A proposal to use Github as PDM environment for a vision based sand analyzer

Jelle Spijker[1]* 495653

Abstract
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Keywords
Product Data Management — Vision Soil Analyzer — Github

[1] *Department of Engineering, HAN University of Applied Sciences, Arnhem, the Netherlands*
***Corresponding author**: spijker.jelle@gmail.com

Contents

Introduction

This article proposes a setup for a PDM[1] environment used during developed of a vision based sand analyzer [2]. This new line of product(s) is currently being developed at IHC MTI B.V. and has the ability to analyze sand by its optical characteristics.

The concept and initial design of this product finds it origin at the minor embedded vision design and will be further developed during the graduation phase mechanical engineering at the HAN. Design and production of this product leans on the following three disciplines: mechanical, electrical and software engineering. Due to its multidisciplinary properties it is relevant to implement a PDM environment which acts as a solid basis for all disciplines and allows for interaction and flexibility.

IHC MTI B.V. is a R&D based company and subsidiary of Royal IHC . Although Royal IHC is currently in the process of implementing a new PDM and ERP[3] environment named IHC ONE, linking all wharfs, departments and subsidiary companies using Siemens Teamcenter and IFS. Current data systems are all still folder and DMS[4] based. This setup doesn't allow for the necessary flexibility and revision controlled, that is needed for the VSA project.

It is therefore proposed to make use of Github as an PDM environment. Github is a web-based Git[5] repository, used in the development of software and build upon the principles of collaboration. Due to the origins in software development this environment is largely unknown in the mechanical engineering branch. But it is gaining traction as an PDM environment according OLEG[5].

Github is strongly rooted in software development, it therefore has a strong toolbox supporting it. This article clarifies the protocols and processes which will ensure support for

[1] Product Data Management
[2] Working title VSA
[3] Enterprise Resource Planning
[4] Document Management System
[5] version control system for software development designed and developed by Linus Torvalds and used for the Linux Kernels

the none native disciplines: electrical and mechanical. Thus ensuring a fully fledge PDM environment. These protocols will also put safeguards in-place to ensure protection of IP[6] but still allows multiple users/parties to work on sub-projects. It does so by defining different user roles and set restrictions to sub projects.

This document will illustrate how to use Github as a PDM tool during the development of a vision based sand analyzer . It does so by describing the regular workflow employed by Github. It will then describe the current product and give a foresight in to certain possible future developments. Afterwards it describes which adjustments need to be made with respect to the normal workflow in order to support the whole VSA project.

This article has the following document structure: In section 2 the main product is described using a product breakdown structure at such a level that it

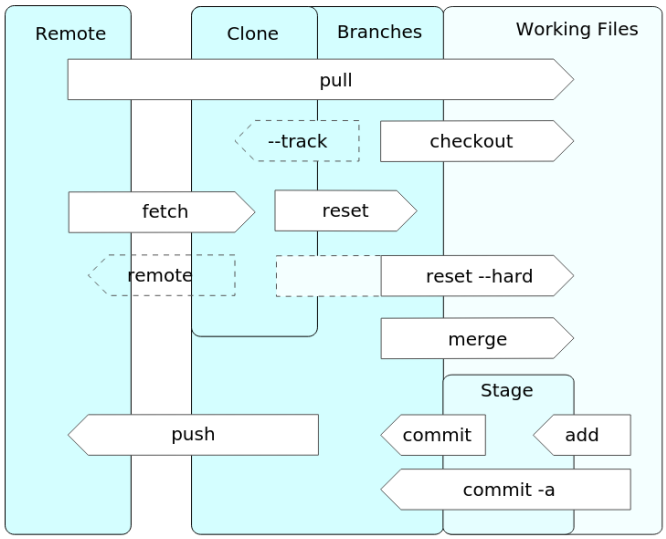

Figure 1. Git workflow. Source: [2]

1. Github

Github is a web-based Git repository hosting service with SCM[7] and distributed revision control capabilities. It also provides access control and multiple collaboration features, such as bug tracking, feature request, task management and wikis. Github has a strong influence in the open-source community. This is because standard ”free” services don’t allow for private repositories. They’re accessible by all and thus allow everyone to view, copy and use them. A lot of company, governments and institutions such as NASA, CERN, Google and Netflix do so under the credo; Dare to share. For those whom want to protect their intellect property they offer paid accounts which allow for private repositories. It is also possible to host your own service using GitLab

Recent developments show a move from strictly software support to other disciplines, to cite Peter Bell and Brent Beer [6] While GitHub is still primarily used to collaborate on the development of software, it’s also a great way for a team to collaborate on a wide range of projects. From the authoring of books and the distribution of models for 3D printing to the crafting of legislation, whenever you have a team of people collaborating on a collection of documents, you should consider using Github to manage the process.

Github is based on the Git workflow, which was developed by Linus Torvald to help with the development of the Linux kernels. Git is a strict command-line tool which emphasizes on speed and data integrity, it allows for a distributed, non-linear workflow. This workflow is explained in the next section.

1.1 The normal Github workflow

As mentioned in the previous sections, Github is Git-based and works with repository which can roughly be described as a directory structure with revision control. This workflow is depicted in figure 1 and can be described as follows: A new repository is created on the remote server, which is then pulled to a local host. A user can make changes in this directory by adding, deleting or modifying files and directories. Each alteration such as the addition of a few lines of code, which represent a new function are added to the staging area. These alterations are then committed to the current branch, which will be discussed further on. These commits are then pushed to the remote server, where they’re stored with meta-data such as comment, blame (who applied the change), and a date. Each commit only stores the change in the file and allows a user to review against previous commits (versions).

If another developer has committed new work, the developer first has to pull these changes to his local host before he is allowed to push his own changes. Git will automatically detect any conflicts and show them to the developer. He will then be allowed to solve these conflicts if needed.

As mentioned before the commits are first stored in a branch. These branches allow for distributive workflow. They do so be running parallel with the master branch. If a new feature is being developed, it is common practice to create a new branch, implement the new features, which probably consist of multiple commits and pushes to the remote server. Which allow other developers to work on the same feature as needed. All the while changes ar still allowed on the master or any other branch. Once the feature is ready to be added to the master or an other branch, a pull request is made to the community.

This will compare the current branch against the other branch and allow for other developers to discus and review the code. If they find it satisfactory and they don’t foresee any major problems the pull request is granted and the branch is merged. This process is shown in figure 2.

The figure 2 also shows the use of tags. It is possible for a developer to tag a certain commit, in essence freezing the state of the project. This is mostly done when versions are

[6] Intellectual Property
[7] Source Code Management

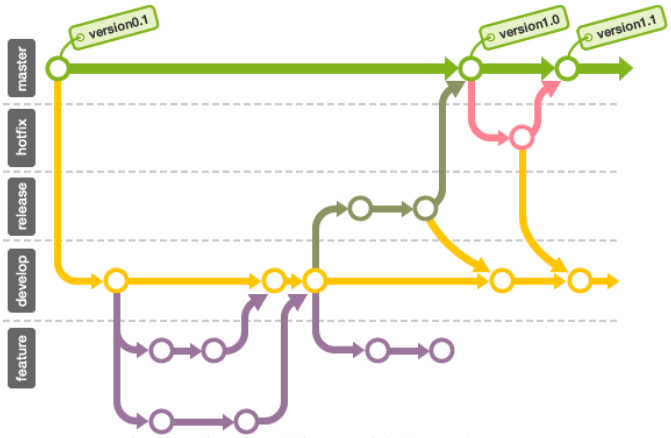

Figure 2. Distributive workflow with branches. Source: Backlog[1]

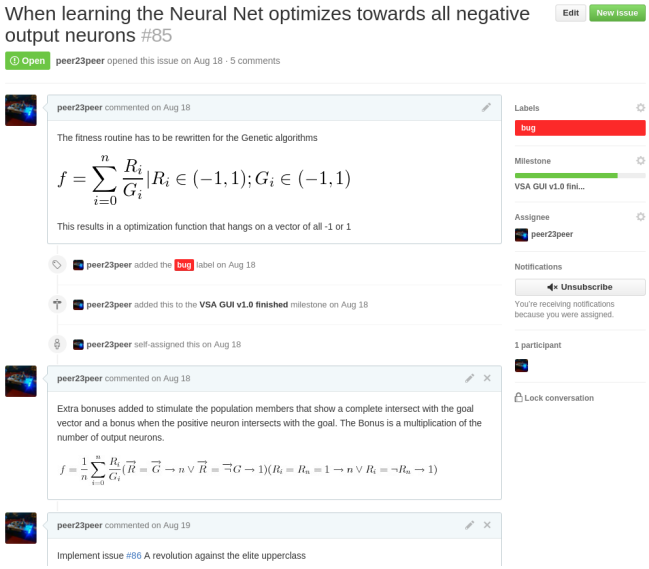

Figure 3. Issues used with Github

released to the public or classification agencies. These tags allow the state of the project to be recalled at that time.

1.2 Issues and waffle

Further active collaboration is supported and encouraged using Issues. These a generic items which could be idea's, task, bug reports or questions for instance, which can be submitted by a developer. An issue is created by describing it, illustrating it with images, links or examples. Assigning a label to it such as ToDo, bug, feature enhancement. Other developers can comment on these issues so as to give it direction. Such an issue and discussion is shown in figure 3. These issues can be assigned to individual developers and be assigned to milestones. Which allows them to focus on further goals.

In order to distribute the workload it is possible to use the third party service Waffle, which integrates completely with Github. It shows the status of all issue, and present four column, namely: Backlog, Ready, In Progress and Done. All open issue start in the Backlog, when preparations are made to start working on them to can be moved to the Ready column. As soon as work is commenced the move to the In Progress column, when they're finished the move towards the Done column. The throughput of the project can be monitored closely allowing to actively steer on optimal team performance.

1.3 Project wiki

Each project has its own wiki[8] It's expected from the developers to maintain an active wiki. All knowledge regarding the project can be gathered here, shared and improved

2. Vision based Sand Analyzer

The vision based sand analyzer is a product that analyzes sand sample using a digital camera sensor. It does so by taking a snapshot of a magnified soil sample and applying multiple software algorithms on an obtained digital image. A detailed and full description of the first two prototypes are described in the product documentations of the previous build prototypes [3] and [8], where the last iteration serves as the basis for the product breakdown structure depicted in the next subsection.

2.1 Product breakdown structure VSA

The product breakdown structure, depicted in figure 4 differentiates between three structures: hardware, software and the casing. This demarcation is made to better serve the different disciplines. This breakdown is purely done on individual parts, where each part serves a function. The intricate web of interfaces and connections are set out in section 2.3.

Due to the scope of this document these functions are not specified. The software functionality is covered in detail in the Doxygen documentation. It is important to note that the software is developed with the object orientated language C++ and is divided in different namespaces. Which are depicted in figure 4. Each of these namespaces is a bundle of object each serving its own function and by inheritance it's possible to create complex objects. Integration of the software development environment in Github is seamless.

Whilst function which are fulfilled by the casing and hardware are describes in the product documentations [3]. The individual parts serve as actuators, sensors and interfaces with the outside world.

2.2 The toolboxes

Each discipline uses it's own type of tools, it is therefore relevant to asses all used tools and make sure the are compatible with each other. Each of the tools below are stand-alone versions which run on a operating system, they store their data in files, which in turn can be seen as the output. At the basis of all three development environments lies the hardware. The specifications given below, describe the current development computer. It is guaranteed that the project can be recreated with a similar computer.

[8] A wiki is a website which allow collaborative modifications

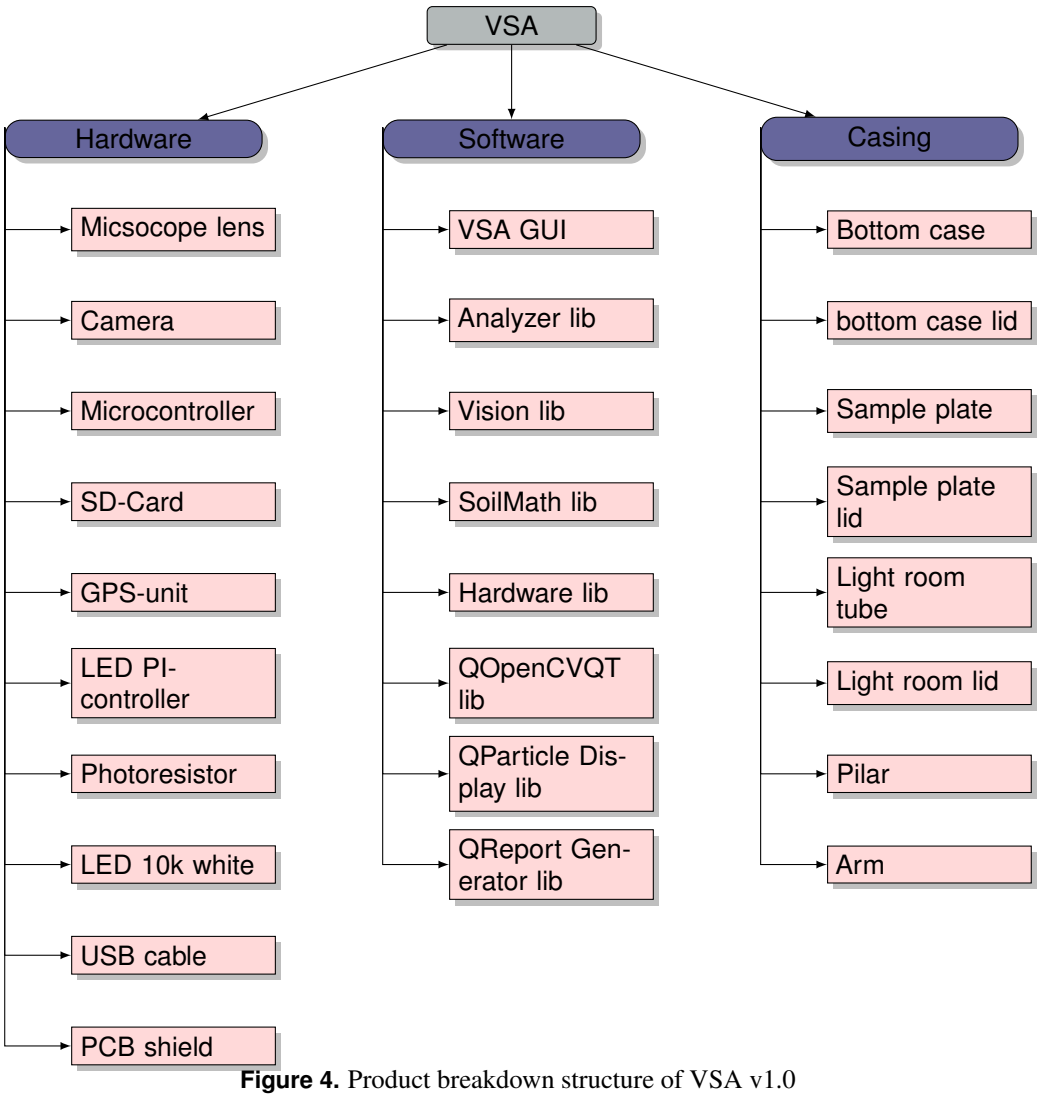

Figure 4. Product breakdown structure of VSA v1.0

- Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz
- 8gb memory
- Nvidia 820M
- SSD 128 gb
- HDD 500 gb
- Dual boot with Kubuntu 15.10 / Windows 10

Software engineering The VSA currently runs on a dedicated embedded Linux environment and is programmed in C++. The IDE[9] consist of a Linux Debian environment running Qt Creator, testing is done with Valgrind and code compiling is done with the GNU GCC compiler or LLVM Clang. The basic list of used libraries is given below.

- Standard C++ Library
- OpenCV 3.0 beta
- CUDA 7.0 SDK
- ZLib
- Boost 1.58
- Video4Linux
- GStreamer

All these packages are chosen such that the can easily be ported to other operating systems, such as Windows, Android or iOS. Although current prototypes are dedicated embedded devices, the ultimate goal is to create a cheap and portable device. In order to achieve that goal, it is important to make use of existing and readily available architecture such as phones and laptop. Software development has to be done with this foresight; Currently all these tools allow for this strategy.

The output of the software engineering toolbox will be mostly source code and binary files. The source code files are plain text format, which can be read and edited from most devices. While the binary files will consist of images and other additional resources needed by the program. This actual program needs to be compiled for each individual target system. This is done with each release and the files are bundled with the version tag.

Future release may save their data in databases, it is possi-

[9] Integrated Development Environment

ble that these database run on the local host, but they could also run on remote host. Databases need special care to put under Git revision control. The best practice is to dump the database and schema and put the output – which is a plain text file – under git control.

Electrical engineering The actuators and sensors, such as the GPS-unit, LEDs and photoresistor are all driven from an embedded Linux device, The Beaglbone Black. This device has the capability to run an OS and can interact with the outside world by pinmuxing[10]. The PCB[11] and circuits are designed with the Circuit Design Suite from National Instruments.

This suite consist of the programs Multisim for circuit design & simulation and Ultiboard for PCB design. Both programs only run in a Windows environment. Multisim works with a binary file with the extension **.ms14*, but it also has an option to output SPICE[12]. Which are plain text files, and can be used with a multitude of different simulation programs. Ultiboard works with a binary file that has the extension **.ewprj*. It can export in the Gerber format, which is a plain text file and the de facto industry standard used with PCB.

Mechanical engineering Each part has a mechanical representation. These parts are modeled in Siemens NX 10, which offers a rich interface for mechanical engineers. They allow for parametric modeling and complex FEM and CFD simulations. Siemens NX output all its file in binary formats. These files aren't backward compatible.

Github recently adopted a new viewer for 3D-models. It can now read and compare previous versions of **.stl* files, as shown in figure 5. These file are easily exported with NX and are used for CAM[13] such as CNC[14], 3D-printing or any other rapid-prototyping technique.

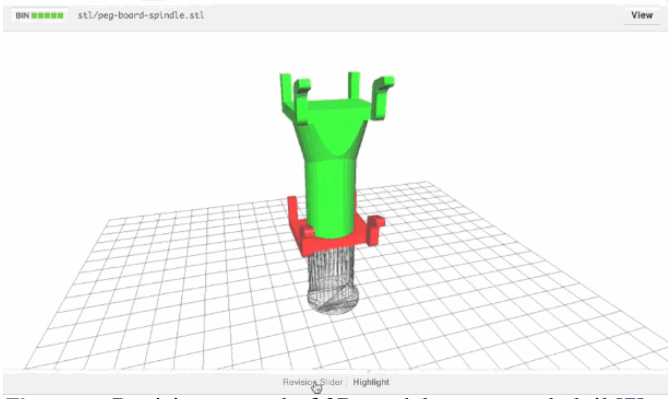

Figure 5. Revision control of 3D models, source: skalnik[7]

Simulations and calculations are done with Matlab 2015a, which has a Git integration. Most Matlab files are plain text files, but special care should be taken with the Simulink models. According to the Matlab website it is necessary to prevent Git from corrupting Simulink models, this is done by editing the *.gitattributes* file and adding the following lines:

```
*.slx -crlf -diff -merge
*.mdl -crlf -diff -merge
*.mat -crlf -diff -merge
*.mlx -crlf -diff -merge
```

Documentation LaTeX

2.3 Interfaces

3. What is a PDM environment

In order to truly understanding the function of a PDM environment, one has to start with the main definition, which is according to La, Hoogeboom, and Konst [4] a strategy which puts the right product and process related information in the right hands at the right time in a product life cycle. It also allows for the creation of processes which guard those data en allows other to use it. All the while generating en secure environment, at which all relevant information, and previous iterations is stored without redundancy.

Github is without a doubt a great tool when used as revision control it save all relevant information and stores it in a secure environment without redundancy. It has potential to be used as a full fledge PDM environment. But in order to maximize Github for this setup it is relevant to look at the mechanics of such an environment a little closer, so that it becomes clear what should be added or changed.

Data vault Each data item should be stored in one single location, where only authorized developers can access that data at a time when its needed. In order to support this system, roles should be defined. Each developer needs to be assigned one or more roles, which allows him access the for that role relevant data. The role should also specify if he has read or modification rights.

Metadata One of the main features of a PDM environment is a possibility to quickly look-up metadata[15], which comes in two flavors: structural and descriptive. The structural meta-data tells something about the container e.g. a file, whilst descriptive meta-data gives a user information with regards to the data content. Searching for this data should be intuitive and simple; Using elementary queries.

Classification Classification of the files is important because not every file uses the same metadata fields. Classification can be done with the use of (sub)groups, narrowing the possibilities and making sure only the relevant items turn up in a search. It is advised to also create

Check-in / -out Developers should not disturb each others workflow. They check-out their work, modify it, and check it in again; Accompanied with a small description of the changes.

[10] Changing the state and function of pins on the CPU, such that it allows for different in- output protocols
[11] Printed Circuit Board
[12] Simulation Program with Integrated Circuit Emphasis
[13] Computer Aided Manufacturing
[14] Computer Numerical Control
[15] Data about data

Bill of materials A complete list of all materials should be easily obtained for each product, preferably from different role perspectives.

4. Implement Github as PDM environment

4.1 Meta-tag all files

Tagsistant

```
git checkout master
git checkout exp1 path/to/file\_a
git checkout exp2 path/to/file\_b

\# save these files as a stash
git stash
\# merge stash with master
git merge stash
```

5. Results and Discussion

Pull out items from other branches Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

References

[1] Backlog. *Branching workflow using topic and integration branch [branch] | Git Beginner's Guide for Dummies | Backlog*. 2015. URL: http://backlogtool.com/git-guide/en/stepup/stepup1_5.html (visited on 11/21/2015) (cited on page 3).

[2] *Git (software)*. In: *Wikipedia, the free encyclopedia*. Page Version ID: 691077698. Nov. 17, 2015. URL: https://en.wikipedia.org/w/index.php?title=Git_(software)&oldid=691077698 (visited on 11/21/2015) (cited on page 2).

[3] Jelle Spijker. *Vision Soil Analyzer: Product design of a vision based soil analyzer*. Oct. 10, 2015 (cited on page 3).

[4] Fontaine J. P. La, M. G. R. Hoogeboom, and J. S. Konst. *Product Data Management: A Strategic Perspective*. Geldermalsen: Maj Engineering Publishing, Aug. 24, 2009. 103 pages. ISBN: 978-90-79182-06-0 (cited on page 5).

[5] OLEG. *GitHub PDM: Is It For Real?* Beyond PLM (Product Lifecycle Management) Blog. Sept. 19, 2013. URL: http://beyondplm.com/2013/09/19/github-pdm-is-it-for-real/ (visited on 11/18/2015) (cited on page 1).

[6] Peter Bell and Brent Beer. *Introducing GitHub: a non-technical guide*. O'Reilly, 2015 (cited on page 2).

[7] skalnik. *3D File Diffs*. GitHub. Aug. 17, 2013. URL: https://github.com/blog/1633-3d-file-diffs (visited on 11/18/2015) (cited on page 5).

[8] Jelle Spijker. *Product documentatie: Test opstelling - Soil Analyzer*. Nov. 2, 2014 (cited on page 3).

Appendices

A Latest prototype