There’s never enough time

Doing requirements under resource constraints, and what Requirements Engineering can learn from agile development

Bernd Waldmann, Phonak AG

PHONAK life is on

- This talk is not about
  Requirements Engineering for *Agile Development Projects*

- This talk is about
  Agile Requirements Engineering for any Development Projects

Agile Software Engineering *inspires* Agile Requirements Engineering

- don’t copy terms & concepts
- map terms & concepts to appropriate analogs
- Then translate insights from Agile SE

- Part I: Concepts
- Part II: Case Study

Key terms in Agile Development

- “Working Product”
- “Customer Value”

… more in the paper

“Product”

Agile Software Engineering

Product := the engineering deliverable … i.e. car, webshop

Intended use = “drive from A to B”, “buy things online”

Product Owner := has *market insight*, empowered to decide (e.g. product manager)

Agile Requirements Engineering

Product := the requirements engineering deliverable … i.e. Requirements Spec

Intended use = “build satisfying product, on time, on budget”

Requirements “Owner” := has *engineering insight*, empowered to decide (e.g. engineering manager)

Insight #1: **Working product**

Agile Software Engineering

- Always have a working product
  - Walking skeleton
  - Add features by iteration
  - BUT: Don’t change existing features in every release

Agile Requirements Engineering

- Always have a deliverable requirements specification
  - Publish a table of contents
  - Add “chapters” by iteration
  - BUT: don’t change existing chapters in every release

“Customer Value”

Agile Software Engineering

Customer :=

- Driver of a car
- Shopper in webshop
- …

Customer value :=

Customer can do *necessary* things

Agile Requirements Engineering

Customer :=

- Developer
- Tester
- Product Manager
- …

Customer value :=

Developer gets *necessary* information, reduces risk of building the wrong thing

Insight #2: **what should you work on now?**

Agile Software Engineering

- Rank by customer value
- First build the feature most important to the customer
- Re-order items based on new market insight

Agile Requirements Engineering

- Rank by business risk
- First specify the parts with highest risk of mis-understanding & … highest demand … most urgent …
- Re-order items based on new risk insight

User Stories / Definition of Done

Agile Software Engineering

User Story :=
small item that delivers value to the user

Define “done”:

- Designed, coded, tested, …
- end user can use product,
- product functions as expected
- end user is satisfied

Agile Requirements Engineering

Requirements Chapter :=
small item that delivers value to the business / developer

Define “done”:

- Analyzed, written, reviewed, …
- developer can use req’ts spec,
- project risk as low as expected
- organization is satisfied

Insight #3 **how to structure your work**

Agile Software Engineering

- Each product release contains a small number of new User Stories, each of which delivers value to the customer

Agile Requirements Engineering

- Each release/publication of the Requirements Spec contains a small number of new Chapters, each of which delivers value to the business

Case Study

Agile RE at Phonak: **Organization**

- R&D organization
  - In-house development
  - many of them were involved in previous platforms generations, too
  - many of them have some domain knowledge

… more detailed information in the paper

Agile RE at Phonak: **Project**

- “Platforms System Definition” project
  - System requirements for generic hearing instruments, accessories, configuration software
  - 4th generation, brown field, inherit >70% from previous generation

… more detailed information in the paper

Business value

Remember: “Customer value” of RE :=
reduced business risk (wrong/late implementation)

Our “triage” questions:

- high business risk without good system requirements?
- lower business risk with good system requirements?

… these are not trivial questions!

Triage strategy

Condition	*Added value thru detailed req'ts*	*Typical example*
was it implemented in previous generation? ("same as …")	low	"classical" hearing aid features: amplification, signal processing
is it new and complex externally visible behavior	HIGH	usability / behavior / interaction with user
is it new and involves multiple products (hearing aid, accessory, …	HIGH	Connectivity between hearing instrument and mobile phone
is it new and must be explored with prototypes	low	PC software features, with focus on GUI

Multi-stage delivery

1. requirements with apparent impact on ASIC definition
   – needed by chip designers, longest lead time
   ~6 months after start
2. requirements with apparent impact on software architecture
   – needed by software architects, long lead time
   ~ 4 months after 1st cut
3. requirements with impact on system (more than a single component), needed by implementers for multiple components – ongoing
4. all other requirements (reverse-engineering), needed by testers – ongoing

- at each stage: triage according to business value

Lessons learned

Lessons learned: **inform your stakeholders**

- inform stakeholders early about multi-stage process
  - avoid “is this final?” questions
  - avoid “xyz is missing” response
  - Table of contents !
- inform stakeholders early about triage strategy
  - avoid “xyz is missing” response
  - avoid “why will xyz not be implemented?” misunderstanding

Lessons learned: **deliver even more frequently**

- 4-6 months release cycle too long
- Enough to deal with limited resources
- Not enough to provide frequent learning opportunities for requirements authors
- New approach: ~monthly release

Lessons learned: **handle multiple levels of detail**

- Slice it horizontally and vertically:
  - Vertical: feature A, feature B, feature C, …
  - Horizontal: high-level concept, mid-level feature list, low-level details

Conclusion

- Applied concepts & insights from agile development to agile RE
- We would do it again, even more thorough
- Know your “customer” & environment,
  i.e. the readers of requirements specs:
  domain knowledge? Green field / brown field?
- Turn a constraint (limited RE resources) into an asset (really focus on what the “customer” wants)

?