jamk.fi

DEVELOPER FEEDBACK MECHANISM

Case: The Qt Company

Mika Holm

Bachelor's Thesis
May 2015

Degree Programme in Media Engineering
School of Technology, Communication and Transport

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author(s)	Type of publication	Date
Holm, Mika	Bachelor's thesis	22.05.2015
		Language of publication: English
	Number of pages: 54	Permission for web publication: x

Title of publication
Developer Feedback Mechanism
Case: The Qt Company

Degree programme
Media Engineering

Tutor(s)
Huotari, Jouni

Assigned by
Need for Speed-program: JAMK, Marko Rintamäki; Digia Oyj, Mika Myllymäki

Abstract

This thesis was carried out as a part of Need for Speed Program and it was assigned by The Qt Company, a subsidiary owned by Digia Plc. Qt is an open source cross-platform application development framework marketed by The Qt Company, which is co-developed by a community. Although there are free versions of the software available, The Qt Company sells commercially licensed versions of the software that include additional features and support services.

In software development, the user feedback has great significance in the functionality of the software. The aim of this thesis was to study the feedback channels available for software developers and customers and to find possible ways to improve the feedback gathering process by using i.a. service design tools. Also, the possibility of automated usage reporting and crowdsourcing incentives were considered. In this thesis the feedback channels were mostly limited to those that could be used to report failures in the product or to request features.

The current situation was analysed by comparing the feedback channels to those of the competitors. Also, the feedback process was charted by using a data flow diagram and touch points of the feedback event were visualised from the customer's point-of-view in a customer journey map.

It was noticed that the feedback system was built well in most parts, however, there were still different ways to improve the user experience of the feedback process and the presentation of feature requests.

Keywords/tags (subjects)
Service design, user experience, feedback, bug, issue, bug report, customer journey, software development, Qt

Miscellaneous

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Holm, Mika	Opinnäytetyö	22.05.2015
	Sivumäärä 54	Julkaisun kieli Englanti
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Developer Feedback Mechanism
Case: The Qt Company

Koulutusohjelma
Mediatekniikka

Työn ohjaaja(t)
Jouni Huotari

Toimeksiantaja(t)
Need for Speed-ohjelma: JAMK, Marko Rintamäki; Digia Oyj, Mika Myllymäki

Tiivistelmä

Opinnäytetyö toteutettiin osana Need for Speed -ohjelmaa, ja sen toimeksiantajana oli The Qt Company, Digia Oyj:n omistama tytäryhtiö. Qt on The Qt Companyn markkinoima avoimen lähdekoodin alustariippumaton ohjelmistojen kehitysympäristö, jota kehitetään yhteistyössä yhteisön kanssa. Vaikka ohjelmistosta on saatavilla ilmaisversioita, The Qt Company myy tuotteeseen kaupallisia lisenssejä, jotka sisältävät lisäominaisuuksia sekä tukipalveluja.

Ohjelmiston kehitystyössä käyttäjien antaman palautteen merkitys on suuri ohjelmiston toimivuuden kannalta. Opinnäytetyön päämääränä oli tutkia ohjelmistokehittäjien ja asiakkaiden käytössä olevia palautekanavia sekä etsiä mahdollisia keinoja parantaa palautteenkeruuprosessia käyttäen mm. palvelumuotoilun työkaluja. Myös automatisoidun käyttäjätiedonkeruun mahdollisuutta sekä joukkoistamisen kannustimia pohdittiin. Työssä palautekanavat rajoitettiin pääosin koskemaan kanavia, joiden kautta käyttäjät voivat raportoida virheitä tuotteessa tai antaa kehitysehdotuksia.

Nykytilanteen analyysi toteutettiin vertailemalla palautekanavia kilpailevien yritysten tarjoamiin vastaaviin kanaviin. Sen lisäksi kartoitettiin palauteprosessin kulkua tietovirtakaavion avulla sekä tutkittiin palvelun kosketuspisteitä palautetapahtuman aikana visualisoimalla palvelupolku asiakkaan näkökulmasta.

Työssä havaittiin, että palautejärjestelmä oli toteutettu pääsääntöisesti hyvin, mutta palautetapahtuman käyttäjäkokemusta sekä parannusehdotusten esitystapaa voitiin vielä parantaa erilaisin keinoin.

Avainsanat (asiasanat)
Palvelumuotoilu, käytettävyys, palaute, vika, ongelma, vikaraportti, palvelupolku, ohjelmistokehitys, Qt

Muut tiedot

Contents

Figures

Tables

1 Finding Better Practices to Collect Feedback

1.1 Background

Multi-platform software frameworks are very complicated and although they are thoroughly tested, there may still exist annoying or even fatal errors in the code that end up for the users to find. In a way, the user is often doing the final testing of the product. Faulty programs may lead to loss of customers, when upset users search for alternative products. Therefore, it is very important to effectively collect bug reports from the users and quickly provide fixes or workarounds to ensure user satisfaction.

Also, the product may lack some features, which the end users would expect it to have or which would positively differentiate it from the competition and fetch new paying customers for the company. Customer feedback is a good way to find out, how the product should be developed further. Therefore, it is important to make sure users can easily give concise feedback of the product and that every relevant piece of information is collected.

The aim of this thesis is to analyse the already existing feedback channels of The Qt Company and to find out better practices to collect feedback. This paper compares the practices of The Qt Company with those of its rivals and visualises the map of current feedback channels. New practices are then discovered and proposed by using e.g. service design tools. This paper also ponders the possibility of automated usage reporting as a way to collect feedback and the incentive system of the Qt community.

1.2 The Qt Company and the Qt Framework

This task was assigned by The Qt Company, which is a subsidiary of Digia Plc. The Qt Company develops a cross-platform application and UI development framework - simply known as the Qt. The main idea behind Qt is that the developed code can easily be run on desktop, mobile and embedded platforms and on different devices. (Qt - About Us 2015.) The framework is developed together with the Qt Project under open governance terms. The Qt Project consists of a number of companies and individuals. (KDE Free Qt Foundation n.d.)

The Qt Company sells the commercial licences of the Qt platform, but there are free open source licences available for the users. The commercial licences also include various support services, modules, tools and other features that are not available for the free community version users. (Download Qt 2015.)

1.3 Need for Speed-Program

This thesis was carried out as a part of Need for Speed research program (N4S), in which JAMK University of Applied Sciences is participating. The program is established by Digile Oy and it consists of 11 large industrial organizations, 15 small and medium-sized enterprises and ten research institutes and universities - including the assigner of this thesis, Digia. N4S is a four-year program (2014-2017) that is funded by Tekes - the Finnish Funding Agency for Technology and Innovation. (Digile N4S n.d.)

The main goal of the program is summarised as: "N4S adopts a real-time experimental business model, and provides capability for instant value delivery based upon deep customer insight (ibid.)."

The program focuses on three major areas i.e. work packages (WP) (ibid.):

- WP1 - Delivering Value in Real Time,
- WP2 - Deep Customer Insight,
- WP3 - Mercury Business.

WP1's goal is to find new approaches, methods and tools, to quickly and cost-efficiently design, create and evaluate mock-ups or prototypes of new products and services. The customers can then test and evaluate the products without heavy investments in development. (Paradigm Change – Delivering Value in Real Time n.d.)

The Deep Customer Insight (WP2) is about gaining a deep understanding of customer needs and behaviour, and using the information to improve the business hit rate. WP2 aims to find out the true customer value of services and features. This is achieved by creating new mechanisms and tools to collect usage and behavioural data and feedback. Mechanisms can be used to i.a. test and validate new ideas with live users, analyse and visualise collected data, and predict customer behaviour by

using the existing data. One way of achieving customer insight is to develop and take advantage of customer feedback systems. (Deep Customer Insight n.d.)

Mercury Business focuses on adapting new business conditions and searching for new business opportunities with minimum effort. WP3 i.a. detects trends, predicts the future business environment, finds ways to eliminate unnecessary activities, and captures new business ideas by gaining deep customer understanding. (Mercury Business n.d.)

This thesis fits mainly in the area of Work Package 2, because it studies new and existing ways to collect user feedback and usage data. Also, understanding customers' needs and behaviours is one of the key elements required to complete this task.

2 Feedback and Service Design

2.1 Services

2.1.1 Characteristics of Services

Kandampully and Khanh (2004) think that in IT business, the service quality of a company has become an important factor when trying to achieve market differentiation and competitive advantage. Service quality affects customer satisfaction and the prices the customers are willing to pay. (P. 390.)

Edgett and Parkinson (1993, 22) sum up four characteristics of services that distinguish them from goods: **intangibility**, **heterogeneity**, **inseparability** and **perishability**. These are commonly known as the IHIP-characteristics. However, it is debatable if services should be dissociated from goods, e.g. Lusch and Vargo (2004, 9) see goods as "distribution mechanisms for services".

"'Intangibility' implies that a service is experienced; it is rendered; physical ownership cannot occur (ibid. p. 23)." Edgett and Parkinson also quote Bateson (1977), who states that services cannot be touched, tasted, smelt or seen and they can be even mentally intangible (op. cit. p. 22).

Berry, Parasuraman and Zeithaml (1985, 34) describe the heterogeneous qualities of services by stating that: "the quality and essence of a service can vary from producer to producer, from customer to customer, and from day to day." Edgett and Parkinson also add that services are hard to standardise because of that variance (op. cit. p. 26).

Perishability means that unlike goods, services cannot be stored for a later use nor they can be produced before they are required. A service must be used when it is available or the service capacity is wasted. (Carson & Rushton 1989, 26.)

According to Edgett and Parkinson, Kotler (1982) states that a service cannot exist without the provider of the service. With services, production and consumption occur at the same time. (Op. cit. pp. 24-25.) Because of this inseparability, services are first sold, then produced and consumed, whereas goods usually are first produced, then sold and lastly consumed (op. cit. p. 25).

Because of these characteristics, a user cannot test the outcome of a service beforehand. Also the outcome cannot be repaired or returned to the provider in case it appears to be unsatisfactory. (Kandampully & Khanh 2004, 390.)

2.1.2 ITIL and Services

Information Technology Infrastructure Library (ITIL) is a framework that describes best practices in IT service management, including the governance, management and control of IT services. ITIL aims to continually measure and improve the quality of an IT service from the perspective of business and a customer. The current version of ITIL is known as the ITIL 2011, which is a revised version of ITIL Version 3 of 2007. ITIL consists of five core publications: **ITIL Service Strategy**, **ITIL Service Design**, **ITIL Service Transition**, **ITIL Service Operation** and **ITIL Continual Service Improvement**. Each of these publications covers a part of the service lifecycle (see Figure 1). (Cartlidge, Rance, Rudd, Shaw, Smith, Wigzel & Wright 2012, 6-7.)

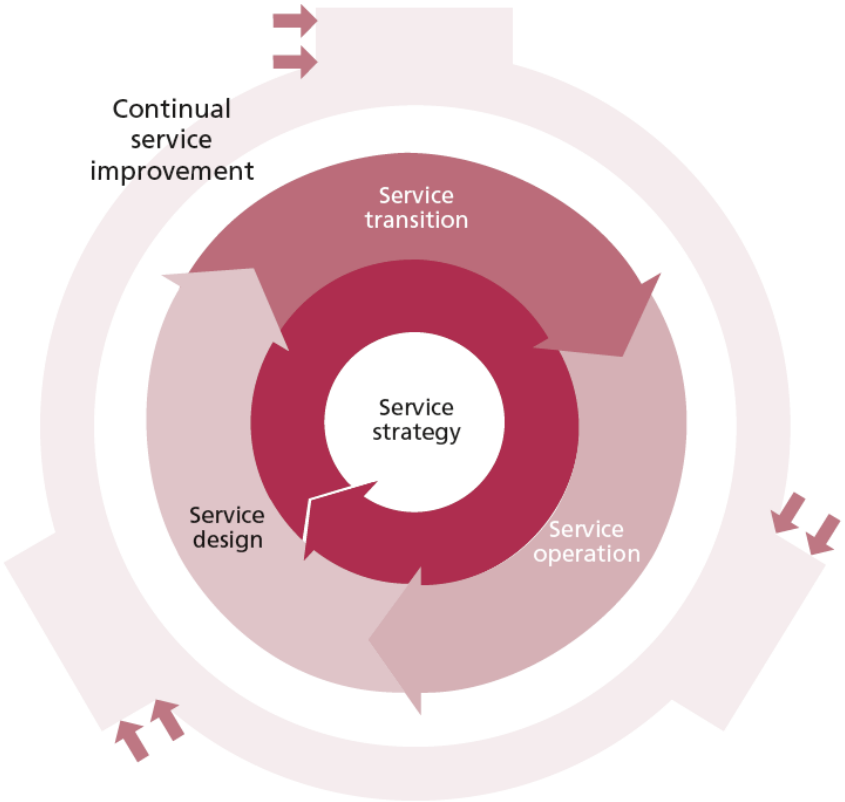


Figure 1. The ITIL service lifecycle (Cartlidge et al. 2012, 7.)

Service Strategy establishes an overall strategy for services and service management (ITIL® Glossary of Terms, Definitions and Acronyms 2007).

Service Design contains guidance on designing and development of services and service management practices. It also contains design methods for converting strategic objectives into a portfolio of services and service assets. (Cartlidge et al. 2012, 21.)

Service Transition ensures that new or modified services meet the service strategy and service design. This stage includes planning and managing changes and releases, risk management, knowledge transfer and ensuring that expected business value is delivered. (Ibid., 31.)

Service Operation aims to deliver the agreed levels of service to users and to actually deliver value to the business. Service Operation also includes following functions: service desk, technical management, application management and IT operations management. (Ibid., 40.)

Continual Service Improvement maintains value for customers by continually evaluating and improving the quality of services (ibid., 50).

According to Cartlidge et al. (2012) the ITIL definition of a service is “a means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks.” Service management is defined as “a set of specialized organizational capabilities for providing value to customers in the form of services.” (Pp. 4-5.)

2.2 Feedback

The importance of feedback is undeniable. For example the lean start-up methodology values customer feedback over intuition and it starts collecting customers’ views and experiences in a very early phase of the product development. The lean start-up uses minimum viable products (MVP) – products that contain only critical features and can be developed rapidly – to collect feedback from customers. The feedback is then analysed and the MVP is revised to meet the customers’ needs and after that,

the development cycle starts over. (Blank 2013, 66-72.) Customers' interaction with the product produces both qualitative and quantitative feedback and data. This build-measure-learn feedback loop is one of the most important components of lean start-up methodology and it aims to save time and money. (Ries 2011, 63.)

In their study, Mattila, Tambyah and Wirtz (2009) learned that negative feedback is more important for service improvement and performance evaluation than positive feedback, because it effectively highlights the weaknesses in the service delivery system. Customers may also give important new ideas for developing the services further in their complaints. Positive feedback, on the other hand, has mostly motivational effect on employees. (Pp. 371-379.)

The solicitation of feedback can be active or passive. In active solicitation, the customer is motivated by direct interaction. The tools of collecting active feedback may include surveys, interviews and focus groups. The feedback is collected passively if the customer is the one to initiate the process. (Sampson 1996, 602.)

When the user accesses an issue tracker to report a fault in a product or a service, it is considered as passive solicitation of feedback, because the service provider does not ask for feedback. The focus of this thesis is on passive feedback channels, such as bug reports and support portal, where the customer contacts and interacts with the service. This information is then use to improve the product, thus closing the feedback loop.

2.3 Service Design

2.3.1 Service Design Thinking

Service design concept was first introduced in Cologne in the early 1990s and its main goals are to make the service useful, usable and desirable from the client's point of view and also ensure that it is effective, efficient and distinctive from the supplier's point of view (Mager 2009, 32-34).

In service design the main focus is on the user experience instead of the technical aspects of the service: “The purpose of service design methodologies is to design according to the needs of customers or participants, so that the service is user-friendly, competitive and relevant to the customers (What is service design? n.d.).”

It is also important to think about the future of the service. According to Mager (2008), the service designer’s duty is to visualise, formulate and choreograph solutions to problems that may not exist yet – service designers must interpret and observe requirements and behavioural patterns and transform them into possible future services (p. 355). Common tools of service design include customer journey maps, storyboards, character cards and mock-ups of the product.

2.3.2 Designing for Services

Designing for services is closely related to service design. While service design concentrates on designing of the services itself, designing for services focuses on what design can do for services.

Manzini (2011) emphasises that designing for services does not mean designing services. The end result is not what is being designed, but an action platform – “a system that makes a multiplicity of interactions possible” (p. 3).

Kimbell (2011) sees designing for services as a different kind of approach to service design: it is an exploratory process that proposes and creates new kinds of value relations within a socio-material configuration.

Human-centred approach is common in designing for services - the main source of inspiration for redesigning or creating new services is to investigate and understand people’s experiences, interactions and practices (Meroni & Sangiorgi 2011, 203).

According to Meroni and Sangiorgi (2011, 204), there are four areas of application in human-centred approach of service design (see Figure 2):

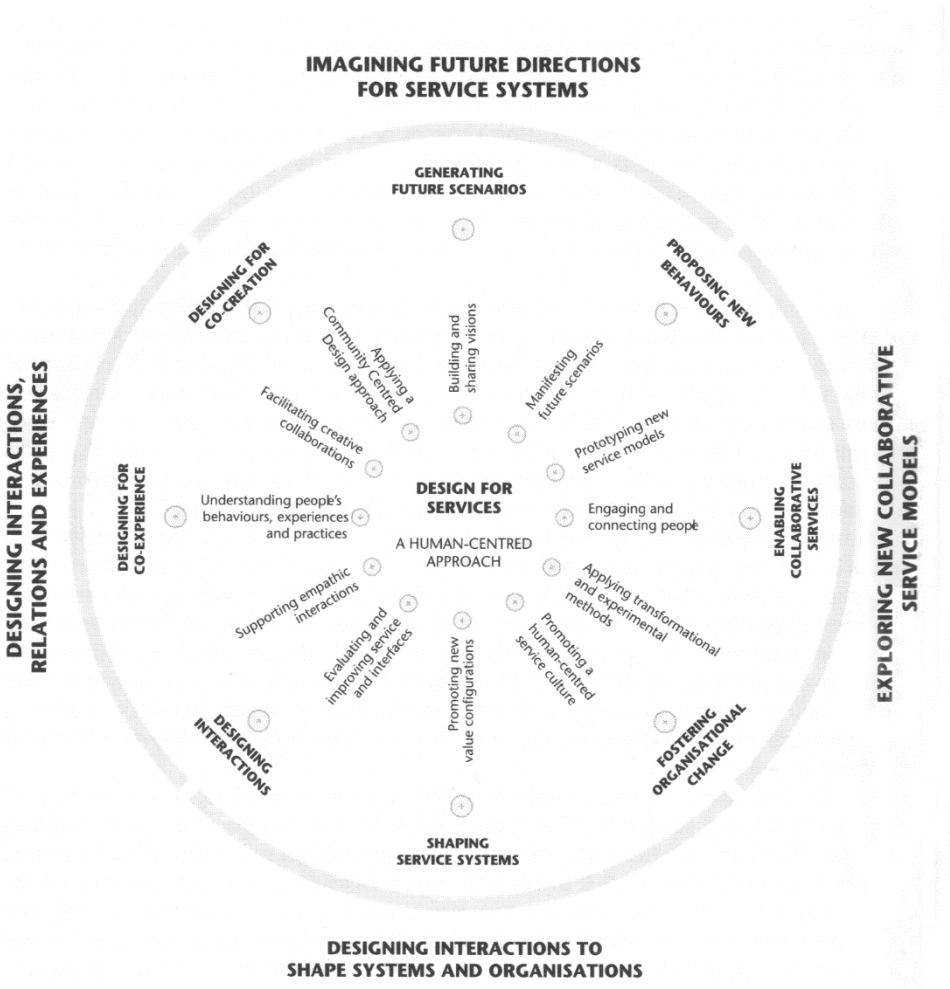


Figure 2. The design for services map (Meroni & Sangiorgi 2011, 204.)

Designing interactions, relations and experiences. Designers try to deeply understand people's behaviours, practices and experiences and then use this knowledge to design new or improved services and ways to interact with them. This can be done in co-operation with the users. Tools for this area include customer journey maps, emotional maps and user diaries. (Ibid. 206.)

Designing interactions to shape systems and organisations. This area redesigns the service interactions and interfaces to improve the service and to foster the organisational change and business development. The aim is to improve usability, explore new service ideas and to bring people's needs to the centre of service provision and development. (Ibid. 206-207.)

Exploring new collaborative service models. Designers develop and experiment new service ideas and study their feasibility on social, economic and technological levels. The focus is on social networks and collaborative solutions. Designers can use service prototypes to let people experiment with and co-create new service models and solutions. (Ibid. 207-208.)

Imagining future directions for service systems. Designers try to generate and share scenarios of the future service systems. Scenarios can be visualised and manifested through e.g. stories and service ideas. (Ibid. 208-209.)

2.4 Issue Management

2.4.1 Service recovery

In general, reliability is considered to be the most important feature when judging the quality of a service (Berry 1995, 79). A service failure has a negative effect on perceived service quality, but the effect can be reversed with service recovery. The main objectives of service recovery are to regain customer satisfaction, and to identify and correct weaknesses in service processes, thus preventing them from occurring in the future. (Kandampully & Khanh 2004, 392.)

2.4.2 Events, Incidents and Problems

ITIL defines an **event** as a significant change of state in an IT service (Steinberg 2011, 58). An event can be one of three types: informational, warning or exception. **Event management** handles events and if it detects that something is not functioning correctly, the event may be qualified as an **incident**. (Cartlidge et al. 2012, 41.)

An incident is defined as an unplanned interruption to a service or a reduction in the quality of it. Incidents are often detected by event management or the users of the service. (Ibid., 42.) **Incident management** then tries to restore the service to the users as soon as possible, often through workarounds or temporary fixes. Incident management does not try find a permanent solution to the underlying cause of incident **–** which is known as a **problem –** its aim is solely to restore the service. (ITIL – A guide to incident management, n.d.)

Problem management aims to find the root cause of incidents and to find a resolution to problems. It also tries to prevent problems and incident occurring. If incidents cannot be prevented, problem management tries to minimize their impact. (Steinberg 2011, 97.)

According to Yuson (n.d.), an incident can raise a problem, but it cannot become one. An incident is caused by a problem, however, a problem may exist even if there are no related incidents present.

To summarize events, incidents and problems: an event can be a part of the regular operation, such as someone logging into a service or it can be an exception that interrupts or reduces the quality of a service. In the latter case the event is an incident and it is caused by a problem. Incident management tries to restore the service, but it is not interested in the problem. Problem management aims to resolve the problem and tries to prevent new incidents from emerging. In IT business, both incident management and problem management can be seen as crucial parts of the service recovery process.

2.4.3 Bugs and Bug Tracking Systems

According to International Software Testing Qualifications Board, a **bug** is another term for a **defect** which is defined as “a flaw in a component or system that can cause the component or system to fail to perform its required function” (ISTQB:n testaussanasto 2007).

Bug tracking systems have been used in software projects since the 1970s to report and resolve bugs. A typical bug report is filed by a user and it contains a description of the defect, information about the user’s system, and the steps to reproduce it. In bug resolution process, the developers have to diagnose and confirm the defect before it can be resolved. In general, there are three possible resolutions:

- **fixed** – the bug is confirmed and corrected,
- **won’t fix** – the bug cannot be reproduced, it is not relevant or it is not a real failure,
- **duplicate** – the bug has already been reported.

Until the bug is resolved, the status of the bug is **open.** (Czarnecki, Lotufo & Passos 2012, 3.)

2.4.4 Issues and Issue Tracking Systems

An issue is a broader term related to bugs. Issues are used in software development to help organize project work, provide customer support, and to control the quality of the software. “An issue can capture and track work items, bugs, defects, tasks, feature requests, etc.” Similarly to bugs, issues are stored in an issue repository that can be used to track the progress of them. (Gu, Shu & Zhao 2011, 86.)

There are several **issue tracking systems** (ITS) available in the market, including Bugzilla, JIRA and Trac. Some issue trackers can also be used as bug trackers. Issues can be reported by the users, developers or testers of the product.

An issue can have several statuses during its lifespan. The statuses can usually be customised in an ITS to better meet the needs of the project. Figure 3 depicts the default statuses and workflow of JIRA issue tracker. When an issue is created it is in **open** state.

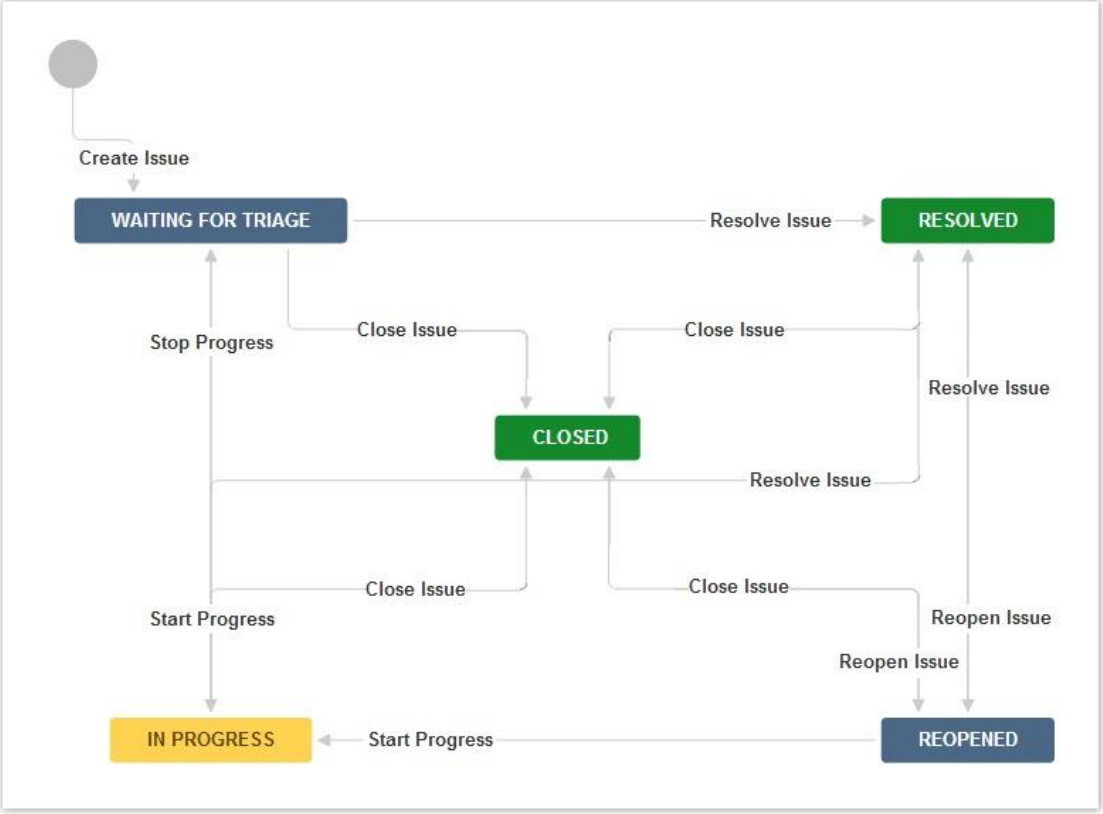


Figure 3. Default workflow of JIRA issue tracker (Configuring Workflow – JIRA 6.4 EAP n.d.)

Before developers start resolving issues, issues are usually **triaged**. In triaging process issues are evaluated – if possible, they are reproduced, the priority of the issue is determined, and lastly they are channelled to the correct development team. A triager can also check if the issue has already been reported, in which case it can be marked as a duplicate. Also, if the issue cannot be reproduced or its description is vague, the triager can ask for additional information from the reporter of the issue. Irrelevant or otherwise obsolete issues can be **closed** in triaging process. (Triaging Bugs 2014.)

When the resolution process of an issue is started, its status is set as **in progress** and after the issue has been resolved, the status will also be **resolved**. In an ITS, an issue has a property called **resolution**. When an issue has been resolved, it does not necessarily mean that the issue is fixed – resolution can be e.g. won't fix, incomplete, du-

plicate, cannot reproduce or won't do (What is an issue n.d.). The issue may be waiting for new information, it may not be fixable or the issue just is not relevant. Likewise statuses, resolutions can be tailored to suit the project.

Once the issue is resolved it can be closed. Often the resolution is reviewed before the closure. Closed or resolved issues may be **reopened**, e.g. if the issue is still present after the resolution or there is more information available concerning the issue. Reopened issues can be set as in progress, resolved or closed.

2.4.5 Problems with Bug and Issue Management

There are some widely recognised problems with the bug or issue reports. In their study, Bettenburg, Just, Premraj, Schröter, Weiss and Zimmermann (2008, 10) found out that most severe problems in reports are "errors in steps to reproduce, incomplete information, and wrong observed behaviour". Low quality and incompleteness of the reports can mislead the developers thus wasting valuable development time (Czarnecki et al. 2012, 2; Hu, Jiang, Luo, Ren, Wu, Xuan & Zou 2015, 264). Poorly written reports are also less likely to get the attention of developers or to be fixed at all (Bettenburg et al. 2008, 10; Petersen 2013).

Duplicate entries of a bug or an issue can provide important additional information for the developers (Bettenburg et al. 2008, 5); however, the information would be more useful if it was posted as an additional comment to an already existing issue rather than filing an independent new one (Gu et al. 2011, 86).

Another time consuming phase in bug or issue management is issue assignment. In triaging process issues are assigned to a developer primarily manually and often they are assigned to a wrong person. Sometimes an issue can be reassigned several times before it is processed. (Jeong, Kim & Zimmermann 2009, 9.)

In 2013, the bug reports of Unity 3D–game engine were studied and out of 505 reports 30 (5.94%) could be reproduced and confirmed as bugs right away. 37 reports (7.33%) were duplicates, 72 reports (14.3%) were too incomplete and were not reproduced, and 181 reports (35.8%) were not qualified as bugs. 185 reports (36.6%) needed more information and it was requested from the reporters. 57 reporters

posted additional information concerning the bugs and the bug count was increased by two, finally totalling to 32 (6.34%). (Peterson 2013.)

3 Analysing Current Situation

3.1 The Qt-ecosystem

The Qt is a platform that is developed by the Qt Project – an open source development community consisting of individuals and companies – including The Qt Company and KDAB – a Swedish consulting company. Although there are free versions available, The Qt Company sells commercial licences for the software. At the moment, the available licences are Community, Indie Mobile, Professional and Enterprise (Download Qt 2015).

Although most of the development work of the Qt is done by the employees of The Qt Company, the community members actively contribute to the project. Roughly 25% of the code is submitted by the community. Also, the majority of the development costs (85%) are covered by The Qt Company. (Knoll 2014; Myllymäki 2014.)

At the time this paper is being written, the content of the community portal – qt-project.org – is being merged with the new qt.io –site that is maintained by The Qt Company. The reason of this fusion is to unify the Qt ecosystem, therefore making it stronger against the competition. The new website is also meant to give a broad overview of both commercial and open source sides of the Qt. Also, the Qt packages are meant to be unified, so that the licence migration and the release of new versions of the software would become easier. (Knoll 2014.)

3.2 Comparison with Competitors

First, it was decided to compare the feedback channels and practices of The Qt Company and four other similar companies. The aim of the comparison was to see, what kind of feedback collecting methods are common today and how the practices of The Qt Company compare to them. Mika Myllymäki, former Director of Ventures of Digia, suggested the following companies to be included in the comparison because they compete with Qt on certain markets:

- Telerik – a Bulgarian firm providing cross-platform development tools and services.

- Unity Technologies – a company developing Unity3D cross-platform game engine.
- Xamarin – likewise Telerik, a cross-platform coding tool developer based in San Francisco.
- Vaadin Ltd. – a Finnish developer of a web application framework.

All of the gathered information is based on the free versions of the software. Purchase of licensed version of the software or usage of paid support was not considered to be feasible in this study.

3.2.1 Issue Trackers

The bug or issue tracking systems and practices of the companies have been collected in Table 1.

Table 1. Comparison of bug and issue trackers

	The Qt Company	**Telerik**	**Unity**	**Vaadin**	**Xamarin**
Public Tracker	JIRA	Bug forum	Bug portal	Trac	Bugzilla
Internal Tracker	JIRA	N/A	FogBugz	N/A	N/A
Way of Submission	Tracker	Bug forum, ticket	Reporting application in editor	Tracker	Tracker
Issue Types	Bug, Epic, Research, Suggestion, Task, User Story	N/A	Problem w/ editor, Problem w/ player, Feature Request, Documentation, Crash Bug	Defect, Issue, Enhancement, Task,	Bug, Enhancement
Triaging	Registered users	N/A	Priority voting for registered users (10 votes / user). All public bugs have been confirmed internally.	Ticketmaster	Bug verification outsourced to 360Logica
Notes	Higher priority included in support packages	Known issues can be found in docs. Dedicated help available in bug forums.	Some bugs have not been made public because of Unity3D's privacy policy.	Priority bug fixes included in support packages (Gold, Platinum)	

Incidentally, all of the companies use different bug trackers. All of the companies let the customers to access the public bug tracker directly, except for Unity and Telerik. Unity has built a separate web-based front-end for bug submission, tracking and priority voting. Also, the Unity3D has a built-in bug reporting application to aid the users. The staff of Unity verifies and moderates the bug reports before they are made public. Telerik's only public way to submit and search bugs are the forums.

By purchasing better support packages or licences, the users of Qt or Vaadin also get higher visibility and priority to their bugs. High priority bugs have a better chance to be fixed in the next release. However, the final priority level is always decided in the triaging process.

The triaging process is usually carried out by the staff of the company, but The Qt Company uses crowdsourcing to aid the in-house triagers. Practically anyone can sign in to the bug tracker and start triaging bugs. Interestingly, Xamarin has outsourced the bug verification to another company.

3.2.2 Feature Requests

Some of the companies handle feature requests as issues alongside with bugs, while other companies have separate tracking systems for them. The different feature request practices have been collected in Table 2.

Table 2. Comparison of feature requests

	The Qt Company	**Telerik**	**Unity**	**Vaadin**	**Xamarin**
Tracking	JIRA	Ideas & Feedback Portal (Telerik TeamPulse)	Feedback Portal	Trac	UserVoice / Bugzilla
Way of Submission	Issue tracker (as a suggestion)	Ideas & Feedback Portal post	Feedback Portal post	Issue Tracker (as a feature request)	UserVoice post
Notes	Higher priority included in support packages	Priority voting for registered users	Priority voting for registered users	Feature votes included in Gold and Platinum support packages	Priority voting for registered users

Feature requests are new ideas and suggestions how the product should be developed further. The Qt Company and Vaadin handle the feature requests as issues and the customers can report them directly to the issue tracker. Telerik, Unity and Xamarin, on the other hand, have web portals for feature requests (see Figure 4). Similarly

to bugs, the priority of a feature requests can be affected by voting. However, the significance or amount of votes may depend on the purchased licence or support package.

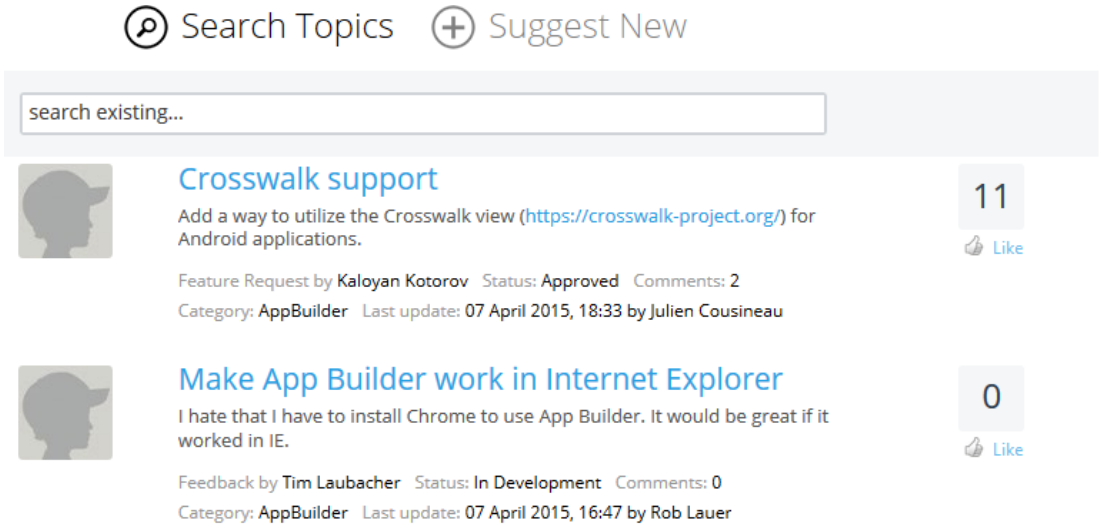


Figure 4. Screen capture of the Telerik Ideas & Feedback Portal

3.2.3 Support Forums

Support forums are popular places to talk about features and problems of the product. The forums are usually open for everyone and they have a wide range of topics from detailed technical help to talk of the town. However, some of the topics may require a licence to be accessible. In Table 3 are sub-forums or topics from each of the companies.

Table 3. Comparison of support forums

	The Qt Company	Telerik	Unity	Vaadin	Xamarin
Topics / Sub-forums	Development, Learning, Talk, Group, International, Behind the Scenes	End-to-End, Mobile, HTML5/JS, Web, Desktop, Productivity & Quality, Debugging, Reporting & Data Access, DevCloud, ALM & Testing, General discussion	General, Commercial and Collaborative Work, Community Support, Services, Community Platform Support	News & Announcements, Using the Forums, Miscellaneous Discussion, General Help, UI Components, Data Bindings, Themes, Vaadin Testbench, Add-ons, Vaadin 7, Events, Forums in multiple languages	Job Listings, Xamarin Test Cloud, Xamarin Insights, Xamarin.Forms, General, Cross Platform, Visual Studio, Xamarin Studio, Component Store, Android, iOS, Mac, Community, Prerelease, CocosSharp
Notes		Includes bug reporting and suggestions for AppBuilder			

The Qt forum featured in the table is actually hosted by the Qt Project. If an actual bug is found in a post, it is not automatically reported to the bug tracer by the staff, but the poster is prompted to do so. Unlike the Qt forum, Telerik's forum may be used as a medium to report bugs.

3.2.4 Paid Support Services

Support services can be seen as additional ways to get help and give feedback. These packages may include help desk and consultancy, ad hoc or even on-site services. Also the response times of the help desk or the amount of feature votes may depend on the package. The support services may be tied to the level of licence or they can be purchased separately. There is a summary of the paid support services in Table 4.

Table 4. Comparison of paid support services

	The Qt Company	Telerik	Unity	Vaadin	Xamarin
Packages	**Bronze** (incl. in Professional) Support Portal, high priority bug reports	**Most licenses** (e.g. DevCraft $1 299 / dev / yr.) incl. forums, feedback portals, unlimited ticket support, 72 hr. response time	**Indie** 4 hr. / mo dedicated support, 1 ticket, 24 hr. response time, 10% consultancy discount	**Silver** €4 000 / yr. incl. 2 day response time, email & web channel support, 15 hr. dedicated support	**Business** €83 / mo. incl. email support
	Silver (incl. in Enterprise) Support Portal, high priority bug reports, consultancy services	**Priority support** (e.g. DevCraft $1 499 / dev / yr.) 24 hr. response time	**Small Team** 15 hr. / mo. dedicated support, 2 tickets, 12 hr. response time, 15% consultancy discount	**Gold** €6 000 / yr. incl. 1 day response time, email & web channel support, 25 hr. dedicated support, bug fixes & feature votes	**Enterprise** €158 / mo. incl. email support, Technical Account Manager, 1 day response time
	Gold (Tailored support) Support Portal, high priority bug reports, consultancy services, virtual remote on-site support, dedicated support persons, on-site support	**Ultimate support** (e.g. DevCraft $1 999 / dev / yr.) 24 hr. response time, phone assistance, remote web assistance, issue escalation w/ 16 hr. response, ticket prescreening w/ 4 hr. response	**Pro** 40 hr. / mo. dedicated support, 3 tickets, 8 hr. response time, 20% consultancy discount	**Platinum** €12 000 / yr. incl. 1 hr. response time, email & web channel support, phone support, 50 hr. dedicated support, bug fixes & feature votes	

Table continued on next page.

Table 4 continued.

	The Qt Company	Telerik	Unity	Vaadin	Xamarin
Packages			**Enterprise** unlimited dedicated support, 5 tickets, 2 hr. response time, 25% consultancy discount		
			Educational 10 hr. / mo dedicated support, 2 tickets, 24 hr. response time, 10% consultancy discount		
Other			**Consultancy Services** $350 / hr., $2 800 / day	**Express Support** €850 / 4 hr. dedicated support, 1 day response time	
Notes		Fair Usage Policy applies to all support services			

All of the companies offer several service packages: some of them are bundled with the purchased licence, while the others can be acquired separately. There are also some on-demand services available with hourly or daily rates. Unsurprisingly, the price of a support package is related to the level of the support.

The Qt Company and Xamarin offers two basic levels of support that are included in licences. If those packages are not satisfactory, The Qt Company also provides a custom package, to which the customer can select the required services, including dedicated support persons and on-site support. Telerik's more expensive support plans promise quicker responses from the support personnel, new ways to contact support

and faster handling of support requests. Telerik does not limit the amount of support tickets, as long as the usage is moderate.

Unlike Telerik, Unity has heavily limited the number of support tickets, but on the other hand, a certain amount of dedicated support is available on all licence levels. Unity also offers additional consultancy services with daily or hourly rates. The discount level of these services depends on the purchased licence.

Similarly to Telerik and Unity, Vaadin also has shorter promised response times and more dedicated support on better packages. Feature votes and high priority bug fixes are, however, only available in gold and platinum packages. In case that all of the support time has been exhausted, additional dedicated support is available at hourly rates.

3.2.5 Other Feedback Channels

Table 5 lists other miscellaneous feedback channels. Although some of these channels are out of the scope of this thesis, they were decided to be included in the comparison, so that the current feedback collecting practices of the companies could be studied.

Table 5. Other channels

	The Qt Company	Telerik	Unity	Vaadin	Xamarin
	Contact form, email feedback, IRC, electronic mailing list, surveys, sales, gatherings, blog comments, business analytics, Usabilla (site feedback), social media, StackOverFlow	Blog comments, gatherings, site feedback (contact form), phone, email, social media	Contact form, Unity Answers (AnswerHub), blog comments, gatherings, phone, mail, social media, StackOverFlow	Contact form, blog comments, gatherings, wiki comments, Zopim-chat (consulting services), phone, email, social media, StackOverFlow	Contact form, email, phone, social media

Electronic mailing lists are a way to send emails to multiple recipients. Subscribers of the list get all of the messages sent by the other subscribers. According to Myllymäki (2014), electronic mailing lists are quite a popular way of communicating amongst the Qt community members and it is used to help the development of the Qt.

Usabilla is a tool that was briefly used to collect and analyse user feedback concerning the new Qt.io-website when the site was launched. With Usabilla users could easily select a single element on the webpage and tell their feelings about it.

The Qt Company also monitors the social media, but it does not automatically collect feedback from those channels. The company's policy is to intervene and correct only when someone posts incorrect or otherwise harmful information about the company. Positive posts may, however, be highlighted and reposted on the company's social media sites. (Myllymäki 2014.)

There are various ways to contact the companies for potential customers and other people interested in their products. On their websites, all of the companies offer contact form as the primary way of contact. Telephone numbers and email addresses

are also available for general inquiries on all of the sites except for The Qt Company, which only has contact information for research and development and local sales offices. The Qt Company also has an email address for generic feedback and Vaadin has implemented a chat application in their consultancy service section. In case none of the team is available to answer, the customer will be contacted by the next business day at the latest.

Stack Overflow is a popular question-and-answer –type website, in which people can ask programming related questions. Of the companies compared, The Qt Company, Unity and Vaadin promote Stack Overflow as a way of getting support, although the site is not affiliated with those companies. According to Myllymäki (2014), Qt developers often solve Qt related problems on that site.

3.3 Visualising Feedback Channels

3.3.1 Data Flow Diagram

All possible feedback channels have not been included in this thesis, because it would not have been reasonable considering the scope of a typical bachelor's thesis nor the time allocated for this work. The feedback channels that were chosen, are unsolicited channels that Qt-users, i.e. developers and business owners, can use to have an influence on the development of the Qt-platform. Basically the feedback either is or it can be converted to bug reports and feature requests.

These channels – and also some channels, such as surveys and sales analytics, that were out of the scope of this thesis – were depicted in a data flow diagram (DFD, see Figure 5). This DFD is illustrative and does not strictly conform the DFD-standard. There is also a standardised DFD available in Appendix 1.

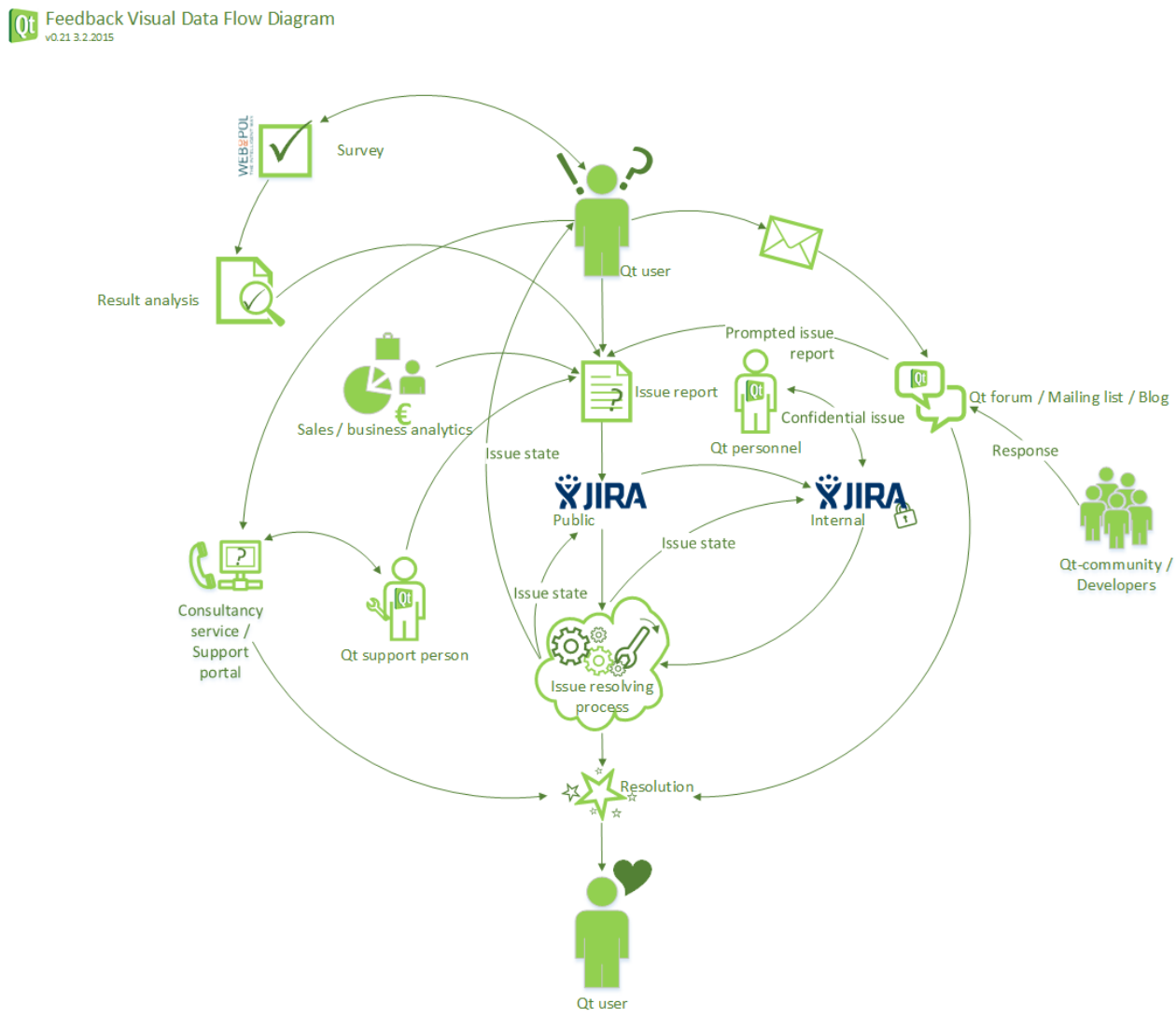


Figure 5. Visualised data flow diagram of the chosen feedback channels

As the diagram depicts, issue resolving process is in the centre of the feedback flow. Issues can be reported by the users or the personnel of The Qt Company. The diagram also shows two separate issue tracking systems (ITS): one that is public and accessible for everyone, and the other that is accessible only for the employees of The Qt Company. The internal ITS contains confidential issues – issues containing information that may harm The Qt Company or the reporter of the issue i.e. the cus-

tomer, if they were leaked to the competitors. Confidential issues may initially be reported to the public tracker, but after evaluation they can be moved to the internal one only by employees of The Qt Company. (Myllymäki 2014.)

The support portal is the preferred way to seek help or report issues, if the customer has purchased a commercial licence. The available support methods depend on the purchased licence or support package (see Table 4). The portal conforms ITIL's incident and problem management guidelines (see Cartlidge et al. 2012, 42-44). After a support request, the reporter is contacted within a reasonable amount of time. Depending on the issue, the support person may resolve the issue, or in case of bug or feature request, the resolution may be available in the next release of the Qt software at the earliest. (Qt Legal – Terms & Conditions 2015.)

The support forum is a popular place where community members or open source version users can chat and solve problems. General help and workarounds are not provided only by the community members, but by Qt-developers as well. Problems can be just about anything Qt related – from programming tips to faults in the product. However, if a problem that can be classified as a bug is posted to the forum, it is not automatically reported to the ITS, but the poster is prompted to file a bug report.

Some issues may also arise from the results of surveys, business analytics etc., but those issues may rather apply to the corporate level and they do not necessarily concern the product itself. Therefore those channels were left out of this thesis, although they are present in the DFD.

3.3.2 BPMN-diagram of the Issue Resolution Process

Business Process Model and Notation (BPMN) is a graphical notation standard developed by the Object Management Group that can be used to depict different business processes. The BPMN aims to be the definitive business process modelling notation that can be easily understood by all business users. (Business Process Model and Notation (BPMN) Version 2.0 2011, 1.)

Because the issue resolving process is in the centre of the feedback processing mechanism, the process was decided to be charted and analysed (see Appendices 2 and 3). The diagram depicts how the process advances in the public tracker. There are

three participants in the diagram and since they may not be part of the same organisation, each one has their own pool:

A reporter is anyone who has found an error in the product or has a feature request in mind. Reporter can be a Qt-user, either a paying customer or a free version user, an employee of The Qt Company, a potential user or a product tester.

A triager is responsible of the reproduction and verification of the issue. The triager also determines the severity of the issue and chooses the correct assignee to the issue. The triager can be an employee of The Qt Company or anyone who has registered to the public issue tracker.

An assignee resolves the issues. An assignee can be an in-house developer working for the Qt Company or someone contributing to the open source Qt project. Even the reporter can resolve issues. If someone wants to be a code contributor, he or she must be an active member of the community and has to have a good reputation.

The workflow basically follows the JIRA default workflow (cf. Figure 3), however, there are some custom statuses and resolutions.

In The Qt Company's internal tracker, the process is similar, but because the issues are confidential, the reporters, triagers and assignees are employees of the firm. Issues can also be moved from the public tracker to the internal one, if they are considered to contain classified information – such as code or other information that could harm the customer or The Qt Company, if they were available for the competitors.

3.3.3 Customer Journey Map

Customer journey map is a tool to visualise a customer's interaction with a service step by step. The map is always depicted in the customer's point of view and it spans through the whole service interaction. Although there are several templates available (see Figure 6, full image available in Appendix 4), a customer journey map can be formed quite freely depending on the service at hand. However, there are some elements that are commonly included in the map.

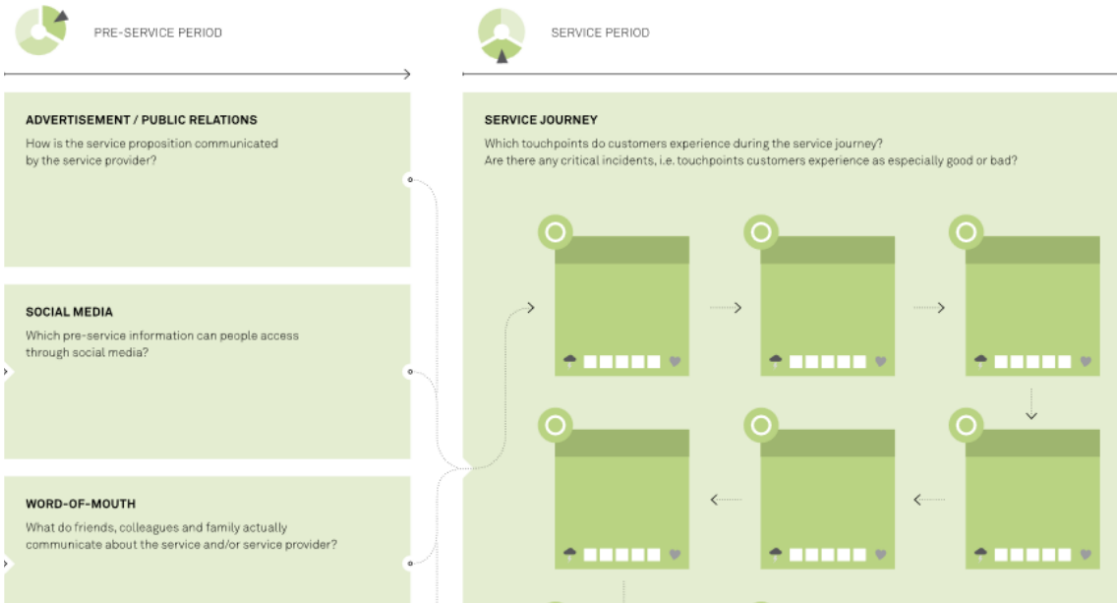


Figure 6. Detail of a customer journey canvas (Schneider & Stickdorn n.d., modified.)

A **customer journey** is formed of **service moments**. These moments contain both customer's choices and service provider's actions. The customer journey may vary from customer to customer, because they can behave differently. Each service moment is experienced and perceived through a number of **service touch points**. Touch points can be divided into channels, objects, processes and people. Channels are environments or media where the service production is experienced. Objects are tangible items that the customers or the personnel use, need or get during the service. Processes are practices that affect how the service is experienced. A process can be e.g. the way how the user interface of a program functions. And lastly, people consist of both customers and contact persons. (Koivisto 2009, 143-147.)

A customer journey map depicting touch points of The Qt Company's feedback process in the customer's point of view is available in Appendix 5. There are three feedback channels in the map: issue tracker, support portal and Qt-forums. The support portal consists of several support solutions, but in this map it is considered as a help desk-type solution. There are also six phases in the service experience that may or may not be present in the customer journey:

- Common tasks are mundane things that are still required. In these cases, a user has to log in to use a service.
- Research is a phase where the user spontaneously searches possibly existing resolution to the issue.
- Feedback submission.
- Information exchange. In this phase the user and service provider interact with each other.
- Resolution. The issue is resolved in this phase.
- Evaluation of resolution. The issue has been resolved and the user checks the results.

The journey begins when the customer chooses the feedback channel. All of the channels require logging in to the medium, if the user wants to post feedback. In the forums and ITS, it is also possible to browse and search the existing topics or issues without a login.

Although the research phase is optional, it would be beneficial for the support system, if the user would do a search for already existing similar issues, before reporting a new one. In forums, a common policy with duplicate threads is to either close the thread or merge it with the already existing one. Duplicate reports in issue tracking systems are not considered to be a big problem, but they would be more helpful, if they were posted as a comment to an existing report. In any case, an unnecessary report will increase the work load of the support personnel.

Information exchange takes place in all of the channels. Forum posts, issue reports or service requests are responded in reasonable time. Responses may contain resolutions, request for additional information or workarounds to resume the service while the issue is resolved. In issue trackers, the status of the issue is automatically reported to the reporter or anyone, who is watching the issue. The community may not be able to help if the issue is a bug or feature request, in which case the user has to file an issue report.

In resolution phase, if the support has not already found a resolution, the only options for the customer is to either wait for a resolution or use a possibly existing

workaround. Bugs fixes and new features are naturally available in the future releases of the software.

In the final phase the user checks if the resolution actually resolves the issue. If the resolution is correct, the feedback process ends and the customer is satisfied. However, if the issue is still present, the user may want to try another support channel and the resolution process starts over again. Also, if the user thinks the issue is not relevant anymore, the journey ends.

The forums are mostly aimed for the community members and free version users and it may not lead to a resolution at all - especially if the feedback concerns bugs or feature requests. The issue tracker, in the other hand, is the correct medium for that kind of feedback, but the process is somewhat complicated. The support portal is a collection of different means to communicate with the Qt-developers (see Table 4) and it is the most efficient way to give feedback. Also, the feature requests and bug reports get better visibility and higher priority when reported through the portal. However, most of those features are available only for the paying customers.

4 Conclusions and Propositions for New Practices

4.1 Automated Usage Reporting

The possibility of automated usage reporting as a way of collecting feedback was discussed. In automated usage reporting, the integrated development environment (IDE) reports automatically, which parts of the software the users are using most and which parts are becoming obsolete. The collected data can then be used to determine how the product should be developed further. This method does not require any actions from the user.

Digia used automated usage reporting earlier, but they gave up the system because the users were concerned about their privacy. At the moment The Qt Company is monitoring the usage of the Qt documentation, to see which parts of the product are most popular. (Myllymäki & Perälä 2015.) While this paper is written, Vaadin is implementing automated usage reporting to their product. Their system is developed by Codetrails, a German company focused on intelligent software engineering tools and data analysis. Vaadin is concerned about the privacy of their data, because by default, the Codetrails' system stores and analyses the usage data in their servers. Therefore Codetrails is developing a custom system which only collects the usage data, but the data is then stored in Vaadin's database. (Perälä 2014.)

Automated usage reporting is a way to collect indirect feedback from the users. There probably is not a complete third party solution available – e.g. Codetrails' system is only available for Eclipse IDE – but since there has been a usage reporting tool for Qt before, the possibility of implementation of the tool should be considered. Some users probably do not want to be monitored, and therefore, there should be an option to opt out of the service. Also, the privacy concerns of the users should be considered, when planning out the privacy policy of the service.

4.2 Crowdsourcing Incentives

The possibilities of crowdsourcing and incentives was one of the topics suggested by the assigner of this thesis. Howe (2006) defines crowdsourcing as "the act of taking a

job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call". Hossain (2012) summarises two kinds of motivation to participate in a crowdsourcing project:

- Intrinsic motivation is driven by the task. This is dominant motivational factor in OSS.
- Extrinsic motivation is driven by external incentives such as status, reputation and fun. (Pp. 502-503.)

Although there are commercial licences available, Qt is an open source software (OSS) project – a form of crowdsourcing. The community participates in the development of the product e.g. by contributing code, triaging bugs and editing documents. The Qt Company has already implemented a non-financial reward system: for their troubles, the community members get mainly points, badges and reputation (Kojo 2015).

The Qt Company has planned their incentive program well: points and reputation act as extrinsic incentives – and there are even some elements of gamification, because amassed points can be converted into status badges. Some people probably do not need any incentives – the challenges of software development are the main motivational factor for them. One possibility to improve the system is to add financial rewards; however, a study shows that monetary incentives may even have a negative impact on the motivation of the participants of an OSS project (Alexy & Leitner 2010, 25). Nevertheless, they are viable in some cases. They can be an effective incentive on mundane or non-interesting tasks (Alexy & Leitner 2010, 25; Hossain 2012, 505). Also, infrequent rewards, such as prizes for specific tasks or events are plausible. (Alexy & Leitner 2010, 25.)

4.3 User Experience Enhancements

4.3.1 Bug Reporting

There are some unnecessary steps and annoyances in the bug reporting process. Of the companies that were compared in chapter 3.2, Unity has designed this process exceptionally well and The Qt Company could take a cue from them.

In the menu of Qt IDE is an option to report a bug, but selecting it will only open the web browser and navigate to the projects JIRA-site (see Appendix 6). After that the user has to log in to the JIRA, and if the user is accessing the tracker for the first time, also sign in. The user may already have a Qt-account credentials, which are used to access e.g. support portal and licence information, but these identifications are not usable in the bug tracker. For example, Unity has one account policy, where the same user credentials can be used in all services.

Unity 3D has also integrated a bug reporting tool in their IDE (see Figure 7), so there is no need to navigate to the tracker. The tool has also options to post a feature requests or to make the issue confidential, in which case it will only be accessible by the employees of Unity. If such a tool were to be included in Qt, there could also be a feature to pinpoint problems with the IDE itself – similarly to Usabilla: a tool that can be used to select a single element on a website, e.g. a button, and then give feedback concerning that one element. Also, it would be useful, if the user could include screenshots or video in the report to clarify the problem.

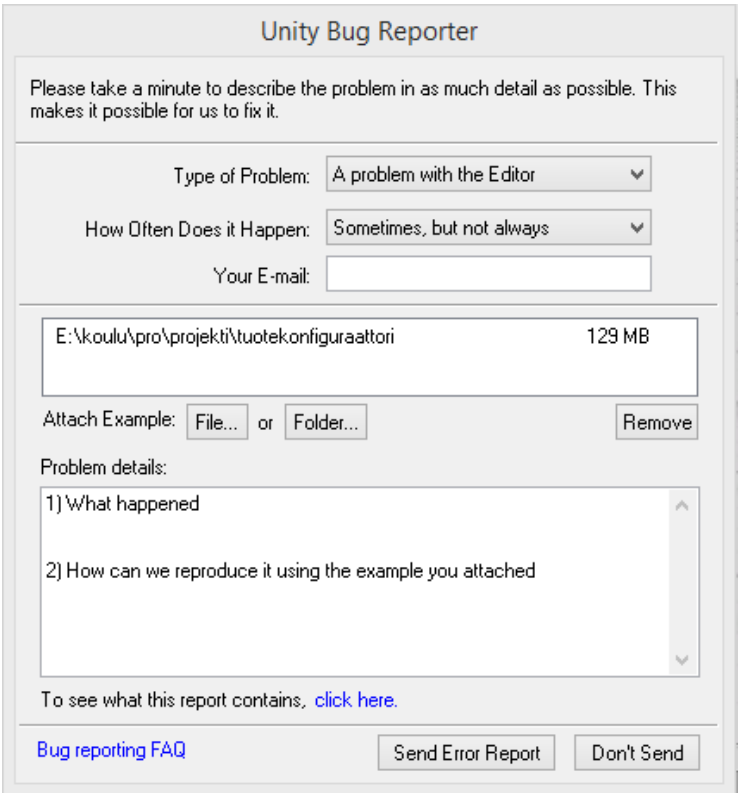


Figure 7. Screen capture of the Unity Bug Reporter

At the moment the user is asked to do a search for already existing issues in the Qt issue tracker. There are plugins available for JIRA that automatically search for similar issues, while the user writes the summary. These plugins would eliminate one touch point from the customer journey (see Appendix 5) and also reduce the number of duplicate issues.

4.3.2 Feature Requests

Feature requests are a great way to find out, what the customers want from the product and in which direction it should be developed in the future. At the moment, the reported feature requests are located in the issue tracker amongst all the bugs and other issues. Although there are filters that can be used to list and search the suggestions (see Figure 8), the presentation could be better. Unity, Telerik and Xamarin have separated feature requests from other issues and have implemented web-

based dedicated portals for them (see Figures 4 and 9), where users can browse, comment and vote the issues easily. Both Telerik and Xamarin use a third party solution, UserVoice, as their ideas platform.

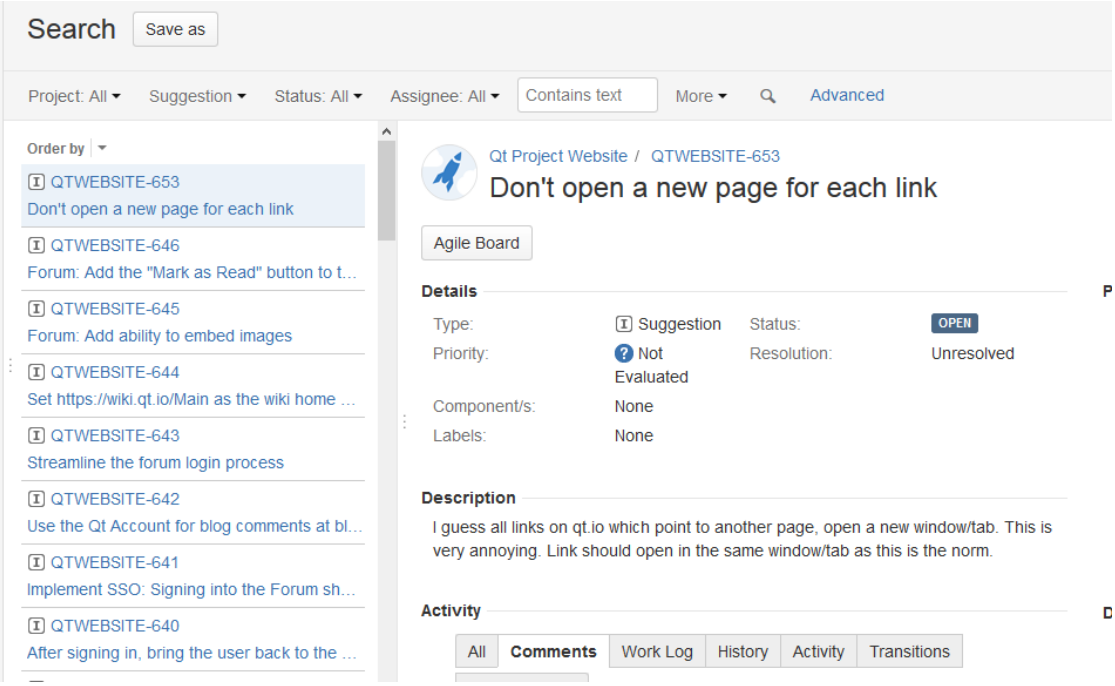


Figure 8. Feature Requests in the Qt Issue Tracker

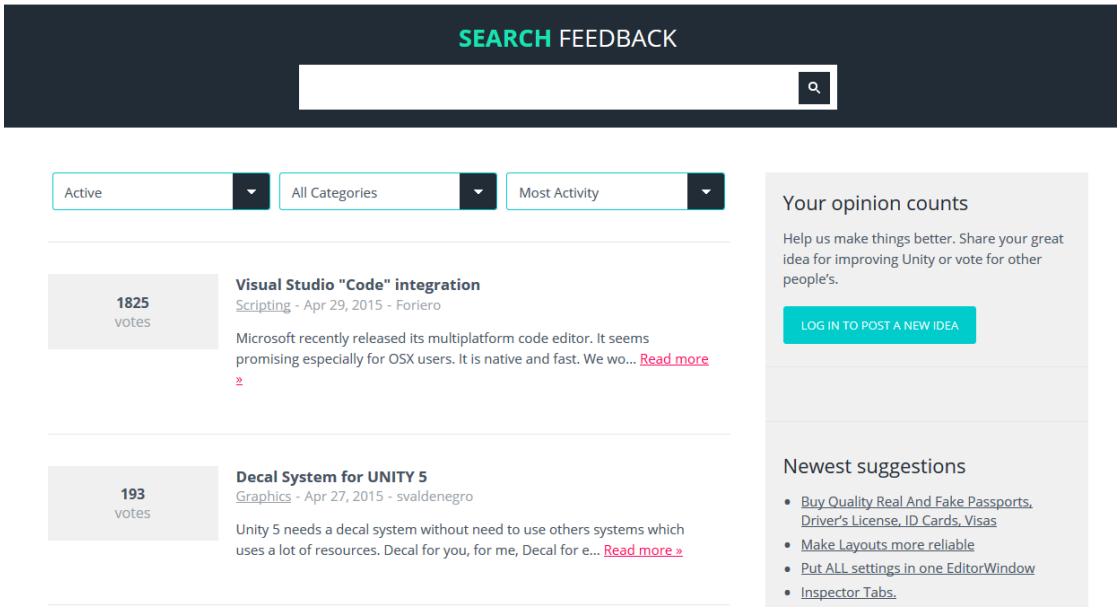


Figure 9. Unity feedback portal.

Similar solution could be beneficial for The Qt Company. Although the main functions would be the same as in JIRA, such a portal would be more than a mere front-end for the tracker. The ideas of other users would be presented attractively and it would be easier for users to contribute by commenting and voting existing issues. Also, the ideas of others could inspire and encourage the users to come up with their own suggestions. If a feature is decided to be included in Qt, an estimation of release date and version could be listed in the portal.

4.3.3 Miscellaneous

There is a plain email-link for general feedback concerning the website on the Qt main page. The link is probably aimed for visitors and potential customers, because the existing Qt users would most likely use the issue tracker as their feedback channel. Email is the most cost-efficient way to collect feedback, however, since the two Qt-sites have merged – now including e.g. the forums and documentation, the content and traffic on the site has increased. Therefore the site could utilise a more efficient solution for collecting feedback.

For example UserVoice is a popular tool for gathering users' suggestions. UserVoice widget can be integrated in the site (see Figure 10) and collected feedback can be categorised as a bug, suggestion etc., and then channelled straight into JIRA as issues. Of course, if no frequent feedback is expected, the viability of paid third-party solution is questionable.

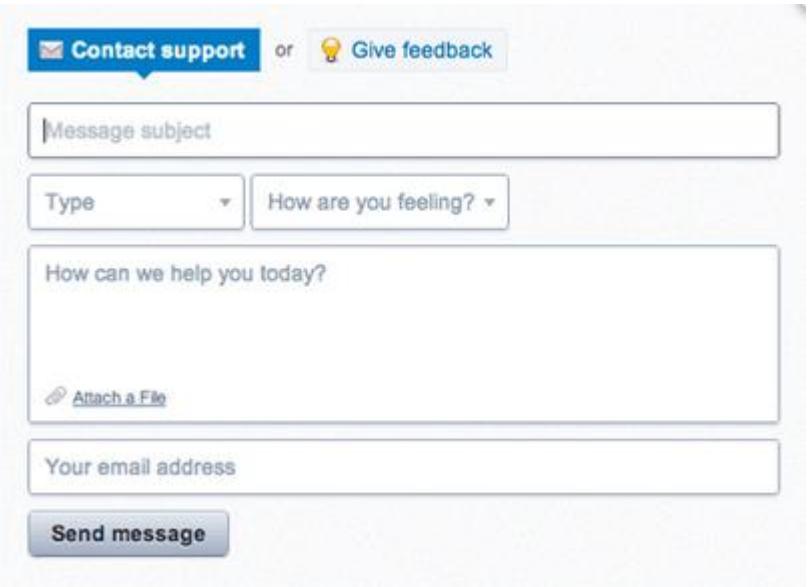


Figure 10. Screen capture of UserVoice feedback widget

5 Discussion

In general, The Qt Company has built their feedback system well – which was expected from a company of such calibre. Therefore, no ground-breaking new discoveries can be expected from this paper. One of the goals of this thesis was to visualise the feedback channels of the Qt ecosystem. The included channels had to be limited to those that can be used give feedback concerning the Qt platform – usually in the form of a bug report or feature request. Although the illustrations may not be quite complete and include all the channels or steps, they show the complexity of the system. These diagrams and maps were then studied to see, if there were room for improvement.

Comparison with competitors was another useful way to see, how feedback systems could be planned out. The comparison showed, that other companies have implemented pretty similar channels with some differences – mainly in the contents of support packages.

Another goal of the thesis was to propose new feedback collecting practices based on the visualisation, comparison and literature review. Especially the comparison proved to be a very useful source of new ideas. All of the propositions were made from the customer's point-of-view, because one of the goals of this work was find ways to encourage the users to report bugs and come up with new ideas. The technology to implement most of these suggestions exists, however, this paper does not discuss the financial viabilities of the propositions.

To improve the user experience of feedback process, a bug reporting tool was suggested. The tool would make the reporting easier and also remove some unnecessary steps from the process. Another proposition was to separate the feature requests from the bugs. Some of the companies in the comparison – and also some other software development companies not featured in this thesis – have implemented a separate feedback portal for the ideas of the customers. Engaging the customers in the product improvement is a modern trend that is used e.g. in the Lean startup methodology.

Automated usage reporting was one of the topics discussed in this thesis. Valuable information about this kind indirect feedback collecting was provided by Vaadin – a company included in the comparison. Privacy concerns – either in customer level or corporate level – are important issues to consider when implementing this kind of reporting technology. This paper does not discuss the viability of the automated usage reporting, however, there are room for further studies in the future.

Open source software project, such as Qt, is also a crowdsourcing project. A literature review showed that The Qt Company had executed the incentive system well. The possibilities of monetary incentives were discussed, but there is currently no need to change the system.

This thesis has been a mix and match of different studies, visualisation techniques and feedback channels. The assignment of the thesis was vague and there were several ways to approach the problem. The focus of work was lost several times during the process but eventually a kind of consistency was reached. The propositions may not be applicable as such, but hopefully, they will provide a basis for a feasibility study.

References

Alexy, O. & Leitner, M. 2010. A Fistful of Dollars: Financial Rewards, Payment Norms, and Motivation Crowding in Open Source Software Development. Accessed on 11 May 2015. Retrieved from http://emma.polimi.it/emma/events/dimeconference/attachments/oliver%20alexy.pdf.

Bettenburg, N., Just, S., Premraj, R., Schröter, A., Weiss, C. & Zimmermann, T. 2008. What Makes a Good Bug Report? Accessed on 15 March 2015. Retrieved from http://thomas-zimmermann.com/publications/files/bettenburg-fse-2008.pdf.

Berry, L. 1995. On Great Service: A Framework for Action. USA: The Free Press.

Berry, L., Parasuraman, A. & Zeithaml, V. 1985. Problems and Strategies in Services Marketing. In Journal of Marketing 49, 33-46. Accessed on 27 February 2015. Retrieved from http://areas.kenan-flagler.unc.edu/Marketing/FacultyStaff/zeithaml/Selected%20Publications/Problems%20and%20Strategies%20in%20Services%20Marketing.pdf.

Blank, S. 2013. Why the Lean Start-Up Changes Everything. In Harvard Business Review, May 2013, 63-72. Accessed on 30 March 2015. Retrieved from http://www.jamk.fi/fi/Palvelut/kirjasto/Etusivu/, Nelli Portal, Business Source Elite (EBSCO).

Business Process Model and Notation (BPMN) Version 2.0. 2011. Accessed on 28 March 2015. Retrieved from http://www.omg.org/spec/BPMN/2.0/PDF.

Carson, D. & Rushton, A. 1989. The Marketing of Services: Managing the Intangibles. In European Journal of Marketing 23, 8, 23-44. Accessed on 28 February 2015. Retrieved from http://xa.yimg.com/kq/groups/28671191/2014129071/name/ServMkt_Intngbles.pdf .

Configuring Workflow – JIRA 6.4 EAP. N.d. Accessed on 13 March 2015. Retrieved from https://confluence.atlassian.com/display/JIRA/Configuring+Workflow Licensed under a Creative Commons Attribution 2.5 Australia License. http://creativecommons.org/licenses/by/2.5/au/.

Czarnecki, K., Lotufo, R. & Passos, L. 2012. Towards Improving Bug Tracking Systems with Game Mechanisms. Accessed on 10 March 2015. Retrieved from http://www.jamk.fi/fi/Palvelut/kirjasto/Etusivu/, Nelli Portal, IEEE Xplore.

Deep Customer Insight. N.d. Accessed on 9 March 2015. Retrieved from http://www.n4s.fi/en/work-packages/deep-customer-insight/.

Digile N4S. N.d. Accessed on 24 February 2015. Retrieved from http://www.n4s.fi/en/.

Download Qt. 2015. Accessed on 24 February 2015. Retrieved from http://www.qt.io/download/.

Edgett, S. & Parkinson, S. 1993. Marketing for Service Industries – A Review. In The Service Industries Journal 13, 3, 19-39. Accessed on 23 February 2015. Retrieved from http://www.jamk.fi/fi/Palvelut/kirjasto/Etusivu/, Nelli Portal, ABI/INFORM.

Cartlidge, A., Rance, S., Rudd, C., Shaw, S., Smith, M., Wigzel, P. & Wright, T. An Introductory Overview of ITIL 2011. 2012. Eds. A. Cartlidge & M. Lillycrop. UK: TSO. Accessed on 1 March 2015. Retrieved from http://www.best-management-practice.com/gempdf/itsmf_an_introductory_overview_of_itil_v3.pdf.

Gu, H., Shu, C. & Zhao, L. 2011. Analysis of Duplicate Issue Reports for Issue Tracking System. In ICMiA 2011 3rd International Conference Data Mining and Intelligent Information Technology Applications, 86-91. Accessed on 10 March 2015. Retrieved from http://www.jamk.fi/fi/Palvelut/kirjasto/Etusivu/, Nelli Portal, IEEE Xplore.

Hossain, M. 2012. Crowdsourcing: Activities, Incentives and Users' Motivations to Participate. In 2012 International Conference on Innovation, Management and Technology Research. Accessed on 2 February 2015. Retrieved from http://www.jamk.fi/fi/Palvelut/kirjasto/Etusivu/, Nelli Portal, IEEE Xplore.

Howe, J. 2006. Crowdsourcing: A definition. Accessed on 11 May 2015. Retrieved from http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html.

Hu, Y., Jiang, H., Luo, Z., Ren, Z., Wu, X., Xuan, J. & Zou, W. 2015. Towards Effective Bug Triage with Software Data Reduction Techniques. In IEEE Transactions on Knowledge and Data Engineering 27, 1, 264-280. Accessed on 3 March 2015. Retrieved from http://www.jamk.fi/fi/Palvelut/kirjasto/Etusivu/, Nelli Portal, IEEE Xplore.

ISTQB:n testaussanasto. 2007. Accessed on 16 February 2015. Retrieved from http://www.fistb.fi/sites/fistb.ttlry.mearra.com/files/ISTQB_Testaussanasto.pdf.

ITIL® Glossary of Terms, Definitions and Acronyms. 2007. Accessed on 1 March 2015. Retrieved from http://www.best-management-practice.com/gempdf/itil_glossary_v3_1_24.pdf.

ITIL – A guide to incident management. N.d. Accessed on 2 March 2015. Retrieved from https://www.ucisa.ac.uk/~/media/Files/members/activities/ITIL/service_operation/incident_management/ITIL_a%20guide%20to%20incident%20management%20pdf.ashx.

Jeong, G., Kim, S. & Zimmermann, T. 2009. Improving Bug Triage with Bug Tossing Graphs. Accessed on 16 March 2015. Retrieved from https://www.cse.ust.hk/~hunkim/images/6/65/Papers_jeong2009fse.pdf.

Kandampully, J. & Khanh, V. 2004. Market oriented learning and customer value enhancement through service recovery management. In Managing Service Quality 14, 5, 390-401. Accessed 7 March 2015. Retrieved from http://www.jamk.fi/fi/Palvelut/kirjasto/Etusivu/, Nelli Portal, Emerald Journals.

KDE Free Qt Foundation. N.d. Accessed on 24 February 2015. Retrieved from https://www.kde.org/community/whatiskde/kdefreeqtfoundation.php.

Kimbell, L. 2011. Designing for Service as One Way of Designing Services. Accessed on 16 March 2015. Retrieved from http://www.ijdesign.org/ojs/index.php/IJDesign/article/view/938/345.

Knoll, L. 2014. Defragmenting Qt and Uniting Our Ecosystem. Accessed on 3 May 2015. Retrieved from http://blog.qt.io/blog/2014/08/06/defragmenting-qt-and-uniting-our-ecosystem/.

Koivisto, M. 2009. Frameworks for Structuring Services and Customer Experiences. In Designing Services with Innovative Methods. Eds. M. Koivisto and S. Miettinen. Keuruu: Otava Book Printing Ltd, 136-149.

Kojo, T. 2015. New Qt Forum now open. Accessed on 11 May 2015. Retrieved from https://blog.qt.io/blog/2015/03/06/new-qt-forum-now-open/.

Lusch, R. & Vargo, S. 2004. Evolving to a New Dominant Logic for Marketing. In Journal of Marketing 68, 1-17. Accessed on 17 March 2015. Retrieved from http://pages.uncc.edu/jhanse15/wp-content/uploads/sites/37/2015/01/SD-Logic.pdf.

Mager, B. 2008. Service Design. In Design Dictionary: Perspectives on Design Terminology. Eds. M. Erlhoff and T. Marshall. Germany: Birkhäuser Verlag AG, 354-357.

Mager, B. 2009. Service Design as an Emerging Field. In Designing Services with Innovative Methods. Eds. M. Koivisto and S. Miettinen. Keuruu: Otava Book Printing Ltd, 28-43.

Manzini, E. 2011. Introduction. In Design for Services. Eds. A. Meroni and D. Sangiorgi UK: Gower Publishing Limited, 1-6.

Mattila, A. Tambyah, S. & Wirtz, J. 2009. Organizational learning from customer feedback received by service employees. In Journal of Service Management, 21, 3, 363-387. Accessed on 9 February 2015. Retrieved from http://www.jamk.fi/fi/Palvelut/kirjasto/Etusivu/, Nelli Portal, Emerald Journals.

Mercury Business. N.d. Accessed on 9 March 2015. Retrieved from http://www.n4s.fi/en/work-packages/mercury-business/.

Meroni, A. & Sangiorgi, D. 2011. Design for Services. UK: Gower Publishing Limited.

Myllymäki, M. 2014. Director, Ventures of Digia Plc. Interview of 4 December 2014.

Myllymäki, M. & Perälä, P. 2015. Interview of 14 January 2015.

Paradigm Change – Delivering Value in Real Time. N.d. Accessed on 9 March 2015. Retrieved from http://www.n4s.fi/en/work-packages/paradigm-change-delivering-value-in-real-time/.

Petersen, T. 2013. Bug Reports Incidents and Some Bashing. Accessed on 15 March 2015. Retrieved from http://blogs.unity3d.com/2013/10/28/bug-reports-incidents-and-some-bashing/.

Perälä, P. 2014. Key Account Manager of Vaadin Ltd. Interview of 18 December 2014.

Qt – About Us. 2015. Accessed on 24 February 2015. Retrieved from http://www.qt.io/about-us/.

Qt Legal – Terms & Conditions. 2015. Accessed on 28 April 2015. Retrieved from http://www.qt.io/terms-conditions/.

Ries, E. 2011. Build Measure Learn. In Inc. 33, 8, 56-63. Accessed on 5 April 2015. Retrieved from http://www.jamk.fi/fi/Palvelut/kirjasto/Etusivu/, Nelli Portal, ABI/INFORM.

Sampson, S. 1996. Ramifications of Monitoring Service Quality Through Passively Solicited Customer Feedback. In Decision Sciences, 27, 4, 601-622. Accessed on 9 February 2015. Retrieved from http://www.jamk.fi/fi/Palvelut/kirjasto/Etusivu/, Nelli Portal, ABI/INFORM.

Schneider, J. & Stickdorn, M. N.d. Customer Journey Canvas. Accessed on 5 April 2015. Retrieved from http://files.thisisservicedesignthinking.com/tisdt_cujoca.pdf Licensed under Creative Commons Attribution-ShareAlike 3.0 Unported License. https://creativecommons.org/licenses/by-sa/3.0/.

Steinberg, R. ITIL® Service Operation 2011 edition. 2011. UK: TSO.

Triaging Bugs. 2014. Accessed on 13 March 2015. Retrieved from http://qt-project.org/wiki/Triaging_Bugs.

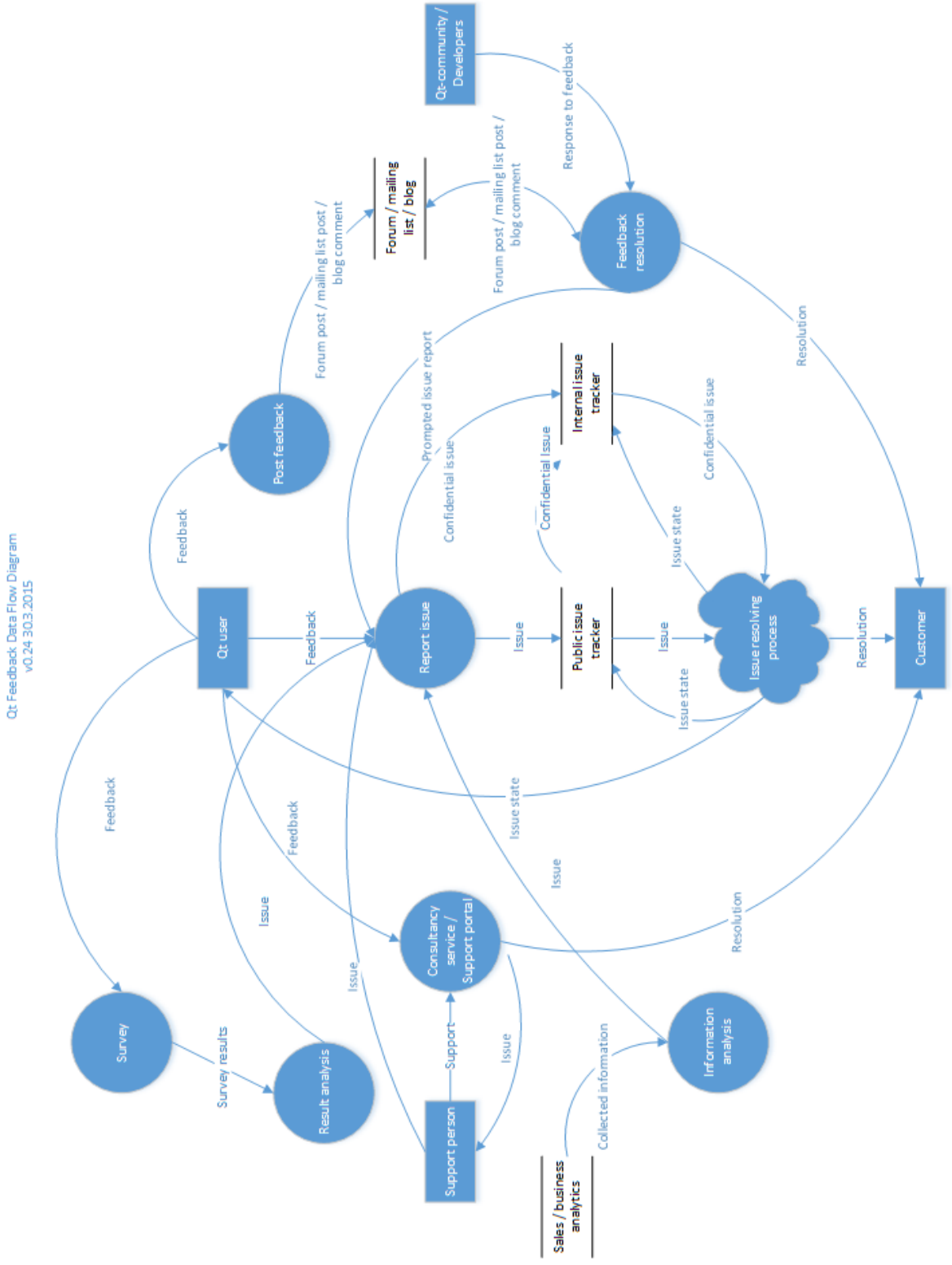
What is an issue. N.d. Accessed on 13 March 2015. Retrieved from https://confluence.atlassian.com/display/JIRA/What+is+an+Issue#WhatisanIssue-Resolution.

What is service design? N.d. Service Design Network. Accessed on 22 February 2015. Retrieved from http://www.service-design-network.org/intro/.

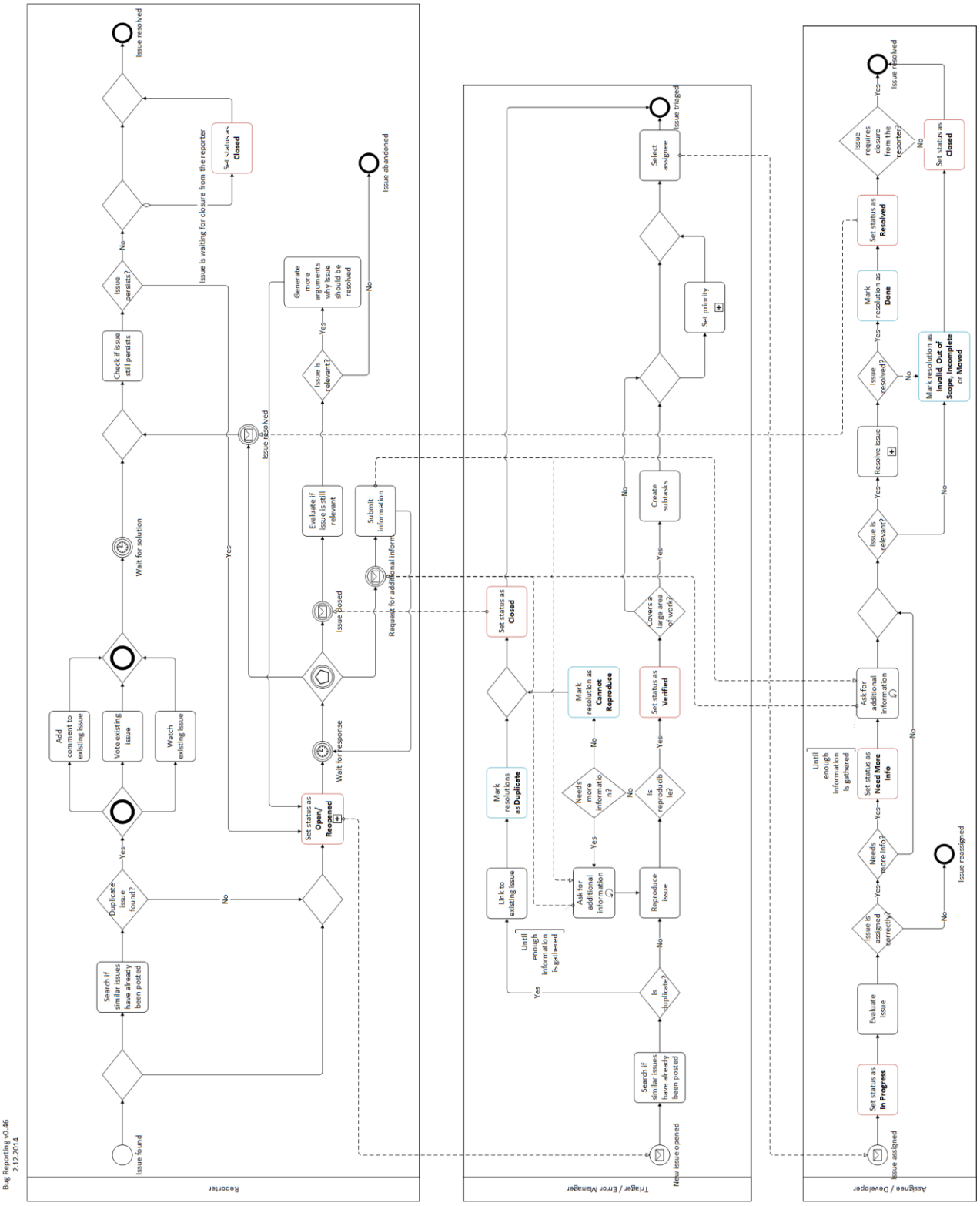
Yuson, P. N.d. Incident and Problem: What is the Difference. Accessed on 2 March 2015. Retrieved from http://www.conceptsolutionsbc.com/it-service-management-mainmenu-60/30-it-service-management/182-incident-and-problems-what-is-the-difference.

Appendices

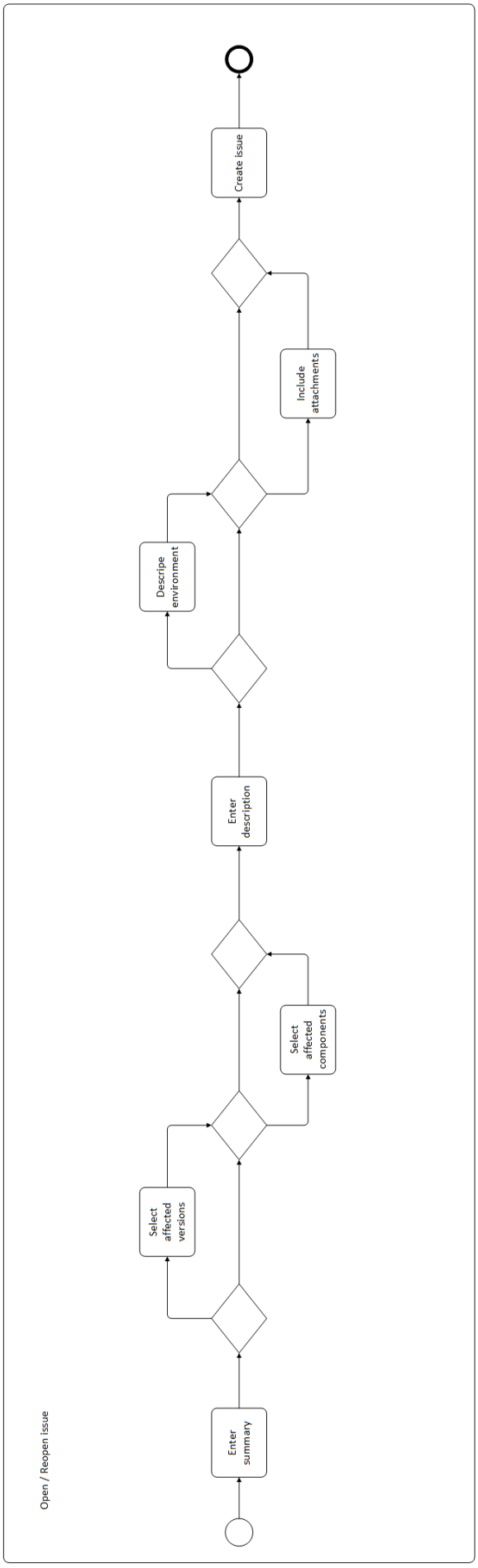
Appendix 1. Standardised Data Flow Diagram of Qt Feedback Channels

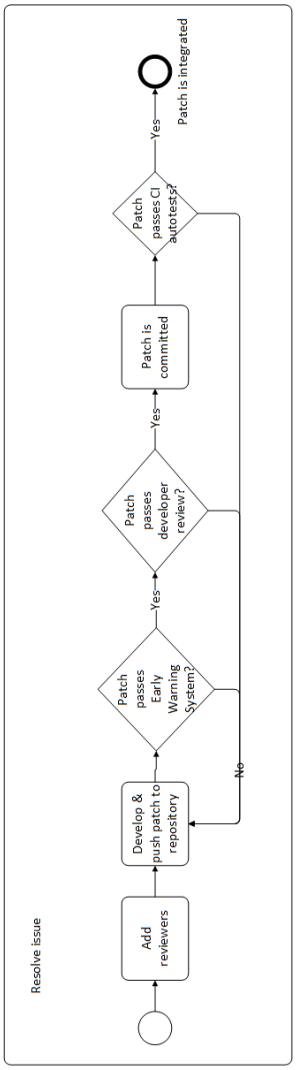


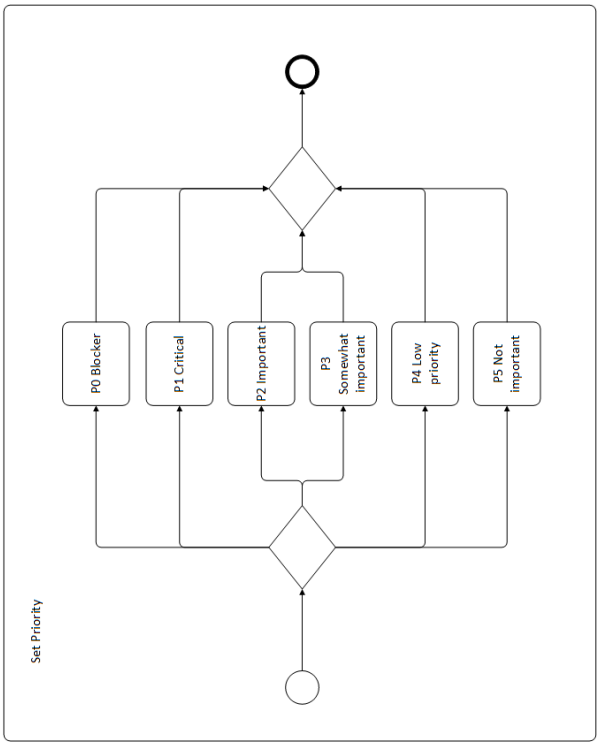
Appendix 2. Issue Resolving Process BPMN



Sub-processes







Appendix 3. BPMN Legend

Start Event
The process starts with this event.

End Event
This event ends the porcess or a part of it.

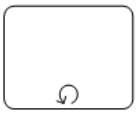
Intermediate Event
Something that is supposed to happen in the process.

Sequence Flow

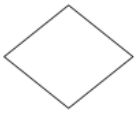
Message Flow

Task
Something that has to be done.

Collapse Sub-Process
Details of this sub-process are not visible in the diagram.



Looping Task
The task repeats until certain conditions are met.



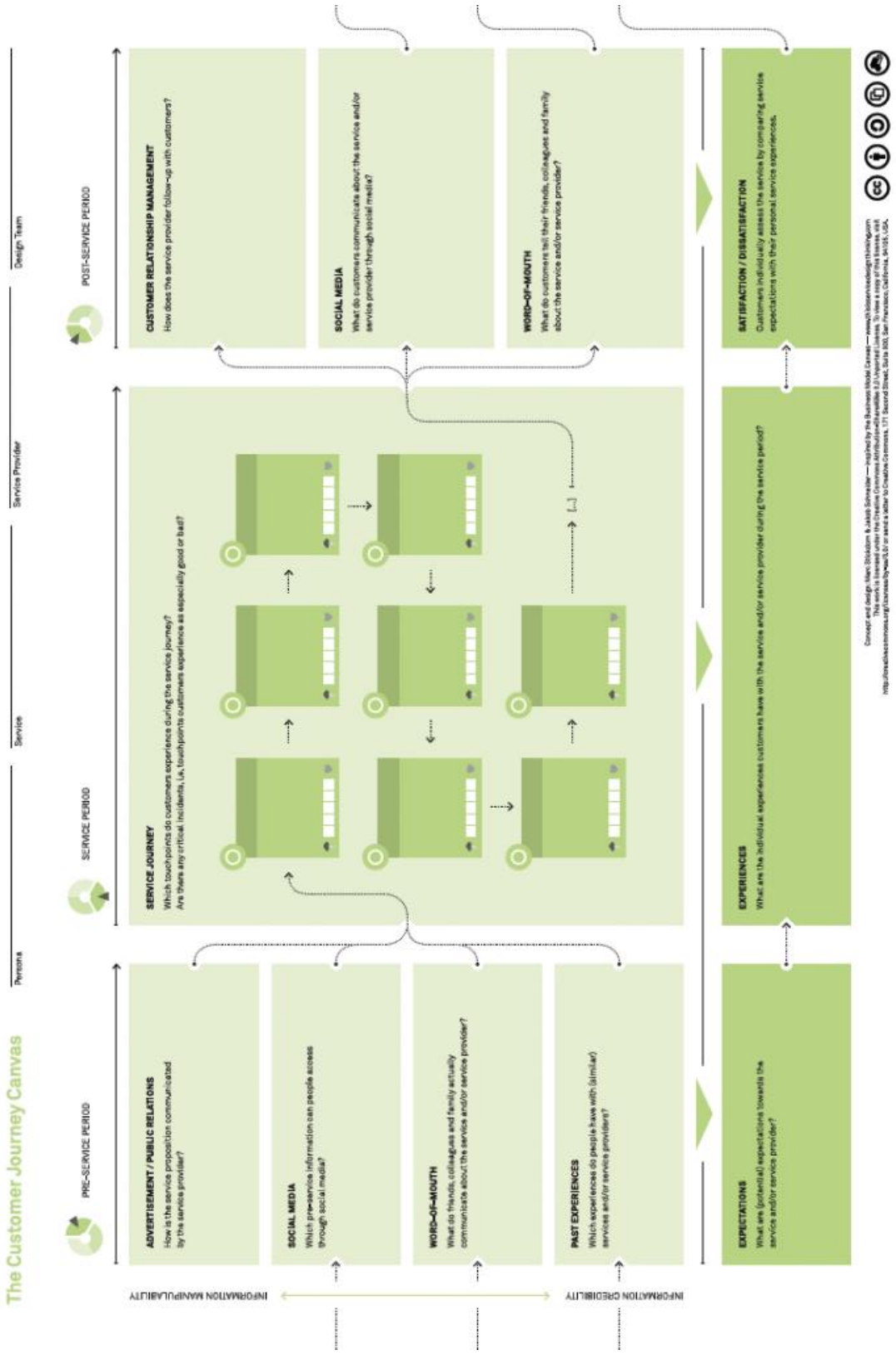
Exclusive Gateway
Point of decision. Only one path can be followed.

Inclusive Gateway
Point of decision. One or multiple paths can be followed.

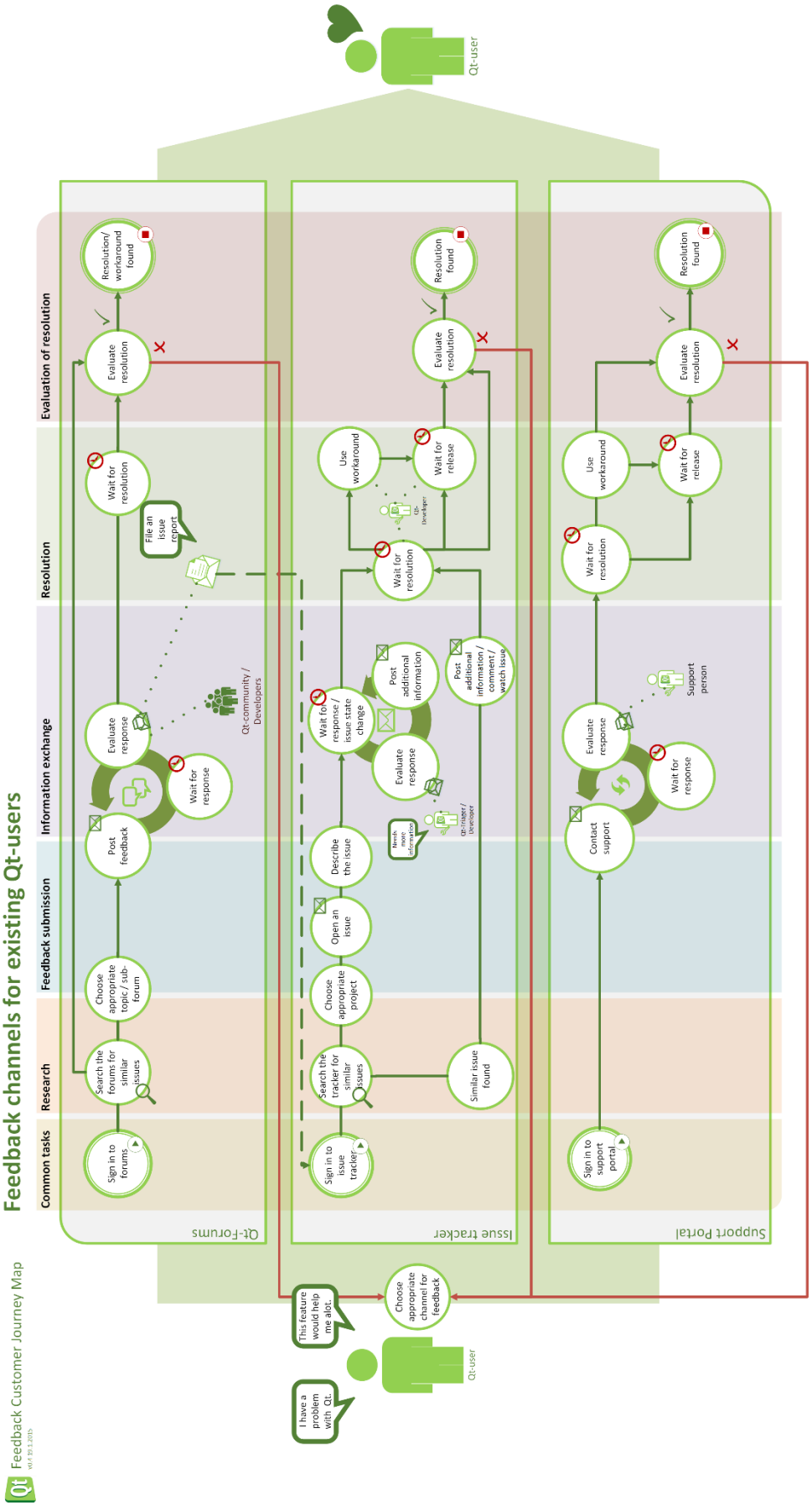
Event-based Gateway
The following event determines the path taken.

Appendix 4. Customer Journey Canvas

Schneider & Stickdorn n.d.



Appendix 5. Customer Journey Map of Qt Feedback System



Appendix 6. Qt Bug Reporting Form