Store og komplekse informasjonssystemer

Gruppetime INF3290 uke 43

kribrae@ifi.uio.no

Agenda

- Interactive Systems: Bridging the Gaps Between Developers and Users (Grudin)
- Software engineering beyond the project – Sustaining software ecosystems (Dittrich)
- Generification as Ecology (Nielsen)

Interactive Systems: Bridging the Gaps Between Developers and Users

Grudin

Interactive Systems: Bridging the Gaps Between Developers and Users

Artikkel fra 1991

En del organisasjonsstrukturer og utviklingsprosesser ble definert før brukermedvirkning var vanlig

Fossefallsmodellen for systemutvikling var dominant, men denne er lite egnet for brukermedvirkning

(Grudin, 1991)

Interactive Systems: Bridging the Gaps Between Developers and Users

Three development contexts provide a framework for understanding interactive software projects:

- Competitively bid
- Commercial product
- In-house/custom development

(Grudin, 1991)

Identifying developers and users

Figure 1 presents three paradigms for software project development based on when users and developers are identified.

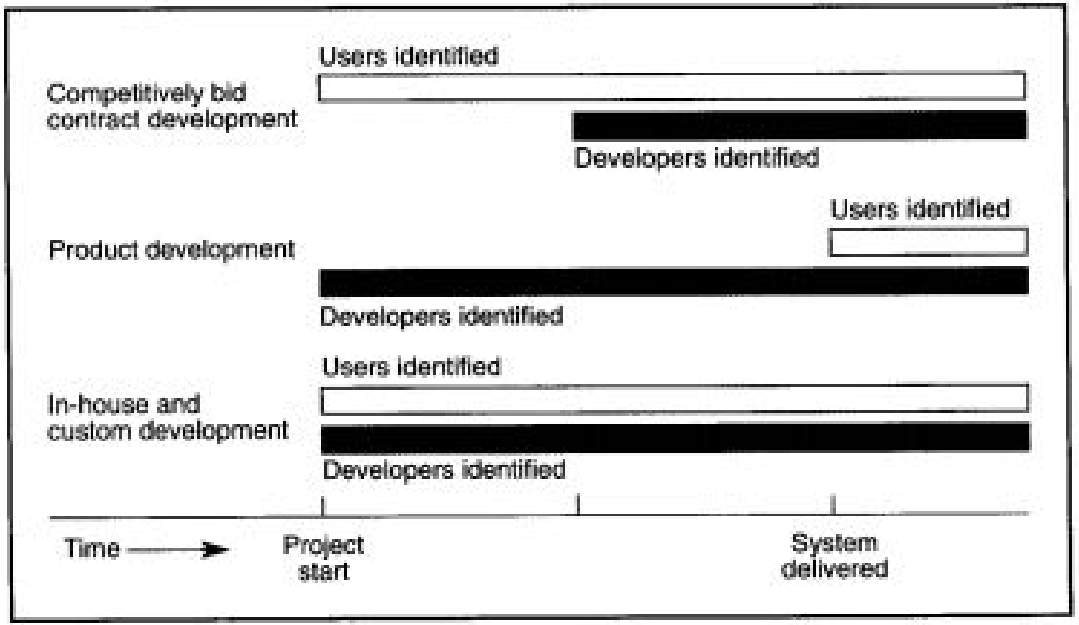

Figure 1. Project time lines with points of user and developer identification.

(Grudin, 1991)

Identifying developers and users

- In contract development or software acquisition, the user organization is known from the outset, but the development organization is identified after a contract is awarded.
- In product development, also called commercial off-the-shelf or shrink-wrap software development, the developers are known from the outset, but the users often remain unknown until the product is marketed
- Finally, in in-house development, also called internal or information systems development, both the users and the developers are known at the project outset.

(Grudin, 1991)

Identifying developers and users

The timing gap between user and developer involvement in a product or competitively bid contract development project is an obstacle to collaboration.

Many aspects of development practice have evolved to “communicate across time” - to bridge the gaps shown in Figure 1 - as well as to bridge the physical distances that often separate developers and users

(Grudin, 1991)

Three development contexts

- Contract development: focus on software methods
  - Waterfall model
  - The reliance on specification documents imposes a “wall” between users and developers
- Product development: focus on the user interface
  - The market and the customer in product development represent a thick hedge. Information about users’ needs gets through, but it takes time and is muffled. Individual voices are not heard.
- In house and custom development: focus on user participtation
  - This paradigm affords an obvious potential advantage to user participation in design: The developers and the eventual users are known when the project is initiated.
  - Potential obstacles to success do exist. Projects for multiuser systems are more challenging than single-user applications
  - an internal development project must be accepted by a set group of users

(Grudin, 1991)

Factors influencing interactive systems development

- The size of the development company or organization.
- The charter of the company or organization
- Organizational structures and procedures.
- The nature of the system user population
- Mediators: additional partners in the development project
- Commitments and agreements among the groups involved
- Societal conditions and change over time.

(Grudin, 1991)

Focusing on users: Opportunities, obstacles, and mediators

- Contract development. User involvement faces the most formidable obstacles in this context, especially with fixed-cost competitively bid contracts.
- Product development. This context provides a strong incentive to increase usability, but user involvement is a challenge when the potential users are numerous yet faceless.
- In-house development. This development context appears to offer good prospects for collaboration among users and developers, but the challenges are substantial. One challenge is that internal development is often modeled on contract development, adopting methods that work against user involvement.

(Grudin, 1991)

Software engineering beyond the project – Sustaining software ecosystems

Dittrich

Software engineering beyond the project – Sustaining software ecosystems

Context: The main part of software engineering methods, tools and technologies has developed around projects as the central organisational form of software development. A project organisation depends on clear bounds regarding scope, participants, development effort and lead-time. What happens when these conditions are not given? The article claims that this is the case for software product specific ecosystems. As software is increasingly developed, adopted and deployed in the form of customisable and configurable products, software engineering as a discipline needs to take on the challenge to support software ecosystems

(Dittrich, 2014)

Introduction

This article takes addresses a different question: How do software development processes change when developing software in and for a software ecosystem?

… it is apparent that many of the software engineering methods, tools and techniques do not support the development and evolution of software products. The analysis proposes that the problem might be due to the focus on the project by software engineering methods and techniques.

The CMMI defines a project as '... a managed set of interrelated resources that delivers one or more products to a customer or end user. It has a beginning and an end and typically operates according to a plan'

(Dittrich, 2014)

Introduction

Software products

The delivered product is rather a half product that has to be configured and customised to a specific context

There is no decided beginning of the evolution, and the goal is to keep the product useful and attractive to its users over a long time, rather than delivering it once.

(Dittrich, 2014)

The four software product ecosystems

- UIQ – developing a user interface framework for high-end mobile phones
- Microsoft Dynamics – evolving an ERP system for small and medium sized companies
- SIM – Water and Environment Simulation
  - Not for-profit organisation
- UIB – software distribution for local area networks
  - Applications + open source product

(Dittrich, 2014)

Analysis results

- Relating design across different constituencies
  - Interlacing design constiuencies
  - Keeping connected with the context of use
- Architectures maintained by communities of practice
  - Architecturing as skill of a group of architects/core developers
  - Interfaces separating and bridging the different design constituencies
- Cycles within cycles
  - Combining different rhythms to juggle different needs
  - Conflicting drivers of the evolution process
  - Developting organisational structures to balance different qualities

(Dittrich, 2014)

Software development beyond the project

- Interwining practices of design and use
- Deferring design to 3rd party developers, organisational implementation and use
- Communication between developers, implementers and users
- Managing an overlay of different evolution cycles
- Informal knowledge management practices

(Dittrich, 2014)

Conclusions and future research

Although size, kind of software and business models differ, the commonalities are striking:

– The software products are developed in interaction with a product related ecosystem where part of the design is deferred to other actors closer to the concrete use context. However, real world ecosystems do not only consist of the product developer providing a platform for 3rd party developers together with the product but includes several layers of actors customising and configuring the software product. Often the product can be regarded as part of one or several ecosystems itself.

– Innovation takes place across the whole ecosystem. Contact with users and other actors is therefore important to keep the innovative edge of the software product.

– The technical design and architecture exists as practice of architects and core developers rather than as explicit documentation. (Dittrich, 2014)

Conclusions and future research

Although size, kind of software and business models differ, the commonalities are striking:

– The interfaces for configuration and customisation both separate and bridge the different design constituencies. They are contested and need to be maintained continuously.

– To juggle different and sometimes conflicting development drivers – bug fixes, new features, technical re-engineering – the development processes consist of an overlay of different development cycles.

– As the companies have to balance different qualities of their products, they struggle with finding the right organisation of the development team. The organisation might change depending on different emphasis in the development, which, in turn, depends on the dynamics of the ecosystem.

(Dittrich, 2014)

Generification as Ecology

Nielsen

Generification as Ecology

Making software packages generic and implementing them in organizations is a key activity in systems development. This paper reports from a study of the implementation of a web shop software package in a multinational Telecom company. The aim of the paper is to improve our understanding and conceptualization of such processes.

This paper is an argument to break up the restricted project focus in information systems research and appreciate the non-lonely nature of organisational implementation projects.

(Nielsen, 2016)

Introduction

The challenge of inflexible legacy systems, the cost and scarcity of internal IT resource and the promise of significant cost-savings motivate organisations to use software packages. Developed by software vendors, software packages are not particularly tailored for one organisation, but sensibly designed to serve a range of different organisations and contexts.

To achieve their generic nature, software packages must be based on a future-oriented architecture and capture requirements from a wide audience of organisations over time.

(Nielsen, 2016)

Introduction

The existing body of research on the implementation of software packages in global organizations (standardisation) and the development of standardized software packages (generification) has focused on the conflict between local and global needs and strategies or incorporating local requirements into global standards.

The main contribution of the paper lies in [...] by conceptualizing the implementation of software packages as processes unfolding in parallel with other implementation processes and generification as a process involving the implementing organization beyond being a source of requirements

(Nielsen, 2016)

Generification as Ecology

In this paper, we argue for taking an ecological perspective on generification. The concept of ecology is borrowed from biology and used by information systems researchers to describe networks of diverse actors influencing each other's and being mutually dependent within a specific (eco)system.

Software platforms involve software development beyond making the platform. Not only will platforms be maintained, but they also open up for continuous innovation and customization to meet local needs

(Nielsen, 2016)

Case: Telco

The particular focus in this paper is an initiative by the Industrialisation Unit (IU) in the corporate headquarters in Telco to renew the local, independent and not standardised web shops in the various Telco operations. This initiative was grounded on activities dating back to 2009, leading up to an attempt to implement a standardised web shop software package during 2012. The initiative turned out to be overly challenging and ultimately failed to reach its primary aims, at least in terms of establishing a globally standardised web shop.

Telco has not previously coordinated and standardised their web shops globally. This has led to there not only being 11 different web shops in terms of software, but also 11 different web shop strategies and organisations along several dimensions (Nielsen, 2016)

Local and global requirements

Through meetings and discussions with the operations, a broad range of requirements was identified.

While swapping the software to a standardised one did not seem so challenging, the third parties also provided functions in the value chain, such as stock management, distribution and servicing of mobile phones. This setup is much harder to change.

(Nielsen, 2016)

Relating to other initiatives: Heterogeneity and complexity

The web shop project did not unfold in a vacuum, but in parallel with and in interaction with a range of other local projects and initiatives by IU.

A large parallel initiative was run by the IU to identify ways to collocate server parks as well as maintenance and operations staff globally in Telco.

The lack of an operative shared service centre not only introduced risk in the business model of the web shop project, but also mandated the initiative to request a software package that could be run locally in the short term, and equally important in regionally or globally service centres in the longer term.

(Nielsen, 2016)

Relating to other initiatives: Heterogeneity and complexity

The IT unit of the IU was also running a Telco technology architecture project. Their plan was to introduce a standardised architecture and back-office stack throughout Telco

Telco did not have a global IT architecture and the local architectures of its operations were not optimal, typically struggling with legacy systems and accumulating complexity

the architecture project did not have a fixed functional architecture to support the web shop initiative, but only one in the making.

(Nielsen, 2016)

Relating to other initiatives: Heterogeneity and complexity

Local projects

To summarise, numerous global and local initiatives influenced and partly overlapped the web shop implementation projects as well as other initiatives.

(Nielsen, 2016)

Planning and running the web shop RFQ

- Software package + Telco layer
- Terminating the process
- Another important factor was the failure of parallel projects in delivering e.g. a common architecture, an operative service centre and a web shop for digital content. With these other components in place or in a more mature stage, the web shop project may have ended differently

(Nielsen, 2016)

Discussion

What we have seen is that the generification space is an ecosystem composed of multiple parallel implementation projects and multiple levels of generification activities. These loosely interconnected initiatives and projects are not necessarily well coordinated while at the same time being dependent on another for their success.

This architecture initiative showed interest for the web shop project and offered input to the architecture of the web shops and its relation to other software components in Telco. In doing so, it delayed the web shop project slightly. More important, when the architecture project was terminated, the web shop project was left with uncoordinated architectures in the 11 operations.

(Nielsen, 2016)

Discussion

If the operation of a standardised web shop is not made right, it is likely to fragment as a standard over time because the operations will develop their own functionality at the cost of the “Telco layer” and the software package. The “Telco layer” adds complexity to this, since it is not always trivial to decide where new functionality should be implemented— locally, in the “Telco layer” or in the software package.

Another level of complexity in this setting was the approach to centralisation in the implementation (software) and the operation of the web shops (operational model).

(Nielsen, 2016)

Plan for neste uke

Forelesning: Case: HISP

Gruppetime: Fremføringer