



Correctness by Construction: Putting Engineering into Software



Rod Chapman
Praxis



Contents

- CbyC – a reminder
- Some good and bad signs in software engineering
- A future?



Contents

- CbyC – a reminder
- Some good and bad signs in software engineering
- A future?



CbyC – A Reminder

- A process/philosophy for software engineering developed and fielded by Praxis over last 15 or so years...
- Emphasis on defect prevention and removal at every stage.
- Use of formal methods and strong static verification where possible.



CbyC – A Reminder

- It works...projects such as SHOLIS, MULTOS CA, Tokeneer published.
- But what next? How can we improve?
- Here are some thoughts on good (and not so good) things I see going on in software engineering...



Contents

- CbyC – a reminder
- Some good and bad signs in software engineering
- A future?



Good things (1)

- **Static Verification renaissance**
 - The “security problem” has caused a huge surge in interest in SV.
 - See work from Microsoft Research, and new tools from Coverity, Klocwork, Fortify, Polyspace, and many more...
 - “Annotations” (aka “Contracts”) are back in fashion!
- **But...**
 - Most emphasis on retrospective analysis of existing code – where we were in UK Mil Aero 15 years ago...
 - Some massive wheels being re-invented...



Good things (2)

- Programming Language designers are addressing high-integrity needs, and starting from scratch:
- Watch out for:
 - Sing# from Microsoft
 - BitC from JHU (See www.bitc-lang.org)
 - Fortress from Sun
- But...
 - Not sure if these will ever impact the mainstream languages



Good things (3)

- Agile Methods

- We were surprised how much XP we were already doing on CbyC projects.
- Most XP practices are just common sense made practical and actually implemented in a lean style.

- But...

- Some XP stuff is not appropriate for high-integrity (e.g. “no design”) where architecture is important for non-functional properties.
- The XP community seem to have totally missed the usefulness of static verification.



Good things (4)

- SEI Personal Software Process (PSPSM)
- Probably the single best software engineering training I have ever been on.
- Zero-tolerance approach to defects, like CbyC, but technology-neutral stance.



Good things (4)

- PSP uses measurement and statistics in a scientific way to judge performance and improvement.
 - What a novel concept! 😊
- Focus on personal professional behaviour is critical.
- But...
 - There is exactly ONE authorized PSP instructor based in the UK...this will take time...



Good things (5)

- Formal model-driven design
- First-generation model-driven design seemed to be based on highly dubious or non-existent semantics.
- Next-generation languages and tools, like UML 2.0, SCADE, and Alloy have much stronger formal underpinning.
 - Very interesting synergy here between these tools and formal programming languages



Contents

- CbyC – a reminder
- Some good and bad signs in software engineering
- A future?



A future?



A future?

- What happens if we put together:
 - CbyC,
 - SPARK and strong static verification,
 - Agile methods,
 - Lean engineering,
 - PSP/TSP style process???



A future?

- What happens if we put together:
 - CbyC,
 - SPARK and strong static verification,
 - Agile methods,
 - Lean engineering,
 - PSP/TSP style process???

- Don't know! But we aim to have some fun finding out...



Praxis

20 Manvers Street

Bath BA1 1PX

United Kingdom

Telephone: +44 (0) 1225 466991

Facsimilie: +44 (0) 1225 469006

Website: www.praxis-his.com, www.sparkada.com

Email: sparkinfo@praxis-his.com