business decisions
- Business analyst – author and reviewer of specifications
- IT architect – supporting and guiding business analysts and product owner
- Developer team member – programmer and unit-tester, reader and reviewer of specifications
- Tester – checking the compatibility of the system against the specifications
- End-user – learning how and when to use the system
- Experienced end-user – requesting changes to functions of the system
- Scope manager – a measurement expert supporting product owner and sponsor
- Maintainer – keeping the system and its environment working and up-to-date

Product life-cycle: when are the documents needed?

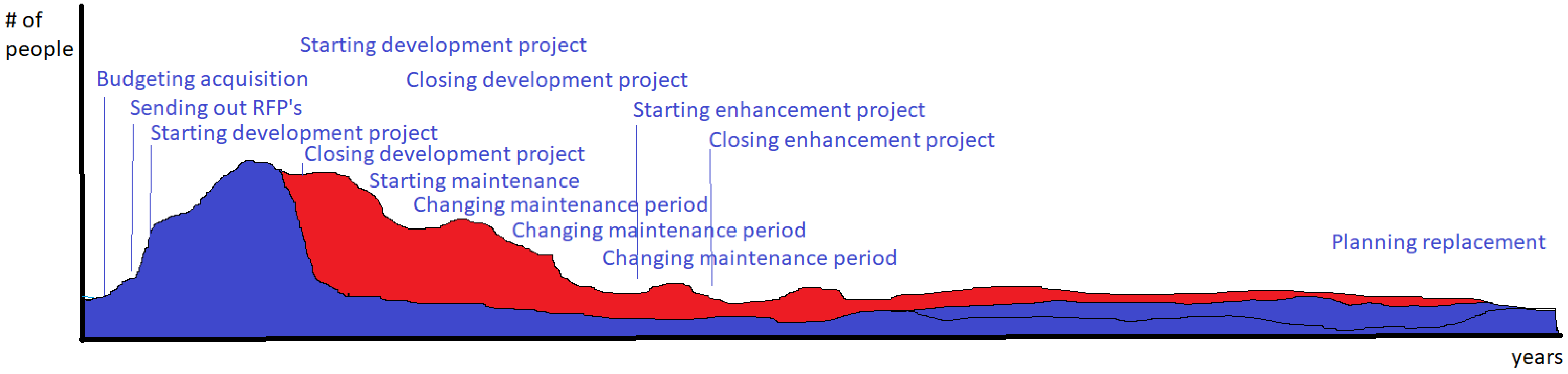
During a software intensive system life-cycle ***major use cases of Product Document*** occur various times.

Budgeting the system acquisition and sending out the requests for proposals to potential vendors happen usually just once. Same in the end of life-cycle, when the product owner starts to plan replacing the whole system with another.

The larger the system, the more different development and enhancement projects occur during the product life-cycle.


Budgeting an acquisition
Sending out RFP's
Starting development project
Closing development project
Starting maintenance
Change of maintenance period
Starting enhancement project
Closing enhancement project
Planning system replacement

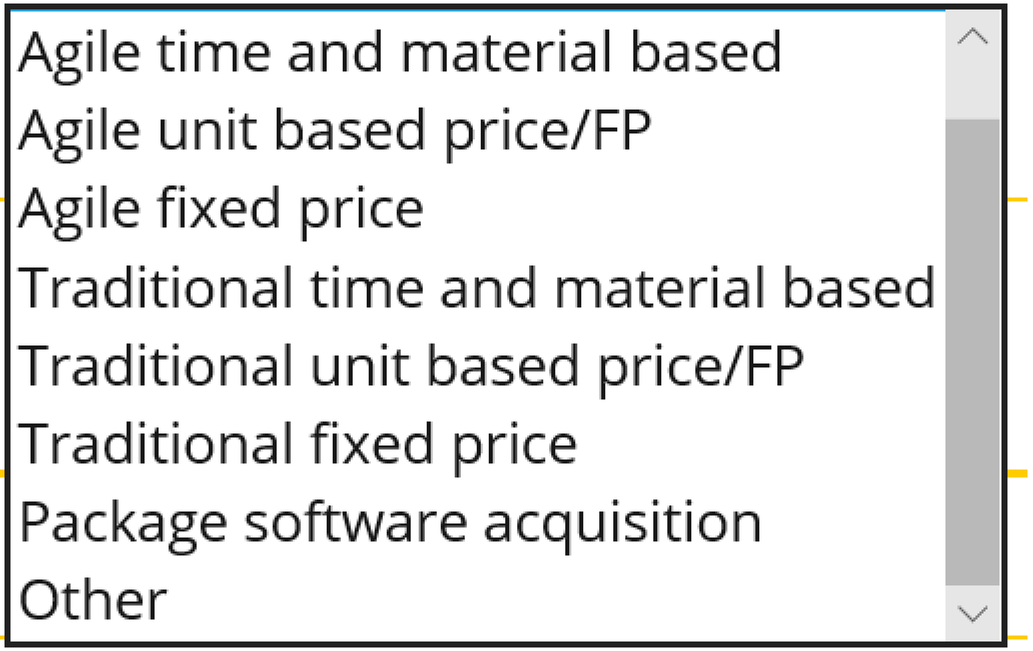
Product life-cycle: Major use cases of Product Document?



Other use cases during the product life-cycle are e.g. acceptance testing, change request, starting to implement a new feature, measuring the achieved productivity of a project or a sprint, changing a developer or maintainer or the whole team.

Impact of development approach: what should be defined to ensure success?

This is relevant only for development and enhancement projects during the product life-cycle. Interesting combinations are:



- Agile time and material based
- Agile unit based price/FP
- Agile fixed price
- Traditional time and material based
- Traditional unit based price/FP
- Traditional fixed price
- Package software acquisition
- Other

Different development approaches allow different maturity for the available specifications. Some of the combinations look insane (i.e. agile fixed price), but they still exist. SCRUM brings into the picture more events to check documents: Starting a new sprint and Retrospective.

What is in a Good Product Document?

This is strong recommendation from FiSMA Scope Manager Forum. Each chapter of the Product Document has great importance for at least one important stakeholder group of the system (users, developers, product owner,…).

- Overview
- User groups
- User stories
- Business processes
- Use cases
- Data functions
- Service functions

You can always challence this recommendation: E.g. “We don’t need overview chapter, because everybody knows what is this system about”, or “We don’t write user stories, because there will be no new users ever, and the testers know how and what to test”.

How to evaluate the “goodness” of Product Document?

FiSMA Scope Manager Forum has agreed about set of best practices, that should be applied to provide good enough system and software specifications, i.e. a Good Product Document.

- Overview
- User groups
- User stories
- Business processes
- Use cases
- Data functions
- Service functions

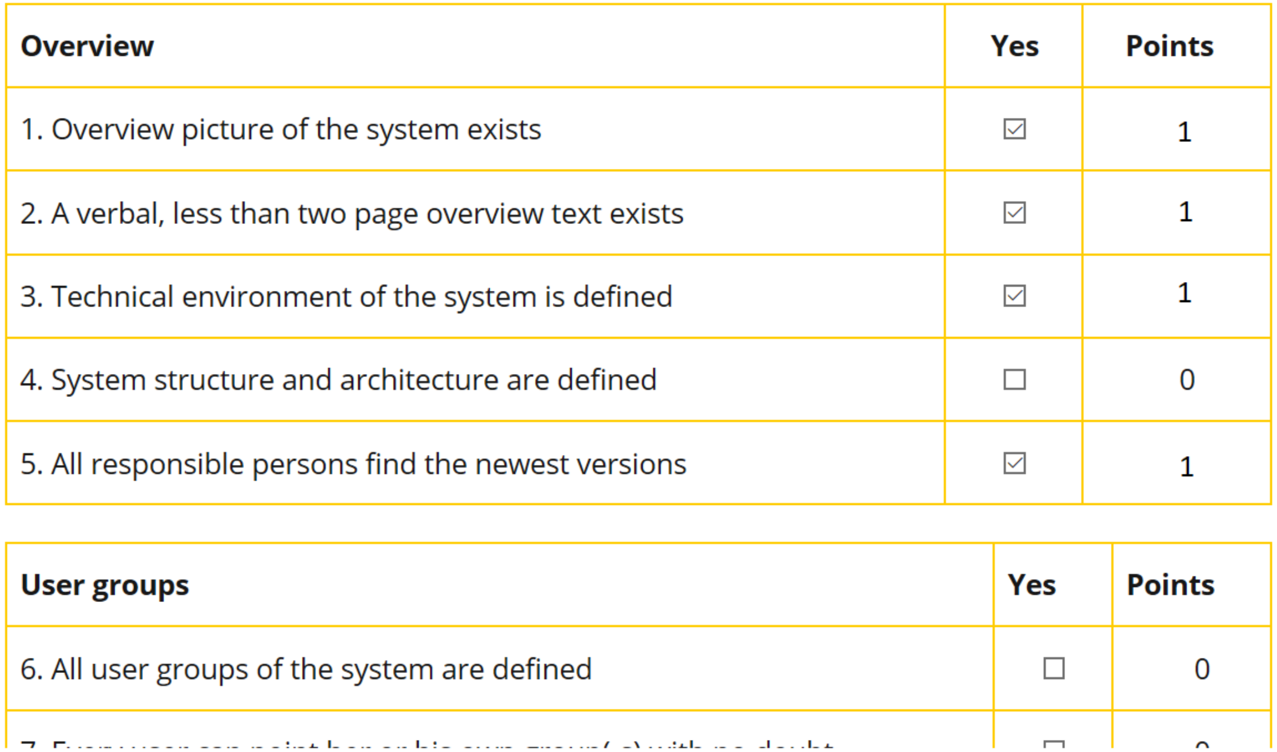
We have collected 50 statements, that are either true or false to all Product Documents. They are divided to all seven chapters: Overview 5, User groups 4, User stories 3, Business processes 8, Use cases 8, Data functions 7, and Service functions 15.

Number of positive answers tells the quality of Product Document (e.g. 23/50)

How does the self-assessment tool work?

Overview with its 5 statements is here the example chapter:

Statements by Product Document chapter



Overview	Yes	Points
1. Overview picture of the system exists	☑	1
2. A verbal, less than two page overview text exists	☑	1
3. Technical environment of the system is defined	☑	1
4. System structure and architecture are defined	☐	0
5. All responsible persons find the newest versions	☑	1

User groups	Yes	Points
6. All user groups of the system are defined	☐	0
7. Every user can point her or his own group(s) with no doubt	☐	0

Number of positive answers is 4/5, telling that there is still some work to do, but not bad at all. However, many people in the "audience" need to get the system structure and architecture definition before they can continue their tasks.

Other examples of best practice statements 1

User stories
10. User stories are written
11. There are stories for each user group
12. All responsible persons find the newest versions

Business processes
13. All pertinent business processes are defined
14. Every user group can be found at least in one process chart
15. The system is one of the actors in every process chart
16. Every activity is allocated to only one actor in a process chart
17. The action chains are unbroken from start to end in every process chart

Use cases
21. All use cases are defined
22. The use cases to be implemented in two next months are defined
23. All use cases are easy to pinpoint in business process charts
24. In most of the use cases there is only one user
25. User and system activities alternate systematically in use case dialogs
26. The dialog chains describe uninterrupted actions
27. Use case definitions point out exceptional quality requirements

Other examples of best practice statements 2

Data functions
29. All data entities are defined
30. The data entities to be implemented in next two months are defined
31. All important business terms are defined and introduced to responsible persons
32. Up-to-date list of implemented entities with their attributes exists
33. All system parameters and their known values are defined
34. Data management plan including all entities exists
35. All responsible persons find the newest versions

Service functions
36. All screens are defined
37. The screens to be implemented in next four weeks are defined
38. All implemented screens are easy to pinpoint in use cases
39. The latest drafts or screenshots are easy to find
40. Menu hierarchy including all the implemented screens exists
41. All different type of printouts are defined
42. The printouts to be implemented in next four weeks are defined
43. The implemented printouts are easy to pinpoint in business process charts
44. All interface services to and from other systems are defined

Conclusions

- Product Document self assessment helps the Product Owner to improve specifications instantly, and gives ideas for process improvement to succeed in long-term.
- A good Product Document allows
  - Effective development
  - Reliable testing
  - Effective user training
  - Effective maintenance
  - Easy functional size measurement
- Through all above, a good Product Document helps Sponsor to
  - Save money
  - Save time-to-market
  - Improve quality

- Note! No documents are written only for measurement purposes, but if you cannot measure functional size of software from documents, no one can develop software from them.

Thank you!

and my apologises that I cannot give you the web-address of the free self-assessment tool now. That's because of minor technical problems with WP, javascrit, new security updates and lack of technical skills. The problems should be solved on next Monday. If you are really interested in trying the tool, please send me an email and I'll send you the link.

Pekka Forselius, Senior Advisor, FiSMA

pekka.forselius@4sumpartners.com

AMMS

Asociación Mexicana de Métricas de Software

CNMES

CONGRESO NACIONAL DE MEDICIÓN Y ESTIMACIÓN DE SOFTWARE

IT CONFIDENCE CONFERENCE

ISBSG

Delivering IT Confidence

Colaboradores:

CIENCIAS

UNAM 60 Años del cómputo en México

CViCom

COSMIC