Designing a metrics model for DevOps at Philips IT

Kruis, S.

Award date:
2014
Designing a metrics model for DevOps at Philips IT

Sander Kruis
Contents

1. Introduction .................................................................................................................... 4
2. Research model ............................................................................................................... 5
3. Theoretical background .................................................................................................. 6
   Agile Software Development ......................................................................................... 6
   DevOps ......................................................................................................................... 7
   Lean Software Development ......................................................................................... 8
   Metrics ......................................................................................................................... 9
4. Methodology ................................................................................................................. 14
5. Current situation (As-Is) ............................................................................................ 17
   IT Delivery ................................................................................................................... 17
   Infrastructure & Operations ......................................................................................... 18
   Linking IT Delivery and I&O ..................................................................................... 19
   Processes ..................................................................................................................... 19
   Other observations ....................................................................................................... 21
6. Design ............................................................................................................................ 23
   Goal definition ............................................................................................................. 23
   Creating a reference framework .................................................................................. 24
   First design .................................................................................................................. 24
   First feedback round and second design ..................................................................... 27
   Second feedback round and third design ..................................................................... 30
7. Discussion ...................................................................................................................... 32
8. Conclusion ....................................................................................................................... 33
References ......................................................................................................................... 34
Appendix A ........................................................................................................................ 37
   Original project description ....................................................................................... 37
Appendix B ........................................................................................................................ 39
   Philips Integrated Landscape: Information Management ............................................. 39
Appendix C ........................................................................................................................ 44
   VIPP .............................................................................................................................. 44
Appendix D ........................................................................................................................ 49
   Lean measurements (Duque & Cadavid, 2007) ......................................................... 49
1. Introduction

Philips IT (Appendix A) is a centralized IT organization servicing three sectors of Philips; Healthcare, Lighting and Consumer Lifestyle. Within IT there exist two large parties. The first is IT Delivery where projects are executed and developed. The second, IT Infrastructure & Operations (I&O) is responsible for daily operations. Daily operations include keeping the IT systems running and providing direct support. IT Delivery has been adopting SCRUM to become more agile with over 100 SCRUM teams. These teams are multidisciplinary and include relevant partners. These partners are located across the world, thus these teams work together in a virtual way.

While IT Delivery is adopting agile methodologies, I&O is working according to an Information Technology Infrastructure Library (ITIL) framework. While the organization is constantly changing, I&O is looking for ways to become more agile with several teams adopting methods like SCRUM or Kanban.

Looking at the two parties, we initially see that IT Delivery is pressing for faster software releases as SCRUM cycles are currently two weeks long. From operations however, system stability is of the highest importance and releases are planned monthly.

IT delivery and I&O are now working together to improve this situation and release more often without compromising on the quality of these releases. Since 2009 (DevOps, 2014), a method called DevOps was introduced that focuses on the way Development and Operations work together. This method originates from lean methodologies and stresses the improvement of flow, communication, feedback and automation. Flow describes how work moves through the process. For example, increasing the speed or decreasing the waiting times of work increases the flow.

Philips IT is looking to implement DevOps strategies, starting with two teams within IT delivery. On their way to implement DevOps it becomes important to validate they are actually doing DevOps and how this implementation process is affecting the organization. For this purpose a set of metrics will be developed to measure whether DevOps is being achieved and how this can be seen within the organization.

To develop these metrics there are a few questions that need to be answered with the main research question being; How can DevOps be measured? To come to a solution for this question, we need several other parts. There are two questions that immediately come up. The first being, what is DevOps supposed to be? And secondly what has Philips IT done so far to implement a DevOps solution?

If we have an understanding of DevOps, and we know how Philips IT is currently working, we can work towards a design to measure DevOps within Philips IT. After that we can follow a PDCA cycle to adapt and improve that design.
2. Research model

Looking at the research questions we can see the path depicted in Figure 1.

![Research model diagram]

This path starts with background research on software development techniques and metrics related to these areas. From this theory some insights are gained from the different methods available. These methods will include DevOps and SCRUM, as well as look at lean development. This is followed by research on available metrics and how metrics can be designed. From this research a conceptual model can be created, which depicts what a DevOps metrics model would entail. Looking at this conceptual model as well as the current situation within Philips IT an initial model is designed, which is validated and adapted several times in a single case-study. Designing the metrics model is done together with people involved in the process in a form of action research.
3. Theoretical background

In this chapter several software development techniques are described to provide insight in the terminology and provide a path towards a conceptual model. The first subject to look into is agile software development, that being the type of methods that IT Delivery is currently working with as well as what IT Operations is currently looking to implement. After that, a closer look at DevOps is provided, corresponding with the goal of the research. From DevOps, some more research follows corresponding to what DevOps is about, leading into a description of lean software development. This is followed by a description of available metrics that are being used in agile and lean areas to provide a reference framework while designing the final metrics model. Finally, some guidelines are described for ways to design metrics.

Agile Software Development

The term Agile Software Development originated from the ‘The Agile Manifesto’ (Beck, et al., 2001) and consists of several values and principles for faster and better software development. The four values they describe are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

While there is not a single definition of agility, most incorporate the idea of adaptability to the environment and quick value creation. One of such definitions is given by Erickson et al. (2005); “agility means to strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like.”

While this definition is focused on software development, similar trends have been previously seen in other disciplines. Conboy and Fitzgerald (2004) for example related agility to “flexibility” and “leaness” as seen in agile manufacturing and lean development. However, several differences exist between the terms. According to Zhang & Sharifi (2000) agility consists of two components, these being flexibility and speed. Hereby stating that flexibility alone is not enough to be agile. Comparing agility to leaness, complements each other with regard to simplicity and quality, but approaches economy differently (Young, Muchlhaeusser, Pigging, & Rachitrangsan, 2001). While leaness attempts to remove ‘waste’ entirely, agility removes waste only to the extent that it does not hinder the ability to change, or be flexible.

While the above definitions describe the values for agile software development, it does not provide a direct method to apply these values. While a multitude of methods have been developed, a few of these have become more prominent and are further described below in Table 1.
Table 1 Agile methods

<table>
<thead>
<tr>
<th>Agile method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Systems Development Method (DSDM) (Stapleton, 2003)</td>
<td>Projects are divided into three phases; pre-project, project life-cycle, and post project and has nine principles for development being: user involvement, empowering the project team, frequent delivery, addressing current business needs, iterative and incremental development, allow for reversing changes, high-level scope being fixed before project starts, testing throughout the lifecycle, and efficient and effective communication.</td>
</tr>
<tr>
<td>Scrum (Schwaber &amp; Beedle, 2001)</td>
<td>The development is organized in sprints (short iterations of about 2 to 3 weeks) by a self-organizing team. Each sprint goes through planning, design, testing and review. Features that need to be developed are stored in a ‘Backlog’ where the product owner decides the items that will be worked on in the following sprint.</td>
</tr>
<tr>
<td>Extreme Programming (XP) (Beck &amp; Andres, Extreme programming explained: Embrace change, 2000)</td>
<td>Focuses on best practice and consists of twelve practices: the planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40h week, on-site customers, and coding standards.</td>
</tr>
<tr>
<td>Lean software development (Poppendieck &amp; Poppendieck, 2003)</td>
<td>Adapted from lean development and based on seven principles: remove waste, amplify learning, decide as late as possible, deliver as fast as possible, empowered teams, build integrity, and see the whole picture.</td>
</tr>
<tr>
<td>Kanban (Anderson, 2010)</td>
<td>Kanban is based on the theory of constraints (Goldratt &amp; Cox, 1985) and comes with six core practices; visualize, limit work in progress (WIP), manage flow, make policies explicit, implement feedback loops, improve collaboratively &amp; evolve experimentally.</td>
</tr>
</tbody>
</table>

DevOps

While development and operations departments have different goals or key performance indicators, DevOps attempt to satisfy both goals. As development wants to deploy more and more often, Operations might want the exact opposite as in order to keep all systems running and stable it might be more beneficial to not deploy anything new at all. As organizations have seen the need for agility, the option to not deploy anything is not realistic at all, as customers most likely want new products or features on a regular basis.

To achieve this agility, not just development should be agile, but the entire organization. DevOps attempt to do this with three core ideas (Kim, Behr, & Spafford, The phoenix project: A novel about IT, DevOps, and helping your business win, 2013) (Kim, 2013):

Table 2 The Three Ways of DevOps

| 1. Systems thinking | Stresses that it is more beneficial to look at the performance of the entire system, than the performance of specific parts of that system. |
| 2. Amplify feedback loops | Allows understanding of the customer and availability of knowledge where it is needed. |
| 3. Culture of continual experimentation and learning | Experimentation and learning helps you in the long-term by being able to more quickly adapt and respond to changes or problems. |
Systems thinking refers to an approach that looks at problems in relation to the entire system it exists in (Ackoff, 1971). The approach ensures that the performance of the system as a whole is more important than the performance of separate parts of the system (IT Delivery or I&O). Like in lean development the performance described here is the performance related to the value of the system. Implementing the first way leads to a few outcomes:

- Never passing a defect to following work centers
- Do not allow local optimization if it creates global degradation
- Always look to increase flow

Amplifying feedback loops leads to early knowledge of issues and problems, so the system can quickly adjust where needed. Implementing the second way leads to the following outcomes:

- Understanding and responding to all customers
- Shortening and amplifying feedback loops
- Knowledge available where needed

The third way mostly supports the other two ‘ways’ to ensure that improvement is a continuous process and it leads to:

- Making time for the improvement of daily work
- Reward teams for taking risks
- Introducing faults to increase resilience

**Lean Software Development**

Leading from systems thinking, as a method that looks at the entire value stream, Lean software development is the application of lean principles and methods to the software development domain. Lean focuses on the creation of value and the elimination of waste in the production process and was first introduced in “The machine that changed the world: the story of lean production” (Womack, Jones, & Roos, 1990). Some changes were needed to translate from production to software development however. As Jailia et al. (2011) described lean software development is based on five principles:

- Specify value
- Identify the value stream
- Create flow
- Customer pull
- Pursue perfection through continuous improvement

In order to identify the value stream and create flow, lean focuses on eliminating waste. For software development projects seven types of waste have been identified (Ikonen, Kettunen, Oza, & Abrahamsson, 2010) (Poppendieck & Poppendieck, 2003):

- Partially done work
- Extra processes
- Extra features
- Task switching
- Waiting
- Motion
- Defects

**Metrics**

The next paragraphs describe metrics that have been used in areas related to DevOps. Starting with measurements in lean manufacturing and scrum, which is mostly related to systems thinking and slightly to feedback loops. Followed by a description of culture of learning and finally about how metrics can be designed.

The first of these areas is lean manufacturing. Duque & Cadavid (2007) gathered a collection of metrics that could reflect the advancement in the implementation of lean manufacturing. They used a collection of case studies and gathered all metrics used in these cases. While not every metric is as useful as the next, it does provide a good overview of what metrics are being used in practise. These metrics are divided into five dimensions, in a similar fashion to earlier research by Karlsson & Ahlstrom (1996) and Martinez & Perez (2001), but adapted for the goals of Toyota Motor company, for whom Duque & Cadavid (2007) performed their research. The five dimensions that were used are; Elimination of waste, Continuous improvement, Continuous flow and pull driven systems, Multifunctional teams, and Information systems. Below is a description of the metrics they found for these dimensions. An additional overview of these metrics can be found in Appendix D.

For the first dimension of ‘elimination of waste’, Duque & Cadavid (2007) describe metrics related to what kind of activities time is being spent on within the process. This describes activities like Setup time, Downtime and Transportation. Along with these time allocations, they mention Space utilization and Work In Progress (WIP). The last two metrics named are Work in Progress (WIP) and space utilization. Although WIP in IT is usually not something tangible as WIP in a development and deployment process mainly consists of lines of code some form of WIP could still be used. For example, scrum (Schwaber & Beedle, 2001) and many other agile methodologies refer to work as user stories, which are small increments of work that in itself have some value in the final product. The amount of user stories and their relative size could provide a good overview of the size of WIP.

For the second dimension, ‘continuous improvement’, several metrics are again described. These metrics are related to either suggestions made by employees and how many of these get implemented or the amount of rework that has to be performed. For the amount of rework two metrics are mentioned. The first is the actual percentage of units that requires rework. The second refers to ‘scrap’, which they define as the percentage of products that need to be scrapped, thus cannot be reworked properly and are discarded.

For the third dimension, ‘continuous flow and pull-driven systems’, there are again several metrics described. The first metric is the lot size. Translating lot size to IT development could refer to the number of user stories that go through the system simultaneously, so for example the average amount
of user stories per sprint or the amount that simultaneously get deployed. The second and third metric refer to the flow time and lead time of an order. The lead time refers to the average time from the creation of an order to the delivery of the order. The flow time on the other hand refers to the time it is being processed. This thus does not include waiting or setup times. The last metric for this dimension deals with measuring the pull-driven system, which is measured by the percentage of processes that pull inputs from their predecessors and the value of their throughput. This would mean that teams that are located early in the process do not simply work up to their capacity, but only as much as needed to properly supply the following teams.

The next dimension deals with ‘multifunctional teams’. This relates to the flexibility of the team to do each other’s tasks and how much responsibility they take for the product they deliver. The first metrics for this dimension are the percentage of quality inspection the team performs themselves and the percentage of tasks that are performed within the team as opposed to other teams. These together describe how much influence the team itself has on the product that is delivered and how much the team depends on the work of other teams. Other metrics that are mentioned are ‘Cross training’ and ‘Number of employees capable of assignment rotation’ (Duque & Cadavid, 2007). These metrics describe how well the team members can perform each other’s tasks as well as how dependent the team is on each member being present.

For the last dimension, ‘information systems’, the metrics relate to the availability of information within the team and the documentation of procedures. The mentioned metrics however do not deal with the quality of the provided information. The impact of the quality could of course change the measurements done in the other categories. For example, if a team member would not have access to necessary information, this would directly influence the flow time and would be addressed in the daily meetings that are already in place. The metrics named are “the frequency with which information is given to employees”, “percentage of procedures documented” and “the frequency with which the progress boards are updated.” These progress boards refer to kanban-like boards that display the work in progress.

Other metrics related to software development can be found within scrum research. One such research by Downey & Sutherland (2013) describes ten metrics to promote improvements. These metrics are described below in Table 3. While these metrics seem very different to the metrics described by Duque & Cadavid (2007), the reason for that is that the object they measure is different. While the metrics by Duque & Cadavid (2007) look at the entire value chain according to lean principles, the metrics by Downey & Sutherland (2013) are used to measure the performance of scrum teams on their own. As scrum teams usually perform the development before work moves to operations, these metrics thus measure only a part of the value chain.
### Table 3: Ten essential metrics (Downey & Sutherland, 2013)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>The sum of the estimates of all accepted work.</td>
</tr>
<tr>
<td>Work Capacity</td>
<td>The sum of all work reported during a sprint.</td>
</tr>
<tr>
<td>Focus Factor</td>
<td>Velocity divided by Work Capacity, described how much time was spent on planned work.</td>
</tr>
<tr>
<td>Percentage of adopted work</td>
<td>Describes how much extra work the team has done. (Adopted work divided by original planned forecast)</td>
</tr>
<tr>
<td>Percentage of found work</td>
<td>Describes how much extra work the original work has cost. (Found work divided by original planned forecast)</td>
</tr>
<tr>
<td>Accuracy of estimation</td>
<td>Describes how well the forecast was estimated, thus how much time work took in contrast to the time planned to do it.</td>
</tr>
<tr>
<td>Accuracy of forecast</td>
<td>Describes in how far the forecasted work was equal to the actual work done.</td>
</tr>
<tr>
<td>Targeted Value Increase</td>
<td>Current sprint’s Velocity divided by original Velocity.</td>
</tr>
<tr>
<td>Success at scale</td>
<td>Accepted attempts to scale divided by the total attempts to scale.</td>
</tr>
<tr>
<td>Win/Loss record</td>
<td>Each sprint is a success if:</td>
</tr>
<tr>
<td></td>
<td>• A minimal of 80% of original forecast is accepted.</td>
</tr>
<tr>
<td></td>
<td>• Found plus adopted work remains at 20% or less of the original forecast</td>
</tr>
</tbody>
</table>

Metrics for the third way, a culture of continual experimentation and learning, are lightly found within the second dimension of ‘continuous improvement’. However, some further research can be performed here. Research by López, Peón and Ordás (2004) provides a ‘definition’ of organizational learning. While not directly providing metrics, this definition helps the reference framework by better understanding what organizational learning is composed of. Their research defines four different constructs of organizational learning.

- Acquisition of knowledge, either through external sources or internal development
- Distribution of knowledge
- Interpretation of knowledge or how the organization shares and uses knowledge
- Organizational memory or how the organization stores the knowledge

To design metrics, Pfleeger (1993) describes some lessons learned in developing a metrics program at a telecommunications firm called Contel. The main lesson there is that “Metrics are welcome only when they are clearly needed and easy to collect and understand.” Besides this general statement Pfleeger describes ten lessons, which can be seen below in Table 4.
Table 4 Pfleeger’s lessons learned (Pfleeger, 1993)

<table>
<thead>
<tr>
<th>Lesson</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin with the process</td>
<td>The process helps finding problems and identifying what the metrics should</td>
</tr>
<tr>
<td></td>
<td>be used for.</td>
</tr>
<tr>
<td>Keep the metrics close to the developers</td>
<td>Develop the metrics with the people who will work with them. This leads to</td>
</tr>
<tr>
<td></td>
<td>better understanding.</td>
</tr>
<tr>
<td>Start with people who need help; then let</td>
<td>Start with some success stories, which will spread themselves. Thus makes</td>
</tr>
<tr>
<td>them do your advertising for you</td>
<td>other teams follow more quickly.</td>
</tr>
<tr>
<td>Automate as much as possible</td>
<td>Minimize the time needed to collect data as it is more useful to spend</td>
</tr>
<tr>
<td></td>
<td>time analyzing the data than collecting it.</td>
</tr>
<tr>
<td>Keep things simple and easy to understand</td>
<td>Keep the metrics close to the problem and do not create too many metrics.</td>
</tr>
<tr>
<td>Capture whatever you can without burdening</td>
<td>Capture as much as possible without interfering with general work. Some</td>
</tr>
<tr>
<td>developers</td>
<td>things might be useful for other problems than the one you are currently</td>
</tr>
<tr>
<td></td>
<td>solving.</td>
</tr>
<tr>
<td>If the developers don’t want to, don’t make</td>
<td>The quality of the resulting data might not be very good otherwise.</td>
</tr>
<tr>
<td>them</td>
<td></td>
</tr>
<tr>
<td>Using some metrics is better than using no</td>
<td>The general idea is to start with success stories. Measuring one or two</td>
</tr>
<tr>
<td>metrics</td>
<td>useful things might convince people other metrics could be worthwhile.</td>
</tr>
<tr>
<td>Use different strokes for different folks</td>
<td>Keep in mind the products, environments, domains, goals and customers.</td>
</tr>
<tr>
<td>Criticize the process and the product, not</td>
<td>Focus the metrics on the process and/or product.</td>
</tr>
<tr>
<td>the people</td>
<td></td>
</tr>
</tbody>
</table>

While these lessons provide some guidelines in how to proceed while designing metrics, they do not provide a structured way to actually design them. For that the Goal-Question-Metric (GQM) approach (Van Solingen & Berghout, 1999) can be used. This approach defines three levels for a measurement model (Basili, Caldiera, & Rombach, Goal Question Metric Paradigm, 1994), namely; conceptual (goal), operational (question), and quantitative (metric). Starting from a clear definition of the goal, followed by questions that would describe the goal, metrics can be designed to measure the goals.

In closing, the background research has provided some insight in what DevOps is about, followed by an overview of metrics that could provide useful for the design of a metrics model for Philips. Also some guidelines for designing a metrics model have been shown, which will help describing the methodology below. A summary can be found below in Table 5.
<table>
<thead>
<tr>
<th>Subject</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DevOps</td>
<td>• Systems thinking</td>
</tr>
<tr>
<td></td>
<td>• Amplifying feedback loops</td>
</tr>
<tr>
<td></td>
<td>• Culture of learning and experimentation</td>
</tr>
<tr>
<td>Metrics</td>
<td>• Lean manufacturing</td>
</tr>
<tr>
<td></td>
<td>• Scrum</td>
</tr>
<tr>
<td>Designing metrics</td>
<td>• Begin with the process</td>
</tr>
<tr>
<td></td>
<td>• Keep the metrics close to the developers</td>
</tr>
<tr>
<td></td>
<td>• Keep things simple and easy to understand</td>
</tr>
<tr>
<td></td>
<td>• Goal-Question-Metric approach</td>
</tr>
</tbody>
</table>
4. Methodology

This chapter describes the methodology used in this research.

As some background knowledge on related topics is now gathered, DevOps can be described in terms of a conceptual model. This model depicts what factors DevOps is composed off. From research in Chapter 3 we see DevOps is composed of three ‘ways’, namely; systems thinking, feedback loops and culture of learning & experimentation. These three can thus be seen in the conceptual model (Figure 2). When we can measure these three ‘ways’ through the metrics model to be designed in this research we can see the performance of DevOps within Philips IT. After that, the model could lead to new insights to adapt the model.

![Figure 2 DevOps conceptual model](image)

To design this metrics model, some steps will have to be taken. This starts with getting a better understanding of the environment the model will be used in. This means a description of the current situation within Philips IT is needed. To do this properly, a general introduction of the environment is provided first. After which follows a more in-depth look into how the three ‘ways’ can currently be seen within Philips IT. For this, two teams within Philips are available to gain this understanding. To provide more understanding of especially systems thinking and feedback loops as well as following Pfleeger (1993), the process these teams follow is mapped and compared. For the third way, culture of learning & experimentation, a more general impression suffices, by describing the way the teams perform daily activities.

Following the analysis of the current situation, a metrics model can be designed. Because of technical difficulties, it was not possible to get data from the teams about designed metrics or metrics that were already in place. For this reason, the model is designed together with relevant parties in the form of...
action research (McNiff, 2013), where a solution is designed by (or with) those working in the environment itself. Also, this ensures the metrics model will be better understood and closer to those working with it afterwards along the lessons learned by Pfleeger (1993).

The goal of this research is to investigate the situation within Philips IT. Therefore a set of metrics will be designed that together determine how well the Philips IT is doing in terms of DevOps. In order to come to such a design, several steps have to be taken along the lines of the Goal-Question-Metric approach (Van Solingen & Berghout, 1999). First the goal has to be further specified. This would describe what we need metrics for exactly and provides a better insight in which way the design is moving. From there a reference framework should be created. This is primarily done to provide some insight in what is already present in today’s literature to further understand the goal. Following the literature review, a first design can be created. To ensure the design is useful, feedback is gathered within Philips after which this feedback is processed and the metrics are updated. During this feedback process the design is constantly being validated.

![Figure 3 Design methodology](image)

Figure 3 shows the steps taken in this research. The following paragraphs will describe how the different steps are performed in more detail.

While determining the goal and designing the metrics are two separate steps, the Goal-Question-Metric (GQM) approach (Van Solingen & Berghout, 1999) can be used in both these steps. This approach is meant to discover which metrics are suitable for your problem by looking at the goal that the metrics are supposed to be used for. The first part of this approach solely looks at the definition of the goal, which is what is needed here. To help defining these goals a template is used. One such template is suggested by Basili, Caldiera, & Rombach (1994) and is shown below in Table 6.
To create a reference framework, literature research was done to provide insight in what metrics are currently available and used for related or similar goals. This research is focused on metrics that are used for DevOps or similar situations. As metrics for DevOps were not readily available, a look into lean manufacturing and scrum metrics was taken. While GQM technically states that the metrics should be designed with just the goal in mind, those using GQM generally possess a reference framework of the sort, due to them working in the particular environment. This step is there to cope with the lack of such a reference framework for me personally. Also, it could be interesting to see whether following GQM leads to similar metrics as the literature research suggests, or whether these will be different.

After a reference framework has been created, GQM can be further used to design a set of metrics according to the already defined goal(s). The next step in the GQM process is to devise questions for the goal(s). These questions should refine the goal to make it easier to approach. Good questions will also tell you whether the goal is reached. To answer these metrics one or more metrics are suggested which together form a metrics design.

For the feedback loop, several relevant parties are asked to provide feedback proposed metrics. The feedback collected resembles an embedded case study design (Yin, 1989), where several parties in the context of the case study are studied. This feedback is provided in an iterative manner, where feedback provided is processed into the design, after which new feedback can be given. This ensures the design will be more valuable later on, and those working with it have a better understanding of the design and how it was created. Depending on the feedback this feedback loop can be applied multiple times. While this could technically always continue, a stop-criteria is needed. As feedback is processed in each loop, the involved parties should come to a form of agreement in a few cycles. Therefore whenever the design is not significantly changed from its previous iteration, the cycle will be stopped.

Due to the process of designing the metrics model, a separate validation is not performed. Because of the repeated sessions to gather feedback, the model is constantly being validated.
5. Current situation (As-Is)

As described in the introduction, Philips IT is divided in two large parties, namely Delivery and I&O. In order to work together it is important to realize the places in which these parties have common ground as well as where the differences lie. In order to gain insight into this, both parties are described in the next paragraphs. After a description of these two parties, the process in which they interact is described, followed by some final observations.

**IT Delivery**

Philips IT Delivery is the party responsible for creating and acquiring IT solutions. The solutions are then passed on to I&O for maintenance and support. As Philips has a wide array of IT services, IT Delivery is divided into several delivery centers based on the end to end value chain (Philips IT, 2013), namely:

- Idea to Market (I2M)
- Market to Order – Digital (M2O – Digital)
- Market to Order – Customer Relationship Management – CRM (M2O – CRM)
- Order to Cash (O2C)
- Information Management (IM)
- Integration
- Enabling Processes (Human relations, Finance, Procurement)
- Common Infrastructure & Operations

Figure 4 describes shows these eight delivery centers to provide some background on where they exist in the end to end (E2E) process.

![Figure 4 Philips Integrated Landscape E2E process](image)

IT Delivery has been adopting Scrum to become more agile, along the lines of the Scaled Agile Framework (SAFe) (Leffinwell, LLC., 2014). This framework described three layers. The top layer is called the portfolio layer, which describes the alignment to the business and mainly consists of epics. These epics are large business needs or goals. For IT there are two different types of epics specified, namely;
Business epics and Architectural epics. The first, business epics are customer focused and deliver direct value to the customer and thus the business. In contrast, the architectural epics are more supporting needs that ensure the ability to develop and support future needs.

The middle layer is called the program layer, which describes the alignment of agile teams that each deliver parts of the above epics. These epics are further specified here into features that can be developed separately.

The bottom layers, the team layer, describes the workings of the actual development teams. The features are described further into user stories. These user stories are work items that can be completed within a specified sprint (which is usually two weeks). The teams work on these user stories and complete an amount of these within each sprint, so that at the end of each sprint some working software/development can be delivered.

The scrum teams within Philips IT are Multidisciplinary teams (MDTs). This means that there is regular contact with every stakeholder of the project they are working on. For example, such a team consists of developers, testers, a product owner, a relevant partner. Also, there is an Infrastructure and Operations Manager (IOM) within these teams, who exists as a link between these MDTs and IT Operations. These teams hold short daily meetings to discuss the current status and issues of their current project and the way they perform their work.

Infrastructure & Operations
While IT Delivery is responsible for developing new solutions, I&O is responsible for maintaining the existing systems as well as providing direct support to customers.

While process-wise work flows from Delivery to Operations and it would be beneficial to work together, historically these two parties are quite far apart. With work figuratively being thrown over a wall to Operations by Delivery. While this does not directly seem to be the case within Philips, with the IOMs within the delivery teams, there is still a clear gap between how both parties work. On the one hand there is IT Delivery working according to SAFe, thus attempting to work agile, while on the other hand I&O works according to ITIL (ITIL 2011 - The big picture, 2011). This way of working is more structured and based on following clear processes than working in an agile way.

From ITIL there are several categories of work that exist within IT Operations. After initial meetings with people working within I&O these categories have also been clearly visible.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description (Nissen, Christian Feldbech; CFN People, 2010)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incident Management</td>
<td>The process to restore service to users in case of an incident, error or outage.</td>
</tr>
<tr>
<td>Service Requests</td>
<td>The process of responding to and managing user requests, for example password resets.</td>
</tr>
<tr>
<td>Change Management</td>
<td>The process of applying changes to software or hardware.</td>
</tr>
</tbody>
</table>
## Problem Management

The process of analyzing the root cause of incidents or errors and for attempting to determine where future incidents might happen.

## Release and deployment Management

The process of planning releases to environments and ensure that the live service remains stable during those releases.

While I&O is still working in this way, there are initiatives within Operations to become more agile. This initiative is called AgileOps within Philips and is setting up teams similar to the MDTs within IT Delivery. By looking at the above types of work several are more suitable for an agile way of working. These types of work are mainly the problem management and the release and deployment management. The other types of work are tasks that are in general very small and tasks that are unplanned. The two types that are suitable are types that can be planned and worked on at any time.

### Linking IT Delivery and I&O

While both parties have different types of work and their way of working is quite different, these two parties do have to work together in some areas. Probably all parts could benefit from working together in some way to make sure IT Delivery accounts for issues Operations might encounter and Operations delivers feedback when something does go wrong. However, there are parts that have a more direct link than others. Traditional IT development depended on a method known as waterfall (Royce, 1970). This method describes several steps for the development of software. In short, these consist of requirements, analysis, design, development, testing and deployment. As scrum teams take care of all these but the deployment into operations, this is where the most important link between IT Delivery and I&O exists. Also, as DevOps focuses on quickly delivering and deploying IT solutions, this is the most obvious focus to take.

### Processes

To better understand the process of deployments, two teams were available. These teams, respectively Philips Integrated Landscape: Information Management (PIL IM) and VIPP, are quite different. The VIPP team is a settled team which focuses on development for legacy systems, while the PIL IM team is relatively new and focuses on development for the new landscape (PIL).

Going from lean perspectives described in Chapter 3, the first step was to determine the value to be delivered by these teams. Speaking to these teams, the value they deliver comes down to the service or product that has to be deployed to the respective platform. While this is the initial value, the system will also be supported and maintained, thus apart from the deployment and the value to the end-user, there is also an internal customer, namely I&O. In the end, I&O is also delivering value in terms of support to the end-user.

Both teams follow a process to deliver this value (Appendix B & Appendix C). This process however is different for both teams. While the teams were working in these processes, there was no clear view of this process and it therefore had to be created.
The processes were created over several review sessions where the process was discussed step by step with those working within the teams. The feedback was then processed and reviewed again in the following session. This continued until a point where the process was agreed upon and could provide the members with a clear understanding of how the process works and who are involved. This does mean the process is a perfect representation, as the goal here was to create an understanding of the process and not to for example implement this process into a workflow system.

The process for the PIL-IM team is shown in Figure 5 and more closely in Appendix B. The process starts where a feature is placed on the team’s backlog for a new Information Factory (IF). Before development can begin, some preparation has to be done, where models and specifications are created. When this has been created and approved the actual IF is requested. This practically means that if the request is then approved the development can start. From there a partner of Philips, in this case Cognizant, starts development and deploys the IF on the Dev environment. Here the IF will undergo unit testing before it can be transported to the QA environment. In this environment a system integration test (SIT) will be performed as well as a user acceptance test (UAT). From there some preparation has to be done to prepare for the release of the IF as well as training and knowledge transfer to support teams. Finally the IF can be deployed to the production environment, after which some final adjustment have to be made, as well as loading the appropriate data. For some time the Delivery team stays responsible for early life support before the final handover to I&O.

Figure 5 PIL-IM E2E process for the creation of an Information Factory (IF)

The process (Figure 6 & Appendix C) of the VIPP-HC team works similarly. It starts with a feature on the team’s backlog. For this feature several user stories are created and some preparation is done before development. When a user story is then picked up in a sprint, it gets developed and unit testing is done. From there a SIT is performed on the Dev environment. If these tests are approved the development is transported to QA and another SIT is performed followed by the UAT. If these are approved a final regression test is done and preparation for deployment to the production environment starts. When this has been done the deployment has to be approved and is then performed. Some checks in production are done for the integrity of the system and data is loaded. Then when the release is accepted there is the final handover to I&O.

Figure 6 VIPP-HC E2E process
While the processes of both of these teams are different there are some clear similarities between them as well. Using these two processes a general flow can be deduced that both processes follow and shows us in general steps what happens within these processes. Figure 7 shows this more general flow which starts with a feature on a backlog that needs to be developed. When this item is picked up there is some preparation that needs to be done, referring to the different specifications that need to be determined. From there it is passed to the delivery team which does some additional preparation before actual development begins. In the VIPP team, this first preparation is also done by the delivery team however. After this has been done, the actual development begins. Development can take place within the team or with an external party. After development, some testing is done, which includes at least a SIT and a UAT. When these tests have been accepted the service or product can be deployed in the next release, after which it can be supported by I&O.

![Figure 7 Delivery flow](image)

To further describe the status of the processes, some data should be collected, relating to lean measurements. That way bottlenecks and issues can be identified to provide a direction for a possible redesign. Within Philips most user stories are kept within a system called Rally (Rally Software Development Corp., 2014). However, due to technical difficulties, it was not possible to get this data.

The creation of these processes however, was a clear step towards systems thinking, as by understanding the entire process it becomes more likely actions can be taken to improve the entire process, opposed to just parts of the process. For feedback loops the amount of approval steps can be gathered from the process itself. However, without data about process times, this in itself does not provide much information.

**Other observations**

Besides the way both parties work as well as the processes they work with, there are a few other observations that can be made. While these are not directly linked to DevOps, an impression about the culture can be gained from these observations.

The first of these can be seen in the way sprints are handled. What scrum tells us is that work is divided up into small user stories that each have value to the user or customer. Each of these user stories can then be picked up within a two week sprint. Within this sprint the user story goes through the full design cycle, meaning it will be dealt with in a small waterfall-like manner. Thus, a design is made, it is developed, tested and preparations for deployment are made. However, what can be seen in the processes is quite a linear flow. As well as from some indications by team members is that what is done within a sprint is mainly the development and perhaps some unit testing. Besides this, it can also be seen in the planning of sprints. This planning usually starts with a sprint that is dedicated to the actual
planning of the project. The next sprint will then deal with designing the solution, after which one of more sprints are planned for the development. In case there are more sprints for this we can see something that looks like actual scrum. Then after development there is a sprint for testing and then finally there is a deployment. The only part of this that is actually scrum is perhaps the actual development part. For the rest it looks exactly the same as a traditional waterfall method, with the exception that every two weeks there is a report about what has happened in the previous two weeks.

The second observation relates to the PIL-IM team. At this time, the actual generation of the Information Factory (IF) is done by an outside party, Cognizant. The agreement with the partner is that whenever a request is made by Philips, cognizant has two weeks (sprint length) to deliver on this request. This two week period holds for both small changes as well as an entirely new IF. What happens here is that leadership encourages that the specification that is sent to Cognizant is 100 percent correct (one-time-right). However, DevOps encourages a culture of experimentation and learning. This is however conflicting with the one-time-right attitude. This approach makes sense when every change takes two weeks, after which it is tested within Philips, and has to go through another two week cycle if something is wrong. However, should this agreement be based on the actual amount of work needed for the change, thus a more agile agreement, the focus does not have to be on one-time-right. This could then considerably speed up the preparation phase as well as the development phase.

From the situation and observations described in the previous paragraphs, there are definitely things that show DevOps thinking, however there are also observations that in a way contradict the message of DevOps. There are three areas in which there could be some clear improvements. These areas are:

- **Scrum and waterfall**, whether real scrum actually exists or would be useful
  - This mainly relates to the fact that the content of sprints is still planned like it is waterfall
- **Approach on agreements with partners**
  - While Philips is on its way to work in an agile manner, the most benefit can be gained there if the partners are supportive on that. The agreements with partners should take this agile way of working into account.
- **Metrics on DevOps**
  - While systems thinking seems to already be more or less present in the teams I saw, it would be beneficial for management and the teams itself to be able to see how they are doing.
6. Design

As described in the methodology there are several steps to take to come to a final design. This chapter starts with the definition of the goal(s). Following the goal(s) is an overview of research on possible metrics to create a reference. From there the Goal-Question-Metric (GQM) method is followed to determine which metrics are needed and most valuable for the purpose of DevOps. Finally some feedback rounds are described where the metrics model is updated and validated.

Goal definition

The first step of the GQM process as well as the creation of the design is to define the measurement goals. To help defining these goals a template (Table 8) by Basili, Caldiera, & Rombach (1994) is used.

Table 8 GQM goal definition template (Basili, Caldiera, & Rombach, Goal Question Metric Paradigm, 1994)

<table>
<thead>
<tr>
<th>Analyze</th>
<th>The object under measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>For the purpose of</td>
<td>Understanding, controlling or improving the object</td>
</tr>
<tr>
<td>With respect to</td>
<td>The quality focus of the object that the measurement focuses on</td>
</tr>
<tr>
<td>From the viewpoint of</td>
<td>The people that measure the object</td>
</tr>
<tr>
<td>In the context of</td>
<td>The environment in which measurement takes place</td>
</tr>
</tbody>
</table>

The object in this case in the development and deployment process within Philips IT. This is the process that has been described within Chapter 5. The initial purpose of this research was to design a DevOps solution. While Philips has already started a pilot in this, the purpose for the measurement is to further understand this process and if possible improve it. The focus would be on the three factors that DevOps describes, thus systems thinking, feedback loops and a culture of learning and experimentation. To discover the viewpoint, several stakeholders are involved in the process. There are three stakeholders that can quickly be identified, namely, the Philips IT management, the customer of the process, and the teams within the process (Delivery, test factory, I&O). As the purpose of this research as well as that of the DevOps pilot within Philips is to discover whether DevOps is useful, the measurements are firstly meant for the management.

This leads to the following goal: Analyze the development and deployment process within Philips IT to further understand and improve with respect to the three ways of DevOps; systems thinking, feedback loops and culture of learning and experimentation from the viewpoint of the IT management.

As the quality focus in this goal actually contains three different foci, these three could be seen as sub goals. Thus, the first goal is to understand and improve with respect to systems thinking, the second goal with respect to feedback loops and the third with respect to culture of learning and experimentation. These goals can be seen below in Table 9.
Creating a reference framework

Following the three goals, it becomes clear the reference framework should look into several different metrics, namely; systems thinking, feedback loops and culture of learning and experimentation. Besides this it is useful to look into metrics used in similar fields. This would be looking into metrics related to lean manufacturing or Scrum. The description of these metrics can be found in chapter 3.

Along with the background on metrics mentioned in chapter 3, several additional factors must be regarded. These factors mainly relate to the quality of work. In regard to development being mostly knowledge work, doing work faster usually means a trade-off in quality. This trade-off in quality is however not accepted by Philips. The metrics (Duque & Cadavid, 2007) do not directly measure quality, however, the quality can be seen in the percentage of rework that has to be done and the amount of quality control done by the team itself. Also, Philips I&O is already measuring the amount of incidents that were raised following a deployment. This number can also be used as an indicator for quality. Finally, it must be taken into account that there is not just a single team working on a platform. Again for the identification of bottlenecks and issues the capacity of these teams could be looked at. Again, while this is not measured directly, the Work in Progress (WIP) does show a part of this as more teams in general allows for more WIP.

Using these metrics covers the first way of DevOps, along with a part of the third way. The second way is also covered in the five dimensions as rework. This would however only include negative feedback and not necessarily the positive feedback. While feedback is generally also provided in the daily team meetings, related to agile development, it is important who provides the feedback. To deliver value in the end, the feedback should come from the people who will eventually use the product or service and perhaps from those who maintain it. To measure this, we could count the average number of times contact exists between the team and the end-users per user story.

First design

Now that a reference framework has been established the first design can be created. For this part the Goal-Question-Metric (GQM) approach (Van Solingen & Berghout, 1999) is again used. As the goals have been defined earlier (Table 9), questions and metrics can now be created.

Starting with the first goal of systems thinking, it is mainly defined meant as looking at the performance of the system as a whole opposed to the separate parts. To understand and improve this performance, it is important to know the current performance of the process for reference. Besides the current state, it

<table>
<thead>
<tr>
<th>Goal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal 1</td>
<td>Analyze the development and deployment process within Philips IT to further understand and improve with respect to systems thinking from the viewpoint of the IT management.</td>
</tr>
<tr>
<td>Goal 2</td>
<td>Analyze the development and deployment process within Philips IT to further understand and improve with respect to feedback loops from the viewpoint of the IT management.</td>
</tr>
<tr>
<td>Goal 3</td>
<td>Analyze the development and deployment process within Philips IT to further understand and improve with respect to culture of learning and experimentation from the viewpoint of the IT management.</td>
</tr>
</tbody>
</table>
is important to know if a change in the process lead to an improvement in that performance. This leads to the following two questions:

- What is the current performance of the process?
- Do changes in the process improve the performance of the entire process?

The second sub goal relates to the feedback loops within the system. Thus generally speaking how well feedback can be given within the process and how well the process can deal with that feedback. Structuring this in a similar fashion as the first sub goal, the first thing to know is the current state of feedback loops. Besides that, while you can have a lot of feedback loops built into the process, this only helps if the customer or end-user is satisfied with the feedback he can provide and if the system can respond to the given feedback. This lead to the following questions:

- What is the current state of feedback loops within the process?
  - Is the customer satisfied with the feedback that can be given?
  - How well can the process respond to feedback?
- Do changes in the process improve the state of feedback loops within the process?

For the third sub goal of culture of learning and experimentation, the questions can again begin with the current state of the culture. To answer this, research by López, Peón and Ordás (2004) can help by defining four different aspects of organizational learning. The first construct relates to the acquisition of knowledge, either through external sources or internal development. As a second construct they name the distribution of knowledge followed by interpretation as a third construct. Interpretation describes the way that people within the organization share and use the knowledge. The last construct they describe is organizational memory, or the way the organization stores the knowledge for later use. Besides learning, experimentation is there to further help improve the work and process and reward trying new things (Kim, 2013).

Summarizing this, and again structuring like the previous two goals, this leads to the question what the current state of learning and experimentation is like within the teams. This question can then be divided further into three questions that could answer this, namely:

- What is the current state of learning and experimentation within the teams?
  - Is knowledge being stored?
  - Can knowledge be retrieved?
  - Is the team improving their own process?
- Do changes in the process improve the state of learning and experimentation?

From these questions metrics can be defined that answer these questions and thus answer whether the goal is reached. For the question of the first sub goal; do changes in the process improve the performance of the entire process? the metrics should describe whether a difference exists between the performance before and after a change. To describe the performance of a process Basili et al. (1994) use the current average cycle time divided by a baseline average cycle time. To relate this to a change in the process, and to the situation within Philips we could use the time from the moment a user story is
picked up until the moment it is being deployed and supported. Also the baseline would here be the average before the change occurred. While this first metric only takes the speed into account, a second metric could be used for quality. This is another factor that is important to take into account, as higher quality leads to less rework, which in time leads to a better lead time. A simple metric that is already being used within Philips could suffice here, by counting the average number of incidents after deployment before and after the change in a similar way as the previous metric. A third metric to look at would be the cost of a run through the process. This cost figure should be in there to account for the fact, that if you would use double the resources for everything in the process, you would most likely also see the cycle time being cut in half. Doing this however, will significantly increase or double the cost, which in turn should not count as an improvement. An overview of the metrics related to the first goal is shown in Table 10.

Table 10 GQM Metrics for system thinking

| What is the current performance of the process? | • Average cycle time of a user story  
• Number of incidents after deployment  
• The cost of a user story going through the process |
|----------------------------------------------|------------------------------------------------------------------------------------------------|
| Do changes in the process improve the performance of the entire process? | • \( \frac{\text{average lead time of a user story after change}}{\text{average lead time of a user story before change}} \ast 100\% \)  
• \( \frac{\text{average number of incidents after deployment after change}}{\text{average number of incidents after deployment before change}} \ast 100\% \) |

To gain an understanding of the current state of feedback loops, the first thing that comes to mind is the amount of feedback loops there are. While this sounds like a decent first metric, the issue here could be that an arbitrary amount could be sufficient for a process of length \( x \), but not for a process of length two times \( x \). To take the length of the process out of the metric a different metric can be used, which looks at the average time between feedback moments. To use this metric, the start and end point of the process should also count as a moment. Which means if no additional feedback moments are in place, the time would be equal to the length of the process. Less abstractly, the time describes the time between contact points with the customer. One problem with this would however be if the only feedback moment is located at the end of the process, the average time would be same as if the feedback moment would be right in the middle of the process. To cover this, an additional metric could alleviate that problem, by keeping track of the maximum time within the process without feedback. When this time would be very close to the average, we can see that the feedback moments are evenly spread out over the process.

To answer if the customer is satisfied with the way he can provide feedback a qualitative metric is proposed. This can be done by asking the customer whether or not he would have the next feedback moment quicker than the time since the last feedback moment. This metric should atleast be asked the first few times working with a particular customer, until a clear understanding is reached between the team and that customer.
To answer how well the system can respond to given feedback. A first suggestion would be to look at the amount of work which has to be redone within the process. This can be quantified in the amount of time spent from the moment of feedback until the process reaches the same point again. While this might be slightly difficult to measure in practice, an easier metric could be to measure the total time spent on rework during the process.

An overview of the second set of metrics relating to feedback loops is shown in Table 11.

**Table 11 GQM Metrics for feedback loops**

| What is the current state of feedback loops within the system? | • Average time between feedback moments
| | • Maximum time between feedback moments
| Is the customer satisfied with the feedback that can be given? | • Would the customer have the next feedback moment quicker or later than the time since the last feedback moment?
| How well can the system respond to feedback? | • Time spent from feedback moment until reaching the same point
| | • Total time spent on rework

To answer the questions about the current state of learning, the first and second metric could simply state if new knowledge is actively being stored and if stored knowledge can be actively retrieved. However to determine if knowledge is being shared as well as a mechanism in place to make sure that knowledge is actually being stored a metric could state whether the team reflects on their work and learnings after a project or sprint. Finally, a metric should state whether the team reflects on their way of working, and thus takes time to improve their process or way of working. These metrics are shown below in Table 12.

**Table 12 GQM Metrics for culture of learning and experimentation**

| What is the current state of learning and experimentation within the teams? | • Is new knowledge being stored during the process?
| | • Can previously acquired knowledge be retrieved?
| | • Does the team reflect on their work and learning points after a project or sprint?
| | • Does the team reflect on their way of working after a project or sprint?

**First feedback round and second design**

Following the creation of the first design, two feedback sessions were performed. Both sessions were done in the same setting, where the model was briefly explained per goal. After each goal the attendees were asked what they thought of the goal, the questions and the metrics. After the last goal was discussed, attendees had the chance to provide some additional feedback if they wished to do so. An overview of the feedback can be found in Appendix E. The materials used can be found in Appendix F. The materials included the process picture from Figure 7 to ensure those involved were all talking about the same thing. This was especially helpful when talking about feedback loops. Also, to firstly explain
systems thinking being about the performance of the whole process, and not just the performance of the separate teams. The first of the two sessions was done together with two people namely; Bas Vermeer (Manager Operations, whom the assignment originated from) and Rita Conceicao (Global Demand Manager). As the assignment originated from them, it would be useful to get their feedback. Also, they would be able to adjust the direction if that turned out to be necessary. The second feedback session was done with Laki Ahmed (IT Delivery Manager). Her position was closer to the teams that the metrics are designed for. Therefore she could provide more specific feedback.

For the first goal, there were three main points of feedback. The first point considered the use of the word ‘user story’, while it would be better to use a different unit of measurement. For the process, a collection of user stories moves through the process simultaneously, except for the part where they are developed. To measure this would not provide any additional information about the entire process. For this reason, a new unit of measurement is chosen as the collection of these user stories, called a ‘feature’. The second point of feedback considered the specification of cost within the process, which should be further defined. Considering the fact that this process contains knowledge work, thus no tangible products, the cost of a feature would consist of the hours spent and the amount of people working on it. That together is thus proposed as the cost measurement. The third point of feedback was that by measuring the first question continuously, the second question would be irrelevant. As a continuous measurement would show change in these metrics directly rather than as a change. Therefore the second question has been removed. This led to the metrics in Table 13.

For the second goal, the feedback was quite different over the two sessions. One session mainly agreed with the proposed metrics and suggested some small changes in words and a few clearer definitions. For the other session a different understanding of feedback loops led to some more feedback. The understanding there meant more focus on feedback like how fast would changes in the system be seen at the end of the process, rather than customer feedback. It was also suggested that I change feedback moments to touch points for better understanding within Philips. This feedback led to a change in questions, with more focus on the internal changes than the customer feedback. The changes can be seen in Table 14.

<table>
<thead>
<tr>
<th>Table 13 GQM Metrics for system thinking</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What is the current performance of the process?</strong></td>
</tr>
<tr>
<td>• Average cycle time of a feature</td>
</tr>
<tr>
<td>• Average waiting time of a feature</td>
</tr>
<tr>
<td>• Number of incidents as a result of the feature after deployment</td>
</tr>
<tr>
<td>• The cost of a feature going through the process</td>
</tr>
<tr>
<td>• Hours spent</td>
</tr>
<tr>
<td>• Number of people</td>
</tr>
<tr>
<td>• Number of features being worked on</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 14 GQM Metrics for feedback loops

<table>
<thead>
<tr>
<th>Question</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the current state of feedback loops within the system?</td>
<td>- Average time between touchpoints</td>
</tr>
<tr>
<td></td>
<td>- Maximum time between touchpoints</td>
</tr>
<tr>
<td>How well can the system respond to feedback?</td>
<td>- Time spent from feedback moment until reaching the same point</td>
</tr>
<tr>
<td></td>
<td>- Total time spent on rework</td>
</tr>
<tr>
<td>How fast can the system respond to changes in the process? (How fast are the changes visible in the end?)</td>
<td>- The average time a change is seen at the end of the process?</td>
</tr>
<tr>
<td></td>
<td>- Cycle time divided by 2</td>
</tr>
</tbody>
</table>

For the third goal, it was initially more difficult to find decent metrics. Therefore the feedback mainly stated that the metrics were still mostly questions and should be further developed. Some feedback included the addition of metrics related to the capabilities of the team members, and how well people could perform the activities of other team members. Also, a suggestion was made that the number of value propositions should be counted. Here, a value proposition would mean a member making a suggestion for change in the process or team with an estimated value that is gained by implementing the change. Both these suggestions however, have their own problems. By just measuring the capabilities, it would mean that you can get a culture of learning by simply hiring the people with the most capabilities. While this is certainly important, it does nothing directly to measure your culture of learning. The amount of value propositions would be a nice metric to look at. However, you would also not like your people to spend too much time on this, thus making it very difficult to say when this metric is at a decent value.

However, some metrics are still proposed as further development was necessary. The focus for learning lays on the time spent on improving daily work. This does not mean improving work done, and improving it, but rather improving the way the teams perform their daily work. Thus measuring their time spent on learning and partly in experimentation. For experimentation specifically some ideas were suggested during the feedback sessions, which were added as well. The resulting questions and metrics can be seen below in Table 15.

Table 15 GQM Metrics for culture of learning and experimentation

<table>
<thead>
<tr>
<th>Question</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the current state of learning within the teams?</td>
<td>- Percentage of teams spends time improving their daily work?</td>
</tr>
<tr>
<td></td>
<td>- How much time do teams spend on improving their daily work?</td>
</tr>
<tr>
<td></td>
<td>(percentage of total time)</td>
</tr>
<tr>
<td></td>
<td>- Is the time spent adequate according to the team?</td>
</tr>
<tr>
<td>What is the current state of experimentation within the teams?</td>
<td>- Amount of intentional faults introduced per feature?</td>
</tr>
<tr>
<td></td>
<td>- Percentage of faults caught before Live?</td>
</tr>
</tbody>
</table>
Second feedback round and third design

For the next feedback loop one single feedback session was held. This session was again held with Laki Ahmed (IT Delivery Manager). The feedback in this session mainly consisted of small updates and clarifications. Only for the third goal actual feedback was given to actually change things. This was however more validation feedback than feedback to actually change the metrics. An overview of the feedback can be found in Appendix E. The materials used for this sessions can be seen in Appendix F.

For the first and second goal, this meant a small clarification in two specific terms. First, cycle time was changed to lead time and Second, the cost of a feature was further specified by adding service costs. For the metrics regarding experimentation, the feedback was that it might not be very useful to introduce faults. Also, concerning there will be a pre-Live environment that is kept in the same state as Live for testing purposes, the fault would be caught there anyway, and only cause extra work without much gain. While this feedback is certainly valuable, it does not have to change the metric, but rather as a validation step for the usefulness of the design.

The metrics model after these changes is depicted below in Table 16.

<table>
<thead>
<tr>
<th>Table 16 DevOps Metrics Model</th>
</tr>
</thead>
</table>
| What is the current performance of the process? | • Average lead time of a feature  
• Average waiting time of a feature  
• Number of incidents as a result of the feature after deployment  
• The cost of a feature going through the process  
  • Hours spent  
  • Number of people  
  • Service costs  
  • Number of features being worked on |
| What is the current state of feedback loops within the system? | • Average time between touchpoints  
• Maximum time between touchpoints  
• Number of approvals |
| How well can the system respond to feedback? | • Time spent from feedback moment until reaching the same point  
• Total time spent on rework |
| How fast can the system respond to changes in the process? (How fast are the changes visible in the end?) | • The average time a change is seen at the end of the process?  
  • Lead time divided by 2 |
What is the current state of learning within the teams?

- Percentage of teams spends time improving their daily work?
- How much time do teams spend on improving their daily work? (percentage of total time)
- Is the time spent adequate according to the team?

What is the current state of experimentation within the teams?

- Amount of intentional faults introduced per feature?
- Percentage of faults caught before Live?

As so few changes were actually made, this is the last feedback cycle performed. Also, in this last cycle, part of the feedback belongs more closely to validation than feedback. As during the sessions an agreement was reached on each part except for the culture of experimentation. This suggests the proposed design consists of metrics that would be useful within Philips IT to measure the progress on DevOps. For the state of experimentation, the feedback however was that it would not be very useful to actually do this experimentation. For this reason, it is still kept in the model, as it is a part of what DevOps is about. However, for the usability of the metrics it could be excluded by Philips if there is no perceived value. The reason for this is that the faults would only cause extra work, and not actually create any additional value.
7. Discussion

In this chapter some theoretical and practical points are discussed. This starts with a small discussion on the criticism in literature regarding the GQM approach. Following that, the creation of process models as a start to design metrics is discussed. Finally the final metrics model is discussed in contrast to the metrics found earlier in the literature.

While GQM is an approach that has quite some positive response in literature, criticism states that the outcome is rather unpredictable as it is still possible to come to many different metrics that describe the goal. While this is true, this research shows that by performing a few feedback cycles you can adapt those first metrics into metrics that can be validated and used within the environment.

As suggested by Pfleeger (1993), this research also started to design metrics by first describing the process. As someone not working daily in these processes the creation of these process models provided a much better understanding of what was going on. However, besides the value of creating an understanding what the people in the process are doing, the models were hardly used during the design of the metrics model. The main use came from illustrating where the metrics could be applied and to clarify concepts like feedback loops and systems thinking. The amount of detail the processes were modelled in, was probably not needed for this particular purpose. A quick and small overview of the process is most likely enough to use as illustration while designing the metrics.

While not all metrics in the final model can be directly found within the literature research, some clear similarities can be found. For the first goal of systems thinking parallels are found in the average lead time and the amount of features being worked on simultaneously (WIP). Both of these were also included in the research by Duque & Cadavid (2007). The other metrics in the model were not directly found. However, these metrics represent costs and quality, thus might not have been included in the found literature because they are not specific to the domain of lean manufacturing or scrum. For the second goal the parallels are more hidden, but they are most certainly present. While different, the time spent on rework is mentioned again in literature as the percentage of units sent for rework (Duque & Cadavid, 2007). Other than that, the metrics found, like approvals, are more closely related to software development and were less present in manufacturing. For the third goal, a culture or learning and experimentation, the metrics turned out quite different than what was previously found in literature (López, Peón, & Ordás, 2004) as during the first feedback sessions, this seemed to be too abstract. Therefore some simpler and more direct metrics were chosen.

Looking at the metrics model from the literature, the lean manufacturing metrics turned out to be quite helpful in especially the first goal. The metrics found for scrum development however have not been used in the designed metrics. The reason for this is most likely the focus of these metrics. While the lean manufacturing metrics focused on the entire process, similar to the focus of the designed metrics model, the scrum metrics focused more on teams working within this entire process. Therefore these were too specific to answer the goals.
8. Conclusion

This research has provided a metrics model to measure the progress of DevOps within Philips IT. While the validation currently showed low interest for the metrics related to culture of experimentation, this could become more useful when DevOps is further developed within Philips. As more automation would be implemented and more time becomes available to actually try this experimentation.

While the designed metrics model could not yet be implemented within Philips to gather some measurements, this research has shown that within a few feedback loops a clear metrics model can be designed using the GQM approach.
References


Appendix A

Original project description

_Design DevOps for Philips IT_

Are you interested to implement DevOps in Philips IT through designing the new working processes across IT Development and Operations based on Logistical and Lean principles?

**Background**

Philips IT is a fully centralized IT organization servicing the 3 sectors of Philips (Healthcare, Lighting and Consumer Lifestyle). Within the IT organization IT projects are executed in IT Delivery, where daily Operations is executed within Philips IT Infrastructure & Operations (I&O). Over the last 2 years Philips IT made a significant change towards Agile/SCRUM project management. At this moment, Philips is one of the leaders in the industry with more than 100 SCRUM project teams delivering working software every 2 weeks. We do this in full multidisciplinary teams together with our partners in an completely virtual way across the globe (teams working together between Europe, US, India on the same software products for instance).

Philips IT Operations (I&O) is responsible for the day to day operations. With every release of new functionality IT Delivery hands over to IT I&O for day to day operations. In IT I&O we work according to ITIL, which is the Industry best practice operations process framework. ITIL is sometimes conflicting with the Agile/SCRUM mindset. Where Agile/SCRUM demands faster software releases and shorter cycle times, from an operation perspective this may lead to additional risks to system stability, security and performance. An explicit risk is that scrum teams do not spend sufficient time on the full test cycle (regression testing, stress testing, performance testing) leading to production issues and outages.

To overcome this IT Delivery and IT I&O are working together to introduce DevOps into the IT organization. DevOps is a new way of working between Delivery and Operations based on 3 principles: **Fast feedback loops** through multiple releases per week and even per day creating Robust solutions; **Perfect deployment** through fully automated testing and deployment procedures for UAT, regression and stress testing and release mgt; and **Rebuild instead of Repair.** A full description on DevOps thinking is described in the _Phoenix Project (Gene Kim, Kevin Behr and George Spafford)._  

To implement DevOps properly within Philips IT we are looking for students that want to support us in implementing this new way of working in Philips.

**Assignment**

We have an open position to support this change in Philips around the design and implementation of DevOps. How should delivery and operations change their Agile/Scrum and ITIL work processes to enable a full DevOps way of working within IT? How should they adopt principles from **Logistics** and **Lean** to coordinate the workload and ensure high quality delivery of new software?
The assignment is to Design the new working processes between Delivery and Operations based on Logistical and Lean operating principles to enable daily releases of new functionality in production.

Candidate Profile

For this assignment we are looking for a student on a university level with a background in Industrial Engineering, Logistics and/or Information Technology with a clear interest in Logistical design and using Lean principles on work processes. Student should be in his/her final year of a Masters Degree.

Location

Location of the assignment is Eindhoven, High Tech Campus building 42/43.

Further info

For further information please contact Bas Vermeer (Bas.Vermeer@Philips.com).
Appendix B

Philips Integrated Landscape: Information Management
Appendix C

VIPP

VIPP: improved management decision support system
## Appendix D

**Lean measurements (Duque & Cadavid, 2007)**

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elimination of waste</td>
<td>- Work in Progress</td>
</tr>
<tr>
<td></td>
<td>- Setup time</td>
</tr>
<tr>
<td></td>
<td>- Machine downtime</td>
</tr>
<tr>
<td></td>
<td>- Transportation</td>
</tr>
<tr>
<td></td>
<td>- Space Utilization</td>
</tr>
<tr>
<td>Continuous improvement</td>
<td>- Number of suggestions per employee per year</td>
</tr>
<tr>
<td></td>
<td>- Number of suggestions that get implemented</td>
</tr>
<tr>
<td></td>
<td>- Scrap %</td>
</tr>
<tr>
<td></td>
<td>- Rework %</td>
</tr>
<tr>
<td>Continuous flow and Pull-driven systems</td>
<td>- Lot sizes</td>
</tr>
<tr>
<td></td>
<td>- Order flow time</td>
</tr>
<tr>
<td></td>
<td>- Order lead time</td>
</tr>
<tr>
<td></td>
<td>- Pulling processes</td>
</tr>
<tr>
<td></td>
<td>- Pull Value %</td>
</tr>
<tr>
<td>Multifunctional teams</td>
<td>- Autonomous control</td>
</tr>
<tr>
<td></td>
<td>- Workteam Task Content</td>
</tr>
<tr>
<td></td>
<td>- Cross training</td>
</tr>
<tr>
<td></td>
<td>- Number of employees capable of assignment rotation</td>
</tr>
<tr>
<td>Information Systems</td>
<td>- Frequency with which information is given to employees</td>
</tr>
<tr>
<td></td>
<td>- Percentage of procedures that are documented in the company</td>
</tr>
<tr>
<td></td>
<td>- Frequency with which the line or cell progress boards are updated</td>
</tr>
</tbody>
</table>
## Feedback session 1 (Round 1)

Date: July 16th 2014  
Participants: Bas Vermeer (Manager Operations) & Rita Conceicao (Global Demand Manager)

<table>
<thead>
<tr>
<th>Goal or Metric</th>
<th>Remarks</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Systems thinking</strong></td>
<td>Made specific use of the process picture to ensure we were all talking about the performance of the entire process opposed to separate parts within the process</td>
<td></td>
</tr>
<tr>
<td><strong>The cost of a user story going through the process</strong></td>
<td>Should be further specified, what does a cost consist of</td>
<td>Added: Hours spent &amp; number of people</td>
</tr>
</tbody>
</table>
| **Feedback Loops**                         | More focus on how the system responds to feedback and not so much on customer feedback | Added: How fast can the system respond to changes in the process? (How fast are the changes visible in the end?)  
Removed: Is the customer satisfied with the feedback that can be given? |
| **Culture of Learning**                    | Metrics were still pretty much questions and not really metrics. Suggestions were to measure the capabilities of team members (capability matrix) | No direct changes during the session |
| **Culture of Experimentation**             | Some suggestions were made regarding introducing faults into the system and measuring those. | Added:  
  - Amount of intentional faults introduced  
  - Percentage of faults caught before Live |
**Feedback session 2 (Round 1)**

*Date: July 21st 2014*

*Participants: Laki Ahmed (IT Delivery Manager)*

<table>
<thead>
<tr>
<th>Goal or Metric</th>
<th>Remarks</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systems thinking</td>
<td>Made specific use of the process picture to ensure we were all talking about the performance of the entire process opposed to separate parts within the process. User story as an object was deemed too small.</td>
<td>Change user story to feature (collection of user stories)</td>
</tr>
</tbody>
</table>
| What is the current performance of the process? | • Cost should be further specified (same as session 1).  
• Add a measure to see how much more can be improved.  
• Work in progress or done simultaneously could also be important | Added:  
• Average waiting time of a feature  
• Number of features being worked on                                                                 |
| Do changes in the process improve the performance of the entire process? | Unnecessary to measure. By measuring the performance continuously, changes can be seen much faster.                                                                                                       | Remove question                                                                                   |
| Feedback Loops                              | • Concept explained along the process model  
• Internal use of the term “Touchpoint” instead of feedback moment | Change “feedback moment” to “Touchpoint”                                                           |
| Culture of Learning                         | Metrics were still pretty much questions and not really metrics. Suggestions were to measure the amount of value propositions made by team members                                                                 | No direct changes during the session                                                                |
| Culture of Experimentation                  | Showed interest in the concept but no suggestions were made                                                                                                                                            |                                                                                                    |
## Feedback session 3 (Round 2)
**Date:** September 17<sup>th</sup> 2014  
**Participants:** Laki Ahmed (IT Delivery Manager)

<table>
<thead>
<tr>
<th>Goal or Metric</th>
<th>Remarks</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Systems thinking</strong></td>
<td>Cycle time was incorrectly used as a term.</td>
<td>Change cycle time to lead time</td>
</tr>
<tr>
<td></td>
<td>Should change to lead time</td>
<td></td>
</tr>
<tr>
<td><strong>The cost of a user story going through the process</strong></td>
<td>Should be further specified, should include service costs</td>
<td>Added: Service costs</td>
</tr>
</tbody>
</table>
| **Culture of Experimentation**          | • Live should be pre-Live as an extra environment will exist as a copy of Live, but for testing purposes.  
• Did not perceive much added value in introducing faults, especially because of the pre-Live environment | Change Live to pre-Live      |
Appendix F
1st Feedback round

DevOps
Graduation project: Metrics model

Goal:
- Relevant metrics
- Gain final input

Background:
- Philips IT: Systems & T&L
- Systems thinking
- Feedback loops
- Culture of learning

Improvement options:
- Simpler and adaptive
- Approach agreements with partners
- Metrics on DevOps

DevOps metrics
Graduation project

Determine metrics:
- S PUT (Simplifying, Test, Understand, and Learn) (Schibeci & Gregoire, 1999)
- Goal definition template (Roche, Cadbury, & Berntson, 2003)

Analysis
- The development and deployment process
- Understanding and improving
- The context of Philips IT
- From the viewpoint of the IT management

Goal:
- System thinking
- Feedback loops
- Culture of learning and experimentation

DevOps metrics
System thinking

Goal:
- Evaluate the development process within Philips IT to further understand and improve with respect to systems thinking from the viewpoint of the IT management.

DevOps metrics
Feedback loops

Goal:
- Improve the development and deployment process within Philips IT to further understand and improve with respect to feedback loops from the viewpoint of the IT management.

DevOps metrics
Culture of learning and experimentation

Goal:
- Evaluate the development and deployment process within Philips IT to further understand and improve with respect to culture of learning and experimentation from the viewpoint of the IT management.

What is the current state of learning and experimentation?
- Are people actively learning and experimenting?
- What do people believe about learning and experimentation?
- How do people reflect on their work and sometimes about other projects or methods?
- How do they reflect on their way of working after a project or sprint?

53
2nd Feedback round

DevOps metrics
Graduation project
Goal:
- Present metrics
- Gain final input

Background:
- PHIL IT - Scrum & ITIL
- DevOps
- Systems thinking
- Feedback loops
- Culture of learning

Current situations
VOP, IC, & FRIM end-to-end process to a general model.

Improvement options:
- Scrum and ward fall
- Approach vs agreements with partners
- Metrics vs DevOps

DevOps metrics
Graduation project
Goal:
- Present metrics
- Gain final input

Background:
- PHIL IT - Scrum & ITIL
- DevOps
- Systems thinking
- Feedback loops
- Culture of learning

Current situations
VOP, IC, & FRIM end-to-end process to a general model.

Improvement options:
- Scrum and ward fall
- Approach vs agreements with partners
- Metrics vs DevOps

DevOps metrics
System thinking
Goal:
- Analyze the development and deployment process within PHIL IT to further understand and improve with respect to systems thinking from the viewpoint of the IT management.

What is the average performance of key metrics:
- Average cycle time of changes
- Average cycle time in MoSiC
- Number of changes made in the feature of deployment
- The cost of changes going through Development
- Sprint gap
- Ratio of DevOps

DevOps metrics
Feedback loops
Goal:
- Analyze the development and deployment process within PHIL IT to further understand and improve with respect to feedback loops from the viewpoint of the IT management.

What is the average time it takes to respond to feedback:
- Average time from feedback received until working on the issue
- Time spent on feedback response activity
- Time spent on feedback activity
- Time spent on feedback

What is the difference between changes in the process:
- Changes in the process
- Changes in the process

DevOps metrics
Culture of learning and experimentation
Goal:
- Analyze the development and deployment process within PHIL IT to further understand and improve with respect to culture of learning and experimentation from the viewpoint of the IT management.

What is the average time:
- What percentage of team reach time improving their daily task?
- How much time do teams spend on improving their daily work? (percentage of total daily work)
- How much time does each team spend on daily task?
- Average of tasks completed within hours
- Percentage of tasks completed before deadline

What is the average time:
- What percentage of team reach time improving their daily task?
- How much time do teams spend on improving their daily work? (percentage of total daily work)
- How much time does each team spend on daily task?
- Average of tasks completed within hours
- Percentage of tasks completed before deadline

DevOps metrics
Graduation project
Goal:
- Present metrics
- Gain final input

Background:
- PHIL IT - Scrum & ITIL
- DevOps
- Systems thinking
- Feedback loops
- Culture of learning

Current situations
VOP, IC, & FRIM end-to-end process to a general model.

Improvement options:
- Scrum and ward fall
- Approach vs agreements with partners
- Metrics vs DevOps

DevOps metrics
Graduation project
Goal:
- Present metrics
- Gain final input

Background:
- PHIL IT - Scrum & ITIL
- DevOps
- Systems thinking
- Feedback loops
- Culture of learning

Current situations
VOP, IC, & FRIM end-to-end process to a general model.

Improvement options:
- Scrum and ward fall
- Approach vs agreements with partners
- Metrics vs DevOps

DevOps metrics
System thinking
Goal:
- Analyze the development and deployment process within PHIL IT to further understand and improve with respect to systems thinking from the viewpoint of the IT management.

What is the average performance of key metrics:
- Average cycle time of changes
- Average cycle time in MoSiC
- Number of changes made in the feature of deployment
- The cost of changes going through Development
- Sprint gap
- Ratio of DevOps

DevOps metrics
Feedback loops
Goal:
- Analyze the development and deployment process within PHIL IT to further understand and improve with respect to feedback loops from the viewpoint of the IT management.

What is the average time it takes to respond to feedback:
- Average time from feedback received until working on the issue
- Time spent on feedback response activity
- Time spent on feedback activity
- Time spent on feedback

What is the difference between changes in the process:
- Changes in the process
- Changes in the process

DevOps metrics
Culture of learning and experimentation
Goal:
- Analyze the development and deployment process within PHIL IT to further understand and improve with respect to culture of learning and experimentation from the viewpoint of the IT management.

What is the average time:
- What percentage of team reach time improving their daily task?
- How much time do teams spend on improving their daily work? (percentage of total daily work)
- How much time does each team spend on daily task?
- Average of tasks completed within hours
- Percentage of tasks completed before deadline

What is the average time:
- What percentage of team reach time improving their daily task?
- How much time do teams spend on improving their daily work? (percentage of total daily work)
- How much time does each team spend on daily task?
- Average of tasks completed within hours
- Percentage of tasks completed before deadline