Relevance driven development of analysis software for information systems based on data with a hierarchical structure and a long-tail distribution

Wolfgang Fahl

June 2021

1 Abstract

Quite a few IT projects suffer from trying to achieve too much too early and getting stuck in a situation where there is no delivery for a long time. We assume the reason is that prioritization decisions are not based on honest relevance considerations.

The MVP (Minimal Viable Product) agile approach has tried to mitigate this situation but often lacks a systematic approach to what the MVP should contain.

We propose a relevance driven approach to systematically increase the coverage of a system to the viable point and then to an economically reasonable state thereafter.

We assume this approach will work well in the context of migration projects with a hierarchical structure of entities and a long tail distribution of the data.

2 RQ Wiki Text

`https://rq.bitplan.com/index.php/Relevance_driven_development`

3 Background

The need for information systems based on data with a hierarchical structure and a long-tail distribution is quite common. This is due to the long history of applying traditional classification logic to analyze systems in many fields. Examples are the biological hierarchy based on Carl Linnaeu's Systema Naturae [**Linne¨1758**]

3.1 Goals

4 Related Work

4.1 Minimum Viable Product (MVP)

"Value-related feedback from real customers is needed during software development and maintenance, and decision-making should be increasingly based on empirical evidence acquired through experiments. Getting such value-related feedback usually requires a so-called minimum viable product." [**Muench2013**]

4.2 Traceability

Traceability allows to connect to software engineering artifacts in a top-down or bottom-up style. The top-down style is called post-requirements-traceability and allows to trace from a requirement to the artifacts that were generated later based on the requirement. The bottom-up style is called pre-requirements-traceability and allows to find out the motivating requirement for an artifact. Examples for for a typical traceability chains are:

requirement -¿ acceptance criterion -¿ testcase -¿ database record -¿ database column requirement -¿ model -¿ database table -¿ database column bug report -¿ scenario -¿ database column -¿ database table -¿ model -¿ requirement

In most software engineering project the traceability chains are incomplete / broken leading to extra effort e.g. in the case of deciding questions like "can this database column be deleted?"

[**Ramesh˙2001**]

4.3 Information Quality

[**DBLP:conf/iq/JarkeV97**]

4.4 Other

https://ieeexplore.ieee.org/document/6612885?arnumber=6612885 https://www.growingagile.co.za/wp-content/uploads/2013/06/Black-Swan-Farming-using-Cost-of-Delay.pdf

Feature Driven Development

https://link.springer.com/chapter/10.1007/1-84628-262-4$_9 http://ceur-ws.org/Vol-1115/src5.pdf$

Prioritzing software requirements

A systematic approach for prioritizing software requirements https://www.sciencedirect.com/science/article/

[**Lehtola2004**]

Value Based Software Engineering - Barry Boehm

5 Relevance

5.1 Definition

According to [**websterRelevance**] relevance is "the ability (as of an information retrieval system) to retrieve material that satisfies the needs of the user".

5.2 Relevance levels

We define three different levels of relevance for usecases: standard cases, corner cases and exotic cases.

There are two main criteria to classify the relevance of a usecase:

- usage frequency
- effort/benefit ratio

The usage frequency describes how many the usecases instances / use case scenarios for the usecase exist.

Example classification based on the "long-tailedness" of a usecase:
0-8080-9696-100

The Pareto rule may be applied here several times in a row to select the quantiles that define the relevance levels.

The first 8 deciles would therefore include the most common/ordinary standard cases while the last two deciles would have the corner and exotic cases. Applying the Pareto rule on the last two deciles would then lead to the 15 centiles from 80-95 for the corner cases and the last 5 centiles for the exotic cases.

A rare use case in the corner case / exotic case category can still be promoted to be a standard use case if the effort/benefit ratio is very good that is although it occurs rarely there is a high value compared to a low effort in implementing it in as a service of a digital system.

An example would be a yearly report of activities. This use case might be used only once a year but still be very valuable and not much effort to implement.

5.3 Motivation and Acceptance

Acceptance (in sociological terms) is defined as the vector addition of projection and identification.

Projection is the believe in the performance/functionality/efficiency of an entity (in this case a system) Identification is the confidence/trust into the entity/results given

Its important to note that digital systems mostly take the acceptance from projection. Identification is usually drawn from social relationships.

The use case "recommend a conference" for instance will have much more acceptance when it's offered as a service of a peer then when it is offered by

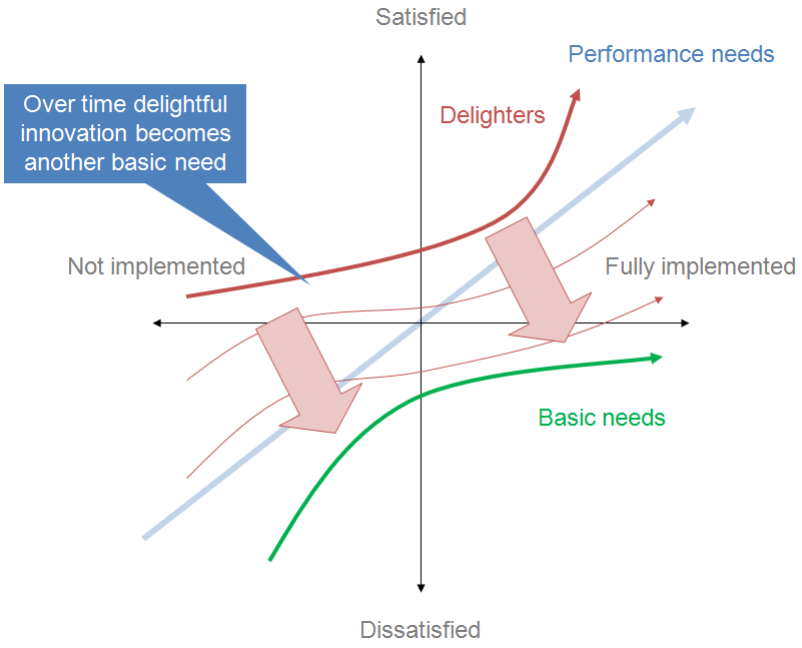

Figure 1: Kano Model

a digital system. The higher acceptance of the social peer is based on the much higher identification. Even if a system would have much better results to improve the projection part it will be hard to compete with the traditional human based interactions.

5.4 Applying the Kano Model

Another aspect of the use cases to be selected for implementation in a digital system is the Kano Model[1] (Figure 1). There is some text required. What? Why? How?

[1] `https://en.wikipedia.org/wiki/Kano_model`