Hypotheses Engineering: first essential steps of experiment-driven software development

Jorge Melegati, Xiaofeng Wang, Pekka Abrahamsson

RCoSE/DDrEE 2019
Joint 5th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution
Montreal, Canada
Goals

- Propose Hypotheses Engineering
  - A discipline to better handle hypotheses in experiment-driven software development

- Present a set of research questions to develop practices for this discipline
Experiment-driven software development

- The use of experiments to build features that the user really wants

- Experiments: testing product assumptions applying scientific methods with the purpose of supporting or refuting these assumptions

- Bosch et al. (2018) identified three approach to software development
  - Requirement-driven development
  - Outcome/data-driven development (experiment-driven)
  - AI driven development
Requirement-driven development

- Requirements Engineering is an important component of the process
  - Concerns the identification and documentation of stakeholders needs

- Even in agile methodologies, it is still present
  - Done more often and with closer and more frequent contact with the customer
  - User stories are an artifact for requirements
Experiment-driven development models

HYPEX (Olsson and Bosch, 2014)

RIGHT (Fagerholm et al., 2017)
Experiment-driven development models

- All were influenced by the Build-Measure-Learn loop from the Lean Startup methodology (Ries, 2011)
- A common denominator:
Requirement vs Experiment-driven software development

1. Requirements Engineering activities
2. Identify, specify, and prioritize hypotheses
3. Design the software
4. Design the experiment
5. Code
6. Execute
7. Test
8. Analyze

Learning
Hypotheses Engineering

- A discipline to handle hypotheses in a similar way requirements were handled in requirements-driven software development
- The team should be able to:
  - Identify hypotheses
  - Analyze them regarding meaningfulness and duplication
  - Prioritize them in order to minimize waste of time and resources
  - Communicate hypotheses to the development team
Hypotheses Engineering

- A discipline to handle hypotheses in a similar way requirements were handled in requirements-driven software development
Hypotheses generation

- Discussed briefly in the literature so far
- HYPEX: first practice is feature backlog generation
  - Product management and product development staff “based on their understanding of customer needs and strategic business goals” generate features that may bring value to customers (Olsson and Bosch, 2014)
- RIGHT
  - “analysis and product owner work with a data scientist role [...] to communicate the assumptions of the roadmap and map the areas of uncertainty which need to be tested” (Fagerholm et al., 2017)
- Common theme: business goals and vision to determine which assumptions have to be experimented
RQ1: How can software development teams systematically define hypotheses based on business goals and vision, and own previously accumulated learning?
Hypotheses documentation

- In traditional requirements-driven development is really important
- Agile
  - There is not formal documentation
  - Long requirements documents are replaced by user stories
- HYPEX: the team should specify how the feature adds value to the customer and support
- RIGHT: experimentation plans and learning as information artifacts not necessarily formally documented
Hypotheses documentation

RQ2: What artifact could be useful to represent hypotheses and support experiments creation?

RQ3: How could a hypothesis artifact be used to keep experiment useful information?
Hypotheses analysis

- Concerns similar to requirements:
  - Hypotheses not well-explained
  - Not possible to perform an experiment
  - Consistent and not-duplicated
Hypotheses analysis

RQ4: How could teams understand if a hypothesis can be practically tested using an experiment?

RQ5: How could teams understand dependencies among different hypotheses?

RQ6: How do hypotheses evolve over the time?
Hypotheses prioritization

- A common theme among scientific and industry authors
- In startups, it is critical to test the most crucial hypotheses first
- Several techniques:
  - Leap-of-faith assumptions (Ries, 2011)
  - Prioritization matrix (Gothelf and Seiden, 2013)
- No work has evaluated these techniques
Hypotheses prioritization

RQ7: Are current assumption prioritization techniques effective?

RQ8: Could requirements prioritization techniques be adapted to hypotheses in experiment-driven development?
Conclusions

- In this position paper, we proposed a Hypothesis Engineering discipline
  - Tailored to experiment-driven software development
  - Contrast to Requirements Engineering

- Notes:
  - The steps does not mean that it will follow a linear approach
  - Maybe there are different techniques for different stages of product development