UNIVERSITY OF LJUBLJANA
FACULTY OF ECONOMICS

MASTER'S THESIS

IT PROJECT MANAGEMENT FOR INTERNATIONAL STUDENT ORGANIZATIONS

Ljubljana, Avgust 2018 DINO MEMOVIĆ

AUTHORSHIP STATEMENT

The undersigned Dino Memović a student at the University of Ljubljana, Faculty of Economics, (hereafter: FELU), author of this written final work of studies with the title IT Project Management For International Student Organizations, prepared under supervision of Prof. Dr Talib Damij.

D E C L A R E

1. this written final work of studies to be based on the results of my own research;
2. the printed form of this written final work of studies to be identical to its electronic form;
3. the text of this written final work of studies to be language-edited and technically in adherence with the FELU's Technical Guidelines for Written Works, which means that I cited and / or quoted works and opinions of other authors in this written final work of studies in accordance with the FELU's Technical Guidelines for Written Works;
4. to be aware of the fact that plagiarism (in written or graphical form) is a criminal offence and can be prosecuted in accordance with the Criminal Code of the Republic of Slovenia;
5. to be aware of the consequences a proven plagiarism charge based on the this written final work could have for my status at the FELU in accordance with the relevant FELU Rules;
6. to have obtained all the necessary permits to use the data and works of other authors which are (in written or graphical form) referred to in this written final work of studies and to have clearly marked them;
7. to have acted in accordance with ethical principles during the preparation of this written final work of studies and to have, where necessary, obtained permission of the Ethics Committee;
8. my consent to use the electronic form of this written final work of studies for the detection of content similarity with other written works, using similarity detection software that is connected with the FELU Study Information System;
9. to transfer to the University of Ljubljana free of charge, non-exclusively, geographically and time-wise unlimited the right of saving this written final work of studies in the electronic form, the right of its reproduction, as well as the right of making this written final work of studies available to the public on the World Wide Web via the Repository of the University of Ljubljana;
10. my consent to publication of my personal data that are included in this written final work of studies and in this declaration, when this written final work of studies is published.

Ljubljana, _____ Author's signature: _____

(Month in words / Day / Year,
e. g. June 1st, 201

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

ASF – Apache Software Foundation

BEST - Board of European Students of Technology

CMS - Content management system

CPM - Critical path management

CSCW - Computer-supported cooperative work

CVS - Concurrent Versions System

DM – Developer meeting

ECTS - European Credit Transfer and Accumulation System

HR – Human Resources

HTTPD – HyperText Transfer Protocol Deamon

IPF - International Projects Forums

IT – Information Techology

ITA – IT Administrators

ITC - Information Technology Committee

ITD – IT Developers

ITdept – IT Department (refering to IT Department of BEST)

ITID – IT Interaction Designers

ITPM - Information Technology Project Management

LBG – Local BEST Group

LDAP - Lightweight directory access protocol

LTSP - Long Term Strategic Plan

LXC - Linux containers

NGO – Non-Govermental Organization

OSS - Open Source Software

PA – BEST Private Area (intranet of the BEST organization)

PA/PWS - Private Area / Public Website Maintainance project of IT Department of BEST

PM – Project Management

PMBOK - Project Management Body of Knowledge

PMC - Project Management Committees

PRINCE2 - Projects in Controlled Environments 2nd revision

VP – Vice President

WBS – Work Breakdown Structure

INTRODUCTION

This master thesis strives to take a deep look at how one can go about organizing Information Technology (hereafter: IT) Department for a big multinational non-profit volunteer organization. The main motivation for this comes from the fact that all present-day organizations, no matter what their purpose and line of work are, can significantly benefit from constantly developing and emerging information technologies.

I will start by defining IT project management and look into different theoretical approaches to project management. First, I will look at more traditional project management styles, that are based on Project Management Body of Knowledge (hereafter: PMBOK) and Projects in Controlled Environments 2nd revision (hereafter: PRINCE2) project management methodologies commonly referred to as waterfall or cathedral style of project management. They focus on planning early in order to ensure that the project on the big picture stays on track and fulfills its goals (Hartney, 2016).

Next I will look at more recent agile methodologies; in particular, I will look at Scrum. In contrast to traditional project management, agile methodologies don't focus on strictly defining what has to be done at each stage and instead focus more on how at every moment the participants can increase customer satisfaction as well as overall quality of the end product. I will also take a look at the particular challenges of organizing and managing online geographically distributed teams, as the organization this paper focuses upon is relying on such teams.

Then we will describe how a number of organizations organize their IT departments and more specificity their developers in this way I will provide examples of success stories from the real world. Next, we will describe the international student organization Board of European Students of Technology (hereafter: BEST) in order to provide contrast as to the environment in which the IT Department of the organization operates.

Following that, I will look at the historical development of the department, its different organizational structures as well as reasons and environment in which they changed and evolved over time into its present state.

The thesis strives to show how to go about IT project management in big non-profit voluntary organizations by answering following reaserch questions:

- What is IT Project Management?
- What examples of successful implementations of IT project management of distributed online teams from both IT industry and open source projects?
- How is IT department of student organization BEST organized?
- How does current IT project management in BEST look like?
- How could I improve the IT project management at BEST?

The main motivation behind the research is the ever-present need of modern day organizations to incorporate the latest in information technology, Due to the specificity of work in big international distributed voluntary organizations many of lessons and best practices from the industry and research community don't apply. A thesis focusing on this

area would thus provide the already mentioned organizations with a much needed real-world example by which they could organize or improve their IT departments and projects.

The primary benefit of this thesis is that it would bridge the research gap and provide an example of how to organize a semi-professional voluntary IT Department and how to do IT Project management in such organization. Another goal is to provide the origination BEST with improvement points by contrasting their practices with the best practices in the industry as well as providing them with a theoretical framework which explains and insights various concepts and practices currently at use in the organization.

Primary data collection will be done using the insider researcher approach. In a nutshell, I as the researcher will spend time inside the organization, doing work as a productive member of the organization. This will allow an unimpeded look into the organization and its unique strengths and weaknesses that more traditional data gathering methods wouldn't uncover.

Although this method is useful in very few use cases, as directly involving the researcher in the subject of research as the fact that researcher perceives through his own senses directly adds a certain level of subjectivity to the overall picture. In this case, as I am focusing on the organization and its impact on the productivity of the department it is appropriate and quite commonly used in such settings (Stanleigh, 2016).

Secondary research method applied is the statistical analysis of Git log using R statistical software. Git is an open source version control system that started as a side project in order to better manage already mentioned Linux kernel project and evolved to be one of the best ways to track versions conceived by humans (Software Freedom Initiative, 2017a). The log itself holds records of every change done inside the code-base of the department since 2003. Besides looking at IT Department's git log I will also take a look at two open source projects which have similar scope in order to better contrast and find better improvement points for the current IT department's state. The outcomes of the analysis will be described in the last part of this thesis while the full analysis will be included in Appendix 1.

1 IT PROJECT MANAGEMENT

Information Technology Project Management (hereafter: ITPM) is a subsection of project management that focuses on the specifics of managing project in the IT industry. Project management as defined in PMBOK guide is the application of skills, knowledge, tools, and techniques for the goal of managing and successfully completing projects (Project Management Institute, 2013a).

There are some key differences that differentiate ITPM from regular project management. Many of them are directly related to the technological environment in which such projects exist. More often than not the technical complexity makes the communication with stakeholders more difficult as they tend not to share the technical background of the IT experts (Project Management Institute, 2013a). ITPM requires managers to have a firm grasp of technical sides of the project, the business side as well as project management side that is required to actually manage the project.

1.1 What constitutes a project

But to actually start talking about project management I have to define what constitutes a project. I will start with the basic definition of the project from PMBOK: the project is a temporary endeavor with the goal of creating a unique product, service or result (Wuttke & Zandhuis, 2015). Best way to actually present what projects are is to contrast them to another activity commonly found in such an organization, operations or in other words operational work.

What this means is that projects have a defined start and end, compared to operations that are continuous and part of the organization's daily activity needed to ensure the organization fulfills its goals (O'Connell, 2011). While operations deliver a constant output for the firm, whether it's a manufacturing line, an orchard or a pharmaceutical corporation, projects deliver something unique. The output of a project is meant to bring a tacit improvement to some aspect of the organization that initiated it.

The outcomes of the projects commonly ether improves existing processes in the company, introduce new ones, or create a unique product or service for the company. The main outcome of a project is, therefore, a tangible outcome. From this, I can logically infer that project management is the art of successfully completing projects within the bounds mandated by the immediate environment in which they are conducted.

Projects vary in scope, size, and complexity. The way I manage projects will thus depend on what is needed for the actual project. For instance, it would be ludicrous to use the same project methodology for a small elementary school project that you would for creating a new nuclear power plant. Over the years many project management methodologies were developed in order to fit the needs of different types of projects.

In its inception ITPM was simply viewed as just another project, however, failures, budget and deadline overreaches have shown that ITPM and IT projects, in general, have some particular caveats that the managers have to take care of in order to ensure successful termination of projects (Picarus, 2012).

Currently, there are two main project management styles on which modern ITPM rests, namely traditional PMBOK/PRINCE2 based approach sometimes also called waterfall methodology. Or a more recent agile approach, which is characterized by its flexibility and ability to change scope and specifications of the project on the fly if the project demands it. In the following text, I will analyze the two approaches to ITPM.

1.2 Classical project management methodologies

When I talk about classical project management methodologies I am generally referring to a PRINCE2/PMBOK approach to project management. The main philosophy behind traditional methodologies is that longer into a project you are more expensive is to change something. Therefore they strive to predict and plan the project as much as possible and to try to formalize, standardize and document as much of the project as possible.

According to PMBOK projects as defined by PMBOK guide can be divided into the following phases:

- **Project conception and initiation** - I examine the project suggestion to ensure it benefits the organization.
- **Project definition and planning** - I define the scope of the project followed by defining budget and schedule as well as planning for all the resources and tasks that will consume them.
- **Project launch or execution** - In this phase I start working on the actual project following the already planned schedule.
- **Project performance and control** - I evaluate how our project compares to the actual plan from the planning phase.
- **Project close** - I finish all the tasks and ensure that our client has approved the project outcome.

PMBOK style of management strives to heavily document the project progress as well as to ensure proper knowledge management for both the project implementation and the final outcome of the project. Also, PMBOK splits the project management knowledge into different areas:

1. Project Integration Management,
2. Project Scope Management,
3. Project Time Management,
4. Project Cost Management,
5. Project Quality Management,
6. Project Human Resource Management,
7. Project Communication Management,
8. Project Risk Management,
9. Project Procurement Management,
10. Project Stakeholder Management.

Each area focuses on different aspects of project management in order to ensure that the project managed finishes on time, on budget and that its positive impact can be released afterward in the organization. Now I will quickly go over different parts of project management knowledge as defined by PMBOK.

1.2.1 Project Integration Management

Project Integration Management has the goal of identifying and unifying different aspects of the project such as its processes and activities. The main point is to define and make choices concerning resource allocation, competing demands, alternatives as well as tailor the processes and activities to match needs of the project.

During this phase according to PMBOK Guide we should define and document the following:

- **Project charter** - the document that formally authorizes the existence of the project and provides the PM with authority to utilize organizational resources for the project. Here we define the purpose and overall objectives of the project as well as defining required completeness of different aspects of the project.

- **Project management plan** - is a comprehensive document that is the basis of how all work will be done in the project.
- **Manage project knowledge** - we have to look for relevant knowledge that is relevant to the project as well as to specify the ways we will in the future document the knowledge from the project.
- **Monitor and control project work** - details ways we will track review and report the progress of the project as well as its individual parts. This process is performed throughout the project.
- **Perform integrated change control -** details how we will deal with changing circumstance and relevant change to the project documentation, tasks, budget, and deliverables.
- **Close project or phase** - how will we finalize and close the project at the end, how will we detail the project deliverables as well as distributing the leftover project resources after the project is closed.

1.2.2 Project Scope Management

Scope management is there for us to know what is included in the project and what isn't. It details different work, deliverables as well as what is at the end required for our project to be successful.

In the project scope phase, we start by defining how exactly we will go about scope management. How will we define and develop the scope, and later on how exactly will we monitor, control and at the very end validate that our target scope was achieved? Following stages are all done with respect to the scope management plan (Project Management Institute, 2016a).

Next step is to collect all requirements so that we actually know what the project aims to achieve and what it doesn't aim to achieve. We gather requirements from our stakeholder's either through brainstorming, interviews, focus groups, surveys or through benchmarking. We do all this in order to gather the best information possible based on which we will define the actual project scope.

Project scope is there for us to define what exactly will be delivered by the projects and which criteria will be used to measure the fitness of the deliverables against the specification. Also, it is important at this stage to define what is not part of the project as wasting time on things not relevant to the project can be a big waste of resources and time.

Next thing we plan is work breakdown structure (hereafter: WBS). WBS is a document in which we break down every single thing that needs to be done in individual tasks. We start with top-level deliverables and go down into more detailed and specific subcomponents so that we can account for all work required on a project (Harris, 2016).

Following the creation of WBS, we validate the scope of the project. In short, we go over the whole scope with its individual parts with the customer or project sponsor so that we can ensure that we all agree on what will be the desired scope of our project (Project Management Institute, 2016a). It is important to get the write off from the customer as not agreeing on what the actual scope of the project is a recipe for disaster later on.

The last step is actually a process that will be ongoing throughout the project, and that is controlling the scope of the project. In a nutshell, we constantly monitor progress and deliverables against the requirements, WBS and the client-accepted scope document. In this stage, we will also do any scope changes in case they are required by methodology specified in the project scope management planning phase.

1.2.3 Project Schedule Management

The main point of the schedule management is to ensure that our project finishes on time. The process starts with the team selecting the scheduling methodology, an example of such methodology would be "critical path management" or CPM. After that based on constraints, dependencies, milestones and other project information management team create the project schedule (McCall, 1973).

As with most things in PM we start with planning how we want to manage the whole process. What exact project methodology are we going to use, with what accuracy are we going to measure our estimates, what thresholds are acceptable for the projects as well as how and how often are we going to report everything during the project and after its completion (Project Management Institute, 2017).

Based on scope and WBS, in particular, we will define all the activities that will happen during the project. After we get the list of activities we will analyze their interdependence and relationships. With the end goal of plotting all the activities and arranging them in such a way as to ensure their logical order of precedence related to their start and finish times is respected.

To find the critical path for a project we start by plotting all activities based on their order of precedence arranging them in a cascading graph. The shortest path through which we can traverse to the end of the project is the critical path, which in a nutshell means that if we have any delay in any activity on this path will cause the whole project to get delayed (McCall, 1973). Difference between critical and non-critical paths is in that non-critical paths may tolerate delays without causing the whole project to get delayed. Commonly this activity is performed using project management software such as Microsoft's Project software package (Harris, 2016).

After we have an idea of how all the activities fit together we can start estimating how much time we have to allocate to each individual activity. One of the best ways to go about estimating is to get input from people who are most familiar with the nature of work of that specific activity. There are many ways of estimating the time required, most common ones are analogous (using historical data), parametric (using an algorithm based on historical data and project parameters) and three-point estimating in which case we sum up most likely, optimistic and pessimistic estimate and divide them by three thus getting an average amount of time for the activity (Project Management Institute, 2013).

Next step is to actually develop the schedule. We start by putting together the outcomes of previous phases and creating a schedule based on the model we selected earlier. At this point, we also have to consider doing the critical path analysis in order to ensure that project's most critical activities don't get delayed as well as some other optimizing activities such as resource optimization, network analysis, and schedule compression.

Controlling the schedule happens throughout the project. We strive to determine how well we are currently following the schedule and then take actions to either ensure that project activities are on schedule or if that proves impossible we change the schedule and accompanying documentation accordingly.

1.2.4 Project Cost Management

Cost management has to do with the financial side of the project. In short, we try to answer the question of "How will we pay for the whole project?" We have to consider planning, budgeting, estimating, financing, funding, managing and controlling costs inside the project.

We start by planning how exactly we will manage costs in this project. Which units of measure are we going to use, and for which resources; what is the level of precision that we need for estimating resources and what are reasonable thresholds for realistic cost estimates (Wuttke & Zandhuis, 2015). How we will procure, use and report said resources are also an important consideration at this stage.

Next step is estimating costs with the goal of getting a picture of how much this particular project will cost the organization. We start with a quantitative estimate of different resources that are needed to finish the project followed by estimating or predicting real word price for said items.

After estimating individual costs we can go on to aggregate them into a comprehensive budget. From it, we can infer an authorized cost baseline which we can use to baseline the cost against the project performance related to costs later as the project progresses (Hartney, 2016).

Throughout the project, we have to take care that we control the costs. It goes from authorizing expenditures to managing changes to budget when they happen as well as preventing overrunning the budget and wasting resources throughout the project.

1.2.5 Project Quality Management

Quality management strives to incorporate organization's quality policy regarding different project requirements as well as stakeholders' objectives in order to ensure that the end product satisfies the former and latter.

We start out by planning how to manage quality throughout the projects. Starting from the scope baseline we also have to incorporate requirements management, risk management as well as stakeholder engagement into the greater picture. A simpler approach may include using some of the already renowned international standards such as the ISO standard family (International Organization for Standardization, 2017a).

After we make the plan we have to translate the said plan into executable activities that we then incorporate into organization policy as well as WBS (and in turn rest of the activity and time planning activities). In the end, we strive to also control quality or in other words compare the planned and executed quality related activities as well as the end or in progress product against the requirements, specifications, regulations, standards as well as

internal documents. Our main goal is to ensure that the quality fits our needs and in case it doesn't take action to ensure quality compliance (International Organization for Standardization, 2017b).

It is important to make a distinction between managing quality and quality assurance. Part of our strategy for managing quality can be quality assurance (as a job position, team or department depending on the needs of the organization). But we should never limit the managing quality process to just quality assurance (Santos & Belo, 2013).

1.2.6 Project Resource Management

Resource management as the name suggests is the part where we dice how to find, acquire, and use all the resources in the project. We start by planning how we will manage the resources of the project, starting with the methodology for resource management and ending with the reporting process.

Estimating activity resources is the phase where we determine the types and quantities of resources that the project will need in order to perform activities required for its successful completion. The end product is closely tied with the cost estimation of the project cost management section, as we need to know said quantities in order to estimate the cost of quantities (Hartney, 2016).

Following estimation, we proceed on acquiring the said resources. It takes quite some effort and time to find and acquire resources of sufficient quality that also fit project's budget. The acquisition process is carried out through the project as needed.

An important part of the resources acquisition is assembling the team that will perform the actual project. Matching of team members to their roles and to the overall teamwork inside the project is quite crucial as it ensures that we complete the project up the spec. Following the acquisition, we also have to develop the team.

Teamwork is crucial in order to ensure efficient and smooth operation of activities throughout the project. Going through team development stages, ensuring all team members have enough knowledge and skills for the project, taking special consideration for virtual teams as well as partially remote teams are important considerations for a project manager (McCarthy, 1987). Remote teams require a special set of skills as well as tools in order to ensure proper communication inside the project. Please refer to section 2.2 for more details on specificities of managing software projects in an environment where teams are remote geographically distributed.

After we develop the team we have to manage it through the project, this includes tracking performance, feedbacking, resolving conflicts and assigning to tasks in order to optimize project performance. This step requires the manager to be quite experienced in conflict resolving as well as to have strong decision-making skills.

Controlling resources means that we ensure that resource allocation, utilization and at the end release and redistribution is done on time as well as optimizing said allocation throughout the project.

1.2.7 Project Communication Management

Communication seems like one of the simplest things to manage, but in reality, it requires special attention as without properly conveying information our project can go down a slippery slope very quickly. We start with the planning of project communication.

We have to plan how we will address each stakeholder group individually, through which channels, what language we will use and do we have to take on some special precautions while sending the message (Earley & Ang, 2007).

During the project communication, we have to take note of collecting, processing and storing all relevant project information followed by dissemination and distribution of said information through relevant channels to relevant stakeholders in a timely fashion. Besides communicating through the project we must also monitor the said communications in order to ensure they are following the communication plan as well as organizational and stakeholder requirements concerning communication.

1.2.8 Project Risk Management

The risk is an ever-present threat that can endanger every project, while we can never make it disappear completely we can take effort in mitigating the risk. We have to consider the size of the project, complexity, importance and developmental approach as for all of them we need to look for a different scope of risk management.

We start by planning risk management this is very dependent on our organization's risk policy, environmental and regulatory factors as well as stakeholders risk appetite. We have to define the process through which we will perform other stages of risk management as well as how we will evaluate and communicate said risks (Wuttke & Zandhuis, 2015).

Next on is identifying risks, describing and evaluating them and putting them into the risk register which is a formal document which lists, describes and provides potential responses to risks. Following identification we do the qualitative risk analysis which prioritizes risks based on how high the chance of them occurring will be and how big of an impact can they have on the organization and the project. After qualitative, we perform the quantitative risk analysis which analyses the combined effect of the previously identified risks and other sources of uncertainty on overall project objectives (García-Álvarez, 2015).

Following the analysis, we have to plan the risk responses, with the priority given to the potentially most damaging risks that can significantly affect the project. Following the definition of risk responses, we have to implement said risk responses. This involves ensuring proper documentation of risk responses as well as communication said risk responses to relevant team members and stakeholders.

Finally, we have to monitor the risks to ensure that new risks don't occur or even worse actually happen. Periodic audits should be taken to ensure that all relevant risks are identified, evaluated and properly documented (Project Management Institute, 2016a).

1.2.9 Project Procurement Management

This stage includes the processes necessary to purchase or acquire resources outside the project team. It starts by specifying who has the authority to purchase what, at the end to whom do we have to take the final procurement or purchase contract to. This is undoubtedly part of the planning phase of procurement management.

The plan covers many things, starting with cost estimates and procurement needs it identifies the short list of qualified sellers (Project Management Institute, 2013) . After selection one, we enter negotiations with the desired seller, evaluate the quality and technical characteristics of the desired resource and in the end, we decide how will we go about finalizing the negotiations and acquiring the needed resource or service.

Following the planning phase we conduct the actual negotiations and in the end, we prepare required facilities to receive the resources and finalize the legal formalities required for each purchase to be legally binding (Yu, Goh & Lin, 2012). Throughout the process, we have to control the procurements to ensure that they meet the specification and the budgetary and temporal requirements of the project.

1.2.10 Project Stakeholder Management

This aspect of project management strives to identify people, groups, and organizations that could impact or in the end get impacted by the projects. It strives to identify the stakeholders of the project then plan and manage their engagement as well as monitor how the planned engagement processes are being fulfilled.

We start by identifying the project stakeholders as well as documenting relevant information concerning their interest, potential level of involvement, influence or potential impact on the success of the project. After we are aware of the stakeholders we have to plan how exactly we will engage them (Wasieleski, 2017).

What channels of communication are appropriate, how our organizational policy influences our communication as well as in what engagement level our stakeholders currently are and to what degree we would want to change their engagement level. After we plan it we have to manage the execution of the planned stakeholder engagement, ensuring that all the planned activities are up to specification. This leads us to monitor stakeholder engagement. Which in essence is the process of monitoring our engagement channels as well as evaluating stakeholder reaction and feedback on project activities based on the engagement of stakeholders in the project (Wuttke & Zandhuis, 2015).

2 AGILE PROJECT MANAGEMENT

The origins of agile approaches to project management can be traced back to 1950s, but their formalization through structured methodologies that have seen widespread use and success is normally tied to 1990s. It is then that the major methodologies were first published and successfully replicated in different organizations (Project Management Institute, 2017). Today when we think agile project management, we mostly refer to the adaptive iterative development cycle, where we start with considerably less planning and

organizational overhead than in traditional PM, but we allocate specific time during the project execution to adapt to changing circumstances.

The term “Agile” itself started being used to describe this family of methodologies in the year 2001 when the authors of major programming and development books and publications got together and created “The Agile Manifesto” (Beck and others, 2001a). Despite the manifesto itself being mere four sentences, it had a significant effect on the software development community. It states that we should focus on the people inside the project, that we should strive to respond to change, that I should collaborate with our customers and that the end goal of a project is the goal is to create something not to document the creation of something. The agile movement can be summarized in its 12 principles (Beck and others, 2001a) :

- Customer satisfaction through early and continuous delivery of valuable software features to the customer.
- Welcome changing requirements, even in late development.
- Working software should be delivered frequently and continuously (weeks rather than months).
- Close daily cooperation between business people, stakeholders, developers and other roles involved in software development process.
- Projects are built around motivated individuals, who should be trusted to deliver high-quality results.
- Face-to-face conversation is the best form of communication (co-location)
- Working software is the primary measure of project’s progress.
- Sustainable development that can be maintained at a constant pace without exhausting the team.
- Continuous attention to technical excellence and good design.
- Simplicity - the art of maximizing the amount of work not done - is essential to project’s success.
- Best architectures, requirements, and designs emerge from self-organizing teams empowered to make a difference.
- Regularly, the team reflects on how to become more effective and adjusts the development process accordingly.

The sharp contrast between agile and classic PMBOK methodologies can be seen in the different focus point of the methodologies. Where PMBOK would focus on planning and controlling the project execution (Project Management Institute, 2013), agile methodologies focus on the actual work required for the project to finish, and on adapting future work to the changing requirements (University of Minnesota, 2008).

This allows for lower project overhead especially because of the way the project reacts to change makes agile practices more appropriate for projects that cope with vague or changing scopes and requirements. As this is commonly the case with software projects, it is not surprising that many agile methodologies ether trace their roots or at least focus on software development industry (Shore, 2007).

There are many different agile software development methodologies, major include extreme programming, lean software project management, scrum, kanban (University of

Minnesota, 2008). Agile unified process etc… Below I will briefly go over few major methodologies before describing scrum in detail:

Extreme programming - this methodology focuses on code quality and avoiding excessive work. At its core is the concept of pair programming where 2 developers work together on one computer. It stipulates test-driven development, where tests are written before the code that they are testing. Other main features include a flat management structure, open workspace that encourages collaboration of team members, constant reviewing of code (hence two people working on it) as well as expecting changes to the user/customer requirements. Its critics mostly point out its rigidity and inflexibility as referring to some of its principles, the concept of pair programming, in particular, has been identified as a big problem especially to managers and companies that are mostly considering ITPM same as that of any other type of project inside the organization (Beck, 2000).

Lean software project management - Is the adaptation of lean manufacturing to the software industry. The main principles are Eliminate waste, amplify learning, and decide as late as possible, deliver as fast as possible, empower the team, build integrity in and see the whole. The main goal of the framework is to empower employees to improve the processes inside the company. The framework has seen quite some success as many big corporations have adopted lean project management in other areas of business and because there is a significant amount of research done on the usage and implementation of lean processes in a company (mainly in the manufacturing sector) (Poppendieck & Poppendieck, 2003)

Scrum - methodology focuses on iterative focused coding sprints, almost all detailed planning is done for next couple of sprints which are time-bound efforts with the goal of producing a partially shippable product. We will look more deeply into Scrum methodology later in the chapter (Rubin, 2012).

Kanban - is a method based on the Kanban board. This is simply a digital or physical wall with different columns on which we move all the tasks through the process. It focuses on visualizing all the work which works very well as it allows all project members to quickly determine the state of every task in the process (Ortiz, 2016)

Empirical research has shown that agile methodologies can improve ITPM, research has shown that lean and agile PM help dealing with project complexity (Sohi, Hertogh, Bosch-Rekveldt & Blom, 2016), while Summers (2011) has indicated that it does improve stakeholder satisfaction which has been identified as one of the leading causes of project failure. In their research (Serrador & Pinto, 2015) shows that agile has a statistically significant effect on efficiency, user and stakeholder satisfaction as well as the likelihood of project delivering on time.

2.1 Scrum

Scrum is an agile project management methodology that strives to deal with project complexity as well as uncertainty by adopting an iterative approach to project management that offloads up-front planning and commitment to as late as possible in the project.

Scrum starts with a product backlog which simply holds all features required by the project. We start by prioritizing a rough backlog and then for the items with the highest priority we analyze and define them in detail. This work is continuous throughout the project as we strive to prepare and plan the following stage when the previous finishes. But unlike the traditional PM methodologies stages are not separated by type of work, they are small self-contained focused efforts with a comprehensible, potentially shippable outcome. Commonly referred to as sprints they last 2-4 weeks and are executed by a diverse team consisting of programmers, designers, testers and any other role required by the project (Rubin, 2012).

By avoiding detailed up-front planning and offsetting the planning process throughout the project we ensure that we waste less time and effort on features that won't be used or that due to change of circumstance. Main philosophy that drives planning in scrum teams is that up-front planning should be helpful without being excessive. Meaning that while we should put the effort in pre-planning the project, we shouldn't waste effort on overly planning said project. Instead of the project, we should ensure that we adapt our plan or completely re-plan if the changing circumstances demand it (Serrador & Pinto, 2015).

While in this approach we lose the rigid and detailed project plan at the beginning of the project, we gain the ability to adapt our solution to the changes and thus provide a better solution for our customer. When we join this with the constant communication and reviewed of work process throughout the project we end up being considerably more likely to ensure customer satisfaction and high-quality final outcome of the project (Beck and others, 2001b).

We will start the description of the core values of Scrum (Scrum Alliance, 2004):

Focus - Scrum focuses on only a few things at the time, making it easier to deliver higher quality outcome at a faster pace due to lack of distractions.

Courage - Scrum teams strive to be more confident when they tackle a challenge due to the cohesive and iterative nature of work organization.

Openness - Making everything as transparent as possible so that everyone has an easy access to information.

Commitment - Scrum teams tend to be more committed to the project as they own it and are able to validate its progress in the short sprints.

Respect - Scrum teams tend to respect themselves as well as rest of the organization, customers, and other stakeholders.

Scrum team is composed of multiple different roles, most important are:

- Product owner,
- Scrum Master,
- The development team.

The project owner is the person who holds full responsibility for the project, he or she has the authority to decide which features go into the product and what priority is assigned to each item. During the project, he is responsible for communicating with all relevant parties

and ensuring that the end customers business needs are properly translated into items that the development team can use to actually develop the product (Rubin, 2012).

Next key role is the scrum master who acts as a coach of the scrum team ensuring that the team doesn't drift from the scrum as well as ensuring that the scrum process gets developed and better adjusted to the circumstances of the organization. Towards the team, he or she acts as a coach and a facilitator. Another important aspect is the fact that scrum master works hard to remove impediments and protects the team from outside interference thus ensuring that the team can focus on the thing they are supposed to do in the first place (Bass, 2014).

The development team consists of various job roles that are relevant to the end product of the project. Depending on the project needs this team can consist of programmers, UX designers, administrators, testers and so on. The main point is to get a diverse cross-functional, self-organizing team of people who can design, build and test the product or project outcome. They are ultimately responsible for the project outcome and are in charge of executing each phase of the project (Rubin, 2012).

Scrum team engages in different activities each adequate for their specific domain of project work. The project starts with the creation of the product backlog which contains a list of all features that go into the final product. Going with the agile philosophy this initial plan is just a rough sketch that does not detail most aspects of the end product, with a rough release train as well as deadlines that should be met for each release. Following this initial plan, we start doing the scrum sprints. This part, in particular, shows how scrum differs from traditional PM approaches (Willes, 2007).

Instead of planning everything in detail at the beginning of each sprint we will take time to go over the features that currently have the biggest priority and we will thoroughly analyze them and plan them for the next time-boxed development effort with the goal of having a potentially shippable product increment at the end of each sprint. This is the concept of progressive refinement which strives to go more into detail on every item as that items development time arrives. Most commonly used format for expressing the business value of an individual item in the product backlog is the User story (Green, 2016).

A user story is a small card that in a short sentence describes what is the goal that user has. Commonly used format for user stories is “As a user <user role> I want to <goal> so that <benefit>.“ The point of the user story isn’t to capture all details of the task, but to act as a conversation starter around which the development team together with the stakeholders and product owner can deconstruct and detail the specific task. Commonly bigger items in the product backlog are referred to as epics, that are gradual as they come closer to the top of the stack split into realize sized items, then sprint sized items and finally into individual user stories for the upcoming sprints (Veazie, 2018).

Before the actual sprint starts we start by planning the said sprint. This is done by taking top items from the prioritized product backlog, defining and estimating individual tasks and creating a sprint backlog. During this, we detail how hard each task is and then we ensure that during the sprint we have enough to fill in a whole sprint worth of work. We can estimate them through different means, with the most popular being story points and ideal work days method. Based on previous sprints we can estimate how much work the

team can do during a sprint and then we can use this knowledge to plan the upcoming sprint. The amount of story points or ideal days that a team does during a sprint is referred to as the team's' velocity We should take care to detail the effort required for upcoming sprints but not also for items lower down the product backlog. For them using a relative measure such as T-shirt sizes (M, L, XL, XXL) is adequate as we don't necessarily need to look for details for the features that are considerably further down the product backlog (Rubin, 2012).

Ideal days estimation method gives a consensus estimate of how much time the team would need to dedicate in order to complete the task. Ideal days differs from actual days in an important aspect as normally developers won't be able to dedicate full work days to just one task and will get distracted with other activities they have to perform in the organization (Alyahya, 2013).

Story point's method doesn't try to put an absolute value on a task; instead, it tries to make a relative estimate of hardness by using a set of values that are not linear. One of the common set of values is the Fibonacci sequence where the next number is the sum of previous two (1, 2, 3, 5, 8, 13…). The main reasoning behind the way story points method is designed is the fact that human brain is considerably better at estimating relative difference than absolute values (Satapathy & Rath, 2017).

Continuously during the project the project owner, scrum master, rest of the scrum team, as well as stakeholders, will work on grooming the project backlog. This is the process of detailing the user stories for upcoming sprints, as well as reevaluating and prioritizing items currently in the backlog. In case it's required we can remove current or add new items to the backlog during backlog grooming. When and under which circumstance the grooming takes place depends on the organization, but it's most commonly performed by the scrum team and product owner in a timeslot predetermined relative to the sprint (Veazie, 2018).

Each day during the sprint the team meets for a few minutes in order to share how the progress is going and discuss any problems encountered. The goal is to have a quick overview as to ensure that the whole team is on the same page and that team members can jump in and help a struggling teammate. At the end of the sprint, we are left with a potentially shippable product increment (Pauly, Michalik & Basten, 2015).

Potentially shippable product increment doesn't necessarily imply that the sprint outcome should be shipped, but the fact that we strive to work on a self-contained part of the system. Point of this is for us to be able to showcase this outcome to the stakeholders and scrum team in order to be able to gather feedback from them as having the ability to show something that runs (Rubin, 2012). Different people visualize things differently which is why being able to sit them down in front of a working product (or increment of a product) is the best way for them to see if this product satisfies their needs and to get how it can satisfy the stakeholder needs better (Earley & Ang, 2007).

At the end of each sprint, we also do a sprint review, which gathers scrum team, project sponsor, stakeholders, customers and interested members of the organization. It focuses on the reviewing the work completed in the previous sprint and how it fits into the overall development effort (Bass, 2014). Following the review, the scrum team does a sprint

retrospective. Sprint retrospective is conducted with the goal of analyzing how the last sprint went and what we can change to ensure that the following sprint gets incrementally more suited to the organization's needs (Rasnacis & Berzisa, 2017). This step is the key to continuous improvement and refinement of the development process that comes with scrum.

2.2 Specificities of managing distributed online projects

An important consideration to note is the fact that the organization we are going to discuss works primarily with distributed remote teams. That is, all members work from home and use teleconferencing and remote collaboration tools in order to coordinate their daily activity as well as for long-term planning.

There are some key differences between managing traditional on-site project teams and teams that are geographically distributed. Distributed teams are a relatively new occurrence as we first see them able to actually be productive thanks to computer-supported cooperative work (hereafter: CSCW). CSCW is a generic term generally used for the concept of using technology and technological improvements in order to enable teams to communicate and collaborate remotely (Alyahya, 2013).

Some of the core components of collaborative work identified by CSCW researchers (Harper, 2016):

- **Awareness -** the extent to which members are aware of their college's work and progress.
- **Articulation work -** the ability to split work into pieces handled by individual team members and their ability to combine it into a cohesive end product.
- **Appropriation -** ability to adjust technology to the team or group needs or organizational norm.

Research has shown that managing and participating in distributed online projects requires specialized communication and cooperation knowledge from both project manager and the team. Getting participants familiar and comfortable with each other is considerably harder if teammates work from remote locations (Domschke, Bog, Uflacker & Zeier, 2009). This can make the manager's job of creating a safe work environment considerably harder, especially if the team lacks knowledge of using technology and norms that are specific to online-only communication.

Ensuring proper awareness, in the CSCW sense, that is that all members of the team are aware of what their colleges are working on and what their progress is especially challenging (Wei, Crowston, Eseryel & Heckman, 2017). Other considerations that are not relevant to regular companies can become crucial for a distributed software project. Things such as cultures, time zones, availability & technological issues can cause significant issues for a project (Ryder, 2017).

Although in recent years there has been a new wave of tools aimed at supporting distributed work it has been noted that there has been a distinct lack of empirical research evaluating these tools in a distributed team. According to the Rodriges and Vizcaino as

much as 75% of tools made for CSCW lack proper empirical evaluation (Portillo-Rodríguez, Vizcaino, Piattini & Beecham, 2012).

More practical approaches to researching distributed teams determined that communication is the key focus for distributed team project manager. Following communication needs arose as being crucial to the project (Foss, Frederiksen & Rullani, 2016):

- Asynchronous direct detailed communication (email).
- Need to store knowledge in a commonly accessible location (wiki, blog).
- Video Conferencing (Skype, Google Hangouts, Viber, WhatsApp and other video conferencing tools).
- Direct concise communication (instant messaging).
- Sharing periodic status updates with other teammates (newsletters).

This chapter gave us the answer to the first research question: "What is IT Project Management?" We defined it as the sum of tools and techniques grouped in individual methodologies that an organization utilizes in order to ensure successful project completion. Now I will take a look at how different generally acclaimed organizations go about organizing their projects.

2.3 Open Source project management

I will start by looking at how in general Open Source Software (hereafter: OSS) projects are managed. This is quite a vibrant research area as many researchers are interested in trying to understand how groups of volunteers free of charge are able to create high-quality software that compares to and even surpasses commercial offerings.

As Belenzon and Schankerman research suggests that there is no important distinction in team productivity between open source software communities and their more closed source counterparts that still operate on a voluntary basis (Belenzon & Schankerman, 2008), I can safely refer to metrics used in distributed OSS projects and projects of the IT Department of BEST.

This means that looking at research done on OSS software project and a project with a closed license share the same set of fundamental concepts and practices, allowing us to look at OSS projects and draw conclusions that are relevant for the IT department of BEST. The sharp contrast in the way OSS software is created compared to traditional software industry was noted in Eric Raymond's book "The Cathedral and the Bazaar". His metaphor has since been embedded in the OSS community (Raymond, 2001).

He compares the classic software industry as a medieval cathedral building site. Skilled craftsmen of all trades work through their guilds in order to achieve the grand vision conceived and directed by the master architect of the project. The metaphor does a good job of describing the traditional PMBOK style management as everything starts and is conceived thanks to the master architect.

OSS development process, on the other hand, he describes as a bazaar. People come and go on their own, in case something interest them they talk to the person who brought the thing and they try to negotiate a more or less mutually beneficial deal. At the end of the day,

everyone goes home with his needs a bit more satisfied. It seems counterintuitive that this approach would yield results, not to mention working software. But if I push the metaphor bit further we get the answer. Why do people go to the bazaar? To Trade! They give something in order to get something they need more in return. So to return to the OSS, the idea is the development of software, the people are programmers who give their time, knowledge and effort and in return, they get working software (Beck and others, 2001b).

Well not exactly, there have been numerous studies looking at the motivation of contributors of OSS software. The notable study was done by Belenzon and Schankerman in 2003 looked into motivators behind contributors of the Linux kernel project. They surveyed the participants and used econometric analysis based on sociological models to analyze what are the main motivators behind the community (Belenzon & Schankerman, 2008).

Linux Kernel is the most important component behind Linux operating system, which is an open-source implementation of Bell lab's Unix operating system (The Linux Foundation, 2016). Through its many "distros" (software bundles of the Linux operating system aimed at particular use cases) it presents the largest and most successful open source project that runs on millions of computers maintaining countless mission-critical services and servers.

The research (Belenzon & Schankerman, 2008) showed that main motivators in OSS software participation (sorted by importance):

1. General identification as an OSS user and contributor.
2. Identification as a developer of a particular OSS project or sub-module of a project.
3. Pragmatic motivation (the benefit of improving the software that he/she personally uses). This, in particular, has been significant mostly for developers who plan to increase contribution to the project in the future.
4. Normative motivation (reaction of friends, family, colleges on participation in the OSS project as well as the positive impact it can have on future career prospects).
5. Socio-political motivations related to support of OSS philosophy and way of creating software.
6. Hedonistic motivators (the joy from programming in general which is often found among programmers).
7. Lack of motivation caused due to loss of time due to it being dedicated to developing OSS software (only present if the person doesn't consider himself/herself an OSS developer).

Also, it has been noted that the in the cases of developers who are predominantly working on a sub-module of the project, which is typically served by a small team of 3-5 contributors following motivators also appear as relevant:

1. Subjective evaluation of the goals of the project (sub-modules goals as well its overall importance to the main project).
2. Perceived importance of the impact of the development of the sub-module for the success of the overall OSS project.
3. Perceived personal technical capability as relevant to the technical challenges related to the development of the sub-module.

From the preceding that improving the software for personal use is not the only motivator found, and at the same time it is not the main motivator for people who contribute. But there is more to the project success than just having developers to contribute. Choosing which features to develop, which bugs to fix and distributing all that work requires some sort of organization. What happens in reality as developers are both users and creators of the software they are way more knowledge applicable for the purpose of choosing what to focus in a given moment. The community itself is the main source of feedback as the user forums of OSS projects are full of bug reports and feature requests as well as suggestions on how to proceed with the project (CNSS Committee on National Security Systems, 2004).

Most OSS projects adopt a meritocratic approach ceding authority to contributors who contribute the most. This usually creates a small circle of core contributors that take care of the project as a whole, tracking bugs and features as well as reviewing code that goes into releases. Inside Linux project, the leader is Linus Torvalds, who started the project back in the 90s. He self-dubbed his style of leading the project as being "Friendly dictator" (Torvalds, 2004). He has the supreme authority inside the project but chooses not to enforce it except in special cases, opting to let the community organically come to a conclusion. As we will see in the following section this approach is also used successfully in regular commercial corporations.

It has been noted that in OSS communities artifact based communication was directly correlated to the time till a solution was found (solving a bug, or developing a feature) (Foss, Frederiksen & Rullani, 2016). In general, basing the communication on incremental artifact based problem solving has been the surest way of ensuring efficient and effective communication inside the OSS community. Artifact-based communication, in general, is defined as the practice of basing the communication on actual objects that have certain properties that are comprehended in the same way by all members of the group (Earley & Ang, 2007).

2.4 Corporate online distributed IT Project Management in Automattic Inc.

Automattic Inc. is a US-based corporation created by members of the WordPress OSS project. WordPress is an open-source content management system (CMS) that at the moment powers around 30% of all websites on the internet while being a clear leader with over 50% market share for CMS systems. More notable the company is fully distributed, employing workers to work remotely all over the world.

The company runs WordPress.com which is a commercial offering that offers Freemium hosting for WordPress blogs and websites. Freemium refers to the practice of user segmentation trough offering core functionality for free for all users and then offering certain paid plans with additional features aimed at particular user groups (Seufert, 2014). Automattic Inc. allows anyone to create and maintain a WordPress blog for free as they strive to fulfill their organizational mission to democratize publishing. Besides this, their activity involves supporting WordPress.org, WordPress as a software platform as well as many of its plugins and themes (Berkun, 2013).

The company has around 650 employees that work remotely all over the world. They make a sharp distinction between creative and support roles. The practical outcome of this is that project management is subservient to the software creation, which is an opposite of traditional PMBOK style PM. The company is split into teams which serve an approximate purpose (data, social, servers…) but that are fully autonomous and able to choose what they want to work on. Working with a fully distributed global workforce with very little organizational overhead, not only makes Automattic is a unique company, it also makes them ideal for this research as they mix OSS style of work while still being a profitable corporation (About, n.d.).

Main input on what to develop next comes from users. Through their customer-support teams (internally called happiness engineers) they gather feedback, improvement points, and bug reports. Combining this with the user forums where millions of their users discuss they get a very good starting point for planning future improvements to the platform. Projects get started inside the teams, and the key to their success is gathering and analyzing user metrics combined with very rapid continuous deployment cycle and A/B testing facilities. This allows teams to gather nearly immediate feedback as to the actual impact of the feature they just deployed towards the goal envisioned (Berkun, 2013).

Having to work full time with a team of people requires quite some coordination and communication. Most of the company uses internal team blogs to post important updates on the current status of things the team is working on. Rest of the company is free to comment and suggest improvements on any of its numerous blogs. For more immediate communication teams mostly use IRC (Internet Relay chat) or its modern Instant messaging or hybrid counterparts (such as Hangouts, Skype chat, Facebook chat, Slack…), video conferencing tools for verbal/visual meetings (Skype, Hangouts…) as well as user forums for public announcements. Also, teams have an allowance to travel and have team meetings few times a year in order to bond and work in person (Berkun, 2013).

The company attributes its success to carefully choosing who to hire. In spite of lack of physical contact, low control over employees or centralized planning (Automattic Inc., 2005a); Automattic has been able to remain one of the top tech companies for years while having an order of magnitude fewer employees than other high-profile tech companies. Through their screening and trial process, they ensure that they only hire people who fit the company's culture, the unique challenges of working remotely and having to be self-sufficient and responsible for the overall success of the organization (Automattic Inc., 2005b).

2.5 Study of IT Project Management in Open Source Software

In this section, I will look at how few OSS projects are managed. Namely, I will look at the Linux kernel project and the way Apache Foundation manages their projects. The first example as previously mentioned uses a "friendly dictator" style of management, while the second one has a distributed management style that handles many projects without a central authority.

2.5.1 Linux Kernel project

Linux kernel project as previously mentioned develops a critical piece of software that is at the heart of every Linux computer in the world. The project was started in the 90s by Linus Torvalds with the goal of creating an open-source variant of the well regarded UNIX operating system. UNIX had a strong reputation for being reliable, efficient and well featured operating system that was quite popular in the research University settings. As the project started it became the first open-source release of a Unix-like kernel (closely followed by the development of BSD) and over time became the most popular universal multi-platform operating system, running on everything from desktop computers to android phones, embedded systems on chip, appliances and even the international space station (The Linux Foundation, 2017).

The project itself is organized very loosely. All communication is done through email. There are discussion groups from different parts of the system where contributors discuss whether a feature should be developed and how to best go about developing it. Once someone takes a feature and codes it, he sends a patch. Patch is reviewed by one or multiple reviewers (developers who contributed a lot to the project and who have rights to modify source code), and then either accepted and incorporated in the next release or in case improvement points are identified the reviewer contacts the contributor with the list of things that should be improved before the patch is accepted (Hertel, Niedner & Herrmann, 2003).

The community itself is notoriously known for featuring extremely brilliant engineers are dedicated to finding the best technical solution for a given problem. It often happens that during discussions participant completely disregard the others feelings and focus solely on the dry technical side of the problem (Torvalds, 2004). With only these simple principles and a lot of talented contributors Linux project has risen to be one of the biggest and most important OSS projects in the world, and it's still being actively developed pooling more than one thousand developers all across the globe (The Linux Foundation, n.d.).

2.5.2 Apache Software Foundation

Second OSS organization that will be covered is the Apache Software Foundation. Unlike Linux this foundation manages over 50 distinct open source projects with the best known of Apache family of products is the Apache Tomcat web server which handles around 60% of all websites on the internet (Apache Foundation, 2017a). The foundation started in 1999 in order to create a single entity that would protect contributors to legal issues related to contributing to OSS projects.

It started with an HTTPD web server and over time grew to encompass many well-regarded OSS projects. The foundation differentiates the umbrella organization from the projects of the organization(Apache Foundation, 2017a). The organization is composed of Board of Directors, various officers that take up specific roles (legal, accounting, server maintenance…) and Project Management Committees (hereafter: PMC).

The Board of Directors governs the organization; sets policies and trough resolutions can appoint officers, start or end projects as well as handle the organization as the whole, the Apache brand and ensure that organization's mission and vision are followed. The officers

that are appointed by the board are responsible for handling various areas of the organization as specified by the job description of their position (Apache Foundation, 2016). PMCs are created in order to manage one or more organizations' projects and are comprised of all committees, as defined by ASF a contributor is at the same time a committer) (Apache Foundation, 2017b).

While this appears to be quite close how a traditional organization might operate, there is an important distinction: Once created the projects are completely under control of the PMC responsible for them. They have the sole authority on how the project is conducted, what are its goals and how will the outcomes be disseminated. Project contributors themselves have full authority on how the project proceeds (Apache Foundation, 2017b).

Inside the organization, the main rule of conduct is a meritocracy. More you contribute, greater your authority. As people contribute more they get more and more control. The first level is getting rights to directly push to version control, followed by a right to push directly to the source branch and so on. This philosophy of work allows for people who put the most effort and greatest results to have a status that reflects their dedication (Apache Foundation, 2016). When it comes to decision making all are equal and the community as a group votes by giving a:

- +1 - a positive vote
- 0 - no opinion
- -1 negative vote, for which its mandatory to follow with an explanation or counter-proposal

Proposals with positive votes and no unresolved negative votes are accepted. This way of voting ensures that community as a whole discusses changes looks for improvements and reaches a consensus. Instead of a traditional decision making oriented democratic process, this one is designed to foster conversation even inside the act of making the decision (Apache Foundation, 2006).

In the preceding chapters I took a look at how different organizations approach organizing programmers and trough it I answered our second research question: "What examples of successful implementations of IT project management of distributed online teams from both IT industry and open source projects?" Now I will take a look at the student organization BEST and then at its IT Department.

3 IT DEPARTMENT OF STUDENT ORGANIZATION BEST

Most of the data in the following chapter is derived from official documents as well as the PR materials of the student organization BEST. Besides this the autor being a member of the organization had ample chance to personally expirance and work in the organizations working methods and organizational structure.

3.1 What is BEST?

BEST is a constantly growing non-profit and non-political organization. Since 1989 BEST provides communication, cooperation and exchange possibilities for students all over

Europe. Its 96 Local BEST Groups (LBGs) in 33 countries are creating a growing, well organized, powerful, young and innovative student network (Ivancan, 2015).

BEST strives to help European students of technology to become more internationally minded, by reaching a better understanding of European cultures and developing capacities to work on an international basis. Therefore BEST creates opportunities for the students to meet and learn from one another through our academic and non-academic events and educational symposia. "Learning makes the master", but the final goal is a good career opportunity for students, therefore BEST offers services like an international career center to broaden the horizons for the choice on the job market (Board of Europian Students of Technology, 2016a).

Main activities of the organization are international complementary academic courses that provide participants with ETCS credits upon successful completion. Most of the work including the course organization is done by Local BEST Groups, which are mostly autonomous units maintained and developed by students of the local university on which the LBG is formed.

BEST beside local members also has a number of international teams which provide higher level services to the rest of organization. At the moment BEST has teams focusing on marketing, fundraising, grants, training of members, evaluating the quality of BEST Academic courses, improving European technical education and IT. In the logo of the organization shown in Figure 1, we can see the embodiment of three main stakeholders of the organization: Students (yellow), universities (green) and companies (blue) (Board of Europian Students of Technology, 2016a).

Besides being a mother organization to its 96 LBGs, BEST also has an international side

Figure 1: The logo of student organization BEST.

Source: www.best.eu.org

that is responsible for maintaining and developing both the organization and its services. International BEST is organized in a matrix structure with 10 departments acting as knowledge holders and focusing mostly on operational work. Normally department members lead or work on projects as most experienced members on given topic. All development in the organization is done through projects.

Projects unlike departments are time bound and are serving both as a starting point for BESTies interested in contributing to international BEST. Normal path of an

internationally involved member is to join a project and when he has enough experience he joins a department. Projects are separated into three categories (Board of Europian Students of Technology, 2014):

- **Strategic** - resulting from BEST's Long-Term Strategic Plan that dictates the organization's development for a 3 year period,
- **Department** - owned and closely tied to a given department,
- **Ad-Hoc** - Initiatives by different bodies that develop the organization.

Currently, BEST has 10 Departments (Board of Europian Students of Technology, 2016a):

- **Competitions Department** supports and develops Competitions inside BEST.
- **Corporate Relations Department**: develops partnerships with companies bringing financial resources to the organization, as well as career support is provided to students.
- **Design Department** provides design materials for the needs of the organization and also ensures that every BEST group is following the rules stipulated in the Visual Identity document of the organization.
- **Educational Involvement Department**: works on improving engineering education in Europe, they are responsible for the educational involvement service through our involvement in international organizations for educational matters.
- **Grants Department**: works on applying for and managing grants.
- **Information Technology Department**: listens to the IT needs of the organization, develops concepts of IT applications, implements and provides technical resources and services, in order to support the whole organization in achieving its goals and visions.
- **Membership Department** provides support to Observer and weak Local BEST groups needed for them remain sustainable members of the organization as well as providing support for external groups of students interested in becoming members of BEST.
- **Public Relations Department**: monitors, builds and maintains the external image of BEST by building strong relations with media as well as managing social media channels of the organization.
- **Training Department** aims that BEST members grow and contribute to the organization in an effective way, enabling it to reach its goals, by maintaining and developing the training system of BEST.
- **Vivaldi Department** supervises and supports the organization of the BEST Events defined in the Vivaldi Handbook. Namely, organization's Academic Courses organized by LBGs.

Besides the departments, another identifiable body inside the organization is the management of the organization. It consists of the international board of the organization and international HR team. International HR team at present consists of 10 department coordinators and 2-3 members of the organization appointed by the board. Together with the international board, it fulfills roles that are normally associated with top management of a company.

International board of the organization consists of following positions:

- **President** – leads the organization and is responsible for representing the organization

towards the external world and other organizations.
- **Secretary** – takes care of knowledge management inside the organization and is responsible for coordinating organization's two annual statutory meetings.
- **Treasurer** – responsible for overseeing and managing finances and expenses for the organization.
- **VP for Human Resources** – oversees human resources for the international level of the organization.
- **VP for Services** - oversees and develops services the organization provides to external students and other stakeholders. They include complimentary education, career support, and educational involvement as well as actively taking part in creating or developing new services.
- **VP for Projects** – oversees projects on the international level, also responsible for coordinating project meetings as well as the International Project Forums.
- **VP for Local Group support** – provides support to Local BEST Groups of the organization.

3.2 History of the IT Department

Information Technology Department's (hereafter: ITdept) place in the organization is stated in its statement of purpose:

> "The Information Technology Department is a body of BEST that listens to the Information Technology (IT) needs of the organization, develop concepts of IT applications, implements the systems and provides technical resources and services, in order to support the whole organization in achieving its goals and visions." (Board of Europian Students of Technology, 2015)

In short terms, the department listens to IT needs of other members, provides interaction design knowledge in order to shape said needs and them programming and administration to make and maintain developed applications.

Current IT services provided to the whole organization are:

- BEST Website,
- BEST Private Area (later in text PA),
- @best.eu.org emails,
- Google Analytics,
- Single sign and authentication service for 3rd party apps over LDAP communication standard (Lightweight directory access protocol).

Besides services provided to the whole BEST the department also has specialized development tools that it uses and develops:

- GitLab git code repository software for version control, programming collaboration as well as continuous integration.
- Makumba open source Java web platform that is the backbone of all development effort in the organization.
- VPN services for secure interfacing between development databases, production

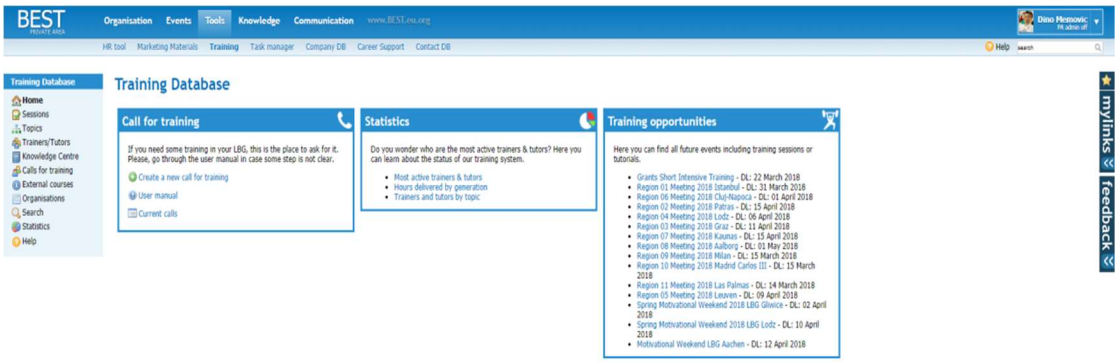
environments, and local development platforms.

- Local Development Platform – provides development resources and simulates the production environment on developer's personal laptops.
- Online Development Platform – which to some degree allows programmers to work through the internet as everything is done inside the web browser.

IT in BEST started in 1994 when the organization realized the need to have a body that would take care of IT needs of the organization. It was formed under the name of ITC, or Information Technology Committee. The first thing it provided was a web contact list of all the members of the organization. Soon after first mail servers was created facilitating the early organization-wide adoption of emails in the year 1995. Soon ITC started implementing all its technologies using a platform called Lotus Notes. It provided quite a robust feature set when it came to algorithms and database interaction but it was later replaced by writing the code in Java/MySQL due to Lotus Note's limited ability to support rich user interfaces. At this point from the original codebases of tools a web platform called Makumba was extracted and from there on further developed as open source project.

By the year 2002, all tools were rewritten in Makumba and unified under the new Private Area. From there on the ITC kept developing and slowly first into the three teams; namely IT Administrators (hereafter: ITA), IT Developers (hereafter: ITD) and IT Interaction Designers (hereafter: ITID) and later on started organizing work in projects. Around 2003 ITC started developing its Interaction design workflow and by 2006 it formalized the workflow and made it mandatory for every project. In Figure 2, you can see the front page of the Training Database inside Private Area, which focuses on handling training resources (types of training, session, and material creation) as well as trainers and training events inside the organization.

Figure 2: Example Training Database tool in Private Area,



Source: BEST, (2018).

Till 2012 the ITC focused on building new tools and then first encountered the risk posed by accumulating technical debt and legacy code. Then it started shifting gears and focusing

more on maintaining and extending current tools in order to make them more stable and modernize their interaction design as well as aesthetics.

In late 2015 due to organizations restructuring the ITC was dissolved and reformed as IT Department. This coincided with the ITC's restructuring of internal working methods in order for them to become more scrum like. And it leads the ITdept to the present they organizational structure.

3.3 Organization of the IT Department

Department is organized into 3 distinct teams: Developers, Interaction Designers, and Administrators. Each three provides a specific set of skills and has its place in the application development lifecycle of the organization.

Currently, most of the work is done online, as members are dispersed all around Europe. Periodically also department has Developer meetings. Developer Meetings are short 3-4 day highly productive live meetings of around 10 programmers. Usually, they focus on finalizing a specific project or developing a tool. Besides these events, ITdept sends representatives to all bigger meetings of BEST in order to provide IT support as well as for members to collaborate with other bodies of the organization on various projects that need input or resources from the ITdept.

Most important of these events are International Projects Forums, which gather around 70 members of the international part of the organization and focus on starting and finalizing various international projects of the organization. Normally during the International Projects Forums, ITdept members have their own working room so that they can code and are also free to join any parallel sessions happening during the event. This is in contrast to the role of other members on IPFs as they come primarily to join sessions that are mostly formatted as workshops and discussion groups, while ITdept members can join them or choose to work in the IT room. IPFs are traditionally followed by a Developer Meeting with the purpose of kick-starting development of tools and improvements discussed on International Projects forum.

Organization structure at present moment reflects the overall matrix organizational structure at BEST. Members of ITdept rarely conduct work directly inside the department, mainly they are expected to join ether departmental or other projects and through them contribute to the development of the organization. At present, the department takes care of its projects and acts as a repository of knowledge stored in its members.

3.3.1 Services

Department has 3 core services it provides to the organization:

- BEST Website
- Private Area
- @best.eu.org email services

3.3.1.1 BEST website

The website was redesigned and recorded in 2016 and currently, it provides information about the organization as well as an application system for events open to students that are not members of a BEST group. The website is aimed mostly at our main stakeholder: European Students of Technology, but it also provides online aspect for our other services such as career support and educational involvement. For career support, it displays job fairs and similar events that our LBGs organize as well as career opportunities for students from our career support partners which are primarily international corporations.

3.3.1.2 BEST Private Area

The intranet of the organization includes a large number of tools developed for different bodies of BEST over the years. At the moment it houses over 50 tools and has over 500 thousand lines of code developed in last 18 years. Most of the projects related to IT at BEST are aimed at creating new or improving current tools. Examples of such tools include Best Application System that manages events, applications, participation and evaluation of individuals in various events organized by BEST and its local groups; Virtual International Plenary: which is the voting system used for voting both on statutory General Meetings and online.

Both website and Private Area are hosted on a cluster of rented servers to ensure constant availability. As they can have a peak of over 10 000 users per second, they are situated on powerful rented servers.

3.3.1.3 Email Service

ITdept also provides some 12 000 students and alumni members with @best.eu.org email addresses. They are designed to integrate well with Gmail and allow students to send and receive emails as well as send emails from their @best.eu.org accounts directly from their Gmail interfaces. Besides personal emails through Private Area ITdept also offers mailing lists, which allow management, access control as well as email archives for different teams and groups inside the organization, at the moment there are over 1700 mailing lists used by the organization. Another service that the department provides to partners is sending company newsletters which are sent to the email database of over 40 000 emails of students that stated that they are interested in receiving such offers. In total best mail servers handle between 100 and 300 thousand emails per day.

3.3.2 Teams in IT Department

As I already stated there are 3 distinct teams inside the department organized by their expertise. Although a member can be involved in multiple teams (eg. developers feedback the design of new tools), normally they focus on one area at the time, potentially switching to different ones periodically in order to pursue particular interests in different technologies they have at the given time.

3.3.2.1 IT Administrators

The smallest team that consists of few highly experienced Linux system administrators. They are the experts of their domain normally involved for more than 5 years inside the department and typically working professionally outside BEST as either system administrators or senior developers.

They maintain our Linux server infrastructure which consists of both rented servers (for mission-critical services such as PA and BEST Website) and around a dozen servers hosted inside university data centers all over Europe. For the university hosted servers ITdept has to provide physical maintenance meaning that one of the administrators or another member who has experience dealing with server hardware has to travel to the location and replace or fix the server hardware in case of hardware failures. This is the reason why mission-critical services are deployed on servers rented from a German hosting provider. Actual services infrastructure is virtualized over a number of virtual machines managed using a technology called Linux containers (LXC) that runs said virtual machines on organization's servers in order to ensure faster deployment and better overview and encapsulation of services.

Although relatively small, the team is invaluable and as system administration is very heavily knowledge-intensive, it is, fortunately, the work is not as labor intensive as other IT areas of work (administrators only have to work when some service goes down, otherwise it's up to them to further develop the infrastructure as they see fit).

3.3.2.2 IT Interaction Designers

The main task of interaction designers is to work with users of a tool in order to determine their needs and based the needs develop specification which department's developers can use in order to program said applications. Interaction design process is formalized and generally follows these steps:

- Idea,
- Users and needs,
- Use cases,
- Data requirements,
- Mock-ups,
- Feedback.

STEP 1 – Idea The first step is to think about what needs to be developed and to formulate a clear idea what should be the desired functionality of the end tool.

STEP 2 – Users & needs We start by identifying end users, or key stakeholders for the tool. Then Interaction Designers together with the identified stakeholders work on coming up with a detailed list of needs that the end-tool is supposed to fulfill (Cooper, 2004).

STEP 3 - Scenarios (use cases) Scenarios represent use cases that the end tool should be able to fulfill, and are generally written down stating explicit steps that user should follow and that system should respond to (Wachs & Saurin, 2018). A scenario is a structured

description of how the user uses/interacts with the system and how the system should respond to each individual interaction between the user and the computer.

STEP 4 - Data requirements Based on the previous steps a list of data fields is compiled. This step is very similar to traditional Database Modeling. Database modeling is the process of applying formal techniques in order to model the data and database for the needs and use cases of an information system (Teorey, 2011).

STEP 5 - Mock-ups Based on the specific definitions you make the mock-ups. Giving visual appearance to specifications helps to make it more understandable for non-IT people. The mockup is a visual representation of a said interface that is used to probe usability and get feedback on the needs of the user of the tool (Halbrügge, 2018)

STEP 6 – Feedbacking Feedback in the interaction design process is the practice of providing return information on work done by interaction designers (Oduor, Oinas-Kukkonen & Alahäivälä, 2017). Oftentimes they were not able to completely define everything in previous steps, and in this step both propose new ideas and clarify the existing ones. This allows for quick changes in the end design before it reaches the developers.

STEP 7 – Translating into tasks During this step department's core team together with its senior developers splits every task into sizable chunks that can be posted on the task list. The result of this final step in the design process is a list of tasks that are ready for developers to take and start developing. With these steps, the interaction design is pretty much finished with the exception of testing the developed application together with stakeholders and providing feedback to possible improvement points. Following the ID process, the development is handed to ITD or the developers.

3.3.3 IT Developers

IT Developers is the team most numerous team inside the department. They are organized around programming projects which have a project leader and are generally organized using a scrum methodology adapted for the organization. At the moment the department has three big projects as well as a number of smaller projects handled by few people independently. The major projects are:

- **Private Area / Public Website maintenance** - is the project that handles small tasks and bug fixes required to maintain organizations public website at www.best.eu.org and its already mentioned intranet Private Area.
- **Bootstrapping Private Area** – has the goal of modernizing the design of organizations' intranet as well as making it responsive and more mobile friendly. This is done by implementing features found in Bootstrap a well-regarded platform for providing responsive modern UIs to websites (Otto and Thornton, 2017).
- **BEST Application System modulization** – has the goal of refactoring one of the biggest tools in Private Area, the BEST application system in order to decouple some of its dependencies and make it more flexible so that later on ITdept can modify it in order to match organizational changes that BEST can go through in the future.

Project leaders are empowered to decide how their team will work internally and are responsible for the development of its members as well as the integration of new members. Projects are managed using 3-6 weak scrum sprints.

All the code developed by programmers inside ITdept must first be reviewed and tested by senior developers who are part of the Merger Team. Only after this phase of extensive testing both locally on developer's personal computers and on the test server can code be deployed on the staging server where it goes through the final testing with the production DB before being put to live servers for users to use.

3.3.4 Human Resources

As BEST is a student volunteer organization, the department has to recruit relatively inexperienced members and internally help them grow into competent programmers. This is a time-consuming process and it usually takes about a year till a new recruit becomes skilled enough to work on non-trivial problems on their own.

Due to this long process very few aspirants actually join the department, which correlates closely to the percentage of IT students who find out that they actually don't want or are not able to spend rest of their lives just programming due to their low skill level at the end of formal education (Aspiring Minds Inc., 2017).

Compared to the rest of the organization ITdept has a very low member turnover, with members staying active for around 3-6 years (compared to retention of an average BEST volunteer which is 2 years). This makes the low retention rate of new recruits as well as the long training period required for new members to be productive worthwhile, as each member ends up being a long-term asset of the department, especially because even when they stop actively contributing they still stay informed about current work and act as a knowledge base for rest of the members of the department.

The first thing a new recruit does is set up local development platform, this step is necessary as it allows the programmer to run and test his code locally and also teaches him basics of the infrastructure on which the whole code-base is built. After setting up local development platform the newbie joins the project of his choice, and the project leader keeps track and assists the new member during his personal development. The most common destination is Private Area / Public Website (PA/PWS) maintenance as this project has an abundance of small, low priority tasks which are perfect for developing new members who didn't have many chances of working in a professional programming environment.

3.3.5 Decision-making process

When we discuss decision making in the context of IT most often we refer to the concept of IT Governance. Which stipulates the way decision making, as well as actions and projects inside the department (or other organizational body tasked to take care of IT), is aligned to the needs of the organization (Sangle, 2015).

In general, most decision making related to IT at BEST is done through IT Monarchy. IT monarchy is the practice of ceding all IT-related decisions to the IT specialist residing

inside the body of the organization dedicated to IT (Weili & Ross, 2005). This is mainly due to the fact that ITdept has considerably more knowledge and experience in IT related matters than rest of the organization. This is mostly due to the fact that rest of the organization as little to no IT knowledge.

Although this way ensures quite efficient and lean ITdept it can present a picture of the department as being quite inflexible and hard to deal with especially by non-programmers who due to their lack of understanding of the underlying technical specificities of IT and especially programming workflows find it hard to understand and even harder to accept limitations and barriers set by the programmers inside the ITdept.

By standard, all relevant stakeholders are involved in software interaction design phases, and only the things they really need are actually built. Rest of the process is solely in the IT hands, except for finances. It purchases have to be authorized by Treasurer of BEST and all events (developer meetings) and travel reimbursements is handled either by Treasurer or VP for Projects. Although these are normally done on the initiative of Senior Department members, as they have the knowledge required to anticipate when hardware needs to be changed.

3.3.6 Leadership structure

Department is headed by the IT department coordinator who is elected for a 1 year period. Duties of the coordinator are following:

As defined in ITdept internal regulations (Board of Europian Students of Technology, 2015):

- Coordinating the development of the systems through the ITdept Projects .
- Coordinating the available resources over the working areas of the department.
- Represent and communicate the work of the department to the rest of BEST.

Also supplemented by Department Regulations (Board of Europian Students of Technology, 2016a):

- Coordinating the work of the department.
- Ensuring that BEST is informed about the work of the department.
- Ensuring good communication between the members of the department.
- Ensuring continuity of the department.
- Planning and foreseeing the organizational and financial resources needed by the Department to work and communicating with the Treasurer about them.

To fulfill his task the coordinator position is supplemented by the position called buddy which takes roles similar to that of a secretary. Buddy is responsible for proper reporting and knowledge management inside the department. Among his roles is to substitute the coordinator when the coordinator is unable to fulfill his duties (as happens often due to events without internet and constant traveling).

Together the coordinator and the buddy constitute the IT Core team of the department. And by tradition members of previous few cores teams are also included in core team and are

still active and provide input, advice, and guidance for current IT Core, occasionally also contributing to the work. The coordinator is also a member of Management of BEST consisting of International Board and all department coordinators. This helps the coordinator keep an overview of the strategic level of the organization as well as help foster better communication between different bodies of BEST.

3.4 Communication

The official channel of communication inside BEST is email. The ITdept has a number of specialized mailing lists dedicated to subsections of work. Examples include MLs for developers, projects, administrators etc…

Department follows BEST standards for formatting emails which dictates using tag keywords in the subject line to distinguish the email (eg. all ITdept mail subject lines start with the [IT] tag) as well as putting dates, links and important information on the beginning of each email when needed. Below you can see an example beguiling of an email calling members to apply for an upcoming event:

Email Subject Line: *[IT][DM] Apply for the upcoming Developer Meeting in Helsinki | DL: 28 Dec 2016*

Start of email Body:

#Application DL: 28 Dec 2016

#Where: Helsinki, Finland

#What: Developer meeting

#Dates: 15-18 March 2015

Source: IT-department@best.eu.org email

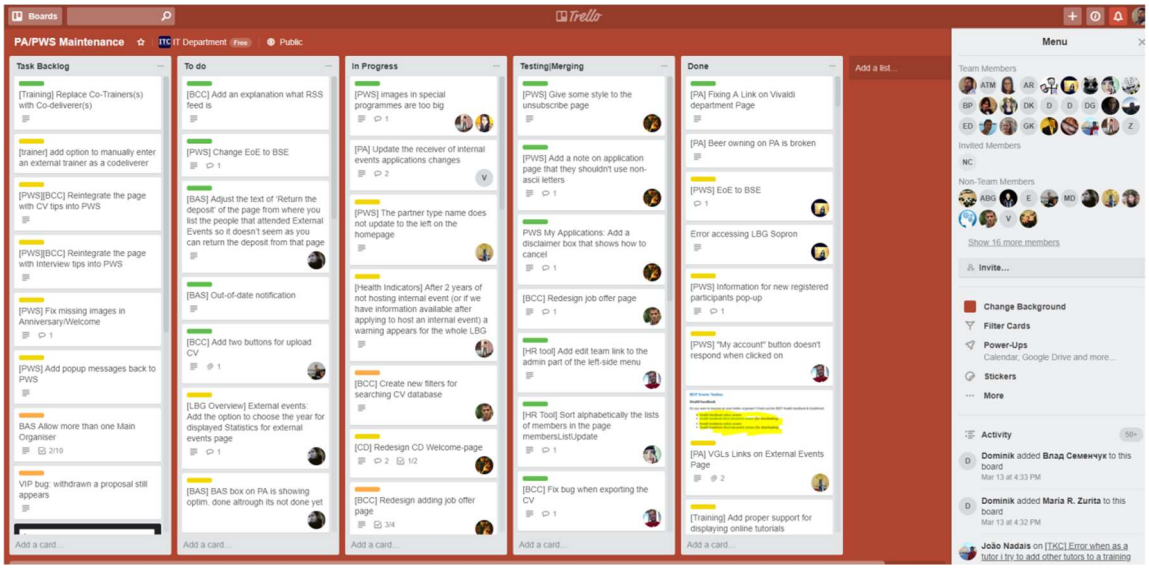
Besides emails from day to day communication ITdept members also use Slack. Slack is an online chat platform that works similar to Internet Relay Chat but also provides both advanced team chat options and variety of very deep integrations with other 3rd party collaboration platforms (such as sending email notifications and pulling data from various sources) (Slack Inc., 2017). As some work is most efficiently done face to face, the department conducts numerous online meetings. Videoconferencing inside the department is normally done using Google Hangouts video chatting tool.

For task management inside the department, ITdept uses Trello. Trello is a free online kanban board, that provides numerous features oriented towards team productivity (Trello Inc., 2016). This is mostly because of the way scrum in general works really well with kanban boards. Kanban board is a visualization technique in which each task has its "post-it notes" which is moved through different stages of the project as it gets more complete. It presents a highly efficient way to track tasks, people, issues and overall project progress (LeanKit, 2018).

The task list of ITdept has a kanban board for each project and all tasks are sorted between boxes: To-Do (priority tasks that are free to be taken by a contributor), Backlog (remaining tasks that are not taken by anyone), In Progress (tasks currently being worked on by

someone), Testing / Merging (tasks whose work is deemed finished by the contributor and currently under review by a senior member) and Done (finished tasks). Below in figure 3, you can find the Trello board of one of the department's projects.

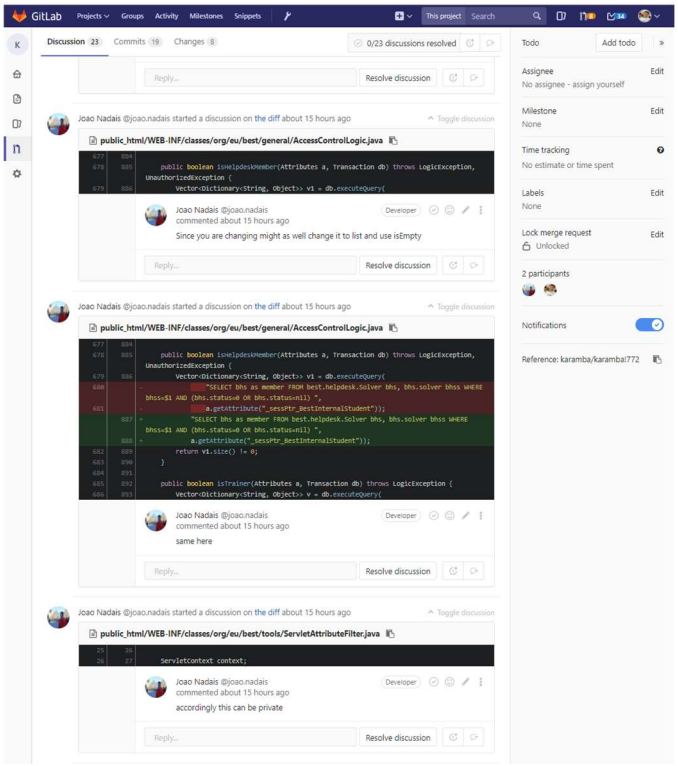
Figure 3: IT Department Tasklist



Source: Trello Inc. (2018).

All code reviews and individual developer contributions are managed using GitLab which is the most popular git repository management web-platform that allows easy tracking of historic development effort as well as keeping all individual features separated and highlighted for easy review and dissemination (GitLab Inc., 2015). An example of such review process you can find in figure 4 below.

Figure 4: GitLab code review interface



Source: GitLab Inc. (2018).

4 IT PROJECT MANAGEMENT IN IT DEPARTMENT OF BEST

The preceding chapter answered the question of: "How is the IT department of student organization BEST organized?". And now I will take a look at the project management inside the ITdept, I will start by first taking a look at the historical development of working methods inside the department from its early days in the late 90s and slowly go to the present day scum based project management methodology.

Data collection is done using insider researcher method with autor being member of the ITDept. Over time work in different areas from developer, systemadministrator to leadership roles starting with being HR manager to being head of the department. Personal observation the transition form classical to agile methodology that consists of the bulk of the following chapter is supplemented with review of internal wiki, official documents and email archives documuneting the transition.

4.1 Previous PMBOK based approach IT Committee

Over the years the way whole organization worked changed quite significantly, and that change also reflected in the way IT was organized inside the organization. At the beginning of the millennium, the whole organization formalized its working methods in a departmental system consisting of committees that are responsible for its core areas of work (courses, educational involvement, marketing, IT, corporate relations, and training).

Committees were the continuous bodies of the organization, thus ensuring proper knowledge management and long-term continuation of operational work.

Another organizational body that existed at the time were the working groups. They were less formalized and were formed to address a specific short-term or medium-term goal after which they are disbanded, or if they needed to operate in the long term they were restructured into committees. Notable examples of this happening are the development of soft-skill training inside the organization, educational involvement and engineering competitions that started as working groups developing an idea and progressed to full-blown committees or were incorporated as part of the daily work of a committee.

The separation of work into separate organizational units meant that work was performed more efficiently inside the individual units, but an organization grew and more than doubled in size it displayed common pitfalls of the purely departmental organization. Namely, the lack of communication between committees as well as the difficulty of organizing projects that required cross-departmental work.

ITC was among the bodies most affected by this phenomenon due to the inherent nature of its work. Getting information such as requirements & feedback on designs and finished tools required input from the bodies of BEST that needed the particular tool or feature being developed. And this communication barrier greatly slowed down the process of getting said information.

Individual modules, better described as self-contained groups of features aimed at a particular need or use case are referred to inside the organization as tools, and as such, they will be referred to as tools in this thesis. Each tool is developed by a project that would take the tool through all the steps of the design and development process till it was ready to be released to the end users.

It was common that for the interaction design of a tool, which is the process of building the whole specification required by programmers to can start programming, said a year to take more than a year. While the programming part that followed usually took few months and was completed by a few programmers, it was again followed by another relatively long testing & validation that could take few more months to half a year, assuming no rework was needed, before the actual tool could be put in production.

What gets developed for a particular year was and still is based on the Long Term Strategic Plan (hereafter: LTSP) of the organization, the annual action plan of the organization as well as the annual action plan of the ITC. The LTSP is a document that guides the development of the organization for a period of 3 years.

LTSP is created through LTSP workshops that strive to include every area of work inside the organization as well as a menu of organization's members. The annual plan of the organization is created by the international board of the organization in conjunction with the international management of the organization (in previous structure management consisted of committee coordinators). The annual plan of the organization stipulates all the areas on which organization will work in the given year.

Previous two plans provided the prioritization for the work ITC had to do in a given year. The internal action plan of ITC was a reflection of said prioritization as well as how

complex and time consuming the work that needed to be done was. Also, the ITC action plan contained internal projects of the committee which ranged from new features to support development process or the IT infrastructure to partial or complete reworks of codebase belonging to particular tools in order to mitigate the constantly accumulating technical debt or make that specific tool more extendable or flexible in anticipation of future work.

Following the internal regulations of the organization, the ITC has to have its annual action plan approved by the organization's members (LBGs) and its annual action plan report accepted at the statutory meetings of the organization. This provides space for the organization to give critical feedback to the work of the ITC (Board of Europian Students of Technology, 2016c).

Early in the development process was partitioned into PMBOK style partitions:

- Idea,
- Interaction Design,
- Programming,
- Code Review, Testing, and feedback,
- Potential rework due to feedback,
- Deployment.

Idea - For the development process to start the initial idea had to be properly formalized and approved by relevant bodies. This was namely getting either the coordinator of a relevant body requesting/being impacted by the tool or the international board of the organization as well as the ITC coordinator to approve of the idea. After that, relevant stakeholders were identified and their representatives allocated to the project as well as project leader and interaction designers for the upcoming stage.

Interaction Design - this stage is handled by the committees' interaction designers, who are responsible for creating detailed requirements necessary for programmers to do their job. The outcome of the Interaction Design stage was a formal interaction design document that contained following:

- who are the users?
- which use cases are defined?
- what are the underlying assumptions for the tool?
- Detailed mockups of pages that the tool will implement.

At this stage, most of the communication was done over email by sharing documents and mockups created using "Balsamiq mockups" mockup tool. Periodically the interaction designers would meet with the stakeholders either in person or in an online meeting in order to gather feedback on current work as well as acquire additional information required to finalize the interaction design document required by developers.

Programming - after detailed interaction design the specification gets translated into tasks that are then put into the main tasklist of the committee. Depending on the crucially and complexity of the task and overall tool (how high is its priority in the long-term strategic plan of the organization, or the annual plan of the committee or the organization) it is

either allocated a specific number of programmers or just left as tasks in the task list to be picked up as by individual developers when they have time for them.

The task list was an online spreadsheet that kept track of all the tasks, while the developers worked using a custom in-house online development tool called a parade. Parade provided an online text editor as well as the ability to test the code that the developer just wrote on a near identical copy of the production environment. Version control was managed through CVS (Concurrent Versions System) , which allowed tracking of all work and changes one on a set of documents (GNU, 2015). Another feature of the parade was the semi-automated system for propagating changed and new files to the production server which was crucial for getting features to users.

Code Review, testing, and feedback - This stage starts by having members of ITC test the new code and ensure it performs up to the specification as well as check the code quality and look deep into it in order to expose any hidden bugs. At this stage, actual users of the tool are invited to test and feedback the tool. Normally a separate publicly available test server was used in cases when a greater number of people were required for testing a tool or feature before it could get approved for a production environment.

Potential rework due to feedback - It often happens that the finished tool while fully following the specification no longer fulfills the needs of the particular body it was developed for. This happened mostly do the very long interaction design process required to generate the specification as well as the changes to the internal working methods of the body that the tool is developed for, as most of the organization strives to continuously innovate and improve the way they do things. In the end, it was often the case that a partial or complete rework of a part of the tool had to happen, which would delay the project by another few months and sometimes more than half a year.

Deployment - The final stage was getting the tool deployed to production servers and providing assistance and training to users so that they can fully incorporate the tool into their daily workflow. Actual deployment is preceded by final code review by senior developers, and was internally referred to as “Putting the code in production”.

Over time some problems in this workflow were noticed. The biggest identified problem was the long interaction design process, which would in some cases span for many years. It was common that by the time interaction design process for a tool was finished; the tool itself became obsolete or required significant rework to be usable by its intended users. Another big issue identified was projected ownership. Every committee of the organization would defend its own turf. This had the practical effect of bodies being unwilling to dedicate resources required to properly finalize interaction design or testing of a tool, as they believed they could be better utilized in other places (ironically in many cases this mint dedicating people to doing manually what the tool intended to automatize). Information required for interaction designers to start working would get delayed by months, and later the feedback on completed interaction design would be similarly get delayed, and this would keep spinning generating a negative feedback loop that could prolong the project by years. The months of delay would mean that the interaction designers couldn’t work continuously on the project as they can’t make progress without information, meaning that they would have to re-learn all tacit knowledge related to the

project every few months which would further delay the tool and make the end design outcome less uniform and less user-friendly.

Another issue was the clear lack of deadlines, or better said lack of having a culture for following deadlines. This issue came mostly from the international aspect of the organization. As members came from all over Europe, a good part of them necessarily came from a culture that has a less strict approach to deadlines. Combined with the relatively short member lifespan of 2 years for most of the organization this meant that between high probability of having a person that doesn't care about deadlines and that you had to constantly put effort into making new people up to date with the development. A notable exception to this is ITC which had an average member lifespan of 6 years ensured continuity of IT services.

One of the more extreme examples of the negative feedback loop was the redesign of the public website project. It took 4 years for its interaction design to finalize. In its first year, the discussions and conceptual planning got so delayed that actually, work didn't start till the next year whose proposed interaction design received such a bad feedback that they decided to do a complete rework next year. In its third year, the project coordinator disappeared in the middle of the process leaving the project in a limbo. The redesign for its first three years was handled as a separate project, not owned by a body of the organization. In its final fourth year, the project was taken by ITC, and it was finalized in less than a year by having a small team design, code, and release for testing the code in increments before releasing it to users.

Besides the issues with the interaction design, there were numerous problems on the technical side of the workflow. The online development tool, called parade, was quite old and slowly grew more unstable. Outrages would stop work for a day or two, which was huge given the fact that developers were volunteers who are not able to dedicate time every day, meaning that if the outrage coincided with the time they dedicated for that week ITC would lose quite some work. Also, the nature of CVS, which was used for version control made it hard to overview changes to many files which meant that it was hardly possible to overview the whole tool when it was ready to be put in production. Another problem was transparency which was quite low as it was not that easy to ascertain who was working on what and who was free to take new tasks at any moment.

At this point, the leadership of the organization was slowly finalizing the planned complete restructuring of the organization to a matrix project based organizational structure. This change coincided with the significant changes to the technological side of the development workflow inside ITC.

4.2 Shifting towards Agile-based approach in IT Department

The restructuring of the organization brought about the dissolution of old bodies of the organization. Thus the ITC was reformed into the IT Department (hereafter: ITdept). ITdept modified the workflow of ITC in order to accommodate the organization-wide shift to project-based workflow. Besides adapting to the organizational shift, the department used the climate of change to proceed with implementing a more agile workflow towards which it the committee started shifting some years earlier.

The organizational change didn't impact the internal workflow that much, as the nature of ITC work very much implied project-based organization and workflow. Instead, the actual effort went into making the work of project teams more scrum-like. While fully implementing scrum couldn't happen inside the department due to scrum's focus on live interaction with the scrum team which was a no-go for a geographically distributed department and teams. The newly created ITdept focused on implementing some important concepts and techniques that were found to be crucial for a more productive workflow.

Most important one was a shift to scrum sprint based workflow. Having shorter loops of design - code - test helped keep everyone on track and incited a culture of deadlines for all participants. It allowed for a mentality of: "If you provide this input by this deadline we can include it in the next sprint, if not it will go in one of the following sprints". And this semi-artificial deadline imposing really helped to ensure that everyone does their part in a timely fashion because they could cognitively process what their delay would do to the overall project in the short term, as compared to them having few days delay in a project that would span years before delivering its first tacit outcome.

Another important methodology adapted was the kanban board, which started being used instead of a spreadsheet-based tasklist. This allowed for greater transparency in the work of the department, as everyone could open the task list and see the current status of every task and member. Also, every change of the kanban board was automatically posted on a dedicated channel in slack so that members could see what was happening without details or effort that was required for opening the actual tool. The kanban board was partitioned into following columns; the numbers in brackets represent a number of tasks for the PA/PWS Maintenance project in December of 2017, which is currently the biggest project in ITdept:

- Task Backlog: containing all non-priority tasks not taken by anyone (typically around 50 tasks).
- To Do: containing tasks with the highest priority that should be tackled in a current or following sprint (typically less than 10 tasks).
- In Progress: tasks currently being worked on by one or more people (10-20 tasks).
- Merging/Testing: tasks currently being tested and reviewed by senior developers (5-10 tasks).
- Done: All finished tasks.

All tasks in the kanban board are color-coded based on their difficulty and complexity. Starting from green for simplest tasks and going all the way to red for more obscure difficult tasks that may require more than a week to finish.

Although individual project can have separate kanban boards for their project, many opt out in to stay in the department's main kanban board (PA/PWS maintenance board) as this makes their work more transparent, easier to follow and allows other members of the department to pick up tasks from the project if they have time and which to do so.

The project leader takes many responsibilities commonly associated with the scrum master, while one stakeholder representative from the body that the project is being developed or a member of the international board of the organization takes the role of the product owner. Day to day communication inside the team typically takes place through email or slack,

although teams are free to choose their preferred communication methods as long as they periodically inform ITdept on their progress through email.

Another important difference is that currently, projects use a variable-length sprint approach. This means that the length of a single sprint is decided at the beginning of the sprint or even adjusted during the sprint. This method is used in order to accommodate the more chaotic availability of team members. This is mainly due to them being volunteers, and often students who tend to be unavailable around exam season and similar university imposed periods of deadlines that they cannot predict in advance. Also as members of the organization tend to travel a lot, and normally attend 5-10 events (each lasting 4-10 days) all over Europe each year. Although these events are a great opportunity to develop ideas and work on interaction design of tools, these constant travels make a standstill of actual programming work.

An important exception to this is already mentioned developer meetings, which due to their nature and unique environment usually push developers to extreme productivity. Although it is quite uncommon to have a whole project team on a single Developer Meeting, normally most teams do a special sprint that tackles more tasks than ordinary sprints and has non-present members join in the effort online.

Technologically the workflow is significantly improved compared to what we had in ITC. Version control is done through git. More specifically using open source git repository management tool GitLab. This allows for very detailed control and traceability of code changes over time, as well as a considerably simpler way to review code changes and code quality through GitLab merge requests.

Current programmer workflow looks like this:

1. Programmer picks up a task.
2. In his local development platform, he creates a new branch (branch represents an independent line of development that separates the work related to that branch from work done on rest of the project).
3. He programs the feature as specified by the task. For every meaningful change, he commits his code with a meaningful commit message.
4. He pushes his branch which contains his code changes grouped by his commits to the GitLab server.
5. He opens a merge request assigning one of the free senior developers to test and review his code.
6. After review (and potential rework/improvement of code) the merge request is accepted and the changes end up in the master branch ready to be put into production.

This workflow allows for quite easy and transparent overview of work done inside the ITdept by its members as well as considerably more detailed look into actual changes done by individual programmers which helps ensure that code with unsatisfactory performance or quality does not end up in a production codebase.

Another significant change is that programmers don’t work on parade any longer. Instead, to contribute they have to install and use local development platform. In a nutshell, a local development platform is a virtual machine that stimulates production environment and is

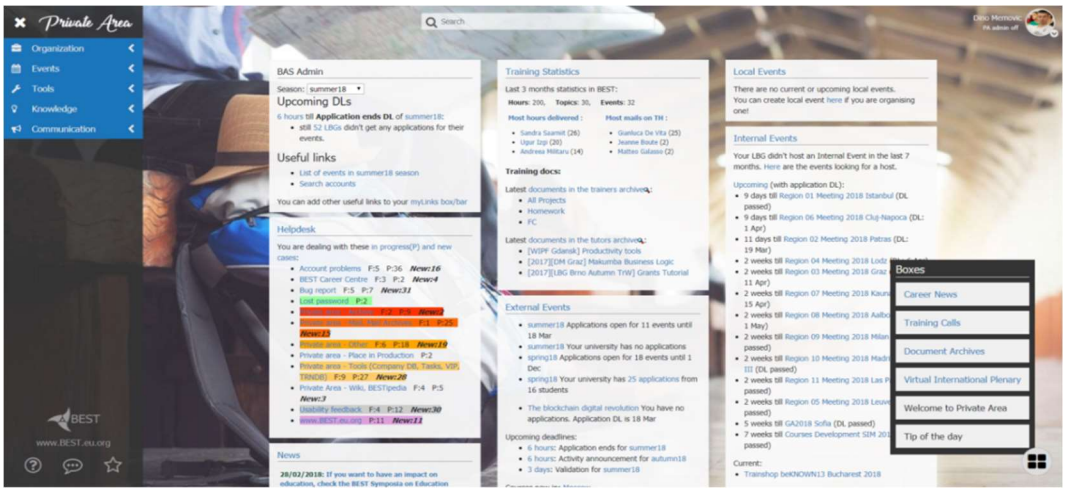
bundled with particular software that helps programmers do their daily programming work more efficiently and effectively. Local development platform has a positive side of allowing programmers to use whatever tools they are comfortable with for development, while it's the main drawback is that it requires quite some effort to properly set up. Its installation is automated but some versions of Windows operating system and notably their updates tend to break the installation process which necessitates constant maintenance and updating of the tool and often causing significant frustration to new developers, especially if this is their first experience with a non-university development setting.

Having looked at the historical development of project management inside the ITC and later ITdept it is time to get an overarching look at how the project lifecycle inside the department looks like at the present.

Every project starts with an idea developed and is initially conceptualized online or on a live event. Being approved by relevant bodies it gets incorporated into the annual plan of the department for the upcoming year (it may be started earlier in case resources are available). At the beginning of the action plan execution (typically beginning of September), call for a project coordinator and team members is sent, Team gets composed by project coordinator with input from department coordinator. International Management of the organization assists in identifying relevant stakeholders as well as finding appropriate product owners to represent the end users of the tool.

At this stage, basic interaction design is already underway and by the time the project team is formed they already have enough information to plan at least one sprint. They proceed to execute the sprint while using emails and online chat tools to prepare the following sprint and communicate their progress. At the end of each sprint period, the team has an online meeting to review the previous sprint and finalize planning and initial work distribution for the upcoming sprint. For projects that don't push directly to master branch, usually, certain milestones are set through which the work gets released to production. Ether through the project, or at the end the code gets pushed to the testing server so that end users can test and provide feedback on features and usability of the tool. In figure 5 below you can see an example of a redesign of the private area currently undergoing testing on the staging server.

Figure 5: New look of Private Area



Source: BEST(2018).

On the end, the project gets concluded by ether finishing all planned work, or by providing a working version of the tool as starting point and input for the next iteration of the project for a future period in case the tool is planned to have multiple releases.

5 CODING METRICS

Another relevant aspect to look at is the activity inside the ITC and ITdept over the years. This is just a short overview of the git log analysis done using R statistical software. Here I will present only the basic overview of the methodology and steps that went into the data analysis, with further details presented in Appendix 1.

In short, the analysis was done on the git log, a file that stores metadata on all changes done to the codebase of a project tracked using git (Software Freedom Initiative, 2017b), in case of ITdept this is the data on its main codebase named Karamba. At present, it stores info on around 7250 commits or contributions to the master branch of the project from 2003 until July 2017. Data from last half a year is omitted from the analysis due to the fact that most of the current work lies on project-specific branches located only on developers personal computers and its work in progress nature makes it hard to utilize for the analysis. As the number of additions, deletions, and files changed corresponds the number of commits, in the further text by contributions I will refer to commits each developer contributed to the said project.

In order to be able to look into the data and find improvement points I first need to have a comparison point. For this purpose besides Karamba, I will also look at the data from two open source intranet projects:

- Motomo – an OSS project that focuses mostly on getting rich analytics from your data. In last 11 years, 256 contributors contributed around 22 thousand commits to the project (Matomo, 2018).
- Plone intranet – an OSS enterprise CMS intranet, to which in last 7 years 67 contributors contributed 8525 commits (Plone, 2018).

5.1 Data preparation and analysis

Data for this project is contained in the git log file, which is a text-based representation of all changes done to the code. From this file I can parse following data fields:

- Commit: the unique key to the commit message,
- Author: name of the author,
- Time: date & time of the commit,
- Message: commit message,
- Effect: text field containing files changed, insertions and deletions.

But the mains of converting fields to proper data formats and by tagging data belonging to each of our 3 projects analysis we can have the meaningful information required to do the whole analysis. Actually, an exploratory analysis is covered in detail in the appendix, while here only calculations relevant to the conclusions will be presented.

Four main comparisons will be covered:

- How long do contributors stay active?
- How much do contributors contribute over their lifespan?
- How meaningful are individual contributions to the project?
- Are DMs cost-effective way of increasing contributions?

In order to analyze how long contributors stay active activity span of each individual contributor had to be calculated by getting the difference between earliest and latest commit said individual committed. After this number of people is calculated for each project and for the following timespans: less than 1 year, 1-2 years, 2-3 years, 3-4 years and more than 4 years. This particular cutout year was selected due to the sharp drop in the number of members in year 4.

For the second metric developers were separated into 3 activity levels:

- Low activity programmers: who on average made less than 20 contributions.
- Average programmers who made between 20-200 contributions.
- Top programmers who made more than 200 contributions.

For each of the clusters average lifespan (the difference between first and last contribution) and a median number of contributions for the group was calculated (sum of contributions divided by the number of contributors).

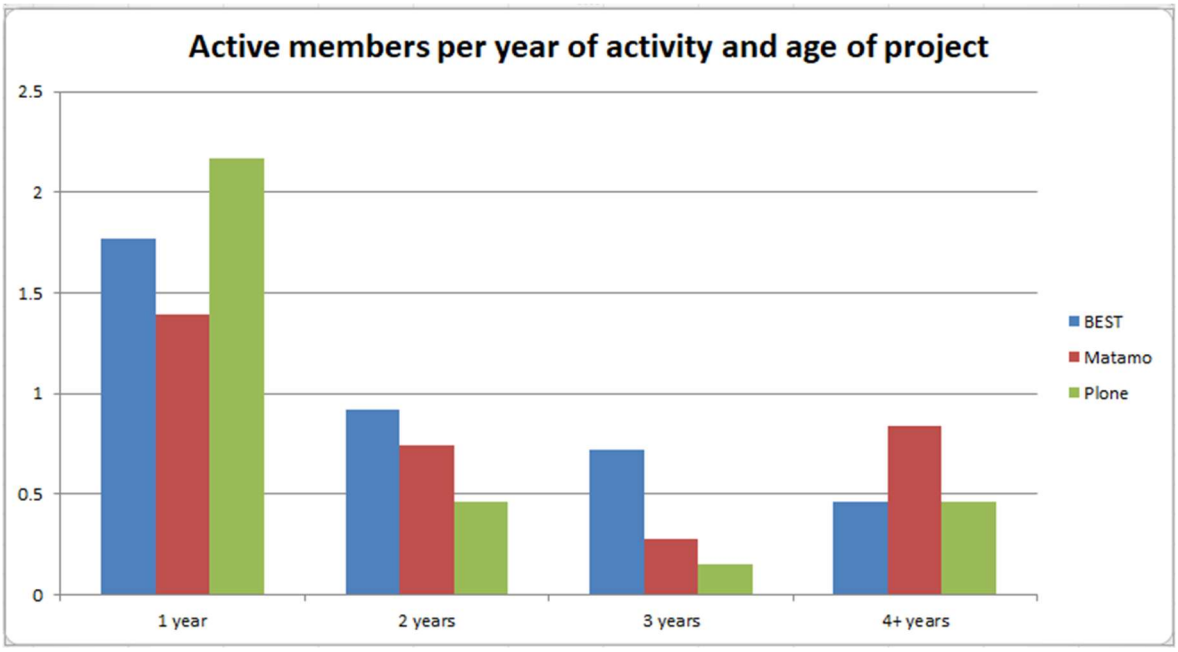
The meaningfulness of each contribution is a measure of how much lines of code each individual contribution modified. While this method has been shown to not be the best estimate for actually programmer productivity significantly big differences in size of contributions can indicate the fact. It is calculated by summing files changed, inserts and deletions and dividing them by the number of commits. After this, a relative measure for setting coefficients of the size of contribution compared to the project with most contributions was calculated.

Further, the question: “Are developer meetings more productive than the regular remote contribution of members”. The answer to this was calculated by time splitting the karamba project data into contributions during developer meetings, contributions around developer meetings (1 week before and after the event) and contributions outside of this period. Then for all three coding metrics were calculated in the same way as described above.

5.2 Results

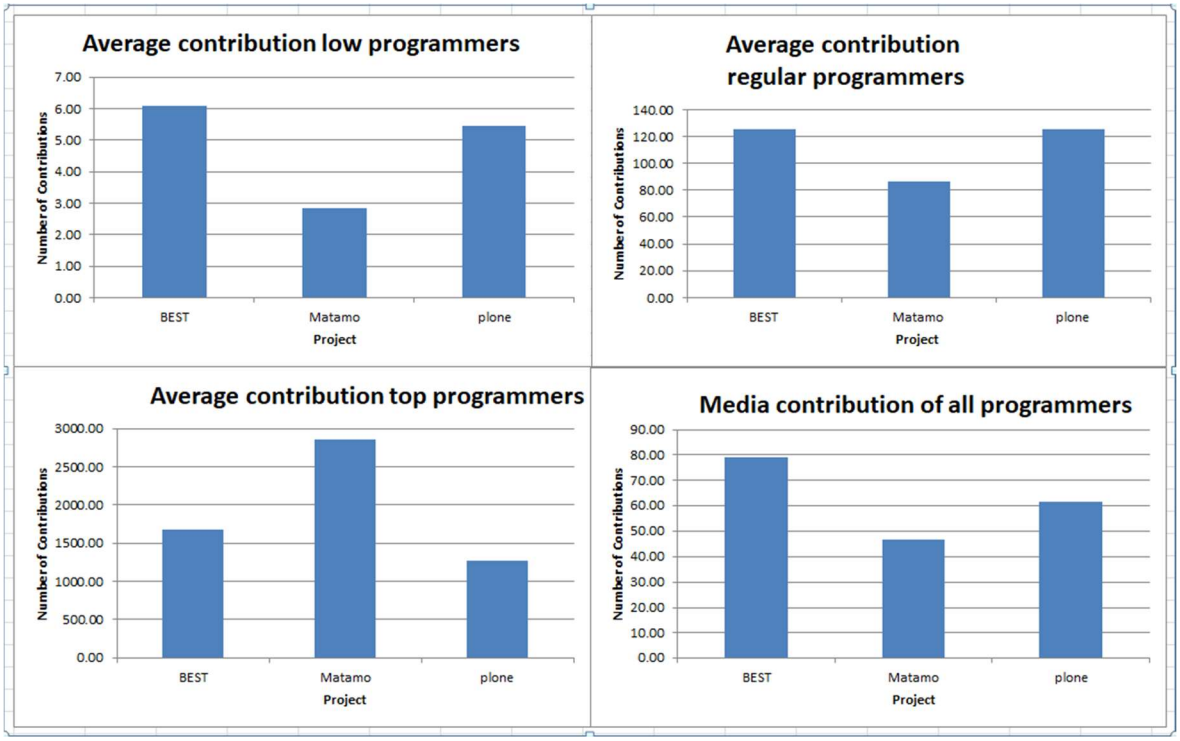
For the first analysis, it is evident that compared to other projects there is a higher in a number of contributors who stop being active in less than a year while contributing less than 20 times to the project itself. Also, it is noted that IT Department gets 35% fewer recruits each year than the mean of all three projects. Another significant difference is that contributors who stay longer than 4 years on average contribute 14% less than it’s the case with other projects. As shown in figures 6 and 7 indicates a clear problem with motivating older more experienced members of the department.

Figure 6: Average member activity per year



Source: Own Work.

Figure 7: Average productivity level for each cluster of developers

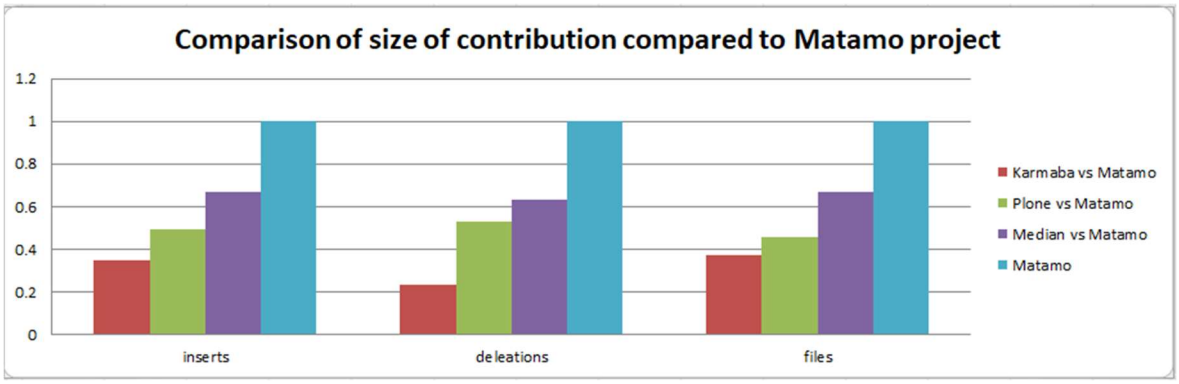


Source: Own Work.

As for the meaningfulness of individual contributions, the IT department's contributors on average contribute 1/3rd as much as much as contributors of Matamo project, and around 50% less than the average of all three projects, which is clearly visible on figure 8. This can be explained in two ways, either the culture mandates smaller more focused commits or members have a tendency of taking smaller tasks than members of other two projects. This is further excruciated by the fact that Karamba is written in Java which tends to be significantly more verbose than PHP and Python which is at the core of other two projects.

Further, the analysis shows that top developers contribute around 8 times more than average developers. This fits well with the long-stated anecdotal maxim that top programmers are an order of magnitude more effective than their less skilled peers.

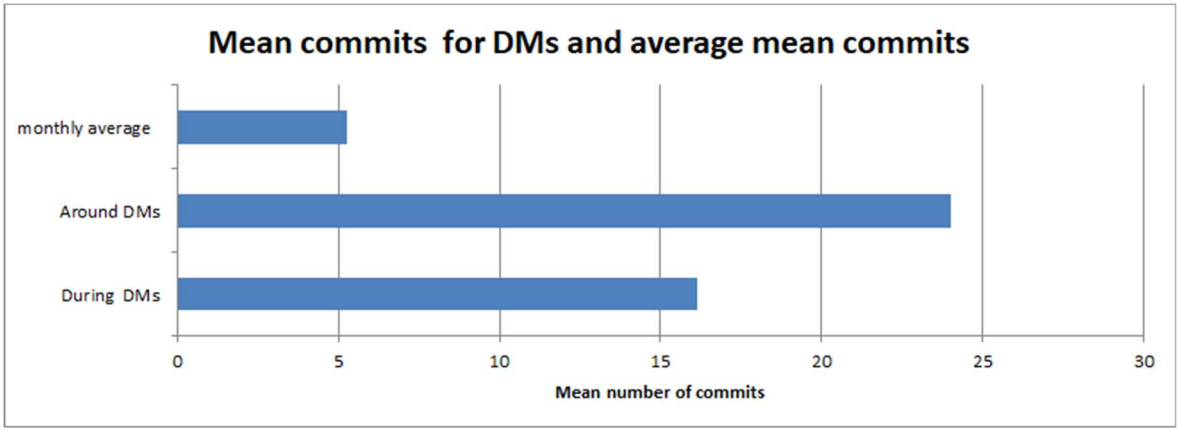
Figure 8: Size of individual contribution



Source: Own Work.

Further, the analysis focuses on the question: "Are developer meetings more productive than the regular remote contribution of members". Results show that during DMs average contribution is around 3 times higher than the average monthly contribution of members of ITdept. This is quite a significant increase as shown in Figure 9 considering that this contribution happens during 3-6 days during which the DM lasts and which I compare to work done by a member during a whole month.

Figure 9: Contribution level during and outside Developer Meetings



Source: Own Work.

Now that I answered the question of: "How does current IT project management in BEST look like?" by looking at both the historical approach and the present day scrum sprint-based approach to project management. Further, I started to analyze the department's work over the years by using statistical methods (again more on this in Appendix 2). For the end of this thesis will use the knowledge I gathered to identify a number of improvement points for the ITdept and suggest potential ways they could be

5.3 Improvement points for the IT department

Looking deeply into the way an organization works, especially from an analytical scientific point of view often uncovers some improvement points or opportunities for the organization analyzed. This section will briefly go over some issues identified through the empirical analysis, theoretical overview, and personal observation as well as though informal talks with the members of the department.

First of the major issues to be addressed is the fact that most people who apply to join the department ether never get active or leave within a year contributing less than 20 times. This is further contrasted by comparison to the other two projects analyzed, which show both higher recruitment rates and lower dropout levels. There can be two main culprits behind the problem, the process a new developer has to go through to get involved and learn all the basics of developing for the system as well as the development workflow and organizational culture. Second are the difficulties involved with installing and running the local development platform, as well as its low performance on older or less powerful computers.

Although the local development platform is designed to be simple to install and use, with few downloads and few commands executed, its main challenge comes with some more obscure bugs appearing on certain versions of virtualization software and Microsoft Windows operating system. This could be in part mitigated by using a platform that is more host independent, as this would ensure that the virtual machine is less influenced by the host.

The performance sadly cannot be addressed directly as low power computers simply cannot run the local development platform properly, an alternative option would be to create something similar to parade that existed in old ITC's days. The online development platform is inherently less taxing for developer's personal computers as it's completely run in the web browser and completely skips the install steps of the local development platform. This, in turn, means that the developers could just start working, instead of having to dedicate time to install everything on their computers.

As for potential non-technical causes of relatively short involvement of members, I can identify a few. One is that for many people this is the first opportunity to actually program in a professional outside of classroom environment. And some people are simply not ready to produce production-ready code at the time when they apply. This mostly happens because of applicants, in general, are first or second-year students of computer science and often lack skills required to produce high-quality code for all except the most trivial of tasks. It has been proven empirically (Aspiring Minds Inc., 2017) and anecdotally by HR professionals in IT industry (Aaronontheweb, 2013) that relatively low number of people who study computer science and related fields can program professionally which has been proven both.

Internally the department can get more accessible, by providing higher-quality tutorials for new recruits that they can take online in their free time. The whole integration process should be reviewed and where relevant streamlined in order to facilitate faster integration times. Also having an ample supply of trivial beginner tasks available will help new developers to get productive sooner, as it is shown that main issue after initial setup problems lies in creating efficient programmers from recruits. Having more live meetings might help as they create a perfect environment for both productive coding and learning from more experienced peers.

Another relevant issue is that experienced members grow less and less active longer they are involved in the department, compared to other two projects analyzed they contribute more than 30% less during their activity period. While they tend to stay available for questions and knowledge long after they stop contributing, it is quite bad for the department when its most productive and experienced developers stop being active. This drop in activity in good part due to the organizational culture which implicitly assumes that at some point the member of BEST will leave the organization. While ITC had and ITdept has members whose average lifespan is several times higher than rest of the organization, this aspect of the organizational culture ultimately contributes to them slowly becoming less and less active. Shifting this culture to be more akin to the open source one where projects have members that are active in the long term would help mitigate or even fix the issue. Sadly this sort of cultural transformation can be quite difficult to execute in practice.

Another big issue comes directly from the restructuring happened in the whole organization. While not affecting the department directly as the old ITC workflow also worked by having a non-formal lead developer spearheading each project, it affected the organization as a whole quite significantly. It created an interesting problem which indicates a symptom of an upcoming leadership crisis throughout the organization.

The numbers of active members on an international level have been quite stable with 220-250 members active annually both during old structure (6 committees and few working groups) and the new current structure (10 departments and around 30 annual projects). But, the leadership required keeping everything running increased drastically.

Under the old structure, each committee had a coordinator, secretary, and some committees also having one or more human resources responsibilities or some other specialized leadership role. On average it would take 20-25 people to run all committees and other bodies of the organization (working groups would have only a coordinator) with additional 6 members of the international board which would give us some 30-35 members who fulfilled leadership roles inside the international part of the organization. That's roughly 13% of internationally involved members who had to focus their time on supportive work compared to rest who focused their time on doing the actual work that fulfills the purpose that body of the organization exists to fulfill. It's very common that a member will take one or two leadership roles during his involvement in the international part of the organization, usually a lower one followed by a higher level role (committee coordinator or international board member), after which the either stop being active or simply would work without taking a leadership role. The culture of the organization tended towards not having long-term position holders in order to ensure better knowledge management.

This tendency continues in the new structure, but the number of positions that require a focus on leadership and similar support work is significantly higher. For the present state, I will assume a flat departmental structure, which implies no leadership required inside the department or project except the department/project coordinator, thesis the structure which the international board of organization is pushing at the moment (at the time of writing departments utilize 2-3 members in leadership roles). So at the moment organization needs 10 department coordinators, 7 members of the international board, 3 additional members of the international HR team as well as project coordinators for each of its average of 30 annual projects. That's around 50 or 21% of internationally involved members focusing on supporting the roughly same amount of work done by the same amount of members active on the international level. This is a 66% increase in leadership needs, which may be more than the organization as a whole can provide. Symptoms of this being a problem are already apparent, at the moment most projects had to extend their calls for project coordinator applicants as in order to find suitable candidates or even because they had no candidates.

Due to the volunteer nature of the organization, members have to segment their limited free time in order to be able to contribute to the organization as well as fulfill responsibilities and goals of their private life. In this case, having members with expertise focus on coordinating project or department effectively means that they cannot effectively contribute to the work in the area where they have expertise due to the lack of time. Having such high demand for leadership coupled with the fact that those leaders need to have a

certain level of expertise in order to lead in the said area of work. So in practice, you would have to constantly utilize experts or emerging experts in leadership roles, leaving you with newer less experienced members to do the actual work that the said body of the organization is doing, which affects both qualities of deliverables and time required to finish the work.

In the long run the organization will have to change its organizational structure to be more lean with less managerial overhead, or face stifling innovation due to the fact that they can't maintain the same number of projects annually or to face having more failed or canceled or severely delayed projects every year or ultimately be forced to completely kill some services or areas of work (which to a limited degree is already happening in the organization).

CONCLUSION

In this thesis, I looked at how one could go about organizing IT for a big international volunteer organization. I focused specifically on organizing IT for a big international non-governmental organization, mainly because there are many international NGOs that are doing great work but that is hampered by their lack of IT knowledge on the inability to properly focus their IT talent to support their goals and fulfill their needs. The main goal of the thesis was to provide both a theoretical background on ITPM as well as examples of organizations who have a long track of successful ITPM from both commercial businesses, NGO and open-source projects while focusing on organizations who operate their IT departments on a geographically distributed global scale. First IT project management was defined, followed by an explanation of intricities that come with working in distributed multicultural teams where team members don't get to meet eachother on regular basis.

What examples of successful implementations of IT project management of distributed online teams from both IT industry and open source projects?

As student organization BEST that this thesis focused on implementing a completely geographically distributed programming workflow I also looked a bit specificities of organizing such distributed programming projects as well as showcased few examples of successful organizations and projects who also implemented similar distributed teams and primarily online workflow.

I looked at the company Automattic Inc. which is behind the biggest online blogging platform in the world. They utilize a less formalized agile workflow by having project teams that have a purpose but that are given full freedom to organize internally and to choose what they will focus on the short and long-term. Then I focused on few examples from the OSS projects. Namely, I looked at the Apache Foundation and Linux kernel project. Both are examples of big international projects that maintain mission-critical code used in the literal majority of computing devices around. Finally, I discussed the impact of computer-supported collaboration work on international geographically distributed teams as well as motivations behind contributing to open source and closed source projects by volunteer programmers.

How is IT department of student organization BEST is organized?

I talked about the student organization BEST. How its matrix structure with 10 departments and some 30 projects on international level supported day to day activities of its 96 local groups and around 4000 members. Then I looked into IT department and its organization.

It was an interesting study to look at how ITdept's members were able to leverage and maintain its quite ancient; more than 20-year-old code base and constantly innovate and react to the organization's needs. I looked at how ITdept went about organizing its teams and knowledge areas inside the department.

I have seen how one can go about organizing an IT department and more specifically how to manage developers and the overall development process for creating features and tools in order to support a large international organization. Moreover, I have seen such an organization evolve its IT department, form it's more primitive early days when only a handful of people bootstrapped different services to its more mature days when its growth and organizations growth make it focus more on maintaining and polishing already existing tools and services instead of creating new services. By looking at ITdept's history I was able to see how different organizational structure of the ITdept affected its work. Starting from its early days when all work was done by few programmers and no real effort was put into figuring out what is actually needed. This approach ended up producing tools that were not as user-friendly and thus they required periodic rework in order to make them more appropriate for the actual use case they were designed for.

As the organization grew and developed so did the need to not waste time and user's nerves on badly designed tools and in response, ITC formalized an interaction design process. The process did its job of reducing the reworks and increasing tool usability and user satisfaction but over time the process ended up slowing down the entire development process as it started to prolong to many months and even years.

How does current IT project management in BEST look like?

That is when the ITdept started thinking of making its development process more scrum like as that would help to keep developers busy and all member of the organization more interested and involved in the whole process. That is when I come to present day ITdept which strives to make its workflow more scrum like. With its current state being that it uses a sprint based development cycle for its internal projects and works ether trough partial or fully-fledged scrum for projects that involve members of rest of the organization. While the tendency to use scrum in all projects is clearly visible, the most that are properly implemented is doing release level planning by focusing on releasing a functional tool or extension of a tool for a development cycle that releases a version of the tool.

At the moment ITdept uses a modified scrum approach for its workflow, organizing all work around 3-6 week sprints that are designed to fit the more erratic schedules of members who besides being volunteers also have to focus on their studies. This aspect of volunteer-based work makes for quite a drastic change compared to a traditional company; the underlying assumption of depending on people having a specific number of hours dedicated to the work in a certain week is not really an option in their case.

How could we improve the IT project management at BEST?

The in-depth analysis of the ITdept's working methods, as well as the theoretical overview of both classical and agile methodologies, gives us some interesting insights on how the department could become better at four fulfilling its mission statement and better satisfying its users.

By analyzing git log data of IT department's main project as well as few similar open source projects I identified four major issues that could hurt the ITdept in the long term. They are:

- Many aspirants apply and then never finish the integration process.
- The local development platform is quite resourced intensive for most computers and it has compatibility problems with many versions of Microsoft Windows operating system.
- Experienced members slowly become less active and the organizational culture implies that they should leave after a certain period of time.
- The transition to matrix structure is showing symptoms of creating a major leadership crisis for the whole international part of the organization.

The first problem is in part due to most applicants being quite inexperienced and not quite ready to work in a professional programming environment. Also, it is noted that streamlining the integration process as well as providing additional support to applicants early on would help achieve higher retention rates.

Local development platform should be moved to a more platform-independent virtualization technology in order to ensure no compatibility problems. Another solution would be to implement an online development platform that would abstract the development work from the device that the work is performed on and thus ensure better compatibility and make it easier for newbies to start working.

The organizational culture itself is the culprit for the problem of experienced members being compelled to leave the department at some point. While its solution is quite hard to implement in practice, ITdept should start working on changing its organizational structure in order to better facilitate long-term membership of its members.

The final problem, while not the originating from the department itself carries the greatest risk as depleting leadership would hamper both its ability to effectively organize projects and the ability to communicate with other bodies of the organization who would end up being too understaffed to provide information required by developers and interaction designers. Creating a leaner organizational structure would mitigate the risks carried by the depleting leadership talent problem. With this I conclude this master thesis, its main goal was to first provide a theoretical background and a framework through which I can look at how to organize programming teams for better support of internal needs of an organization. Following that, I looked at ways multiple organizations go about organizing their IT and programming talent. Finally, I looked in detail at how student organization BEST organized its ITdept, both throughout its development.

The main goal of this thesis was to provide a scientific view on how to organize IT department for an international non-profit organization. It is noted that this area lacks research interest as very little research is done on the topic. So further work on this area of

research is much needed as such work would directly impact the ability of big and small international non-profit organizations to perform their work effectively.

REFERENCE LIST

1. *Basic Phases of Project Management*. (n.d.). Retrieved 7 November, 2017, from http://www.projectinsight.net/project-management-basics/basic-project-management-phases
2. Aaronontheweb. (2013). *The Taxonomy of Terrible Programmers*. Retrieved 7. January 2018 from http://www.aaronstannard.com/the-taxonomy-of-terrible-programmers/
3. About (n.d.). Retrieved 3. December 2017 from https://wordpress.org/about/
4. Alton, R., & Project Management Institute. (2017). *Q & As for the PMBOK® Guide Sixth Edition*. Newtown Square, Pennsylvania: Project Management Institute.
5. Alyahya, S. (2013). *A computer-based holistic approach to managing progress of distributed agile teams*. Cardiff, UK. Cardiff University.
6. Apache Foundation. (2006). *Code of Conduct*. Retrieved 2. January 2018 from https://www.apache.org/foundation/policies/conduct.html
7. Apache Foundation. (2016). *Apache Corporate Governance*. Retrieved 2. January 2018 from https://www.apache.org/foundation/governance/
8. Apache Foundation. (2017a). *Apache Tomcat® - About!* Retrieved 2. January 2018 from http://tomcat.apache.org/
9. Apache Foundation. (2017b). *How the ASF works*. Retrieved 2. January 2018 from https://www.apache.org/foundation/how-it-works.html
10. Aspiring Minds Inc. (2017). *National Programming Skills Report*. Retrieved from http://www.aspiringminds.com/sites/default/files/National%20Programming%20Skills%20Report%20-%20Engineers%202017%20-%20Report%20Brief.pdf
11. Automattic Inc. (2005a). *About Us*. Retrieved 3. December 2017 from https://automattic.com/about/
12. Automattic Inc. (2005b). *Work With Us*. Retrieved 3. December 2017 from https://automattic.com/work-with-us/
13. Bass, J. M. (2014). Scrum Master Activities: Process Tailoring in Large Enterprise Projects. *2014 IEEE 9th International Conference on Global Software Engineering, Global Software Engineering (ICGSE).* Shangai: Institute of Electrical and Electronics Engineers
14. Beck, K. (2000). *Extreme programming eXplained: embrace change*. Reading, MA: Addison-Wesley.
15. Beck, K, Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … & Sutherland, J. (2001a). *Principles behind the Agile Manifesto*. Retrieved from http://agilemanifesto.org/principles.html
16. Beck, K, Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … & Sutherland, J. (2001b). *The Agile Manifesto.* Retrieved from http://agilemanifesto.org/
17. Belenzon, S., & Schankerman, M. (2008). *Motivation and Sorting in Open Source Software Innovation* (CEP Discussion Papers No. dp0893). London: Centre for Economic Performance, LSE.

18. Berkun, S. (2013). *The Year Without Pants: WordPress.com and the Future of Work*. John Wiley & Sons.
19. Board of Europian Students of Technology. (2015). *Internal Regulations of IT Department. BEST*. Retrieved from https://www.best.eu.org/aboutBEST/ITdept/internalRegulations.jsp
20. Board of Europian Students of Technology. (2016a). *About BEST*. Retrieved 6. January 2018 from https://www.best.eu.org /aboutBEST/identity.jsp
21. Board of Europian Students of Technology. (2016b). *BEST Annual Report 2015-2016.* Retrieved 6. January 2018 from https:// issuu.com/ BESTorg/docs/ar_2015_vfinal_web_issu
22. Board of Europian Students of Technology. (2016c). *Internal Regulations of Information Technology Committee. BEST*. Retrieved from https://www.best.eu.org/aboutBEST/ITdept/ internalRegulations _ITC.jsp
23. Brooks, Fred. (1995). *The Mythical Man-month: Essays on Software Engineering*. Addison-Wesley.
24. CNSS Committee on National Security Systems. (2004). *National Information Assurance Glossary* (I no. 4009). Washington, USA.
25. Cooper, A. (2004). *The inmates are running the asylum*. Indianapolis, IN: Sams.
26. Domschke, M., Bog, A., Uflacker, M. & Zeier, A. (2009). Managing globally distributed engineering teams: A case study on virtual industrial engineering. In *IE and EM 2009 - Proceedings 2009 IEEE 16th International Conference on Industrial Engineering and Engineering Management* (pp. 586–589). Beijing: Institute of Electrical and Electronics Engineers.
27. Earley, P. C. & Ang, S. (2007). *Cultural intelligence : individual interactions across cultures*. Stanford : Stanford University Press.
28. Foss, N. j., Frederiksen, L. & Rullani, F. (2016). Problem-formulation and problem-solving in self-organized communities: How modes of communication shape project behaviors in the free open-source software community. *Strategic Management Journal*, *37*(13), 2589.
29. García-Álvarez, M. T. (2015). Analysis of the effects of ICTs in knowledge management and innovation: The case of Zara Group. London: *Computers in Human Behavior*, Elsevier LTD.*51*, *Part B*, 994–1002.
30. Geneca. (2011). *Doomed From the Start? Why a Majority of Business and IT Teams Anticipate Their Software Development Projects Will Fail* (Industry Survey). Oakbrook Terrace, IL, USA: Geneca.
31. GitLab Inc. (2015). *The leading product for integrated software development*. Retrieved 6. January 2018 from https://about.gitlab.com/
32. GNU. (2015). *CVS - Open Source Version Control*. Retrieved 6. January 2018 from http://www.nongnu.org/cvs/
33. Green, M. D. (2016). *Scrum: Novice to Ninja*. Collingwood, VIC, Australia.
34. Halbrügge, M. (2018). *Predicting User Performance and Errors : Automated Usability Evaluation Through Computational Introspection of Model-Based User Interfaces*. Cham, Switzerland: Springer.
35. Harper, R. (2016). From I-Awareness to We-Awareness in CSCW: a Review Essay. Norwell: *Computer Supported Cooperative Work (CSCW)*, (4–5), 295.
36. Harris, P. E. (2016). *Planning and Control Using Microsoft' Project 2013 or 2016 and PMBOK' Guide, Fifth Edition*. [N.p.]: Eastwood Harris Pty Ltd.

37. Hartney. (2016). *The 10 PMBOK Knowledge Area*s. Retrieved 7. November 2017 from http://www.projectengineer.net/the-10-pmbok-knowledge-areas/
38. Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy*, *32*, 1159–1177.
39. Ivancan, D. (2015). *What is BEST*. Retrieved 25. April 2016 from http://www.best.eu.org /aboutBEST/welcome.jsp
40. International Organization for Standardization (2017a). *About ISO*. Retreaved on 25. March 2017 from https:// www.iso.org/about-us.html
41. International Organization for Standardization (2017b). *ISO 27000 series*. Retreaved on 25. March 2017 from https://www.iso.org/isoiec-27001-information-security.html
42. LeanKit. (2018). *What is a Kanban Board?*. Retrieved 6. January 2018 from https://leankit.com/learn/kanban/kanban-board/
43. Matomo. (2018). *Matomo*. PHP, Matomo Analytics. Retrieved from https://github.com /matomo-org/matomo (Original work published 2011)
44. McCall, J. R. (1973). *CPM: a program for critical path management* (Technical No. UCRL-51378).
45. McCarthy, B. (1987). *The 4mat system: teaching to learning styles with right/left mode techniques : Rev. ed.* Barrington, IL: Excel.
46. O'Connell, F. (2011). *What You Need to Know About Project Management*. Chichester, West Sussex, U.K.: Capstone.
47. Oduor, M., Oinas-Kukkonen, H., & Alahäivälä, T. (2017). *Software Design Patterns for Persuasive Computer–Human Dialogue: Reminder, Reward, and Instant Feedback*.
48. Ortiz, C. A. (2016). *The Kanban Playbook : A Step-by-Step Guideline for the Lean Practitioner*. Boca Raton, FL: Productivity Press.
49. Otto, M., & Thornton, J. (2017). *Bootstrap*. Retrieved 6. January 2018 from https:// getbootstrap.com/
50. Pauly, D., Michalik, B., & Basten, D. (2015). Do daily scrums have to take place each day? A case study of customized scrum principles at an E-commerce company. In *Proceedings of the Annual Hawaii International Conference on System Sciences* (Vol. 2015-March, pp. 5074–5083). IEEE Computer Society. Hawaii. Institute of Electrical and Electronics Engineers
51. Picarus. (2012). *IT vs non-IT*. Retrieved 7. November 2017 from https:/iideas4pm. wordpress.com/2012/06/09/it-vs-non-it/
52. Plone. (2018). *plone intranet: Pre-integrated Intranet suite for Plone*. Python, Plone Intranet. Retrieved from https://github.com/ploneintranet/ploneintranet (Original work published 2015)
53. Poppendieck, M., & Poppendieck, T. (2003). *Lean software development an agile toolkit*. Boston: Addison-Wesley. Retrieved from http://proquest.safaribooksonline.com /0321150783
54. Portillo-Rodríguez, J., Vizcaíno, A., Piattini, M., & Beecham, S. (2012). Tools used in Global Software Engineering: A systematic mapping review. *Information and Software Technology*, *54*, 663–685. https://doi.org/10.1016/j.infsof.2012.02.006
55. *Principles behind the Agile Manifesto*. (n.d.). Retrieved 24. November 2017 from http://agilemanifesto.org/principles.html
56. *Project Management Committee Guide*. (n.d.). Retrieved 3. December 2017 from http://www.apache.org/dev/pmc.html

57. Project Management Institute (Ed.). (2013a). *A guide to the project management body of knowledge (PMBOK guide)* (Fifth edition). Newtown Square, Pennsylvania: Project Management Institute, Inc.
58. Project Management Institute. (2013b). *Software Extension to the PMBOK® Guide Fifth Edition* (Vol. Fifth edition). Newtown Square, Pennsylvania: Project Management Institute.
59. Project Management Institute. (2016a). *Construction Extension to the PMBOK® Guide*. Newtown Square, Pennsylvania: Project Management Institute.
60. Project Management Institute. (2016b). *Construction Extension to the PMBOK® Guide*. Newtown Square, Pennsylvania: Project Management Institute.
61. Project Management Institute. (2017). *Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Sixth Edition and Agile Practice Guide (ENGLISH)*. Newtown Square, PA: Project Management Institute.
62. Raymond, E. S. (2001). *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media, Inc.
63. Rasnacis, A., & Berzisa, S. (2017). Method for Adaptation and Implementation of Agile Project Management Methodology. *Procedia Computer Science*, *104*, 43–50. https://doi.org/10.1016/j.procs.2017.01.055
64. Rubin, K. S. (2012). *Essential Scrum: a practical guide to the most popular agile process*. Upper Saddle River, NJ: Addison-Wesley.
65. Ryder, G. (2017). ILO Director-General: BRICS countries play a key role in tackling global world of work challenges. *Premium Official News*.
66. Sangle, D. S. P. S. (2015). Causality in information technology Business value: a review. *Business Process Management Journal*, *Vol. 21*(Iss 3 pp.). Retrieved from http://dx.doi.org /10.1108/BPMJ-06-2014-0061
67. Santos, V., & Belo, O. (2013). Modeling ETL Data Quality Enforcement Tasks Using Relational Algebra Operators. *Procedia Technology*, 9, 442–45 Retreved from: https://doi.org/10.1016/j.protcy.2013.12.049
68. Satapathy, S. M., & Rath, S. K. (2017). Empirical assessment of machine learning models for agile software development effort estimation using story points. *Innovations in Systems and Software Engineering: A NASA Journal*, (2–3), 191. https://doi.org/10.1007/s11334-017-0288-z
69. Scrum Alliance. (2004). *Scrum Values | Agile Manifesto | Scrum Principles - Scrum Alliance.* Retrieved 2. January 2018 from https://www.scrumalliance.org/why-scrum/core-scrum-values-roles
70. Serrador, P., & Pinto, J. K. (2015). Does Agile work? — A quantitative analysis of agile project success. *International Journal of Project Management*, *33*, 1040–1051. https://doi.org/10.1016/j.ijproman.2015.01.006
71. Seufert, E. B. (2014). *Freemium Economics : Leveraging Analytics and User Segmentation to Drive Revenue*. Waltham, MA: Morgan Kaufmann.
72. Shore, J. (2007). *The Productivity Metric.* Retrieved 4. February 2017 from http://www.jamesshore.com/Blog/The-Productivity-Metric.html
73. Slack Inc. (2017). *What is Slack? – Slack Help Center*. Retrieved 6. January 2018 from https://get.slack.help/hc/en-us/articles/115004071768-What-is-Slack-
74. Software Freedom Initiative. (2017a). *Git - git Documentation.* Retrieved 6. January 2018 from https://git-scm.com/docs/git

75. Software Freedom Initiative. (2017b). *Git - git-log Documentation*. Retrieved 6. January 2018 from https://git-scm.com/docs/git-log
76. Sohi, A. J., Hertogh, M., Bosch-Rekveldt, M., & Blom, R. (2016). Does Lean & Agile Project Management Help Coping with Project Complexity? *Procedia - Social and Behavioral Sciences*, *226*, 252–259. https://doi.org/10.1016/j.sbspro.2016.06.186
77. Stanleigh, M. (2016). *Measuring Organizational Performance | Business Improvement Architects*. Retrieved 4. February 2017 from https://bia.ca/measuring-your-organizations-performance/
78. Summers, D. C. S. (2011). *Lean six sigma : process improvement tools and techniques*. Boston : Prentice Hall, cop. 2011.
79. Teorey, T. J. (2011). *Database Modeling and Design : Logical Design* (Vol. 5th ed). Amsterdam: Morgan Kaufmann. Retrieved from http://nukweb.nuk.uni-lj.si/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=356330&lang=sl&site=eds-live
80. The Linux Foundation. (2016). *What is Linux?* Retrieved 2. January 2018 from https://www.linux.com/what-is-linux
81. The Linux Foundation. (2017). *The Linux Kernel Archives - About.* Retrieved 2. January 2018 from https://www.kernel.org/category/about.html
82. The Linux Foundation. (n.d.-a). *Linus on kernel management style [LWN.net]*. Retrieved 3. December 2017 from https://lwn.net/Articles/105375/
83. Torvalds, L. (2004). *Linus on kernel management style.* Retrieved 2. January 2018 from https://lwn.net/Articles/105375/
84. Trello Inc. (2016). *What is Trello? - Trello Help*. Retrieved 6. January 2018 from http://help.trello.com/article/708-what-is-trello
85. University of Minnesota. (2008). *Agile Methodologies.* Retrieved from http://www.umsl.edu/~sauterv/analysis/6840_f09_papers/Nat/Agile.html
86. *Usage Statistics and Market Share of Content Management Systems for Websites,* December 2017. (n.d.). Retrieved 3. December 2017 from https://w3techs.com/technologies/overview/content_management/all
87. Veazie. (2018). Fighting Bottlenecks and Backlogs: Strategies that work. *Receivables Report for America's Health Care Financial Managers*, *33*(1), 5–7.
88. Wachs, P., & Saurin, T. A. (2018). Modelling interactions between procedures and resilience skills. *Applied Ergonomics*, *68*, 328–337. https://doi.org/10.1016/j.apergo.2017.12.013
89. Wasieleski, D. M. (2017). *Stakeholder Management*. Bingley: Emerald Publishing Limited.
90. Wei, K., Crowston, K., Eseryel, U. Y., & Heckman, R. (2017). Roles and politeness behavior in community-based free/libre open source software development. *Information & Management*, *54*(5), 573–582. https://doi.org/10.1016/j.im.2016.11.006
91. Weili, P., & Ross, J. (2005). A Matrixed Approach to Designing IT Governance. *MIT Sloan Management Review*, *Vol.46 No.2*(Winter 2005).
92. *What is Scrum?* (n.d.). Retrieved 29. November 2017 from http://www.scrum.org/resources/what-is-scrum
93. Willes Frank, (2017). *A Guide to Hiring Programmers: The High Cost of Low Quality*. Retreaved on 27. November 2016 from https://www.revsys.com/tidbits/a-guide-to-hiring-programmers-the-high-cost-of-low-quality/

94. Wuttke, T., & Zandhuis. (2015). *A Pocket Companion to PMI's PMBOK Guide Fifth Edition*. Zaltbommel: Van Haren Publishing.
95. Yu, M.-C., Goh, M., & Lin, H.-C. (2012). Production, Manufacturing and Logistics: Fuzzy multi-objective vendor selection under lean procurement. *European Journal of Operational Research*, *219*, 305–311. https://doi.org/10.1016/j.ejor.2011.12.028

APPENDICES

Appendix 1: Git Log Analysis

The following appendix will look into the details of analyzing git log detailing contributions of various members of the ITdept to its main Intranet Private Area, internally referred by its original project name: Karamba. Git log is a text-based representation of all changes done to the files accompanied by who made them and a short explanation for why they happened. In the case of ITdept, it covers 13 years of the contribution of its members. Besides looking at the ITdept I will look also at two open source projects in order to better contrast the department's velocity. These projects are:

- Motomo – an OSS project that focuses mostly on getting rich analytics from your data. In last 11 years, 256 contributors contributed around 22 thousand commits to the project.
- Pione intranet – an OSS enterprise CMS intranet, to which in last 7 years 67 contributors contributed 8525 commits.

While in the most analysis I will only showcase data from Karamba project, this is purely for the purpose of showcasing how the data analysis was conducted and to avoid further duplication from conclusions already depicted in the main body of Master's thesis above.

This analysis strives to answer following questions:

1) How much code can I expect from a programmer and how much work can I expect the department to output in a month, and how does it compare to other projects showcased?

First, I will generate some statistics in order to provide some insight into departments HR situation as well as to get an accurate measure of how much work the department can output in a given month. This information will prove invaluable for the upcoming annual plan of the department as it will help us estimate how many features IT department can expect to code in the following year. Also, I compare this with the productivity of other two projects in order to get a better picture of which improvement points give the greatest potential for change.

2) Does organizing Developer Meetings boost programmer productivity?

Developer Meetings (DMs) are 3-5 day events where programmers meet and work together. As I primarily work online from home, these are rare times I meet up in person. Sadly they are a big strain on organizations recourses as a local best group has to accommodate and feed the programmers and their travel costs have to be reimbursed by the organization.

Having the actual numbers on how DMs affect programmer productivity will allow us to see if organizing these events for the sake of increased output is justified as well as to better understand how organizing DMs can affect how much I as a Department can do in a year.

5.4 Description of data

Datasets used:

- Git log of project “Karamba”, “Pione” and “Motomo”.
- Dataset containing names and dates of developers’ meetings.

Git Log

The main dataset is a large git log containing 19000, 9000 and 22000 commits respectively, logging all changes to the organization’s intranet. It was generated through git using the following command:

```
1. Git.log < git log --date=local --no-merges --shortstat --pretty_format:"%h,%an,%ad,%s"
```

The command resulting in a text file containing single line entries of each individual contribution to the project. Below you can find example output:

```
1061535 dino.memovic Wed May 31 19:31:49 2017 fixed company import from xls/csv 3 files changed, 29 insertions(+), 12 deletions(-)

2295097 dino.memovic Wed May 31 18:29:57 2017 fixed bug in LBG company DB 3 files changed, 3 insertions(+), 42 deletions(-)
```

The information contained in each row is as follows:

- ID: unique id of each commit message.
- Author: a person who committed said change.
- Date & time of the change.
- Commit message: a short description of the purpose of the change.
- No of files changed: how many files were edited.
- Insertions: no of lines of code inserted.
- Deletion: no of lines of code deleted.

The dataset itself was quite complete, excluding a few missing line breaks which were manually fixed by following:

- sorting the table by author inside R after importing it will result in you seeing all funny looking entries (eg. no author starts with a number, so by just looking at imported data it's easy to identify the commits with issues).
- Manually finding the said commit in the git.log and fixing the issue with it.

Also first commit on the projects was removed as it constituted a re-import of the whole project due to the previous repository being lost by a server failure in 2003. The said commit had 1100+ files changed and over 90 000 line edits which would make it a severe outlier affecting in further analysis

Dates of Developers Meetings

It's a dataset extracted by means of an SQL query from the main database of Karamba project. It is a .csv file containing 45 entries of data in the following format:

- Event name,
- Start date,
- End Date.

5.5 Preparation of data

Dates of Developers Meetings

The CSV file was imported into R, the start and end date fields were transformed into Date fields.

An additional field was computed:

- *Event_end*: date 1 week after the developer meeting ended.

The rationale behind the additional field is to encompass the work that usually happens before the event starts, as well as the leftover work that participants have not finalized during the actual event. Both of which are directly caused by the DM itself.

Git.Log

Git.log text file got imported and parsed resulting in following columns:

- *Commit:* unique key for the commit message.
- *Author*: name of the author.
- *Time*: date & time of the commit.
- *Message*: commit message.
- *Effect*: text field containing files changed, insertions and deletions.

Time column was converted into a date using stop time function. Author field had a few conflicting naming patterns which got transformed into the name.surname

Effect field was parsed using regex into 3 distinct numeric fields:

- Files changed,
- Insertions,
- Deletions.

All above were stored in a data.frame called “log” which got subsetted into “Committers” which contained following columns from the log:

- Commit,
- Author,
- Time,
- Files changed,

- Insertions,
- Deletions.

Files Changed, Insertions and deletions were then aggregated based on Author giving us the number of files changed, Insertions and deletions that said programmer did during his/her involvement in the project

Following fields were computed for each Author:

- *Started*: date taken from his first contribution to the project.
- *Ended:* date taken from his last contribution to the project.
- *Months_involved:* how many months the person was active.
- *Commit_perMonth*: No. of commits divided by no. of months.
- *Files changed_perMonth*: No. of Files divided by no. of months.
- *Insertions_perMonth*: No. of Insertions divided by no. of months.
- *Deletions_perMonth*: No. of Deletions divided by no. of months.

Following datasets were subsetted from the log:

- *Low_coders*: coders who contributed less than 20 commits.
- *Average_coders:* coders who contributed between 20-200 commits.
- *Top_coders:* coders who contributed more than 200 commits.

Additionally following 2 data frames were computed:

- *coding_aroundDMs:* all commits whose in between Event_start and Event_end dates from Developer meeting timing dataset (the dates 1 week before and 1 week after actual event).
- *coding_duringDMs:* all commits which were committed during a developer meeting.

5.6 Analysis

The analysis displayed below result either in a table containing values or in a graph. So the process of the actual analysis will be described starting with fields returned by the resulting table followed by a description of how those values were obtained.

Coder Statistics

The goal was to find how what's the productivity of a coder over his/her lifespan. How long that lifespan is as well as what's the difference between low, average and top performers in the project.

Coder productivity

Coder_statistics data.frame returns following values for Low, Average, and Top coders:

- *Activity_level*: Low, Average or Top.
- *Mean_comits*: mean number of commits for a coder.
- *Mean_insertions*: mean number of insertions for a coder.
- *Mean_deletions*: mean number of deletions for a coder.
- *Mean_files_changed*: mean number of files_changed for a coder.
- *Files_changed_by_commits*: no. of files changed divided by the number of commits.

Coders_lifespan data frame looks into how long a member is active on the project its main output is to show a total number of coders involved as well as a summary(Min., 1st Qu, Median, Mean, 3rd Qu, Max.) for the following fields which show aggregated monthly averages for:

- *Months*: how many months members are active.
- *commitsPerMonth*: how many commits are committed per month.
- *insertsPerMonth*: how many insertions are inserted per month.
- *deletionsPerMonth*: how many deletions are deleted per month.
- *files_changedPerMonth*: how many files are changed per month.

Top_vs_average_coder which compares average (contributed less than 500 commits) and top (contributed more than 500 commits) contributors to the project:

- *Mean_avg_code_commits:* gives the mean no of commits an average coder.
- *Mean_top_coder_commits*: gives the mean no of commits a Top Coder.
- *Top_vs_avg*: number of times a top coder is more productive than the average coder.

Besides these, a summary of both Top and average programmers contributions is shown for commits, files_changed and insertions.

Developer meeting productivity

For this, I define two-time periods: "Developer Meeting (DM) which is the coding during the actual events and "Developer Meeting (DM) coding period" which includes the week after the actual event. This is so that I capture all the work done during the event that wasn't finalized during the actual event.

Results of the analysis are stored in the DM_statistics data.frame with following fields (date fields are expressed in days, also ratios and productivity metrics are calculated on daily basis):

- **DM_events :** total time spent on Developer meetings.
- **DM_coding_time:** DM_events + the week after the event.
- **Total_time_days:** Total no of days for which we have data.
- **Total_coded:** total numbers of commits for which we have data.
- **DM_coding:** total number of commits during DMs.
- **SroundDM_coding:** total no of commits during DMs + week before and after.
- **DM_time_ratio:** time spent on DMs / Total_time_days.
- **General_coding_ratio:** total_coded / Total_time_days.
- **DM_vs_total_coded_coding_ratio:** DM_coding / total_coded.
- **Coding_ratio_aroundDM_vs_total_code :** aroundDM_coding / total_coded.

- **Around_DM_time_ratio:** DM_events / Total_time_days.
- **Coding_outside_DM_rato:** DM_coding_time / Total_time_days.
- **Total_coding_ratio:** total_coded / Total_time_days.
- **OnDM_coding_ratio:** DM_coding / DM_events.
- **AroundDM_coding_ratio:** aroundDM_coding / DM_coding_time.

Finally for both Developer meetings and the coding period also I also display the summary functions output aggregate statistics of insertions, deletions and commits for its participants, which is similar to the Coder_statistics data frame which you can find above.

5.7 Results

Coder_statistics

Activity_level	Low	Average	Top
mean_comits	6.41	75.55	715.42
mean_insertions	6.30	72.35	671.50
mean_deletions	5.55	64.35	606.92
mean_files_changed	6.39	75.50	714.67
files_changed_by_commits	1	1	1

```
> summary(low_coders)
   author              commit          insertions        deletions         files_changed
 Length:56          Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
 Class :character   1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.: 2.000
 Mode  :character   Median : 6.000   Median : 6.000   Median : 4.500   Median : 6.000
                    Mean   : 6.411   Mean   : 6.304   Mean   : 5.554   Mean   : 6.393
                    3rd Qu.: 8.500   3rd Qu.: 8.500   3rd Qu.: 8.000   3rd Qu.: 8.500
                    Max.   :18.000   Max.   :17.000   Max.   :17.000   Max.   :18.000
> summary(average_coders)
   author              commit          insertions        deletions         files_changed
 Length:40          Min.   : 25.00   Min.   : 22.00   Min.   : 18.00   Min.   : 25.0
 Class :character   1st Qu.: 38.00   1st Qu.: 36.00   1st Qu.: 33.00   1st Qu.: 38.0
 Mode  :character   Median : 56.50   Median : 55.50   Median : 48.50   Median : 56.5
                    Mean   : 75.55   Mean   : 72.35   Mean   : 64.35   Mean   : 75.5
                    3rd Qu.:101.25   3rd Qu.: 99.75   3rd Qu.: 88.25   3rd Qu.:101.2
                    Max.   :177.00   Max.   :168.00   Max.   :151.00   Max.   :177.0
```

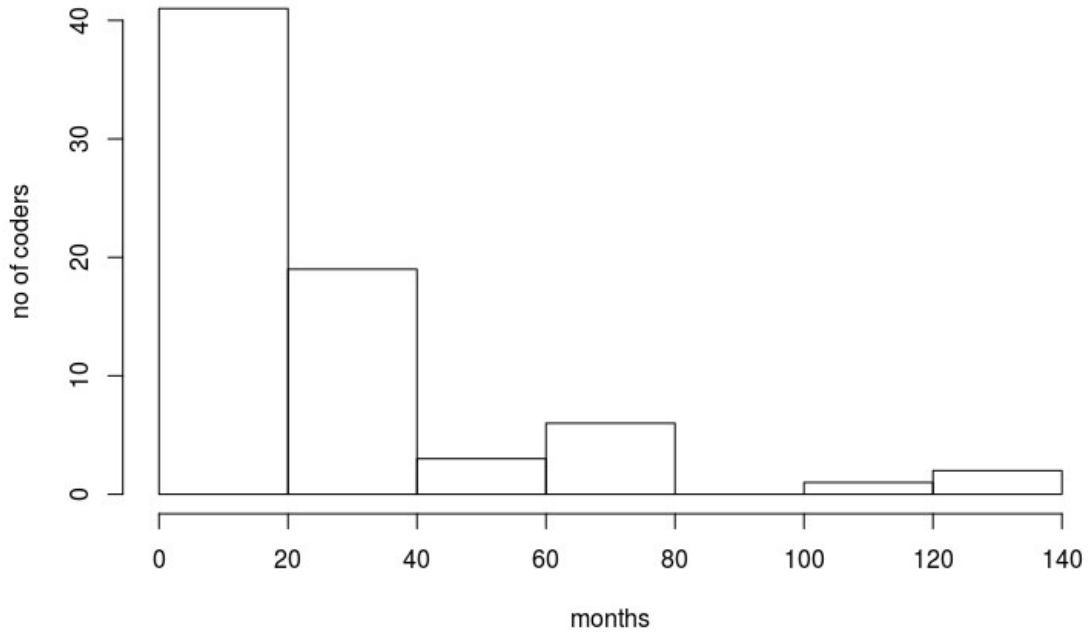
```
> summary(Top_coders)
   author              commit          insertions       deletions        files_changed
 Length:12          Min.   : 204.0   Min.   : 195.0   Min.   : 186.0   Min.   : 203.0
 Class :character   1st Qu.: 303.0   1st Qu.: 290.2   1st Qu.: 257.2   1st Qu.: 301.5
 Mode  :character   Median : 459.5   Median : 438.5   Median : 357.0   Median : 459.5
                    Mean   : 715.4   Mean   : 671.5   Mean   : 606.9   Mean   : 714.7
                    3rd Qu.: 889.0   3rd Qu.: 839.2   3rd Qu.: 802.8   3rd Qu.: 888.8
                    Max.   :2161.0   Max.   :2032.0   Max.   :1791.0   Max.   :2160.0
```

Coder_lifespan and monthly_activity

Figure 3 shows the histogram of the lifespan of departments' members.

```
> summary(Coders$months)
   Min. 1st Qu.  Median    Mean    3rd Qu.     Max.
   2.00    8.00   16.00    25.65   30.00      132.00
> summary(Coders$commitsPerMonth)
   Min.    1st Qu.  Median    Mean    3rd Qu.    Max.
 0.1918  1.9620   3.6020    5.2540  6.6290     28.0600
> summary(Coders$insertsPerMonth)
   Min.    1st Qu.  Median    Mean    3rd Qu.    Max.
 0.1918  1.8540   3.4500    5.0240  6.5060     26.3900
> summary(Coders$deletionsPerMonth)
   Min.   1st Qu.  Median    Mean   3rd Qu.     Max.
  0.100   1.500    3.336     4.543   5.598      23.260
> summary(Coders$files_changedPerMonth)
   Min.    1st Qu.  Median    Mean    3rd Qu.    Max.
 0.1918  1.9620   3.6020    5.2500  6.6200     28.0500
```

Figure 10: Histogram of Length of Coder activity



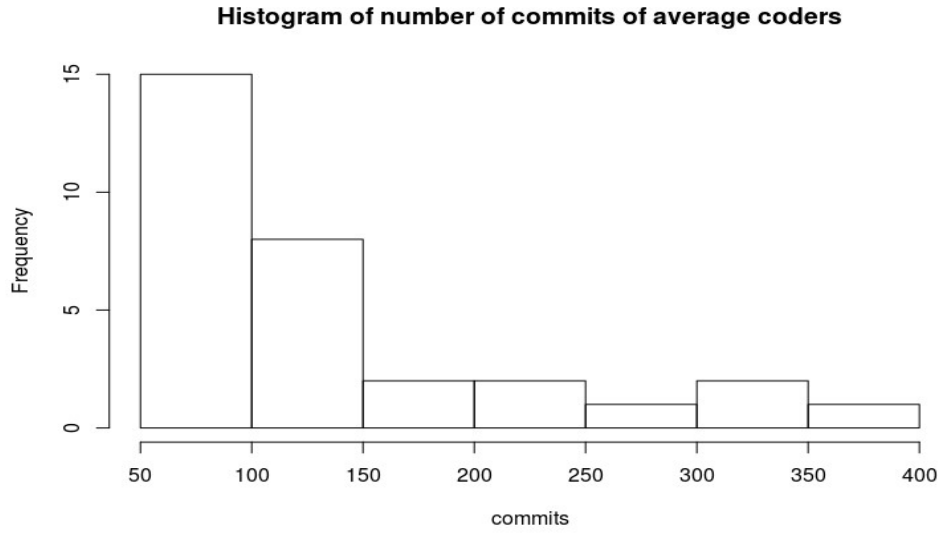
Source: Own Work.

Top_vs_average_coder

Following metric compares the productivity of average programmers and top programmers inside the department, refer to figures 4 and 5 for details.

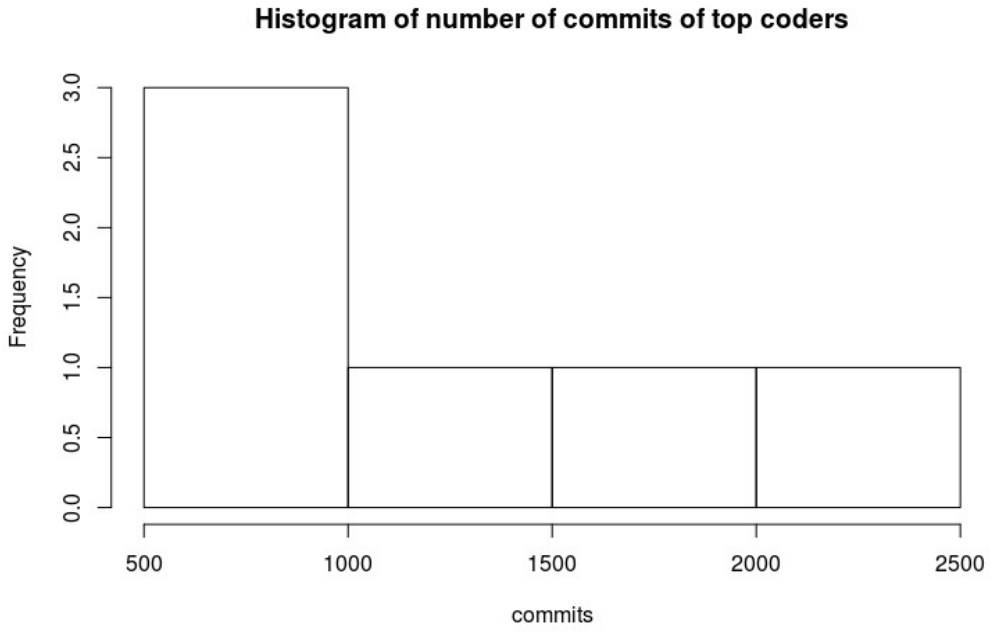
mean_avg_coder	mean_top_coder	top_vs_avg
136.5484	1141.5	8.359674

Figure 11: Histogram of number of commits of average coders



Source: Own Work.

Figure 12: Histogram of number of commits of top coders



Source: Own Work.

DM_productivity

DM_coding_time	500 days
DM_events	185 days
Total_time_days	5256 days
total_coded	11989
DM_coding	839
aroundDM_coding	1538
DM_time_ratio	0.03519787
DM_vs_total_coded_coding_ratio	0.06998082
coding_ratio_aroundDM_vs_total_code	0.1282843
around_DM_time_ratio	0.09512938
coding_outside_DMs_ratio	2.121385
total_coding_ratio	2.281012
onDM_coding_ratio	4.535135
aroundDM_coding_ratio	3.076

```
> summary(Coders_onDMs)
   author              commit          insertions         deletions
 Length:52          Min.   :  1.00   Min.   :  1.00   Min.   :  1.00
 Class character    1st Qu :  3.00    1st Qu.:  3.00   1st Qu.:  3.00
 Mode  character    Median :  7.50   Median :  7.00   Median :  6.00
                           Mean   : 16.13   Mean   : 15.44   Mean   : 13.96
                           3rd Qu.: 17.00   3rd Qu.: 16.50   3rd Qu.: 16.00
                           Max.   :144.00   Max.   :139.00   Max.   :112.00
                                                                  NA's   :3
```

```
> summary(coding_aroundDMs)
   author              commit          insertions         deletions
```

Length:64	Min. : 1.00	Min. : 1.0	Min. : 1.00
Class character	1st Qu.: 3.00	1st Qu.: 3.5	1st Qu.: 3.00
Mode character	Median : 8.00	Median : 8.0	Median : 7.50
	Mean : 24.03	Mean : 23.3	Mean : 20.94
	3rd Qu.: 25.25	3rd Qu.: 24.5	3rd Qu.: 23.00
	Max. :271.00	Max. :253.0	Max. :219.00
		NA's :1	NA's :2

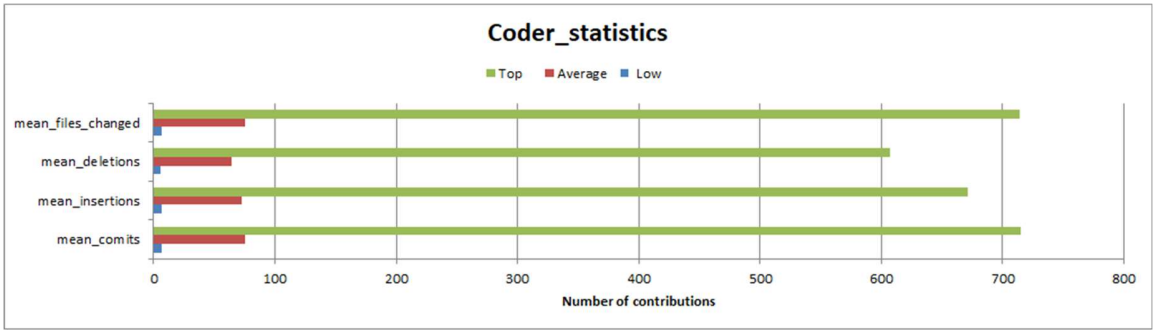
5.8 Conclusions/Discussion

Here I will provide some additional insight into the project karamba, for comparative analysis of three projects please refer to "Coding Metrics" part of the main thesis.

The first part of this analysis had a simple goal of creating some basic statistics so that the department gets a bit better insight into its human resources, as well as what can be on average expected from a member of certain experience level.

What I found out is that a top-level programmer on average outputs 8.35 times as much code as an average developer. Which actually confirms with anecdotal evidence stating that top coders are an order of magnitude more productive than average coders. Below in figure 6 you can find a graph showcasing this productivity gap:

Figure 13: Mean coder productivity



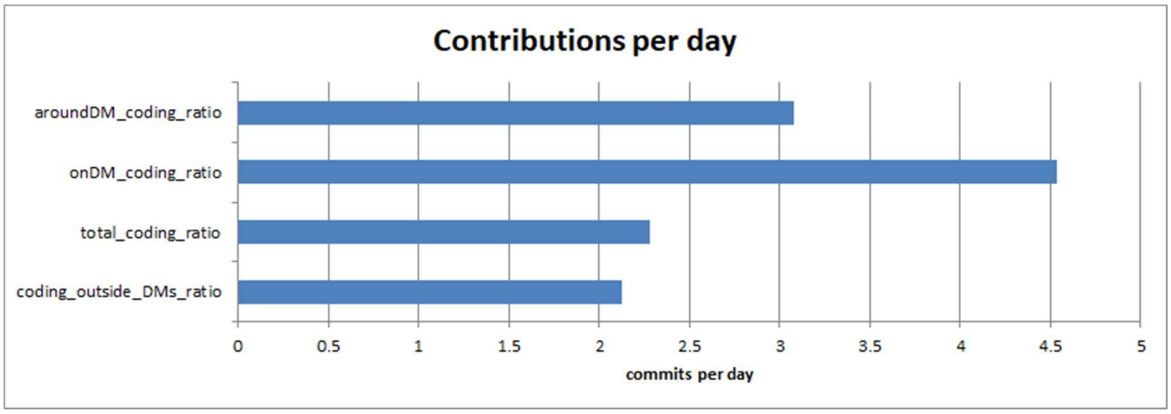
Source: Own Work.

I also now know the average monthly activity of the IT Department which will allow us to better plan the number of features I will implement for the next year. An average programmer contributes around 5 times per month, which roughly translates to five tasks or features. Also

knowing that a programmer will stay active for a bit over 2 years I can estimate how many new members I should recruit in order to fourfold the IT needs of the organization.

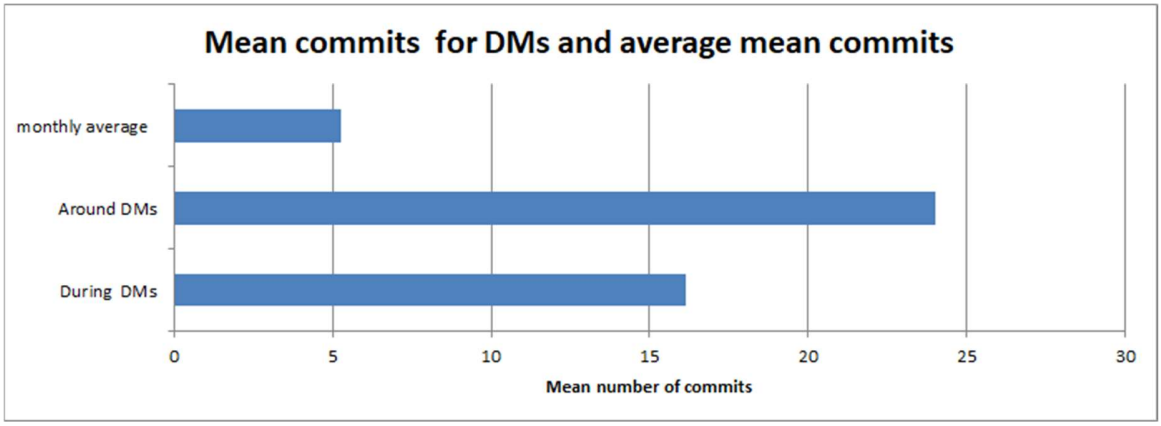
As for the second part of the business problem, I have proven that programmers are more productive on Developer Meetings than on their average work from home. Programmers are twice as productive on developers meetings as outside developer meetings as indicated in the following figures 7 and 8 show:

Figure 14: average daily contribution of members during DMs, around DMs and average daily contribution,



Source: Own Work.

Figure 15: Mean commits for DMs and average mean commits



Source: Own Work.

They show that not only a developer attending a DM will be twice as productive; he will end up contributing 15-25 features during the event and its aftermath. This allows us to estimate

how much work can I estimate out of a developer who is attending as well as allowing us to take note of members who may need further support before becoming averagely productive developers.