Team members

Jordan Griffiths  
Tom Cleaves  
Leo Edwards  
Hiten Dabhi  
Rowan Guyton   
Harry Elphick

Appendix

**1.A. Overview of the problem to be solved**

We are a start-up company very interested in the IoT (Internet of things) and the advantages it can bring in business or the home. Our team has been given the task of developing a monitoring solution; accessible through the web or via mobiles, that takes in a variety of data from different devices (Fridges, Heating systems, Water, etc.) and displays information on those devices, such as current and past status, or other device specific data to the user, in a clear and user friendly way. The main user interface will obviously need to be accessible to the user, through the internet, and this will be a big part of the design choices made.

The solution should provide the following functionality, enabling users to:

* Look through a list of devices currently on the network (at least 3)
* Investigate their devices current and past status
* Add new devices
* See usage details from each device
* Correlate usage patterns

As well as being secure, usable and simple the solution should be scalable, up or down, to accommodate and allow for the monitoring of new devices. This means the interface will also need to allow for this, by including the correct types of visual elements that can extend and scale, while keeping information clustered and readable. Using visual elements with attributes such as scroll bars, or pagination within some parts of the interface may help to organise information if a user has a large amount of devices, and decisions like this will need to be thought about when designing the software itself.

The main devices we will be using to demonstrate the functionality of this system are:

* Raspberry Pi v2 running Ubuntu
* Device 2
* Device 3

In order to demonstrate the solution we have built, we will be including text files on each of the devices that can be read from, by the solution (stub/driver method) to simulate how the application would work in a real world setting, and to provide a proof of concept prototype.

**1.B. Strategy for Team Organisation**

**Our Approach and Why…**

*‘Egoless Chief Programming’*

The following factors have been considered:

* Our team is small and will stay the same size throughout the project
* Our team consists of individuals who have expertise in different areas.
* The project is well defined
* We have strict deadlines

With these factors in consideration, Hierarchal Internal Organisation will not be suitable. Having a small team, it will be difficult to elect a sufficient team leader as members of our team all have similar experience and qualifications. Electing a single team leader may result in disagreement, and runs a risk of a poor relationship between team members and team leader. To resolve this, we could elect multiple team leaders, though this runs the risk of team leaders disagreeing. What happens if the instructions given out by each team leader contradicts? Who do the team members listen to?

Also, since team leaders focus on management over development, we effectively lose man power. With tight deadlines, a focus on development should be prioritised.

Since the project is well defined, most team members have a general idea on what needs to be done and should be able to make decisions for themselves. Therefore it may not be necessary for a team leader to assign specific tasks, nor will a verification process be necessary if a team member decides that a slight change should be made. A verification process will use up precious time, that will be needed to hit our strict deadlines.

Hierarchal has its advantages, such as clear lines of authority and it allows for the team to scale up, but these are not relevant, as clear lines of authority are better suited for larger teams and the size of our team is not subject to change.

With the above factors considered, Chief Programming and Egoless are both suitable. The team size will not change, so the fact they do not scale up well will not change their effectiveness. Chief programming allows for quick decision making, ensuring we hit deadlines. Egoless allows for the reduction of uncertainty, when team other team members are unsure of a specific subject field of the project.

This is why we intend mix the two together. There will be no chief programmer for the project, but a different chief programmer for each task. Whoever feels most confident on the task, will play the role as chief programmer.

With this approach, a team member who is a confident a programmer, can take the role as chief programmer on programming related tasks. A member who is more confident on User Interface related requirements, can take the role as chief programmer on User Interface related requirements. Since the chief programmer will be managing the task they feel most confident with, they should have a clear idea on how they want the end result to turn out.

**Proposed Mechanisms for Communication**

Since we will be using elements Chief Programming approach, we will favour the opinion of whoever is Chief Programmer on the given task, as the Chief Programmer will have the most expertise. The Chief Programmer has the idea (and tools), and his/hers team members have the tools.

However, since we are also using elements of the Egoless approach, it must be considered that disagreements and conflicting opinions may occur, and we should respect that we are using elements of the Egoless approach – everyone’s opinion must be considered and all team members must be acknowledged. This means that if anyone is in complete disagreement with the Chief Programmer, we must seek a resolution. If a major disagreement arises, the following procedure should take place: All parties involved should state their opinion on the matter, with justification. Depending on the conflict, it should be resolved with compromise. If compromise fails, a majority vote should take place. This majority vote should include all members of the team, who must also state who/what they agree with and why. This must all be carried out in a professional manor. This means that anyone who feels anger, or starts to show elements of aggression, he/she will be expected to withdraw from the conversation/discussion. Not only will this give the team member a chance to calm down, but also a chance to re-think about the problem and change their approach to resolving it. It creates a chance to be better prepared for resolving it.

Smaller disagreements will be discussed and should not be ignored. The Chief Programmer should be able to confidently justify why they have made their choices or decision, to put the team member in disagreement at ease. The Chief Programmer must take some sort of role of authority for this to be effective. We do not want smaller disagreements to use up time, especially when considering our tight deadlines. We will define a ‘small disagreement’ as something that prohibits a team member for wanting to continue on with the project.

**1. C. Strategy for Team Task/Role Allocation**

When allocating roles to members of the team, we will be using the Belbin model. This model allows for 9 roles:

* A Coordinator
* Shaper
* A Plant
* A Monitor Evaluator
* One or more Implementers
* Team Workers
* Resource Investigators
* Complete-Finishers

When allocating roles we took into account team-members preferences, as well as their skills and experience on past projects, to decide which role would be best suited to that individual. We decided that whilst a team member’s preference and functional role may not match perfectly, they would not directly contradict each other. For example someone lacking in the confidence, experience, skills and focus required to co-ordinate a team would not be chosen as co-ordinator.

Each role can be described as a “tendency to behave, contribute and interrelate with others in a particular way” [2]. A role is a set of behaviours adopted by an individual at that time, with each having different responsibilities and functions to perform within the team. Allocation of the team roles themselves will be performed by analysing and combining qualitative data attained from two audit reports; a personal and observer assessment, along with the individuals perceived actual skills and past experience. The audit itself can be found at [http://student30266.bucomputing.uk/audit.xls](http://student30266.bucomputing.uk/audit.xls%20). The results of the audits as well as justification of allocation of member’s team roles can be found in the artefacts of the project, when we have actually decided roles, as this is not yet the case.

As we are a relatively small team, we did not have enough team-members to fulfil all the roles singularly, so decided to assign some roles to more than one individual. This would fill the most important roles for the kind of project we were tackling. For example we will not be using a shaper as we have a dedicated co-ordinator and some roles may change over time, should some individuals show strength in areas they are not currently working, or lack in areas they are. We will also ensure that during development we practice the following concepts:

* Encourage team members to develop existing strengths
* Give praise and recognition for team members’ skills and strengths
* Communicate within the team, each individuals skills and strengths

**1.D. Selection and justification for project management and technical methodology**

In terms of the project management methodology for this project, the team chose to go for a combination of scrum and DSDM methods of the agile methodology. Involving elements such as scrum meetings, sprints and product backlog as well as time boxing. We decided to go for scrum as the team are all familiar with the scrum process and so by using this methodology we do not have to go about spending time educating team members about a new way of working such as Prince2, moreover as we are using a egoless team organisation many members could possibly called to take up the role of project lead due to their skill set and so if they have no knowledge of the project leads responsibilities in a Prince2 process, the project could suffer delays or failure. We will use DSDM to decide how to prioritise the requirements and then break requirements down into feasible tasks to be completed through scrumming.

Scrum also gives the team members the freedom to pick their tasks which enables them choose tasks based on their strengths, rather than being assignment a task by a manager that they may not enjoy which could potentially lead to a poorer quality product. A methodology like Prince2 does not give the same level of flexibility as the agile methodologies resulting in the team decision to go for scrum over Prince2.

Furthermore scrum meetings will help the project manager/Scrum Master to measure the productivity of each team member, this will help productivity of each team member as nobody wants to be singled out by the manager for not doing their allocated task(s).

Using this method relies on the team members being committed as well as the Scrum Master trusting the team. Furthermore if a team member is ill for a duration then it could have a huge effect on the project development as allocated task are not completed on time. To counter this possible eventuality each member of our team takes on a secondary role and their sprints will be extended to give them more time to complete their both their tasks as well as their newly inherited ones.

**Technical methodology justification - Waterfall**

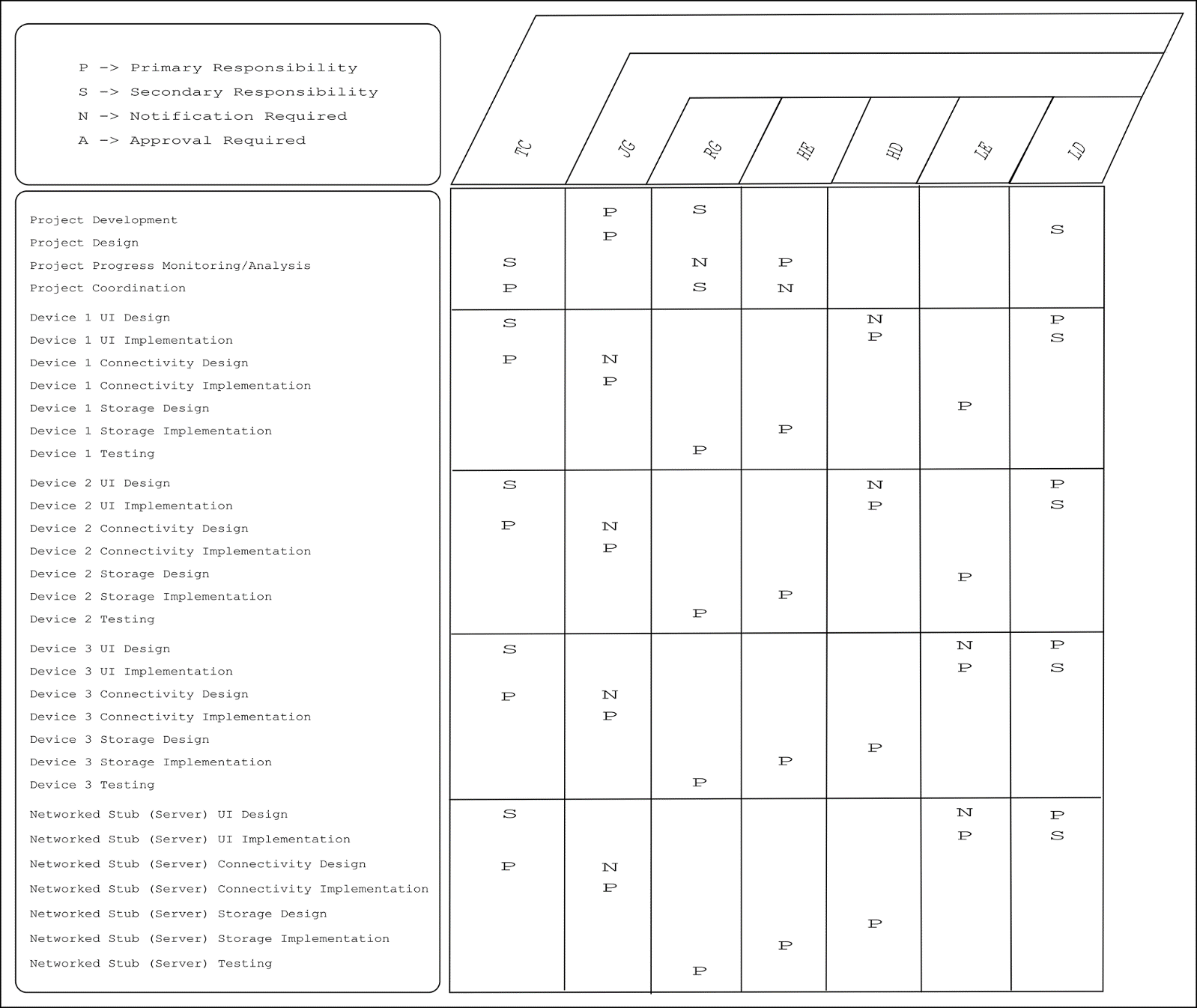
In terms of technical methodology to be implemented during this project, the team chose to use a traditional Waterfall development cycle, comprised of separate design, implementation, and testing phases, to be carried out in a linear order. Further, given that the deliverable requirements are very unlikely to change during development, an agile approach is deemed unnecessary and inefficient. This is in part because the team will be unable to glean feedback from stakeholders (in this case lecturers, whom we will be unable to gain feedback or criticism from during development), which would have been a major hindrance should the team have decided to implement an iterative or agile development methodology. This also implies that stakeholders will be unable to “drop off” new work to the team, thus eliminating the need or ability to add new items to a Scrum-style backlog. However, it is very likely that Scrum-like agile elements will be integrated into both the implementation and testing phases of deliverable development.

The justification for disqualifying a methodology such as Spiral development from our development cycle is thus: Given that the implementation phase of our development will take place over a contiguous period of roughly one week, it would be both incredibly difficult and severely unnecessary to produce a working deliverable at the end of each spiral. This is because the team would essentially be forced to either: Produce a working, iterative deliverable *at the end of each working day*, or to consider the entire work-week as one spiral iteration, produce one deliverable at the end of the iteration, and have no opportunity to iterate or improve on the deliverable, as the project will be completed, thus disallowing any iteration or improvement upon the first deliverable.

A similar set of circumstances also led the team to disqualify a V-shaped technical methodology from the development process. The justification for this is as follows: From the beginning of the project, accessing stakeholders to verify and validate requirements (the first step in a v-shaped technical methodology) will be impossible, since they are one’s lecturers, and essentially the team would be asking them to audit the production of their own assignment briefs, which would technically lie beyond the scope of the project. As a continuation of this logic, verification and validation of all other stages of implementation would be incredibly difficult (or impossible) because, again, the team will be cut off from the influence of stakeholders during development. The question essentially becomes, “How does one verify/validate our deliverables against stakeholder requirements when one has no access to the stakeholder?” As such, the V-shaped technical methodology significantly diverges from the ideal development path, thus disqualifying its use as the primary methodology. Again, however, as in the case of Scrum-like development aspects, V-shaped aspects are likely to be integrated into the overall development methodology package.

**1.E Detailed schedule**

**Responsibility Matrix**

****

**IDENTIFICATION OF CRITICAL PATH**

Project Start

Design Completion

Implementation Completion

Unit/Module/Device Testing Completion

Integration Testing/Deployment Completion

Project Completion

**IDENTIFICATION OF APPROPRIATE MILESTONES**

1. **Design Completion**

* The justification for having completion of all design work packages as a project milestone stems from the fact that it will allow the team to clearly delineate between phases of the project, in accordance with our chosen modified Waterfall technical methodology.

1. **Implementation Completion**

* The justification for having completion of all implementation work packages as a project milestone stems from the fact that it will allow the team to clearly delineate between phases of the project, in accordance with our chosen modified Waterfall technical methodology.

1. **Unit/Module/Device Testing Completion**

* The justification for having completion of all unit and module testing work packages as a project milestone stems from the fact that it will allow the team to clearly delineate between phases of the project, in accordance with our chosen modified Waterfall technical methodology.

1. **Integration Testing/Deployment Completion**

* The justification for a second milestone that includes a form of testing is that while Unit and Module Testing will allow the team to validate the functionality and performance of individual devices or aspects of the system, it will still be necessary to test all the components as a coherent whole. This could also be considered a first successful deployment of the project deliverables, and as such could be considered the final milestone.

**1.F. Risk analysis and contingency plan**

**Identification of risks and associated contingency plans**

As part of the project plan produced by the team for the PMT – Computing assignment, a risk Analysis was carried out to identify any potential hazards or hurdles that were likely to be encountered during design, development, and testing. This consisted firstly of brainstorming, simply naming as many potentially problems the team could think of, primarily from first-hand experience. Secondly, these risks were then assessed to determine whether any variants of a risk could be considered a separate risk themselves. For example, the team concluded that a single team members missing for multiple sprints, and multiple team members missing for a single sprint could (while technically variants of the same basic risk) be considered separate. In most cases, the team determined that indeed variants of originally identified risks could be considered separate risks.

Broadly speaking, the risks identified by the team largely fit into one of three categories:

1. Technical Risks, such as those associated with versioning and Version Control. For example; a risk in this category might be having the team’s online version control repository of choice crash or go offline (for whatever reason) for an arbitrary length of time during development.
2. Personnel Risks, these being risks associated with the attendance, or availability, of team members for an arbitrary period during the duration of the project.
3. Management Risks, largely during the implementation and testing phases of development itself.



The figure above indicates the frequency and severity of some the risks likely to be faced during the project as identified during brainstorming.

As such, while the project is underway preventative measures will be taken to mitigate certain risks upfront, for example the redundancy of development hardware. Further contingency measures include the use of multiple separate instances of the development repository, with latest working builds to be merged/synced at the end of each working day (mitigating risk of catastrophic project loss due to version control issues, etc.). Also, progress audits will be conducted regularly during the project, which will aid in the identification of any risks that have actualised or may do so, allowing the team to plan and deal with them effectively.

**1.G. Quality Assurance plan**

**Introduction**

The Software Quality assurance plan describes the processes, methods, standards and procedures that are to be implemented during the development of this project, to ensure quality is built into each stage of the project.

This document describes 4 main focuses and activities that when complete, will provide deliverables and details to be followed by all members of the project team (where relevant), to ensure consistency, correctness, completeness and accuracy of the project.

**Deliverables**

1. Procedures - A set of rules on how to complete specific processes

* Allocation of responsibilities; **self and peer skills audits** used to allocate Belbin team roles
* Retention and maintenance of records monitor evaluator; **project artefacts, meetings minutes**, etc.

1. Standards – Defines the rules to support policy or project specific activity’s

* Document standards; we will be following the standards used by Bournemouth University when producing academic documents to ensure consistency of content, structure, format and presentation
* Code standards; as we have not decided in which languages to write the system yet, a specific document cannot be named. We will adhere to common naming conventions such as camel case for variables, but will decide on which standard to adopt for this project when the language to use has been formally agreed by the team. (e.g. if we chose java we would consider conforming to standards supplied by oracle here <http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>)
* Design standards; **IEEE Systems Design— Software Design Descriptions**

1. In-Stage assessment audit – At least one audit of deliverables will be completed at each phase of development. An issue will be logged if there is a problem that has no clear solution at the time. Once a list of issues has been logged, they will be taken to the project manager for resolution using an action plan.

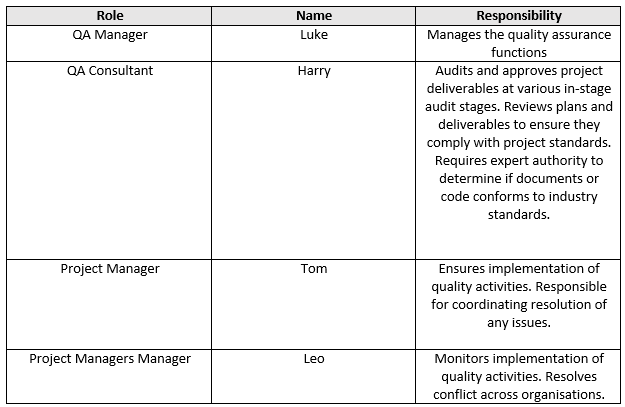
* 4 stages of assessment; schedule the assessment, receive deliverable, conduct the assessment and prepare findings.
* An assessment of risk to the schedule will also be provided by the QA consultant. Risk categories are: High, Medium and Low risk.
* The document used for audits at the “conduct the assessment” stage can be found in appendix (In-stage assessment audit)

1. Peer reviews

* At least one structured walkthrough will be competed at each lifecycle deliverable
* Any issues will be logged and listed, with improvements added to the action plan
* These documents will be included as artefacts

**Quality Assurance roles and responsibilities**

The tables below defines the roles and responsibilities of various team members, who will be responsible for which QA activity. As when organising and allocating team roles earlier in this document; if a team member is for some reason unable to fulfil their role, responsibilities may be transferred to another individual, temporarily or permanently.





Appendix



**In-Stage Assessment Audit**

Name:

Stage of development:

Date:

Reviewer Phone:

|  |  |  |
| --- | --- | --- |
| Issue # | Issues/Concerns | Resolved |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Assessment of risk to schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| Low | Medium | High |  |
|  |  |  | Next stage |
|  |  |  | Remainder of project |

**References**

[1] Mind tools editorial team. (n/a). *How Understanding Team Roles Can Improve Team Performance.* Available: [https://www.mindtools.com/pages/article/newLDR\_83.htm. Last accessed 15/03/2017](https://www.mindtools.com/pages/article/newLDR_83.htm.%20Last%20accessed%2015/03/2017).

[2] Belbin UK. (n/a). *Belbin team roles.* Available: http://www.belbin.com/about/belbin-team-roles/. Last accessed 15/03/2017.

[3] Belbin UK. (2014). *How to… Interpret and make the most of Belbin Team/Group Reports.* Available: [http://www.belbin.com/media/1169/belbin-howtointerpretandmakethemostofteam-groupreports-jan2014.pdf. Last accessed 21/03/2017](http://www.belbin.com/media/1169/belbin-howtointerpretandmakethemostofteam-groupreports-jan2014.pdf.%20Last%20accessed%2021/03/2017).

[4] Bruce Harpham. (2014). *11 Ways to Manage Project Conflict.* Available: https://www.liquidplanner.com/blog/11-ways-mange-project-conflict/. Last accessed 26/03/2017.

[5] DR. E. WALLMÜLLER Qualität und Informatik www.itq.ch, Zürich, <http://itq.ch/pdf/RM__ITProjekteV211.pdf>

[6] SOFTWARE RISK MANAGEMENT Linda Westfall The Westfall Team westfall@idt.net PMB 383, 3000 Custer Road, Suite 270, <http://www.westfallteam.com/Papers/risk_management_paper.pdf>

[7] IEEE. (2009). *IEEE Standard for Information Technology—Systems Design— Software Design Descriptions.* Available: http://cengproject.cankaya.edu.tr/SDD%20ieee%201016-2009.pdf. Last accessed 26/03/2017.