Utrecht University

Optimizing Students Learning Process through the Cognitive Load Theory

A case study on the module Information Systems

Simon van de Fliert
Studentnumber: 5708656
E-mailadres: S.p.j.fliert@students.uu.nl
Bachelor's Thesis
5th of August 2019

Bachelor Information Sciences
Utrecht University, Utrecht, Netherlands

Supervisor and First Examiner: Dr. ir. J.M.E.M van der Werf
Second Examiner: Dr. S.J. Overbeek

Acknowledgements

Before you lies my bachelor thesis "*Optimizing Students Learning Process through the Cognitive Load Theory: A case study on the module Information Systems*". This research has been conducted at Utrecht University. It is the final assignment in the Bachelor of Information Sciences to meet the requirements for graduation. This thesis was written between April and July 2019.

I find many academic topics interesting. This was no different for this thesis. In meetings with dr. Jan Martijn van der Werf, my thesis supervisor, we brainstormed many potential research topics, varying from attempts to enhance information technology to a greater focus on the human application of existing technology. During our discussions, I found that I had a stronger interest in human interactions with technology rather than the technical side of enhancing existing applications. We refined my research question into what has become the topic of this thesis: using the Cognitive Load theory to optimize the learning process of students studying information technology.

This thesis would not have been written without the diligent guidance of my supervisor, Dr. Jan Martijn van der Werf. I would like to thank Dr. Jan Martijn van der Werf for his immeasurable patience and his guidance during this project. Your passion was infectious and after each meeting, I felt renewed passion to continue my work.

I hope you enjoy reading this thesis.

With kind regards,
Simon van de Fliert
5708656
Utrecht, 5th of August 2019

Abstract

This thesis investigates how cognitive load theory might be used to optimize the learning process of students studying information technology. It does so by investigating the risk of cognitive overload in the module Information Systems, offered to first-year bachelor students Information Sciences at Utrecht University (UU). The module Information Systems aims to teach students about the modelling language Petri Nets. To assist students in their learning, they are obliged to participate in practice sessions (also called modelling sessions). However, it remains unclear how the modelling sessions can be set-up to optimally support student learning.

This thesis offers an answer to the following research question: *How can the learning process of students studying information modelling languages in the 10-week Utrecht University module Information Systems be optimized?* This question is answered with a case study. Due to scheduling issues, the case study was only theoretical. It shows how simple changes in how information is taught can help prevent student cognitive overload and help students better grasp more of the information being taught in the module Information Systems. Future research can be conducted on the effect of individual interventions.

Table of Contents

List of Figures

1. Introduction

Since the start of the Industrial Revolution in the late 18th century, companies around the world have attempted to cut costs and out-compete rivals through standardisation and automation. Businesses strived to gain an advantage over competitors, which sparked the need for measurable variables. (Lusk et al., 2005).

As businesses seek to standardize their products, they created production 'lines' and standardized work processes that need supervision and management to run well. Business process management (BPM), the process of optimising business production processes, offers a theoretical description of how to optimise production (Lusk, Paley, Spanyi, 2005).

The development of BPM saw a growing demand for individuals with modelling skills. Over the past half-century, universities have sought to satisfy this demand by enrolling students in business administration and information management classes. Given the great number of modelling languages, modelling methods, and the complexity of many languages, universities need to prioritize which modelling languages they teach. Furthermore, they will need to teach these complex languages within weeks, often to students with no prior knowledge of modelling.

The combination of complexity and time pressure risks exposing students to an overload of information, beyond which their cognitive abilities can cope. Cognitive overload decreases the effectiveness of student learning activities and can impact their study results or even their self-confidence. Therefore, it is in the interest of both universities and students to investigate how classes should present their information so that students will be given efficient learning activities.

This thesis investigates how the learning process of students studying information modelling languages can be optimized. It presents a case study of the mandatory 10-week module Information Systems (Informatiesystemen), taught at the University of Utrecht during the Bachelor of Information Sciences (Informatiekunde). In the module Information Systems, students are primarily taught via classical lectures and modelling sessions. During these modelling sessions, students are given exercises to practice the theoretical information taught during lectures. Due to time constraints, it is imperative that the modelling sessions are optimized without overburdening students. This thesis, therefore, seeks to answer the question:

> How can the learning process of students studying information modelling languages in the 10-week Utrecht University module Information Systems be optimized?

This thesis is divided into 5 chapters. First the biology of learning will be discussed, then the Cognitive Load Theory. The following chapter will discuss Blooms Taxonomy in educational institutions. The final chapter will discuss a case study regarding the module Information Systems.

2. The Biology of Learning

To learn is to be human; such is the description of human nature held by many a philosopher. Like the classic Chinese philosopher Xun Zi already noted more than two thousand years ago: “Learning proceeds until death and only then does it stop” (Hutton, 2016, p. 5) But how is human biology related to this process of knowledge acquisition, processing and storage which we undertake almost every waking moment of our lives? This section describes how human beings learn, and what limits exist in this knowledge.

2.1 Remembering Information

We can think, remember and reproduce knowledge because of our brain. But how does the brain function to allow humans to acquire and remember information? Atkinson and Shiffrin (1968) describe how two important types of memory play a role: long-term memory and short-term memory (also known as working memory). (Atkinson & Shiffrin, 1968)

2.1.1 Short term (working) memory

Short term memory is the area of the brain which concerns itself with the processing of new information. It is limited in both its duration and processing capacity when dealing with new information (Mason et al., 2016). The exact short-term memory capacity limit differs per person and an exact average has been much debated. Some state that the working memory limit to be 4±1 elements (Paas & Ayres, 2014)**,** whilst others determine the range to be 7± 2 elements (Zugal et al., 2011). The duration of information storage is also debated. Some state the limit lies at 30 seconds (Paas & Ayres, 2014), whilst for others, it lies at 20 seconds (Leppink et al., 2015). Part of the difficulty in measuring short-term memory occurs because it is also affected by things such as its physical environment and human emotions. However, irrespective of disagreement about the exact time and storage capacity, there is wide agreement that the working memory can handle only a small number of elements for a short period.

2.1.2 Long term memory

Long-term memory is a key part of human learning. It is the memory which stores information so that it can be recovered at a later date. The long-term memory has unlimited capacity, where information is permanently stored in the form of schemas (Mason, Seton, & Cooper, 2016). Sweller (1994) describes a schema as “a cognitive construct that organizes the elements of information according to the manner with which they will be dealt.” (p.296). Schemas have no limit in size, and the more skilled an individual becomes, the larger its schema becomes – often by combining smaller schemas into a more complete picture (Paas & Ayres, 2014).

2.1.3 Interaction between short-term and long-term memory

The short-term (working) memory can take schemas from long-term memory and use them as required to help conduct all manner of tasks (Mason et al., 2016). A schema counts only as one

element in the working memory (Paas & Ayres, 2014**;** Zugal, Pinggera, Weber, 2011). Because schemas can hold unlimited information (Mason et al., 2016), the limitations of the working memory are just as well removed for more skilled learners when they are dealing with information that is already known. Therefore, more skilled individuals will be able to cope with more complex information that would previously cause cognitive overload (Paas & Ayres, 2014). By approximation, the working memory limit is therefore only relevant when dealing with new information.

2.2 Acquiring Knowledge

Humans are natural information processing systems who obtain information in several ways. One prominent description is offered by Sweller (2016), who offers five distinct phases of knowledge acquisition: *Information store, Environmental organizing and linking, Randomness as genesis, Narrow limits of change,* and *Borrowing and reorganizing.*

2.2.1 Information Store (long-term memory)

As noted, previously learned knowledge is stored in the long-term memory (Sweller, 2016). To be able to function in a natural environment, an individual needs a large amount of information. (Sweller, 2016**;** Leppink et al., 2015).

2.2.2 Environmental organizing and linking

Humans must be able to function in a great many different situations. The first thing humans do when confronted with a problem is to use the short-term (working) memory to search the long-term memory for a solution to the problem being faced (Sweller, 2016). The working memory can at any moment take information from the long-term memory and apply it to the situation it faces. This principle is called the Environmental organizing and linking principle. (Sweller, 2016**;** Tricot & Sweller, 2014)

2.2.3 Randomness as genesis

However, sometimes the solution cannot be found from already known knowledge. In these situations, humans analyse the situation in front of them and determine the possible moves they can make (Sweller, 2016). Due to the lack of information in the long-term memory regarding the problem, the individual is forced to make a random move. The outcome of this move is then analysed for its effectiveness. If it is effective, then the working memory stores it in the long-term memory for later use. However, if the move is ineffective, it is discarded. This process – also known as ‘randomness as genesis’ - is how humans create new knowledge (Sweller, 2016**;** Leppink et al., 2015).

2.2.4 Narrow Limits of Change

The ability to create new information and learn from new experiences is critical to our progress as a species. However, the rate of recording new information learned through Randomness as

Genesis must be restricted. If no restrictions are made, we could damage our long-term memory by flooding it with new information. Human beings are therefore restricted by their working memory, which allows a limited amount of information to be processed at the same time (Sweller, 2016**;** Tricot & Sweller, 2014; Leppink et al., 2015). Due to these 'narrow limits of change', the long-term memory absorbs information much more slowly than that the working memory processes it (Sweller, 2016).

2.2.5 Borrowing and reorganizing

One enormously advantageous short-cut to knowledge acquisition which has been developed by human beings is an ability to pass gained experience onto others. This short-cut essentially attempts to increase the efficiency of human learning by limiting the number of random movements that a person without certain knowledge needs to take to find an acceptable solution to a problem or build a complex schema (see also Figure 1 below). Randomness is still genesis for the individual learner, but by consistently practising an outcome which he/she trusts to be correct, the influence of the 'narrow limits of change' can be limited.

It is for this reason that humans borrow information, listen to each other, and read what others write (Sweller, 2016**;** Leppink et al., 2015; Tricot & Sweller, 2014). The information gleaned by listening to and learning from others is in turn automatically combined with information learned from other sources and now held in long-term memory (Sweller, 2016). In this manner, human beings can build new knowledge from the 'old' knowledge passed on to them by other human beings.

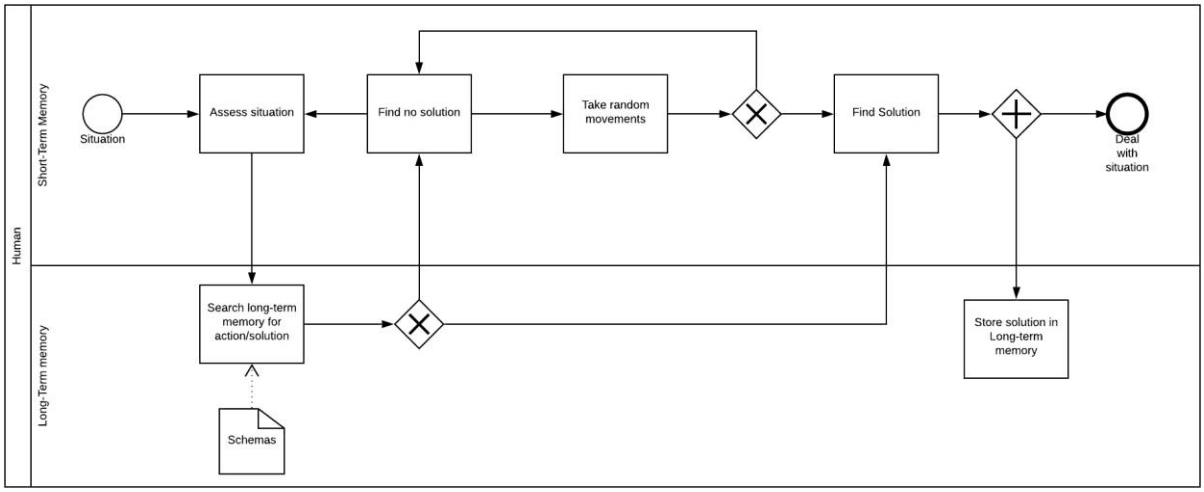


Figure 1: Process of assessing a situation with the five principles

2.3 Types of Knowledge

Given that knowledge acquisition is a deeply biological process, it comes as little surprise that some information is learned more easily than other information. Geary categorizes knowledge into two categories, based on the ease by which it is learned: *Biologically primary knowledge* and *Biologically secondary knowledge* (Sweller, 2016).

2.3.1 Biologically primary knowledge

Biologically primary knowledge refers to knowledge that we as humans have evolved to acquire more easily (Sweller, 2016; Leppink et al., 2015). The information is often complex but easily obtained (Leppink et al., 2015). The information is crucial to our survival (Sweller, 2016). Examples of Biologically primary knowledge are the ability to recognize faces, the ability to acquire general skills, such as social skills (Leppink et al., 2015) and the ability to navigate between places (Sweller, 2016).

2.3.2 Biologically secondary knowledge

Biologically secondary knowledge refers to knowledge which society defines as important, but which we have not evolved to acquire easily (Leppink et al., 2015). Examples of such information include the ability to write or the ability to do algebra (Leppink et al., 2015). There are two important characteristics of this type of knowledge. First, the information is domain-specific (Sweller, 2016), i.e. the information is applicable in some specific situations, but not in others. Secondly, the information taught requires a conscious effort to learn. However, because it is domain-specific, biologically secondary knowledge is learned more easily with explicit guidance (Sweller, 2016). Schools were invented to deliver this type of information to others (Sweller, 2016; Leppink et al., 2015).

3. Cognitive Load Theory

As described in the previous chapter, universities attempt to teach domain-specific biologically secondary knowledge to students in such a way as to significantly decrease the time it takes them to learn these skills compared when students attempt to learn this information through trial-and-error themselves (randomness as genesis). However, in the modern economy, it is not enough to simply teach students new skills and knowledge accumulated through generations. The demand inside and outside academia is to teach these skills within defined schedules so that graduates can fulfil outside business demands at a predictable pace.

Several theories – including the constructivist approaches (Vogel-Walcutt, Gebrim, Bowers, Carper, & Nicholson, 2010), the Cognitive Load Theory and the Cognitive Theory of Multimedia Learning (Mayer, 2005) - have been developed to describe the *rate* at which a person learns, i.e. how a learner might use their biological capacity to create long-term memory schemas as efficiently as possible.

The Cognitive Load Theory (CLT) focuses on obtaining information in well-structured environments. It is less effective in less structured environments. Opposing CLT is the constructivist approach. This approach focuses on individuals processing new information and supporting learners to develop their own learning experience. However, this approach lacks efficiency and fails to consider the human architecture. Individuals faced with this approach often cope with a working memory overload (Vogel-Walcutt et al., 2010). Lastly, the Cognitive Theory of Multimedia Learning focuses on the use of different forms of media. (Mayer, 2005) Whilst this is an interesting theory to study, it is less applicable in the context of this thesis. As this thesis focuses on optimizing the learning process of students studying information modelling languages, the most applicable theory to study is the Cognitive Load theory. Therefore, this theory will be used for this thesis.

Originally developed by John Sweller, CLT focuses on optimizing the acquirement and processing of “domain-specific, biologically secondary knowledge using explicit instructions”, i.e. the type of education performed at schools and universities (Sweller 2016, p. 296).

3.1 Types of Cognitive Load

Cognitive Load Theory distinguishes three types of cognitive load which impact the working memory processing capacity: *intrinsic load, extraneous load, and germane load* (Mason et al., 2016; Paas et al., 2003; Leppink et al., 2015).

3.1.1 Intrinsic Cognitive Load

The intrinsic cognitive load refers to the complexity of the information which must be learned, and the complexity of its relationship to other knowledge schemas. The exact level of complexity is different for each piece of information (Mason et al., 2016; Leppink et al., 2015; Paas et al.,

2003). As the number of interactions between elements of information increases, the material's complexity also increases (Mason et al., 2016; Leppink et al., 2015).

In line with what was described in section 2.1.3, the existence of related schemas in the long-term memory affects the amount of intrinsic cognitive load a user perceives when learning new information (Leppink et al., 2015; Paas et al., 2003). As novices do not have the same schemas which experts do have, novices face a higher intrinsic cognitive load when learning new information about their subject compared with when experts learn new information (Leppink et al., 2015).

Assuming a set knowledge compendium to be learned, the amount of intrinsic cognitive load "cannot be directly influenced by instructional designers" (Paas et al., 2003, p.65). It can only be influenced by the user's pre-existing knowledge of relevant related schemas.

3.1.2 Extraneous Cognitive Load

Extraneous cognitive load "refers to a learner…engaging in cognitive processes that are extraneous to the learning goals" (Leppink et al. 2015). In other words, extraneous cognitive load describes extra information or effort which the learner must process in order to learn the information they want to learn.

In classroom settings, extraneous cognitive load is influenced by instructional design. If the way in which information is presented does not clarify the precise information that should be learned, the cognitive load of learners is higher (Paas et al., 2003; Mason et al., 2016).

This load can be directly influenced by changing the way information is presented. As Leppink et al. (2015) state: "An example, information that should be presented visually is presented verbally (e.g. describing anatomical structures such as the vessels of the heart verbally instead of presenting them visually using a diagram) also contributes to extraneous cognitive load;" (Leppink et al., 2015, p.209).

3.1.3 Germane Cognitive Load

Germane cognitive load is the processing power used to relate new schemas to existing knowledge in the long-term memory (Mason et al., 2016; Paas et al., 2003). As Leppink et al. (2015) state: "Given that in cognitive load theory, learning is the gradual development of knowledge in long-term memory, this third type of cognitive load is the one that arises from relating relevant information from long-term memory or context to new information elements." (Leppink et al., 2015, p.210).

This type of load is also influenced by the instructional design (Paas et al., 2003). When relations between different pieces of information are clearly described, it is easier to build a similar scheme. If, by contrast, the relation between different pieces of information is described diffusely, it takes much more cognitive load to 'lay the puzzle' and develop a complete picture of the information that is being taught.

3.1.3 Relation between different types of Cognitive Load

All three of the cognitive load types are additive. Therefore, the intrinsic cognitive load can be well within the physical limits of the working memory, but if the information is taught inefficiently, the ability of the working memory can be surpassed.

The intrinsic cognitive load cannot be reduced unless the individual gains more and better schemas (Paas et al., 2003). The extraneous and germane cognitive load have thus a set amount of working memory to work with. For example, if the intrinsic cognitive load is ⅓ of the total working memory, the Extraneous and Germane load can use ⅔ of the total working memory before surpassing the maximum threshold (Paas et al., 2003). This is shown in Figure 2 below.

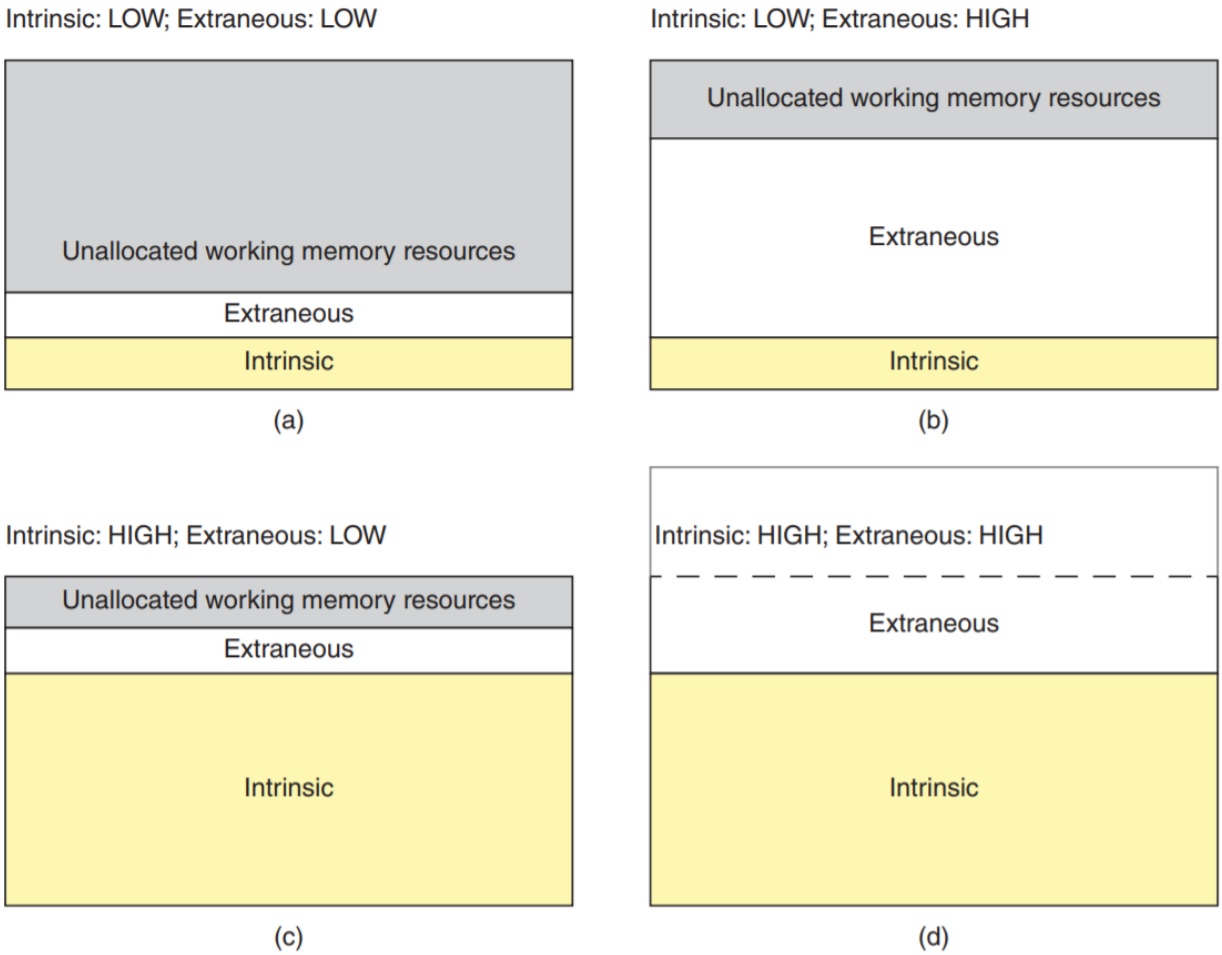


Figure 2: Examples of total working memory consisting of different levels of cognitive load (Leppink et al., 2015)

3.2 Cognitive Overload

As described in section 2.2 above, when humans are confronted with unknown situations for which they have no previous experience, they will experiment to find a suitable solution to the problem which they are presented. In this new situation, their short-term memory will be affected

by the three cognitive loads discussed previously. Furthermore, short-term memory will also be affected by alternative variables, such as emotions and their physical environment.

All these factors are additive, resulting in peak cognitive load or even in cognitive overload. Cognitive overload occurs when humans are confronted with too much complex information to process effectively. This process is enhanced by the way information is presented. This process is also shown in Figure 3 below.
If cognitive overload occurs, the individual will need to take more time to process the information, therefore decreasing the efficiency of their learning process. Cognitive overload does not speed up the learning process but slows it down: it requires students to ask for a break and/or a repetition of explained information.

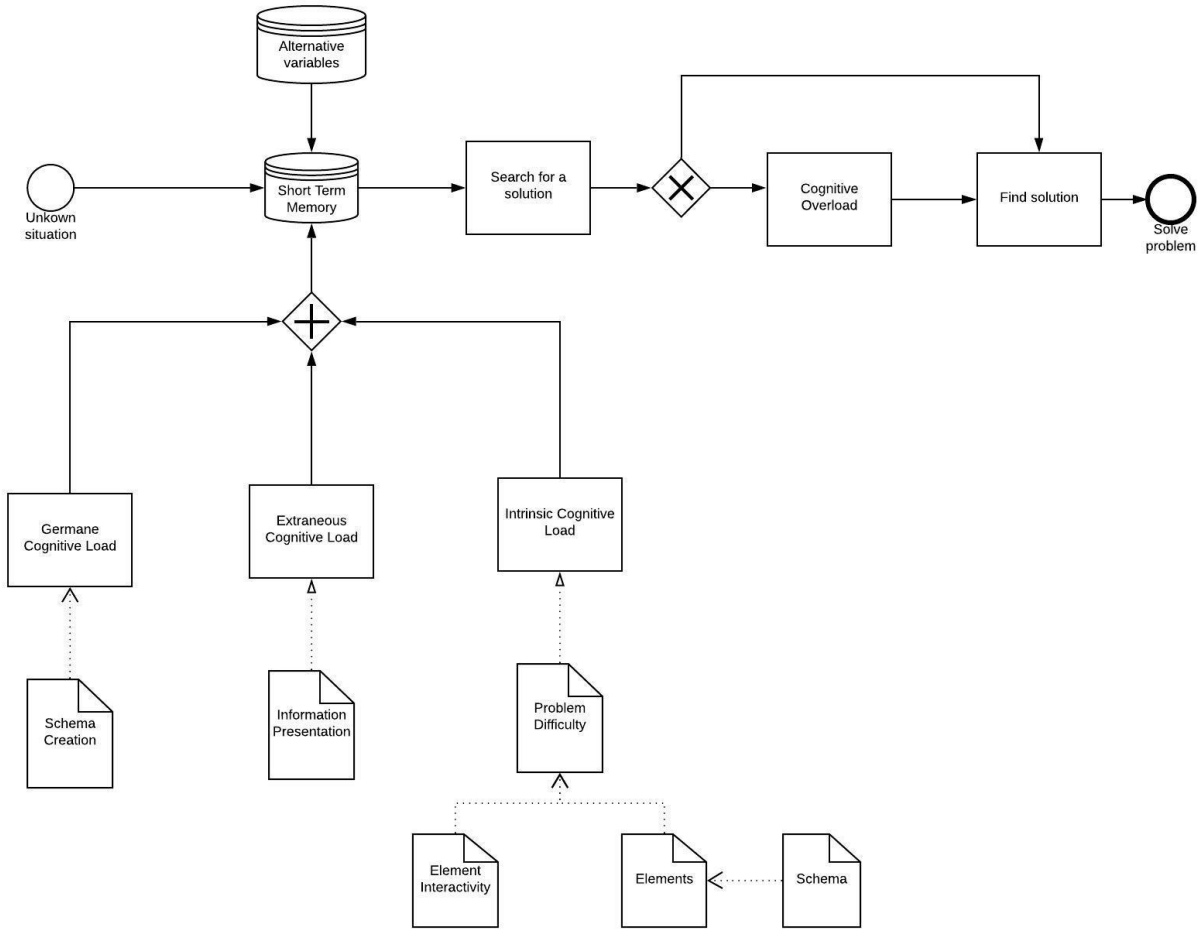


Figure 3: Process of facing an unknown situation

3.3 Optimising use of the cognitive load

Knowing the different cognitive loads and recognising that cognitive overload is not conducive to efficient learning, it is interesting to know how to improve the cognitive load.

Academic literature describes thirteen possible interventions to optimize classroom use of the limited cognitive load of students. Given that in an ideal learning situation, intrinsic cognitive load is high and extraneous and germane cognitive load are limited, these interventions can be divided into those which *adapt the intrinsic cognitive load, adapt the extraneous cognitive load,* and those which *influence the germane cognitive load.* Note that the effects described in this section assume that the learned knowledge is obtained in an educational institute, the knowledge is domain-specific, biologically secondary information that is most effectively transferred with explicit instruction. (Sweller, 2016, p.302)

3.3.1 Adapting Intrinsic Cognitive Load

One of the main ways to prevent cognitive overload is to adapt the difficulty of problems being taught to the level of the student. Several interventions can help to limit or increase the intrinsic cognitive load of practice sessions.

Smaller information chunks (The sub-goal effect and the isolated elements effect)
Manipulating intrinsic cognitive load is all about managing the number of information elements that need to be handled while solving a problem. Whenever students face a problem which is too complicated, a teacher can set smaller goals to make the problem and learning level more manageable. Over time, the number of sub-goals can be reduced to create a larger gap between each sub-goal until the student can solve the complete problem without sub-goals. Teachers can also present the student smaller chunks of information, following the isolated elements effect. Students will create knowledge schemas more efficiently, as the information is presented in more manageable portions and cognitive overload is prevented (Mason et al., 2016).

Step-by-step problem solving (worked example effect)
When students are presented with practice problems to test their knowledge, they will generally use discovery-based problem solving to solve the problem. When the problem relates to new information, they are likely to spend a large portion of their cognitive load on the discovery aspect, and they will use randomness as genesis (see section 2.2.3) to discover the correct answer (Sweller, 2016; Clark, Nguyen, & Sweller, 2011).

When students must discover multiple steps to answer a practice problem, the number of possible answers increases exponentially with each extra step that must be solved. Practice problems that are above the knowledge level of a student therefore require rapidly escalating amounts of energy and cognitive load to solve. Rather than presenting students with open-ended questions, educators can first use worked example exercises which demonstrate step by step how to complete a task (Leppink et al., 2015; Clark et al., 2011).

When a student is presented with a worked example exercise, the student will use less cognitive load on discovery-based problem solving and more on making schemas (Clark et al., 2011). As they use less cognitive load on the discovery of the correct answer, the total cognitive load lowers, meaning there is more room for the germane load. A strong focus on worked examples at the start of a new topic (i.e. for novices) proved more efficient than the conventional focus on solving open question problems and also a more efficient intervention than problem-solving tasks, tutored problem-solving or even a combination of problem-solving and worked examples. (Chandler & Sweller, 1991; Leppink et al., 2015; Mason et al., 2016).

The expertise reversal effect

However, as humans gain expertise, the effectiveness of some teaching methods to offer new information is reduced. For example, whilst the worked example effect is beneficial for new students, it is less effective for experts.

The term *expertise reversal effect* is sometimes used to describe that teaching methods become less effective when an individual gains more expertise (Sweller, 2016). At the core, this effect is the result of the interaction between short-term and long-term memory as described in section 2.1.3: use of long-term memory schema does not require significant working memory cognitive load. As a student gains more knowledge and skill, they therefore need to be presented with more challenging exercises.

Practice problems (the problem completion effect)

One way to present these problems is to ask students to indicate that they want more complicated practice problems. When they do so, a transition should be made and students should be asked to answer partially completed exercises, which are exercises that show new information but do not show previously learned information (Mason et al., 2016; Clark et al., 2011). By presenting partial solutions when a student so indicates, students can couple their learning curve to the difficulty level of the questions being asked, preventing both time wasted on practice questions that are too easy as well as a risk of cognitive overload (Sweller, 2016).

The goal-free effect

With some types of information, it is very difficult to create worked examples. One alternative to limit the cognitive load for students with open questions is to remove a predefined target from problem exercises. The need to resolve a problem to find a single specific answer forces students to think through multiple steps simultaneously, therefore increasing the cognitive load.

Instead of demanding a single answer, students can simply be asked to prove competency of their thought process by finding as many correct answers as possible. For example, in mathematics problems, students might be asked to find as many values as he or she can and showing their thought process at every step without demanding that a specific value for X is described. By taking away the need to ‘hit’ a predefined goal, the student focuses more on technique by which they solve a problem instead of feeling pressure to hit a predefined target (Sweller, 2016).

3.3.2 Adapting Extrinsic Cognitive Load

As noted before, it is not only the amount of information that is presented which affects cognitive load but also how it is presented. Two main decisions which influence the extrinsic cognitive load are how information is presented, and the support that is offered when working through problems.

Presenting information in the most effective manner:
When presenting new information to students, it is important to remove all unnecessary and irrelevant extra information. For example, if the information is presented by two sources, but the sources contain the same information (the redundancy effect), it is best to remove one of the sources of information. By removing sources with identical information, the cognitive load is reduced, as less information clamours for the student's attention (Sweller, 2016; Mason et al., 2016).

Beyond removing redundant information, it is important to as much as possible retain one clear overview of the information which needs to be learned. If related elements of information are presented in two different locations, an individual will have to mentally integrate the information before he or she would understand it. This means the individual is using a portion of his or her cognitive load on mentally integrating the information, which means it is not available for learning (the split-attention effect). Mental integration can be removed by integrating the different sources of information (Sweller, 2016; Mason et al., 2016). The individual can focus more cognitive load on making schemas and learning new information (Sweller, 2016; Mason et al., 2016).

Technology has given educators new methods to present information in a more coordinated manner. However, technology is not a panacea. If not used carefully, it can hinder the learning activity of students. For example, by changing complex written information into spoken information could drastically increase the cognitive load (Sweller, 2016). This phenomenon that cognitive load differs depending on the means of presentation is also known as the transient information effect. It is therefore important to measure the effects of different technology types before implementing them.

At times it is not feasible or practical to fully integrate multiple sources of information. One solution to maximise the use of the working memory in this situation is to present information using multiple sensors. For example, instead of presenting two sources of information in text, it could prove beneficial to present one source of information in text whilst presenting the other source of information via auditory processors. Using different sensors can expand the working memory (Sweller, 2016). However, it is not always beneficial to divide the sources of information across different sensors. Furthermore, if the information is integrated, but is presented in two different formats, for example, audio and visual, a portion of the students' cognitive load will be occupied to ensure that the two sources of information remain coordinated (Mason et al., 2016).

Offering the right support:
Beyond the presentation of new information, how information is processed also influence the use of cognitive load. With very new information cognitive load can be reduced by allowing students to work together in groups (collective working memory effect). Students that work together can theoretically gain a collective working memory – supporting each other by contributing different pieces of knowledge that they have retained to solve problems. (Sweller, 2016).

Whilst novice students learn better with explicit guidance, more skilled students need less guidance (guidance fading effect). Burdening a more experienced individual with too much guidance will interfere with the effectiveness of their learning activities.

It is therefore important to tailor the guidance and support for students so that their knowledge decreases with the increase in expertise – as they start using schemas in line with the following the environmental organizing and linking principle (see section 2.2.2 above). By decreasing the guidance, an individual will not be presented with redundant information, therefore lowering the amount of information he or she must process.

3.3.2 Influencing the Germane Cognitive Load

Of the three types of cognitive load, the germane cognitive load is least susceptible to external influence. It simply takes time for a person to relate newly learned information to older in the memory schemas. However, it is important to recognize that germane cognitive load takes time, and to leave room for this cognitive process to do its work. Given the Narrow limits of change (see section 2.2.4 above), it is important to not teach too much information at once. One technique for which there is some proof that it allows learned information to enter the long-term memory quicker is the ‘imagination effect’, whereby time is taken in a classroom for students to mentally rehearse the learned information and actively find connections with other knowledge they already have (Sweller, 2016).

4. Cognitive Load Theory and Educational Institutions

In the past half-century, universities have sought to satisfy a growing economic demand for individuals with modelling skills by offering students information modelling classes. However, they have not necessarily done so with the perspective of the student in mind.

4.1 Bloom's Taxonomy

First published in 1956, and revised in 2001, Bloom's Taxonomy is a framework for educational institutes to develop learning modules. The framework can be used to determine what students will learn and what knowledge or skills they should be able to produce (Krathwohl, 2002).

Bloom's taxonomy consists of six different cognitive levels: *Remembering, Understanding, Applying, Analysing, Evaluating, and Creating* (Forehand, 2010; Churches, 2008; Adams, 2015)**.** These steps are hierarchical, whereby a student must first be able to describe and interpret a skill before it can be applied. While a student cannot skip ahead a step, he or she can return to previous steps to learn more information (Forehand, 2010).

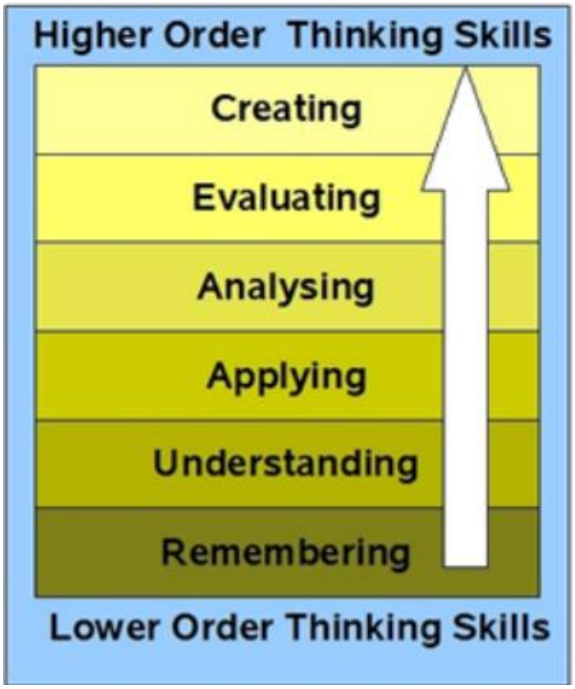


Figure 4: Cognitive levels of thinking according to Bloom's Taxonomy (Churches, 2008)

Examples of what activities student can perform at each level are shown in Figure 5 below. Following Bloom's Taxonomy, a student must not be asked to analyse information before he or she has been offered enough time to remember and understand new knowledge. Furthermore,

as universities deal with a strict time constraint, it is imperative to create a module where students are given the most efficient learning activities.

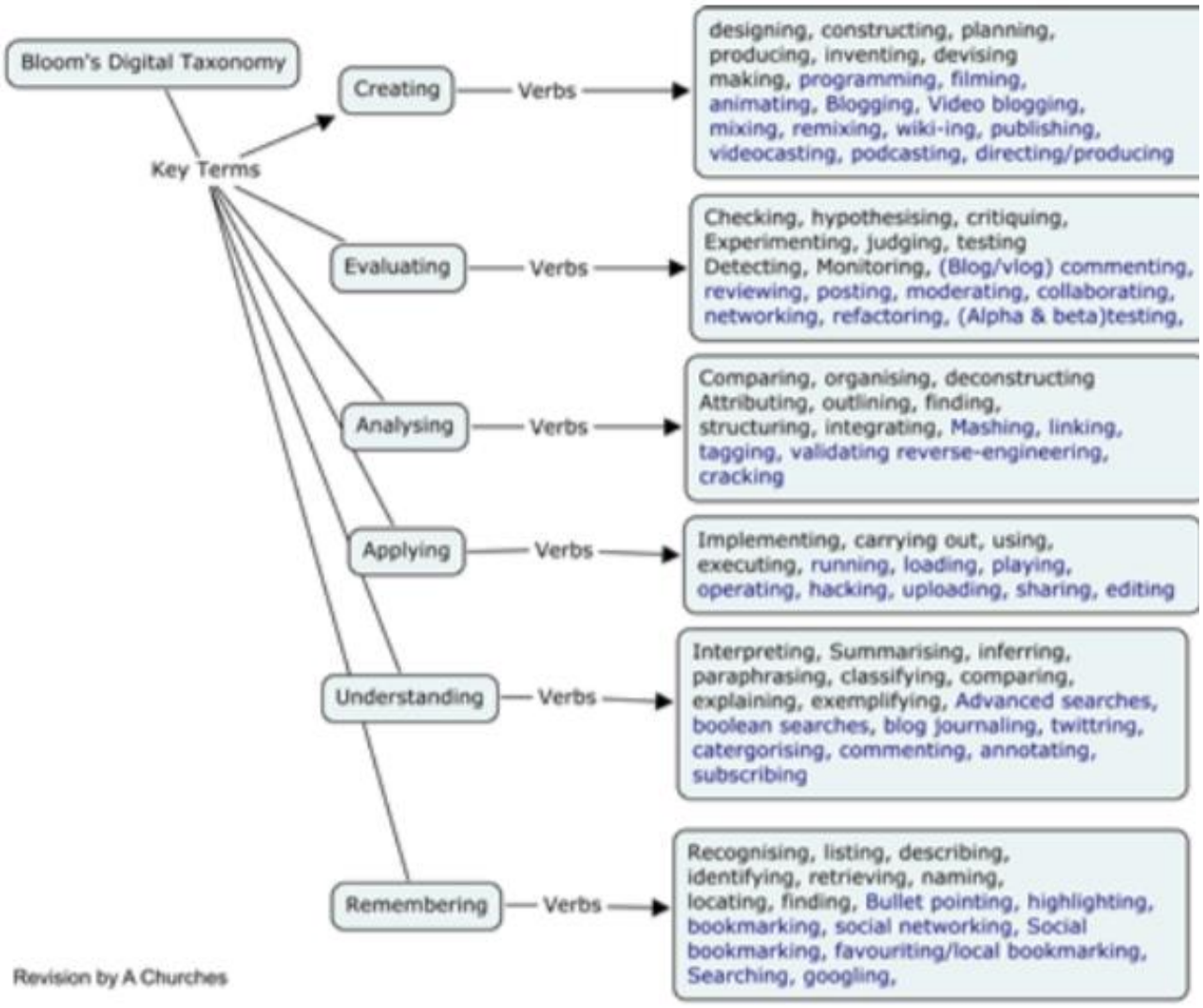


Figure 5: Verbs that show at what level a student's knowledge is (Churches, 2008)

4.2 Applying Bloom's Taxonomy to Information Modelling

To successfully model information systems, it is not enough for students to remember, understand, analyse or even evaluate the information they learn in a modelling class. Students need to reach the pinnacle of Bloom's Taxonomy; they need to create new models with the modelling language(s) they have learned.

A model is a simplified view of a system or reality, graphically showing the complexity of reality through abstraction (Robinson, Arbez, Birta, Tolk & Wagner, 2015; Wand & Weber, 2002; Rodrigues da Silva, 2015; Noran, 2000). Pinggera et al. (2013) describe the creation of a model in three consecutive phases: a *comprehension phase,* a *modelling phase,* and a *reconciliation phase*. During the comprehension phase, a modeller creates a mental representation of the problem or system and develops a mental solution. Knowing the problem and solution, the modeller utilizes his knowledge of a modelling language to model a solution. Having created a draft, the modeller makes the model more understandable, for example by adding or changing labels, in the reconciliation phase. Each system can be modelled in several ways and each modeller can go through the phases differently (cf. Figure 6 below), the important observation is that modelling is an inherently *creative* process that assumes mastery of modelling languages before it can be successfully executed.

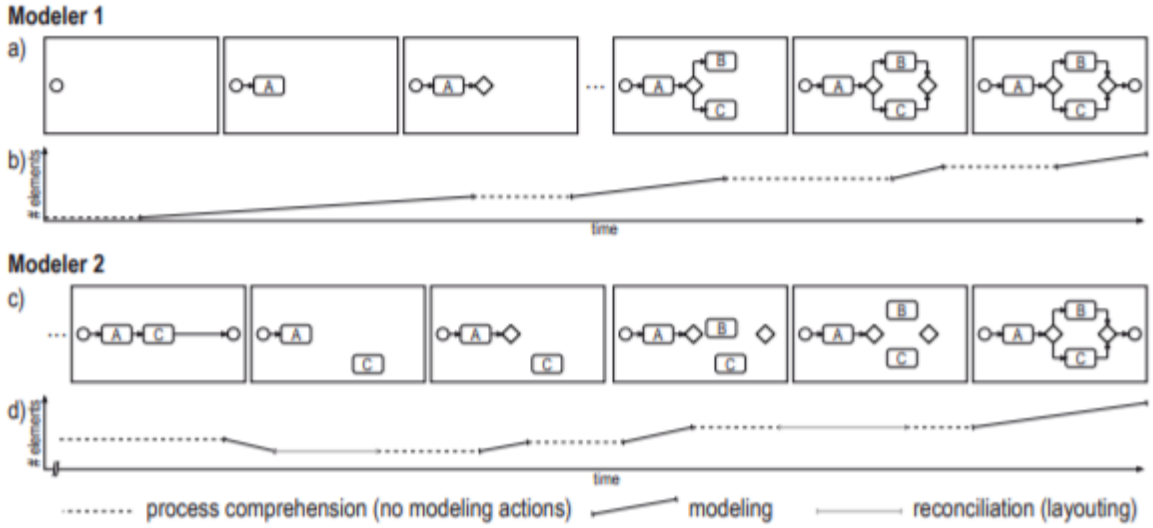


Figure 6: Two modellers modelling in different phases (Pinggera et al., 2013)

Because of the complexity of modelling languages, and the pressure on students and universities to deliver results, information systems modules risk exposing students to cognitive overload.

Cognitive overload decreases the effectiveness of student learning activities and can impact their study results or even their self-confidence. In contrast, effective management of student cognitive load might in theory even help increase the grades and number of successful graduates of modelling systems classes – helping students, universities and businesses alike.

5. From Theory to Practice: A Case Study

However, does this theory also pan out in practice? This can only be discovered by running a case study.

5.1 Utrecht University module Information Systems

Utrecht University offers the module Information Systems (Informatiesystemen), which is mandatory for first-year bachelor's students specialising in Information Sciences (Informatiekunde). The module focuses on how to analyse and model information flows, and how to specify and implement a process-aware information system. In total, around 200 students take the class every year.

An ability to understand and create models is an essential part of information systems development (Wand, Monarchi, Parsons, & Woo, 1995). During the Bachelor Information Sciences, students are taught several modelling languages. These languages each have different notations, rules, strengths and weaknesses. Appendix A shows five commonly used languages and a comparison between them. In the course Information Systems, students are taught the modelling languages BPMN and Petri Nets. A thorough explanation of these languages can be found in appendices A and D. While students have been taught BPMN in earlier modules, they will, in general, have no prior knowledge of Petri Nets.

Information Systems classes take place three days a week during the last 10-week period of every UU academic year. Each week contains three different sections:

1. Lectures
2. Project development
3. Modelling Sessions

During lectures, the lecturer presents the theoretical background of the modelling languages BPMN and Petri Nets as well as other information deemed important for the subject matter at hand. Lectures are conducted with the entire class present. Project development emphasises group work. It is a timeslot where students work together on a prescribed problem for which they have to develop a solution.

The modelling sessions offer time to put into practice the information taught during lectures earlier in the week. All students are divided into groups and scheduled for a session. Each modelling session consists of two problems. Students are given 25 minutes to module a solution to each problem, followed by 5 minutes to individually reflect on each model by using predetermined questions and finally, after reflection, a 15-minute group discussion. A typical question that might be asked during the modelling sessions goes as follows:

> "A group of frogs jump from a bridge into a pond. They swim to one of two sides and return to the bridge to jump again. A princess looking for true love picks up every third frog on the bridge, kisses the frog, and then returns the frog onto the bridge.
>
> Imagine the group consists of five frogs. What happens to your model? How does your model change if the group has two or four frogs? Is it possible that each frog can be kissed at least once by the princess?" (Utrecht University, z.d.)

The material and the set-up of the Information Systems module have been developed with Bloom's Taxonomy in mind but without regard to student cognitive (over)load.

5.2 Problem Description

From a cognitive load perspective, the course set-up still looks far from perfect. For example, students are currently asked to answer questions using discovery-based problem-solving techniques from the very start of every session. As described in section 3.3.1 above, discovery-based problem-solving techniques are not the most effective means to learn new modelling languages – especially as a beginner. Furthermore, the lecture slides and modelling questions have not been integrated into a single information framework. As a result, students practising modelling languages need to integrate two separate sources of information (problem description and lecture slide), causing unnecessary extra cognitive load (cf. section 3.3.2 above). Lastly, students are immediately asked to focus on achieving specific modelling outcomes – instead of focusing on performing the right steps to make a coherent model. This goal-focused effect unnecessarily increases the intrinsic cognitive load following the goal-free effect (cf. section 3.1.1 above).

5.3 Hypothesis

How might cognitive load theory be used to optimize the learning process of students studying the module Information Systems at Utrecht University? Based on the above description, we can hypothesize that the learning process of students studying Information Systems can be improved by reworking the class set-up to incorporate insights from cognitive load theory. Two hypotheses have been created, shown in Table 1.

Table 1: Hypotheses

H0	The Utrecht University Information Systems module is efficiently set up to avoid cognitive overload
H1	By restructuring the Information Systems module, cognitive load can be reduced and the learning process of students studying Information Systems can be improved

5.4 Method

In practice, it is very difficult indeed to measure changes in cognitive load. First, as discussed in chapters 2 and 3, the exact intrinsic cognitive load of any given problem is not objective but subjective: it differs depending on the long-term memory schemas at the disposal of a student. A more experienced student will experience less cognitive load than a true beginner in any subject. The most effective way to – as much as possible – prevent measurement impairment because of different knowledge bases, is to make sure that individuals start a course with the same level of background knowledge (Sweller, 1994). Second, while methods exist to measure the total cognitive load, it has proven impossible to measure intrinsic, extrinsic and germane load separately (Paas et al., 2003).

That said, the measures of cognitive load that do exist can indicate if a student is being challenged at an optimal level.

5.4.1 Measuring Cognitive Load.

Three main ways of measuring cognitive load exist.

First, cognitive load can be measured through the use of questionnaires which ask participants about their perceived level of mental effort, fatigue and/or frustration. However, these questionnaires – also known as subjective rating scales of cognitive load – are very subjective representations of cognitive load that do not necessarily reflect actual student learning progress (De Jong, 2010).

Second, recent scientific advances have led to attempts to create more objective physiological measures of cognitive load. These techniques measure heart activity, brain activity and track eye activity while solving problems. The idea is that physiological aspects change and can be measured depending on cognitive load. Physiological measures are not yet fully developed and remain inconsistent for now, but they show potential to become a more reliable method of measuring cognitive load in the future (Leppink et al., 2015)

A third method is to measure cognitive load indirectly. Among the approaches that follow this track, two stand out. First are simple indirect measures of cognitive load which compare test results after performing certain exercises to test results before these exercises were performed. This type of measurement can take on various guises. For example, it is possible to indirectly measure cognitive load by evaluating the speed at which students complete exercises and then improve on their test scores (Leppink et al., 2015). A different approach is to baseline cognitive load by including a secondary task to any primary exercise a person is performing. By introducing a secondary task, an individual would have to spend a portion of his or her cognitive load on performing said task. If the primary task takes less load to perform then there is more room to perform the secondary task (Leppink et al., 2015; De Jong, 2010). By introducing the same secondary task to different problems, researchers can theoretically measure the relative cognitive load which different primary tasks cost.

While there are therefore different methods of measuring cognitive load, they all fall prey to several flaws. First, as discussed before, it is currently impossible to measure individual cognitive load levels. It is only possible to measure the total cognitive load of an individual (Paas et al., 2003). Second, most methods are performed after a task has been completed, which may be less precise than measurements taken while performing the task (De Jong, 2010). Third, while performing tasks, individuals are confronted with several variables which are near-impossible to control – even in an experimental environment. For example, people's cognitive load can be affected by their emotions (Fraser et al., 2012), their individual characteristics (Chen, Pedersen, & Murphy, 2011), their motivation, their environment (Choi, van Merriënboer, & Paas, 2014), and secondary skills like writing or typing (Tarafdar, Pullins, & Ragu-Nathan, 2014).

In addition, not all mentioned measuring methods are equally suitable to apply in an educational setting. Given a real-life class setting, it would be unethical to purposely hinder students in their learning. Yet by introducing secondary tasks for students to perform while studying would purposely distract them from learning their primary studies by spending a portion of their attention (cognitive load) on conducting an unrelated secondary task. Likewise, physiological measures would require equipment that could prove costly to acquire and distracting in a classroom setting – potentially impacting the real-life study outcomes of students in a negative manner.

In contrast to secondary tasks or physiological measurements, subjective rating scales (questionnaires) and indirect measurements are much less likely to unduly impact student ability to learn in a classroom setting. Both measurements are easily scalable, relatively cheap to conduct, and could offer reasonably reliable results about student cognitive load. When comparing these two, indirect measurements are less dependent on voluntary student participation and require less manual processing of information than questionnaires, while questionnaires offer an opportunity to gain more detailed experiment than is possible with indirect measurements.

Within the scope of this research and the university setting to which it applies, indirect measurements seem like the most promising avenue to inquire if and how cognitive load theory might be used to optimize the learning process of students studying information technology. Several indicators might be used to analyse student improvement, but the most straightforward among these is an analysis of the percentage of students that pass the course, and/or the average grade by which students pass the class. In such an experiment, the application of cognitive load interventions (as described in section 3.3) acts as an independent variable, affecting the dependent variable, i.e. grades of students.

Student grades are well-suited to act as dependent variable give that they should offer a rough reflection of their competence of the subject matter in any given subject. To fail a class means to not have shown a minimum required grasp of the material which has been taught. To pass a class with high marks indicates that the student has shown great proficiency in the subject

matter at hand. In general, if the grade average structurally improves this can indicate that students have obtained a better grasp of the knowledge that has been taught.

5.4.2 Controlling for extraneous variables:

Of course, class grades are affected by much more than just the style and means of instruction. Several precautions should therefore be taken to control for as many extraneous variables as possible.

As noted in section 3.1.1, the exact intrinsic cognitive load caused by any given problem is not objective but subjective: it differs depending on the long-term memory schemas at the disposal of a student. As a result, it is incredibly difficult to measure the intrinsic cognitive load unless we ensure that the case study subjects start a class with the same level of background knowledge (Sweller, 1994).

As noted in section 5.1, the module Information Systems involves lectures on two programming languages: BPMN and Petri Nets. However, students have been taught BPMN in earlier modules, they will, in general, have no prior knowledge of Petri Nets. To measure cognitive load accurately, this case study should therefore focus on the progress made with the modelling language ‘Petri Nets’.

Furthermore, as noted in section 4.2, in order to successfully model information systems, it is not enough for students to remember, understand, analyse or even evaluate the information they learn in a modelling class. Students need to reach the pinnacle of Bloom’s Taxonomy; they need to create new models with the modelling language(s) they have learned. To do so, students need to put into practice what they have been taught and show mastery of their subject. Based on the description of cognitive load optimization in section 3.3, hands-on practice sessions offer significantly more opportunities and intervention possibilities than plain lectures. These two observations support a special focus on the module’s modelling sessions (see section 5.1).

Even if we focus our attention specifically on the modelling sessions that teach the modelling language Petri Nets, a series of extraneous variables relate specifically to the students that attend the module. Examples include class size, the course set-up, schedule, means of instruction and instructor. Many of these variables can be controlled for by creating a control group which teaches the module in the same way that it has always been taught, and a research group that is taught the module set-up using the insights of cognitive load theory. These two groups should be of identical class size, can attend the same class lectures and group projects but would follow separate modelling sessions each week.

In order to ethically populate the control group and research group, students starting the course should be made aware of the experiment and while they should in principle be randomly

assigned to either group, any person with a significant preference for one group or the other should be catered in their wishes.

While the use of control groups and random assignment of students to either the control or the research group will help control for many extraneous variables, there remains a risk that either the control group or the research group is above- or below-averagely talented or motivated or otherwise not average. Likewise, if two modelling sessions are not taught by the same instructor or student assistant differences in teaching styles may affect class outcome. If the experiment were only performed once, such a hidden discrepancy could skew its results. Possible anomalies can, however, be controlled for by running the same experiment for several years keeping its parameters (means of group assignment, group sizes, instruction manual, etc.) the same every year.

Within these parameters, we might assume that any results which show a structurally significant difference in the grade average between the research and the control group is the result of the cognitive load intervention performed during the experiment.

5.4.3 Experimental Set-Up

How would we apply this experiment to the UU module Information Systems? Figure 7 offers a schematic overview of an experiment which can test how cognitive load theory might optimize the learning process of students studying information technology. Note that this experimental set-up and its results have not been applied in practice yet.

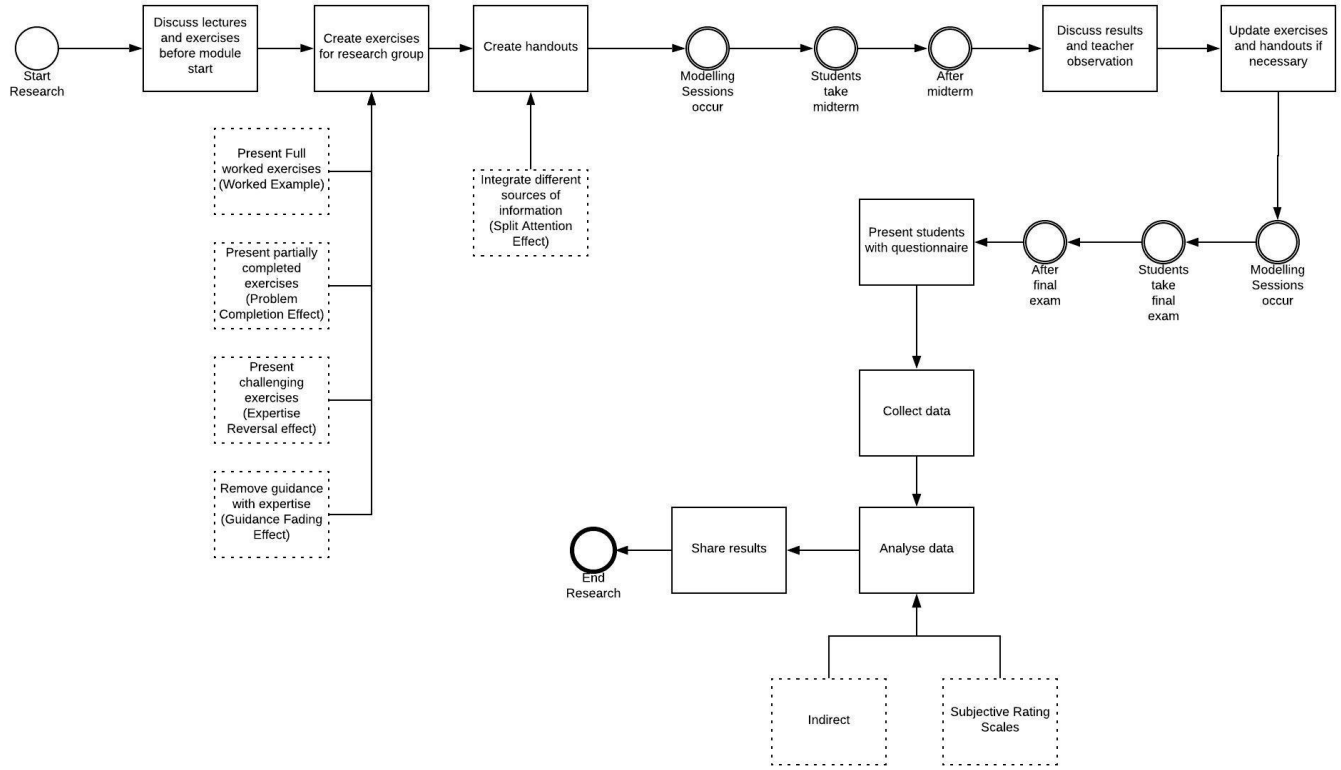


Figure 7: Process of Entire Experiment

Before the start of the module Information Systems, the module coordinator will develop a lesson program for the control group (i.e. lectures and exercises). Barring major program changes, this lesson program can be largely recycled from the module taught in the previous year.

The module coordinator and the research team then take control group program and help to redevelop it in line with cognitive load theory (cf. section 3.3). Examples of how this might be done include:

1) Present a summary of the lectures and the modelling session problems in one single hand-out. By integrating the modelling problems with the class lectures, the need to mentally integrate information from different sources is negated (split-attention effect). As a result, there is no longer any need to apply cognitive load to remember and mentally integrate separate sources of information.
2) Tailor the exercises students need to make during the modelling sessions. Large exercises can be sub-divided and only gradually be made more complex to help students first grasp smaller schemas and only then integrate these into larger schemas. This could be done by presenting small chunks of information and exercises with multiple small goals. (cf. isolated elements effect and sub-goals effect) Students must not have to rely on discovery-based problem-solving, as this costs a lot of cognitive load. As shown in Figure 8 below, it is therefore important that students are presented the right exercises for their knowledge level.

When a student has a schema level of 0, he or she should be confronted with a level 1 problem difficulty, so that the student can make a level 1 schema. It is less effective for a student with a schema level of 0 to face exercises of problem difficulty 2, as he or she lacks the schemas to effectively deal with the exercise. This would force the student to rely on discovery-based problem-solving which will drastically increase their cognitive load. If instead, the student is presented with the correct exercise difficulty in relation with their schema level, the students will effectively create more in-depth schemas. To achieve this, students might first be offered fully worked examples, which require less discovery to solve the problem. Later on, to make sure that they remain engaged and challenged (i.e. prevent the expertise reversal effect), they should be given partially completed exercises. Once students have enough expertise to easily complete partially completed exercises, they can finally be asked to solve full problem exercises. Together, these adjustments should lead to a different program for the research group than for the control group.

These two types of interventions seem most suitable for this experiment because they do not radically change the structure of the class but do apply a significant number of cognitive load theory: one single location of information (split-attention effect), using smaller goals during exercises (sub-goal effect; isolated elements effect), and strategically combining exercises of different difficulty at different stages of the learning process (worked example, Problem completion, and the expertise reversal effect).

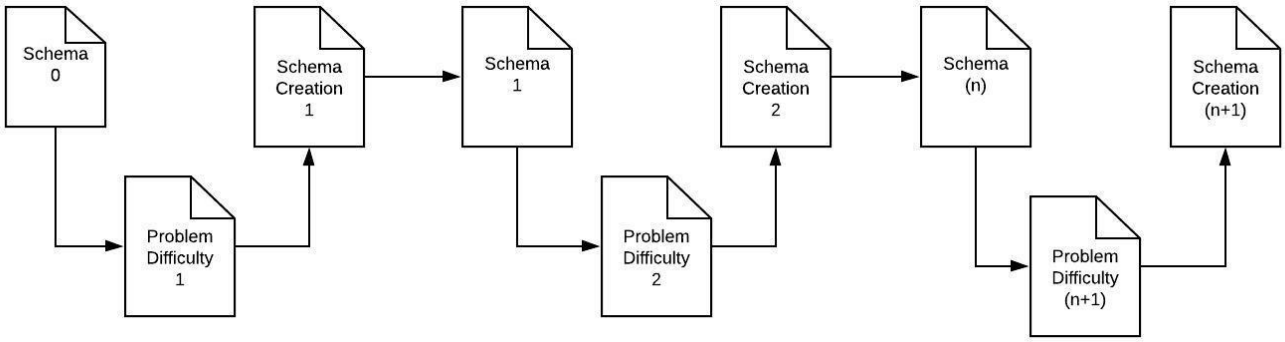


Figure 8: Problem difficulty adjusted to schema level

At the start of the module's first lecture, the researcher explains the experiment to all students and offers them the opportunity to express a strong preference for the control or research group. Two groups of identical size are created for the modelling sessions. All students attend the same class lectures and are randomly assigned to group projects, but they follow separate modelling sessions each week.

During the first modelling session, students have not practised with any exercises. Students will be presented with two types of information. If the students are part of the control group, they will follow the same structure that is currently used. Students are given 25 minutes to solve the exercise, 5 minutes to answer questions about the exercise, and 15 minutes to discuss the

answers as a group. This process is done twice before ending the session. The test group will be presented with different exercises and they will be given handouts First, the students will face worked examples, where they will be given 15 minutes to analyse the model. Hereafter, students are faced with partially completed exercises, where they are given 20 minutes to complete the exercise. Then students are faced with two extra full problem exercises, where they are given 20 minutes for each exercise. Hereafter, students will discuss their results as a group. This is shown in Figure 9.

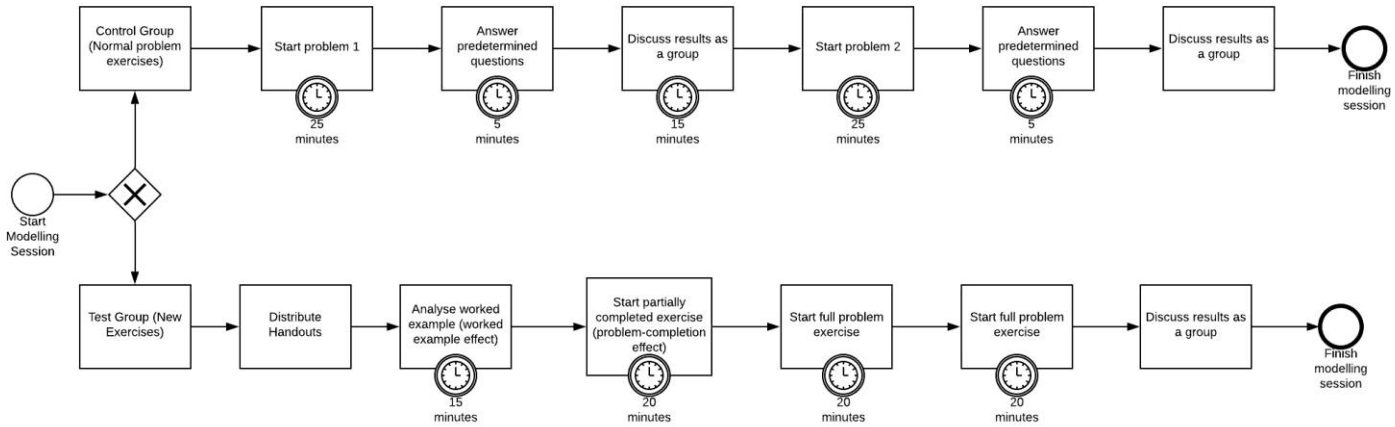


Figure 9: Proposed first modelling session of the module

During the following weeks, excluding the exam weeks, students remain in their experimental group. After the first modelling session and halfway through the course, the research team and coordinator discuss the learning experience and if any changes need to be made to the experiment. These moments are chosen with reason. Discussion after the first week offers an opportunity to adjust to any unforeseen challenges to the experiment. The discussion halfway through the module offers more in-depth feedback as well as the first statistics in the mid-term exam performance.

Alongside the final exam, the researcher will hand out a questionnaire to the students. The students will be asked for their cooperation in filling in this questionnaire after the exam. This questionnaire asks what students thought of the sessions.

Two weeks after the final exam, students will have received their grade. The research team and the coordinator collaborate anonymise the grades and statistically analyse them.

5.5 Results

What might be the results of this experiment? As noted, the experiment has not yet been run. However, the hypothesis of this experiment – (cf. section 5.3) would be confirmed if the experiment offered the following results:

- Significantly more students in the research group than in the control group pass the module.

- Students in the research group attain statistically significant higher grades than students in the control group

5.6 Further Work

The main experiment described above combines many different advised interventions to prove that cognitive load theory can help students more quickly and effectively learn Information Systems modelling languages. However, it is also important to single out each intervention and know its contribution to any effect. As with the main experiment, the effects of these interventions can be measured indirectly (student grades, percentage of students that pass) or with subjective rating scales. However, in order to test each of these interventions individually during a single module, the class would have to be split into five groups (one for each intervention and one control group). Assuming all 200 students participate in the experiment, five groups of 40 students could be created. This does assume that all students are willing to participate in the experimentation.

Figures 10, 11, and 12 show the setup of the experiment of three specific types of individual interventions. The control group stays the same for each experiment. In Figure 10, the difference between the control group and the research group is that the latter receive a clear overview of information located in one place. In Figures 11 and 12, the research group are given smaller, more tailored questions than the control group. These problems are smaller, therefore decreasing the likelihood of student cognitive overload.

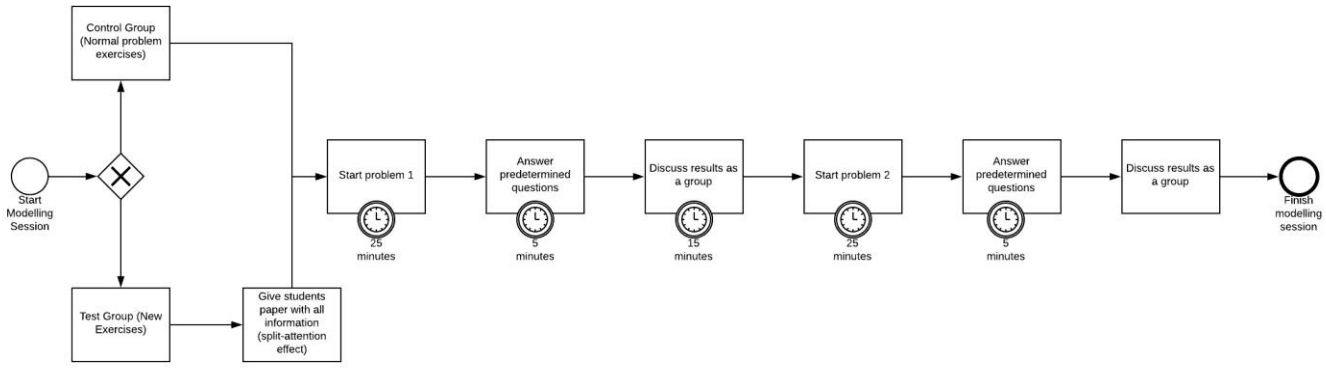


Figure 10: Proposed structure for experiment on split-attention effect

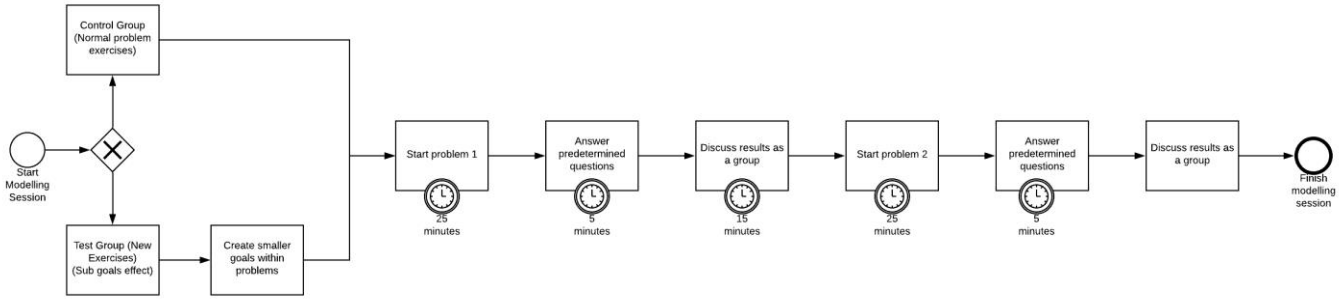


Figure 11: Proposed structure for experiment on sub-goals effect

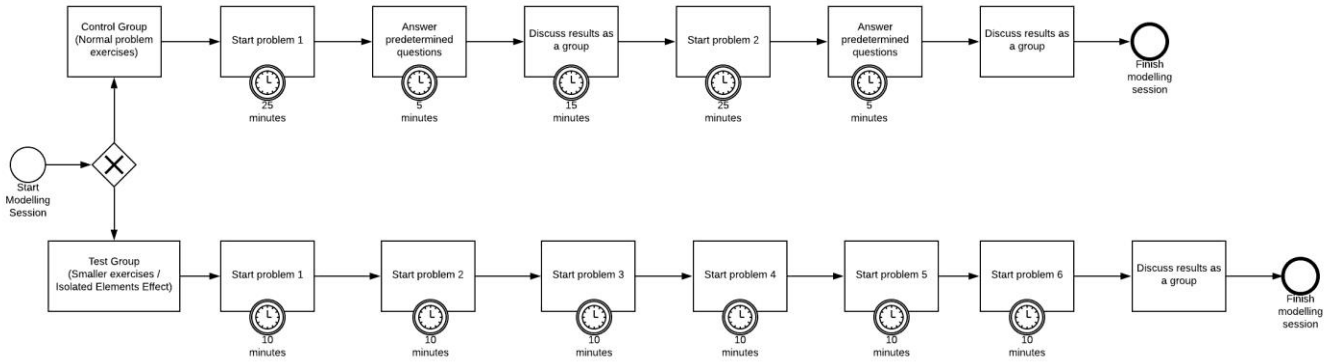


Figure 12: Proposed structure for experiment on isolated elements effect

6. Conclusion

This thesis sought to investigate how cognitive load theory might be used to optimize the learning process of students studying information technology. It does so by investigating the risk of cognitive overload in the module Information Systems, offered to first-year bachelor students Information Sciences at Utrecht University (UU). The module Information Systems aims to teach students about the modelling language Petri Nets. To assist students in their learning, they are obliged to not only attend classic lectures but to also participate in practice sessions (also called modelling sessions).

A literature review on Cognitive Load Theory indicated that simple changes in the manner by which information is taught can help prevent student cognitive overload and help students better grasp more of the information they are being taught in the module Information Systems. By implementing a combination of worked examples, problem-completion examples, and full problems, students could use cognitive load more efficiently – resulting in an improved rate of learning.

To test if this theory also works in practice, a case study was devised around the Utrecht University module Information Systems. This case study focused on proving the hypothesis: *By restructuring the Information Systems module, cognitive load can be reduced and the learning process of students studying Information Systems can be improved*.

The results of this thesis are particularly relevant for coordinators of university modules who seek to efficiently transfer knowledge to a group of students. However, while the proposed interventions theoretically help students grasp and retain the information they learn more effectively, it is necessary to also corroborate these findings with experimental evidence. Unfortunately, a real-life experiment could not take place in the course of writing this thesis because of scheduling issues.

To acquire the necessary empirical proof, the described experiment might be run during the Information Systems module in 2019-2020 – and repeating it in later years. By repeating the data, the coordinator could get a nuanced data set which also controls for intangible factors like IQ, student motivation, etc. In addition, further research might be done to determine the total effect of each proposed intervention, as students are affected by multiple variables that could impede their learning abilities.

Taken together, Cognitive Load Theory can act as a tool which can help universities more predictably satisfy a growing economic demand for individuals with modelling skills while at the same time keeping the perspective, limits and needs of students in mind.

Bibliography

Adams, N. E. (2015). Bloom's taxonomy of cognitive learning objectives. *Journal of the Medical Library Association : JMLA*, *103*(3), 152–153. https://doi.org/10.3163/1536-5050.103.3.010

Atkinson, R. C., & Shiffrin, R. M. (1968). Human Memory: A Proposed System and its Control Processes. *Psychology of Learning and Motivation*, 89–195. https://doi.org/10.1016/S0079-7421(08)60422-3

Chandler, P., & Sweller, J. (1991). Cognitive load theory and the format of instruction. *Cognition and instruction*, *8*(4), 293-332.

Chen, C.-Y., Pedersen, S., & Murphy, K. L. (2011). Learners' perceived information overload in online learning via computer-mediated communication. Research in Learning Technology, 19(2), 101–116. https://doi.org/10.1080/21567069.2011.586678

Choi, H.-H., van Merriënboer, J. J. G., & Paas, F. (2014). Effects of the Physical Environment on Cognitive Load and Learning: Towards a New Model of Cognitive Load. Educational Psychology Review, 26(2), 225–244. https://doi.org/10.1007/s10648-014-9262-6

Churches, A. (2008). Bloom's taxonomy blooms digitally. *Tech & Learning*, *1*, 1-6.

Clark, R. C., Nguyen, F., & Sweller, J. (2011). *Efficiency in learning: Evidence-based guidelines to manage cognitive load*. John Wiley & Sons.

David, R., & Alla, H. (1994b). Petri nets for modelling of dynamic systems. *Automatica*, *30*(2), 175–202. https://doi.org/10.1016/0005-1098(94)90024-8

De Jong, T. (2010). Cognitive load theory, educational research, and instructional design: some food for thought. *Instructional science*, *38*(2), 105-134.

Desel, J., & Reisig, W. (1996, September). Place/transition Petri nets. In *Advanced Course on Petri Nets* (pp. 122-173). Springer, Berlin, Heidelberg.

Dumas, M., Rosa, M. L., Mendling, J., & Reijers, H. A. (2013). *Fundamentals of Business Process Management*. https://doi.org/10.1007/978-3-642-33143-5

Forehand, M. (2010). Bloom's taxonomy. *Emerging perspectives on learning, teaching, and technology*, *41*(4), 47-56.

Fraser, K., Ma, I., Teteris, E., Baxter, H., Wright, B., & McLaughlin, K. (2012). Emotion, cognitive load and learning outcomes during simulation training. Medical Education, 46(11), 1055–1062. https://doi.org/10.1111/j.1365-2923.2012.04355.x

Geambaşu, C. V. (2012). BPMN vs UML activity diagram for business process modelling. *Accounting and Management Information Systems*, *11*(4), 935-946.

Hutton, E. L. (2016). *Xunzi: The Complete Text*. Referenced from https://books.google.nl/books?id=IW6YDwAAQBAJ&pg=PA5&dq=Learning+proceeds+until+death+and+only+then+does+it+stop&hl=nl&sa=X&ved=0ahUKEwjX7Ljwk-rjAhUOGewKHcJwBl8Q6AEIKTAA#v=onepage&q=Learning%20proceeds%20until%20death%20and%20only%20then%20does%20it%20stop&f=false

Krathwohl, D. R. (2002). A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice*, *41*(4), 212–218. https://doi.org/10.1207/s15430421tip4104_2

Leppink, J., van Gog, T., Paas, F., & Sweller, J. (2015). 18 Cognitive load theory: researching and planning teaching to maximise learning. *Researching medical education*, 207.

Lusk, S., Paley, S., & Spanyi, A. (2005). The evolution of business process management as a professional discipline*. BP Trends*, *20*, 1-9.

Mason, R., Seton, C., & Cooper, G. (2016). Applying cognitive load theory to the redesign of a conventional database systems course. *Computer Science Education*, *26*(1), 68–87. https://doi.org/10.1080/08993408.2016.1160597

Mayer, R. E. (2005). Cognitive theory of multimedia learning. *The Cambridge handbook of multimedia learning*, *41*, 31-48.

Mili, H., Tremblay, G., Jaoude, G. B., Lefebvre, É., Elabed, L., & Boussaidi, G. E. (2010). Business process modelling languages. *ACM Computing Surveys*, *43*(1), 1–56. https://doi.org/10.1145/1824795.1824799

Noran, O. S. (2000). Business modelling: UML vs. IDEF. *School of Computing and Information Technology, Griffith University*.

Nüttgens, M., Feld, T., & Zimmermann, V. (1998). Business Process Modelling with EPC and UML: Transformation or Integration? *The Unified Modelling Language*, 250–261. https://doi.org/10.1007/978-3-642-48673-9_17

Paas, F., & Ayres, P. (2014). Cognitive load theory: A broader view on the role of memory in learning and education. *Educational Psychology Review*, *26*(2), 191-195.

Paas, F., Tuovinen, J. E., Tabbers, H., & Van Gerven, P. W. M. (2003). Cognitive Load Measurement as a Means to Advance Cognitive Load Theory. *Educational Psychologist*, *38*(1), 63–71. https://doi.org/10.1207/S15326985EP3801_8

Pereira, J. L., & Silva, D. (2016). Business Process Modelling Languages: A Comparative Framework. *New Advances in Information Systems and Technologies*, 619–628. https://doi.org/10.1007/978-3-319-31232-3_58

Pinggera, J., Furtner, M., Martini, M., Sachse, P., Reiter, K., Zugal, S., & Weber, B. (2013). Investigating the Process of Process Modelling with Eye Movement Analysis (Full Paper). In *Business Process Management Workshops: BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012, Revised Papers* (pp. 438–450). https://doi.org/10.1007/978-3-642-36285-9_46

Reisig, W. (2013). *Understanding petri nets: modelling techniques, analysis methods, case studies*. Heidelberg: Springer.

Robinson, S., Arbez, G., Birta, L. G., Tolk, A., & Wagner, G. (2015). CONCEPTUAL MODELLING: DEFINITION, PURPOSE AND BENEFITS. *Proceedings of the 2015 Winter Simulation Conference*, 2812–2826. Geraadpleegd van https://www.informs-sim.org/wsc15papers/277.pdf

Rodrigues da Silva, A. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, *43*, 139–155. https://doi.org/10.1016/j.cl.2015.06.001

Russell, N., van der Aalst, W. M., Ter Hofstede, A. H., & Wohed, P. (2006, January). On the suitability of UML 2.0 activity diagrams for business process modelling. In *Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling-Volume 53*(pp. 95-104). Australian Computer Society, Inc..

Shen, H., Wall, B., Zaremba, M., Chen, Y., & Browne, J. (2004). Integration of business modelling methods for enterprise information system analysis and user requirements gathering. *Computers in Industry*, *54*(3), 307–323. https://doi.org/10.1016/j.compind.2003.07.009

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction*, *4*(4), 295–312. https://doi.org/10.1016/0959-4752(94)90003-5

Sweller, J. (2016). Cognitive load theory, evolutionary educational psychology, and instructional design. In *Evolutionary perspectives on child development and education* (pp. 291-306). Springer, Cham.

Tarafdar, M., Pullins, E. Bolman., & Ragu-Nathan, T. S. (2014). Technostress: negative effect on performance and possible mitigations. Information Systems Journal, 25(2), 103–132. https://doi.org/10.1111/isj.12042

Ticket vending machine UML activity diagram example describing behavior of the Purchase Ticket use case. (2014, 18 januari). Referenced on 24 July 2019, from https://www.uml-diagrams.org/ticket-vending-machine-activity-diagram-example.html?context=activity-examples

Tricot, A., & Sweller, J. (2014). Domain-specific knowledge and why teaching generic skills does not work. *Educational psychology review*, *26*(2), 265-283.

Utrecht University. (z.d.). Modelling session 2, Information Systems. Referenced on 26 Juli 2019, from https://uu.blackboard.com/webapps/blackboard/execute/displayLearningUnit?course_id=_117690_1

van der Aalst, W., & Stahl, C. (2011). *Modelling Business Processes: A Petri Net-Oriented Approach*. Referenced from http://web.b.ebscohost.com.proxy.library.uu.nl/ehost/detail/detail?vid=0&sid=b4f56393-c3e8-43e8-b861-e8ad6aea2f79%40pdc-v-sessmgr05&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#AN=1683164&db=nlebk

Vogel-Walcutt, J. J., Gebrim, J. B., Bowers, C., Carper, T. M., & Nicholson, D. (2010). Cognitive load theory vs. constructivist approaches: which best leads to efficient, deep learning? *Journal of Computer Assisted Learning*, *27*(2), 133–145. https://doi.org/10.1111/j.1365-2729.2010.00381.x

Wand, Y., & Weber, R. (2002). Research Commentary: Information Systems and Conceptual Modelling—A Research Agenda. *Information Systems Research*, *13*(4), 363–376. https://doi.org/10.1287/isre.13.4.363.69

Wand, Y., Monarchi, D. E., Parsons, J., & Woo, C. C. (1995). Theoretical foundations for conceptual modelling in information systems development. *Decision Support Systems*, *15*(4), 285–304. https://doi.org/10.1016/0167-9236(94)00043-6

White, S. A. (2004). Introduction to BPMN. *Ibm Cooperation*, *2*(0), 0.

Zugal, S., Pinggera, J., & Weber, B. (2011). Assessing process models with cognitive psychology. *Enterprise modelling and information systems architectures (EMISA 2011)*.

Appendix A – Modelling Languages

There is a copious amount of modelling languages created to model business processes. According to Pereira & Silva (2016), the five most prominent business process modelling languages are Business Process Model and Notation, Event-driven Process Chain, Unified Modelling Language - Activity Diagram, Integration DEFinition, and Role Activity Diagram. These languages will be briefly discussed in this section. As BPMN and UML-AD are the most used languages, they will be discussed using examples. Hereafter the languages will be compared. Note that this thesis is by no means an in-depth review of these modelling languages. For more in-depth information regarding each modelling language, refer to the designated sources.

A.1 Business Process Model and Notation

Business Process Model and Notation, also known as BPMN, is a graphical notation that is a widely used standard for modelling business processes (Dumas, Rosa, Mendling, & Reijers, 2013; Mili et al., 2010). The main goal of BPMN is to provide a notation that is understandable for all business users (Mili et al., 2010; White, 2004). BPMN was updated in 2011 and is currently in version 2.0 (Dumas et al., 2013). The notation contains over 100 different symbols, but users can create in-depth models with just a handful of symbols (Dumas et al., 2013). These symbols are shown in a Business Process Diagram, or BPD (White, 2004).
According to White (2004), there are four basic categories of elements, these being *Flow Objects, Connecting Objects, Swimlanes,* and *Artifacts*. These categories will be described below.

A.1.1 Flow Objects

The Flow Objects consists of three different elements, these being *Events, Activities,* and *Gateways.* Events are notated as circles, indicating that something has occurred during the business process.

Events can be seen in three versions, where a start event is shown as a clear circle, an intermediate event is shown as a circle within another circle, and an end event, where the circle has a dark outline (Dumas et al., 2013; White, 2004), This is also shown in Figure 13.

Figure 13: Different Types of events (White, 2004)

Activities represent something a company does. Activities are notated as round-corner rectangles (Dumas et al., 2013**;** White, 2004). These activities can either be a task or a sub-process, where a task is an independent activity, and a sub-process represents a smaller group of elements. A sub-process is represented as a round-corner rectangle with a plus sign in its centre. Figure 14 shows an activity (White, 2004).

Figure 14: An Activity (White, 2004)

Lastly, gateways are used to determine the flow of the process. For example, if a process splits into two flows, representing a simultaneously conducted action, a gateway is used. Gateways are also used for decision making. For example, if a process can only go one of several ways or multiple flows are needed to combine before the process can continue. The gateway can either contain a plus sign or an X-sign. A gateway is depicted as a diamond, shown in Figure 15 (White, 2004).

Figure 15: A gateway (White, 2004)

A.1.2 Connecting Objects

A business process model cannot be made by only using Flow Objects, as the relations between these objects are currently unclear. Connecting Objects are used to create the relations between these objects. Like the Flow Objects, there are three connecting objects, being *Sequence Flow, Message Flow,* and *Association.*

As the name suggests, a sequence flow shows the sequence of the diagram. The flow is depicted with a solid line and a solid arrowhead (Dumas et al., 2013; White, 2004). Like the sequence flows, the name message flow depicts the function of the flow. Depicted by a dashed line with an open arrowhead, message flows show the flow of messages between two parties or participants within a diagram.

Lastly, associations. Depicted by a dotted line with a line arrowhead, an association is “used to associate data, text, and other Artifacts with flow objects.” (White, 2004, p.3). These flows are shown in Figure 16.

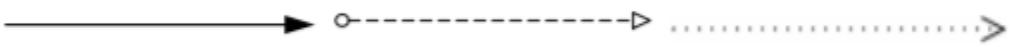


Figure 16: Different types of flow. From left to right: Sequence Flow, Message Flow, and Association (White, 2004)

A.1.3 Swimlanes

By using Flow Objects and Connecting Objects a modeller can create a high-quality BPMN model. However, in some cases a model becomes exceptionally large, containing multiple

participants. This would cause confusion, something BPMN was designed not to create. Swimlanes were created to prevent this type of confusion. There are two different objects within a swimlane, called *Pool* and one or multiple *Lanes*.

A pool is depicted as a large rectangle, wherein the elements of the diagram are shown. A Pool represents a participant in the diagram. Therefore, it is possible to have several pools in one diagram. A Lane is a small part of a pool and is used to “organize and categorize activities” (White, 2004, pp.4). The notation is shown in Figure 17.

Figure 17: Pools and Lanes (White, 2004)

A.1.4 Artifacts

The last category of elements is Artifacts. Currently, Artifacts come in three versions, being *Data Objects, Groups* or *Annotations*. These can be added to add context for the reader of the diagram.

A Data Object is an Artifact that shows what a process consumes or produces to work. For example, during software engineering, requirements are collected to create a complete program for the stakeholder. A by-product of the requirements phase is a document of requirements description. This by-product can be added in a diagram to enhance the context for the reader. Data Objects are connected to activities via associations (White, 2004).

A Group is only used to enhance the readability and understandability of diagrams. As the name suggests, a Group connects different elements. However, a Group function does not alter the sequence flow, it only functions to enhance the understandability of the reader (White, 2004).

Lastly, Annotations. These are used by the modeller to give extra context for the reader via text. Annotations are connected to other elements via associations (White, 2004).

A.1.5 Example

As Dumas et al. (2013) state “A basic BPMN model includes simple activities, events, gateways, data objects, pools, and lanes.” (Dumas et al., 2013, p.109) A modeller can make all forms of models, all varying in depth and complexity. For example, an order processing process can be modelled in several methods. A simplified diagram could contain a few elements, as shown in Figure 18.

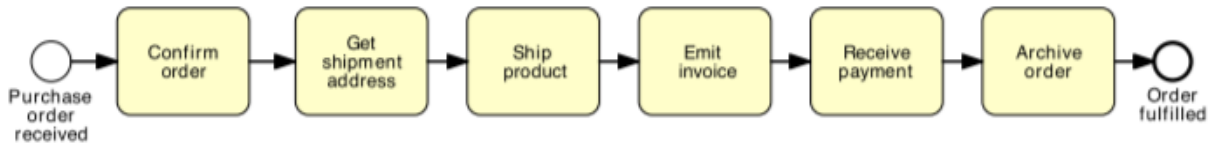


Figure 18: A simplified order processing diagram (Dumas et al., 2013)

However, it is also possible to model the same process in a more in-depth manner, as shown in Figure 19.

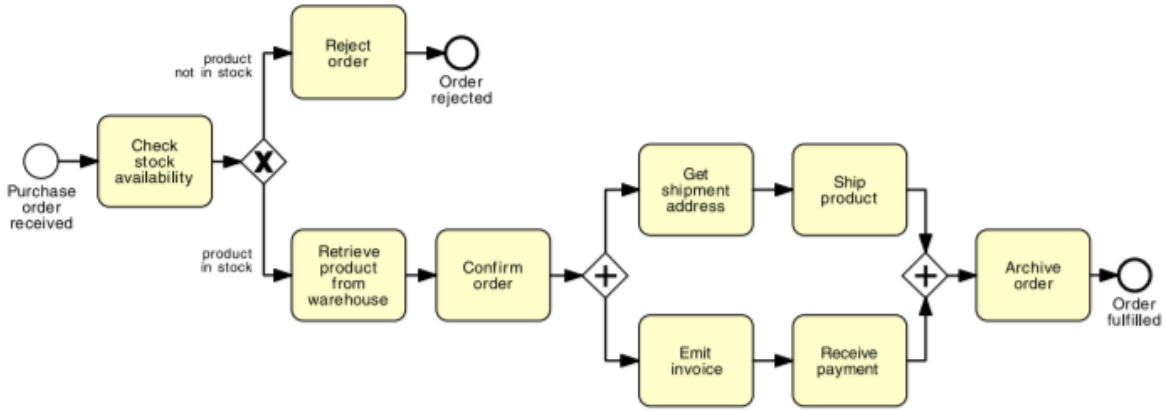


Figure 19: An elaborated order processing diagram (Dumas et al., 2013)

This diagram already shows more complexity, but a modeller can choose to elaborate even further. An example is shown in Figure 20.

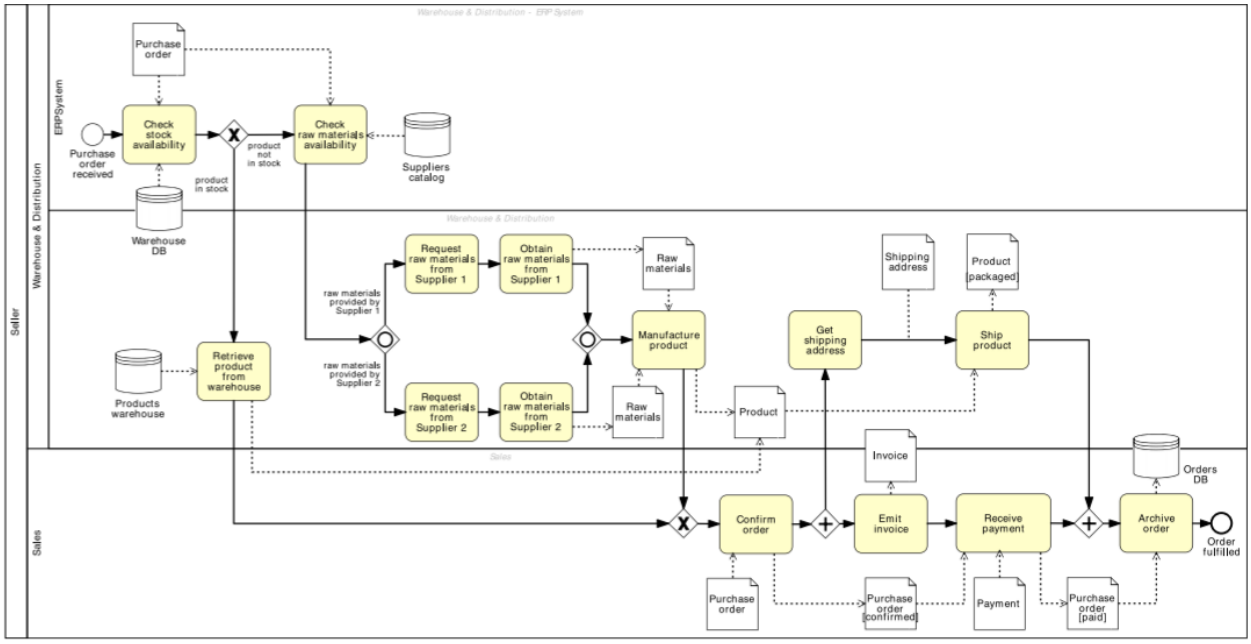


Figure 20: An in-depth, elaborated order processing diagram (Dumas et al., 2013)

As the figures show, BPMN models can become quite complex rather quickly. Depending on the process being modelled and the goal of the modeller, BPMN models can be used to present a wide variety of complex models.

A.2 Event-driven Process Chain

Another business process modelling language is the Event Process Chains language (EPC). Developed by researchers at the University of Saarland and SAP, the EPC is used to "describe business processes at the informal business level, and are meant to support business users rather than formal manipulation." (Mili et al., 2010, p.20). An EPC model consists of business functions, which are triggered by a sequence of events (Nüttgens, Feld, & Zimmermann, 1998). The three components of EPC are 1) an event, 2) a function, and 3) a connector (Mili et al., 2010). These components have some restrictions, as an event can have a maximum of one inbound and one outbound arc, or flow. A function can have exactly one inbound arc and one outbound arc. Connectors are less restricted, as these can have either multiple inbound arcs and one outbound arc, or one inbound arc and multiple outbound arc (Mili et al., 2010). These connectors can be an 'and', 'or' or 'exclusive or'. This gives an EPC the possibility to illustrate complex business systems (Nüttgens et al., 1998).

Researchers have tried to translate EPCs into Petri Nets and use the high-quality analysis of the Petri Net. However, EPCs use ambiguous semantics, which makes it too difficult to translate to Petri Nets (Mili et al., 2010). Whilst EPCs are less rich compared to RADs and Petri Nets, they are widely used in commercial products (Mili et al., 2010).

A.3 Unified Modelling Language - Activity Diagram

Like BPMN, UML-AD, short for Unified Modelling Language - Activity Diagrams, can be used to model processes. The main unit of Activity Diagrams is the element of Action, which can modify the state of the system. Within the language of UML, there are more than 40 unique action types (Russel, Van der Aalst, Ter Hofstede, & Wohed, 2006). These action types are specific to certain situations, and generally, just a few types are used. Like BPMN, an Activity Diagram consists of nodes, activities, and arcs. An example of a UML Activity Diagram is shown in Figure 21. The diagram is split into three sections using swimlanes, each lane representing a participant. Starting with the initial node, represented with a black circle, the diagram describes actions taken with action nodes, represented with rectangles. These are connected by arcs. Gateways are shown in the middle lane, where the process can split into two ways. The flow of the process can be controlled with these gateways. Eventually, the process reaches its end, represented with a partially filled-in circle, called the final node.

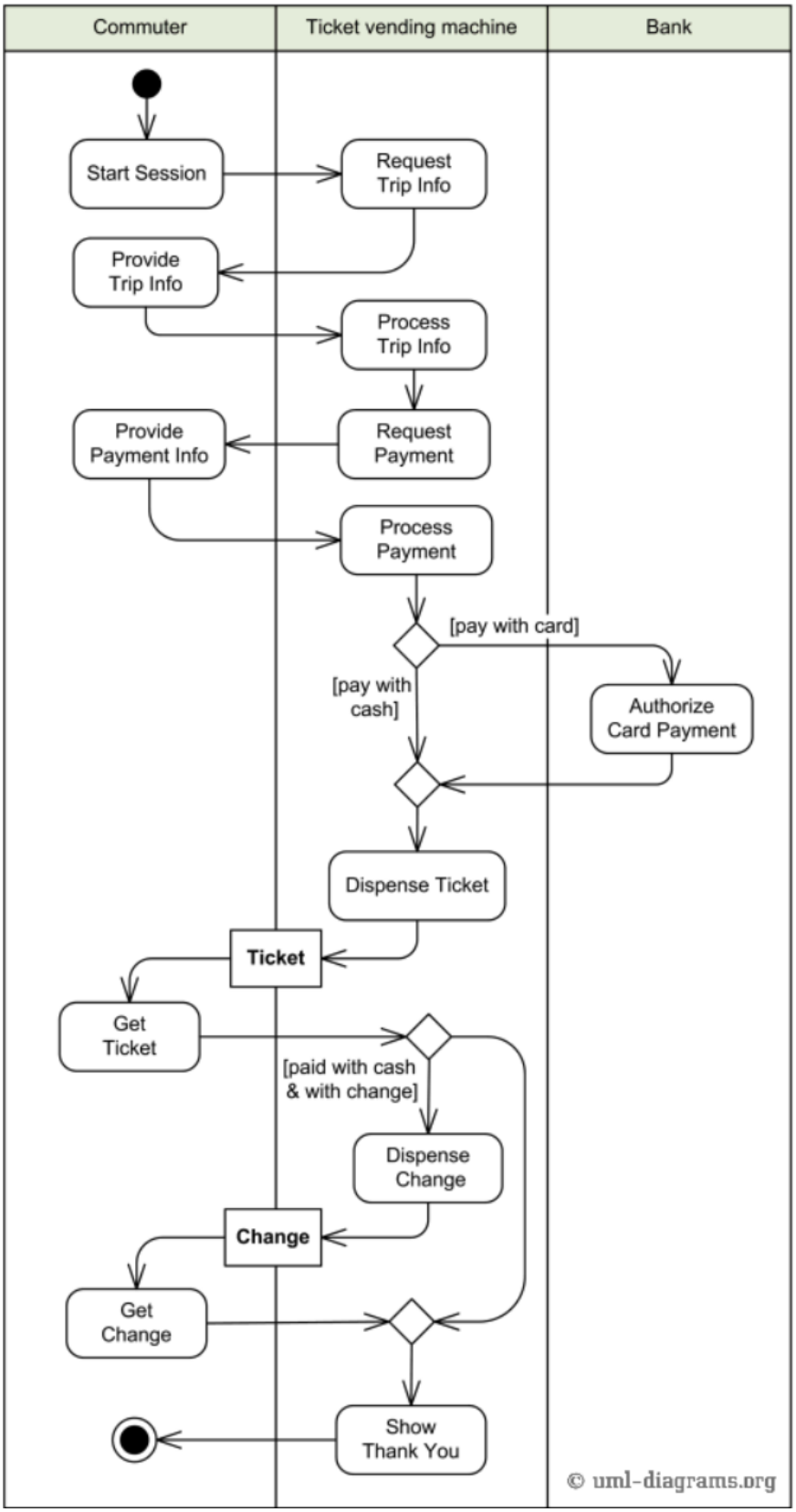


Figure 21: Example of UML-AD describing Purchase Ticket use case behaviour ("Ticket vending machine UML activity diagram example describing behavior of the Purchase Ticket use case.", 2014)

The elements used in UML-AD are very similar to the elements used in BPMN. A comparison is shown in Figure 22 in Appendix C. However, UML contains more elements that can describe certain situations. These are described with Workflow control patterns (Russel et al., 2006). There are at least 20 workflow control patterns that can be used to describe situations; however, these are not within the scope of this paper. For more information, see (Russel et al., 2006) and (Geambaşu, 2012).

A.4 Integration DEFinition Family

One of the largest modelling families is the IDEF family. The IDEF family was created to address the needs of the system design and analysis domain. During the 1970s, The American Air Force Integrated Computer-Aided Manufacturing (ICAM) program designed the IDEF family to address these needs (Mili et al., 2010**;** Noran, 2000). Since its integration, the IDEF have become widespread in use for information system analysis and design, as the models were easy to understand by the business clients (Shen, Wall, Zaremba, Chen, & Browne, 2004).

IDEF originally consisted of three methods, namely the IDEF0, designed for functional modelling; IDEF1, designed for information modelling; and IDEF 2, designed for dynamic modelling (Mili et al., 2010**;** Noran, 2000). Later the methods IDEF 3, designed to address the shortcomings of IDEF2, therefore providing more; IDEF 4, designed as an object-oriented software design method; and IDEF 5, a "knowledge acquisition and engineering method aiming to support enterprise ontologies" (Noran, 2000, p.19) were created (Mili et al., 2010; Nüttgens et al., 1998). IDEF0, IDEF 1, and IDEF3 will be discussed in more detail.

IDEF0

As stated above, the IDEF0 method is used to create functional models (Mili et al., 2010; Shen et al., 2004; Noran, 2000). As Mili et al. (2010) state: "An IDEF0 model describes what a system does—its function—what controls it, what are its inputs, its outputs, and which services or other functions it needs to perform its function." (Mili et al., 2010, p.13). It contains just two graphical notations, being the activity box and interface arrow (Shen et al., 2004). The models are described precisely due to strict syntactical and semantic rules. However, due to its lack of graphical notation, the IDEF0 models can only describe the functions and the connection between them. Other information, such as logical or sequential relations cannot be described with the models (Shen et al., 2004). An important limitation of the IDEF0 is that the model shows the activation of activities and not the sequences of flow. This shows the abstraction of time, as it is unknown how long a process takes place using only an IDEF0 model (Noran, 2000).

IDEF1/IDEF1x

IDEF1 is a modelling method based on the Entity-Relationship approach and the relation model (Mili et al., 2010), where the intent was to define the information about the objects within a business (Noran, 2000). It is imperative to realize that IDEF1 is not a method to design databases. The IDEF1 method can be used by a business to understand what information it is dealing with (Noran, 2000). It does this in three ways. First, IDEF1 can be used to "identify what information is available in the organization" (Noran, 2000, p.20). Second, it can be used to find

lacking information management, which has been causing problems for the company. Lastly, it can be used to “specify information that needs to be managed in the 'to-be' CIM implementation” (Noran, 2000, p.20). IDEF1 mostly uses three concepts, these being: 1) an entity, also known as an information image; 2) relations, which are the associations between entities; 3) entity and relations classes, which are templates for both the entity and relations (Noran, 2000).

Whilst IDEF1 should not be used to create databases, the method IDEF1x should. IDEF1x is designed to support data modelling and create databases (Mili et al., 2010; Noran, 2000). The method also contains three concepts, these being 1) an entity, 2) an attribute, and 3) a classification structure. To use the IDEF1x model most effectively, it should be used after the information requirements have been defined and the decision to use a relational model was made (Noran, 2000).

IDEF3

IDEF3 is a method designed to describe the dynamic environment of businesses (Mili et al., 2010). As Shen et al., (2004) states: “IDEF3 is a process description capturing method whose primary goal is to provide a structured method by which a domain expert can describe a situation as an ordered sequence of events, as well as describe any participating objects.” (Shen et al., (2004, p.5). When a developer uses IDEF3, he can choose between two description modes, these being the process flow and object state transition network (OSTN). The process flow is widely used most (Shen et al., 2004**,** Noran, 2000). The process flow can show activities, described by Unit of Work/Behaviour, it can show temporal precedence, object flow, and relations, and it can implement logic, by using junctions such as “and”, “or”, and “exclusive or” (Shen et al., 2004). The OSTN is used to show the available transitions of an object. An OSTN diagram contains concepts such as object states, state transition arcs, and referents (Noran, 2000). The network is dependent on constraints, where a transition may begin or end depending on the surrounding constraints (Noran, 2000).

Whilst many aspects of IDEF3 are like the IDEF0 method, there are some differences. First, IDEF3 has fewer restricting rules, which makes it easier to model systems (Shen et al., 2004). Furthermore, whilst IDEF0 adopts a single view of a system, IDEF3 uses several user views of a system, therefore getting a more complete picture of the system. According to Mili et al. (2010) the IDEF3 model flourishes in describing several aspects, such as 1) scenarios of organizational activities, 2) roles of user types in the organization activities, 3) use cases, 4) user classes, 5) timing, sequencing, and resource constraints, and 6) user interface objects.

A.5 Role Activity Diagram

Role Activity Diagram, or RAD for short, is a modelling method that is part of the STRIM™ methodology. Produced by Praxis P.c of Bath in the United Kingdom, STRIM aimed to provide for the “elicitation, modelling, and analysis of business processes.” (Mili et al., 2010, p.17). The focus of STRIM, and therefore also RAD, is developing models that are “revealing and communicative” (Mili et al., 2010, p.17). RAD is one of the most influential languages, as there

is a distinct focus on the participants and their interactions within a business process (Pereira & Silva, 2016).

The Role Activity Diagram uses five different items, these being roles, actors, activities, interactions, and entities. These items are used to make a role activity diagram (Mili et al., 2010).

A.6 Evaluation

Pereira & Silva (2016) compared five different business process modelling languages, these being BPMN, UML-AD, EPC, RAD, and IDEF. They evaluated these languages based on *expressiveness, readability, usability, user-friendliness, formality, versatility, universality, tools support, flexibility, ease of learning, innovation inducer, evolutionary,* and *collaborative work.* Each criterion is given a value between 0 and 5, 0 being the lowest and 5 being the highest. Therefore, the maximum amount of points a language can accumulate is 70. These values are visualized in Table 1, shown in Appendix B. They found that BPMN accumulated the most points, with 61 points, followed by UML-AD with 58, EPC with 55, RAD with 49, and IDEF with 39 points.

The researchers found that BPMN was overall the best modelling language based on the criteria, where it either outshined or was level compared to the other modelling languages, with the noticeable exception being collaborative work, where RAD outshined the other modelling languages with several points. Closely following BPMN is UML-AD, which performed similarly as BPMN, only differing one point in several criteria. Following these two modelling languages is EPC, which also generally keeps up with the two leaders. The notable exception is tool support, where EPC has little tool support, therefore having a low value, whilst BPMN and UML-AD have multiple supporting tools, therefore having a high value. RAD, being fourth in line, is generally outshone by the previous three modelling languages, where the difference between each criterion often surpasses one point. The notable criteria are formality and collaborative work. RAD is the only language in this comparison that has no formal definition of its semantics nor a standard representation of its elements. Therefore, RAD has only been given a 1 the other, whilst the other languages have been given a value of 5. However, RAD does outshine the other languages in the criterion collaborative work, where it is given a value of 5, whilst the next highest value is 2. This is because RAD gives users more freedom in working together, whilst other languages place more restrictions. Following these languages, with a large gap, is IDEF. IDEF is outshone by all other languages with multiple points in each criterion, only being up to par regarding formality (Pereira & Silva, 2016).

This evaluation shows that BPMN is the best choice for modelling processes. In almost every criterion, BPMN proves to be an excellent choice. Another good choice is UML-AD. A reason for this gap in quality between BPMN and UML and the other languages can be the rate of update. UML and BPMN are often updated, whilst other languages take far longer if at all, to update. Furthermore, the groups concerning these updates, being the Object Management Group, incorporate the best aspects of other languages in their language, therefore enhancing the language.

Appendix B – Table of compared languages

Table 2: Comparison between business process modelling languages (Pereira & Silva, 2016)

Languages / Criteria	BPMN	EPC	UML-AD	RAD	IDEF
Expressiveness	4	3	4	3	2
Readability	5	4	4	4	3
Usability	4	4	4	4	3
User Friendly	5	5	5	5	3
Formality	5	5	5	1	5
Versatility	5	5	4	3	3
Universality	5	4	5	3	3
Tools Support	5	2	5	2	3
Flexibility	4	4	4	4	3
Concision	4	4	4	4	3
Ease of Learning	5	5	5	4	3
Innovation Inducer	4	4	3	5	2
Evolutionary	4	4	4	2	3
Collaborative Work	2	2	2	5	0

Appendix C – UML 2.0 & BPMN Compared

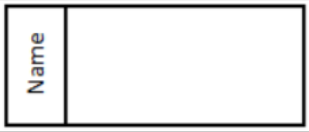
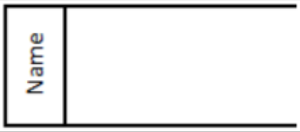
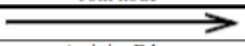
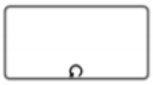
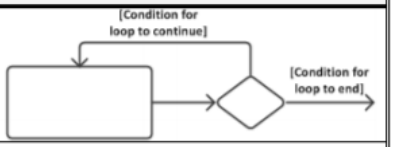
Element of the business process	BPMN 2.0	UML AD 2.1.4
Elements of the business process that are represented in BPMN and UML AD using similar graphical symbols		
Participants to the business process: *Car Service*, *Customer*	 Pool	 Swimlane
The start point and the end point of a process	 Start event; End event	 Initial node; Final node
	Note: In a process with more participants, BPMN uses a start event and an end event for the parts of the process corresponding to each participant, while UML AD uses only one initial node and one final node for the entire process.	
Activities (without loop) performed by the participants: *Require vehicle reparation services*, *Schedule reparation*, *Bring damaged vehicle*, *Create invoice*, *Pay invoice*, *Pick up vehicle from service*	 Task object	 Action node
Occurring of a date that generates the performing of an activity: *Registration start date*	 Timer event	 Time event
Synchronize (combine) parallel flows	 Parallel gateway	 Join node
Flow elements	 Sequence Flow	 Activity Edge
	Note: The Sequence Flows cannot cross the boundaries of a Pool. The interaction between Pools is shown through Message Flows: 	
Representation of objects and data: *Invoice*	 Data Object	 Object node
Elements of the business process that are represented in BPMN using one symbol and in UML AD using a group of symbols		
Activity that repeat sequentially: *Perform reparation*	 Task object with a standard loop marker (eventually with the loop condition shown as a text annotation)	 Action node & decision node & activity edges

Figure 22: Comparison between BPMN and UML-AD elements (Geambaşu, 2012)

Appendix D Petri Nets

As the previous section showed, there are several business process modelling languages a modeller can choose from. These languages have different concepts, and some are stronger than others. However, what all languages above lack, is the ability to analyse the properties of a system. This is where Petri Nets thrive.

Defined by German mathematician Carl Adam Petri in 1962, Petri Net modelling is a method to model dynamic systems and business processes (David & Alla, 1994). Petri Nets are an excellent method to model business processes and due to its mathematical foundation, it can be used to analyse and validate dynamic systems. Furthermore, Petri Nets are capable of modelling large and complex systems with few elements (van der Aalst & Stahl, 2011). This section will continue with the basic concepts of Petri Nets and its analytic potential.

D.1 Basic Concepts

Graphically, a Petri Net consists of four different elements, these being *Places, Transitions, Arcs,* and *Tokens*. A place is a static element, as it can only store, accumulate or show other elements (Reisig, 2013). A place is depicted as a circle (Desel & Reisig, 1996; van der Aalst & Stahl, 2011; Reisig, 2013; David & Alla, 1994). A transition is an active element, as it can change the state of a Petri Net by producing, consuming or transporting elements (Reisig, 2013). A transition is depicted as a square (Desel & Reisig, 1996; van der Aalst & Stahl, 2011; Reisig, 2013; David & Alla, 1994). Arcs show the relation between places and transitions (Desel & Reisig, 1996; Reisig, 2013). Arcs are depicted as arrows (van der Aalst & Stahl, 2011; Reisig, 2013; David & Alla, 1994). Furthermore, places and transitions can only follow each other, creating a bipartite graph. Two places or two transitions cannot follow each other (David & Alla, 1994). These elements are shown in Figure 23.

Figure 23: Legend of Petri Net elements (van der Aalst & Stahl, 2011)

Mathematically, a Petri Net is described as a 3-tuple (P, T, F), where *P* refers to the set of places, *T* refers to the set of transitions, and *F* relates to the set of flows (or arcs). These sets are contained in *N,* which refers to the Net (Reisig, 2013).

Whilst places and transitions only have to be noted once in a set, flows have to be described twice. This is necessary, as it is imperative to know how places and transitions are connected. This is mathematically written as

$$F \subseteq (P\ x\ T)\ U\ [T\ x\ P)$$ (van der Aalst & Stahl, 2011; Reisig, 2013).

As seen in the formula, there are two parts. First, the flows from place to transition. These are also called *input places*. Mathematically, an input place can be written as (p, t) ∈ F. All input places combined can be placed in a set, called •t. If the entire set needs to be referred to, then the formula becomes

$$\bullet t = \{p \mid (p,t) \in F\}.$$

Similarly, the same can be made for all *output places*. The formula then becomes

$$t \bullet = \{p \mid (t,p) \in F\}.$$

An input and output places are respectively also known as preset and postset (van der Aalst & Stahl, 2011).

Tokens are depicted as black dots, and these are distributed across the Petri Net (van der Aalst & Stahl, 2011). Tokens can only be located within places. These tokens can move to other places when they are fired through a transition. This is only possible when a transition is *enabled*, meaning that the input places contain enough tokens for the transition to fire, thereby generating a certain number of tokens in the output places (van der Aalst & Stahl, 2011). When multiple transitions are enabled, they can fire at any time and in any order. There is no predefined sequence. This is also known as a non-deterministic choice (van der Aalst & Stahl, 2011). The number of tokens needed for a transition to be enabled is determined by the *weight* of the flow connecting places to transition.

The weight of flows is formally known as *w*. The weight of a flow can never be lower than 0. Let's look at an example. Figure 24 shows a small Petri Net.

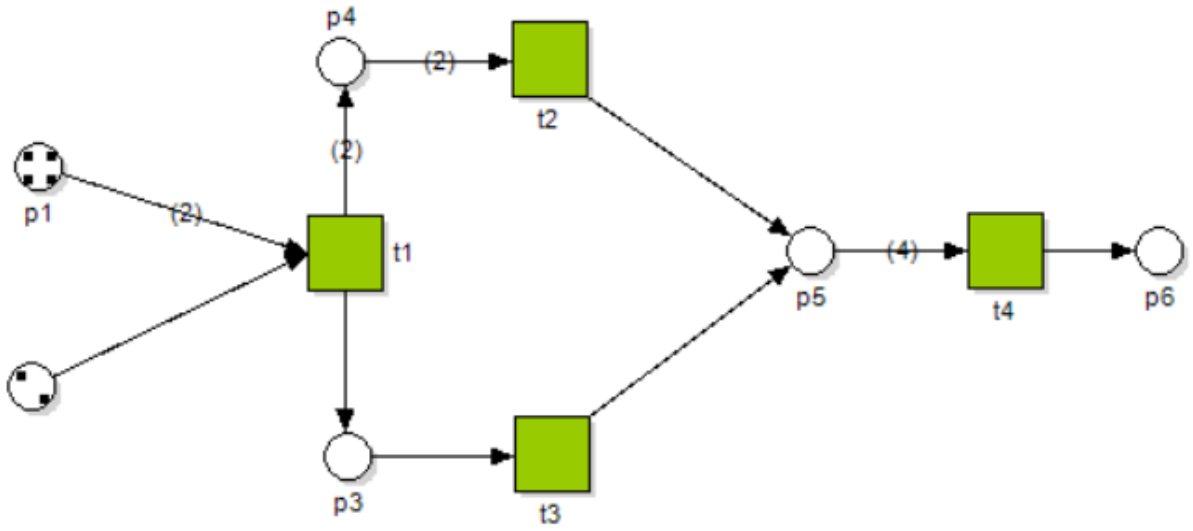


Figure 24: Petri Net 1

Following the formal description, a net is N (P, T, F). Mathematically, Petri Net 1 can be described as follows:

- P = {p1, p2, p3, p4, p5, p6}
- T = {t1, t2, t3, t4}
- F = {(p1, t1), (p2, t1), (t1, p3), (t1, p4), (p3, t3), (p4, t2), (t2, p5), (t3, p5), (p5, t4), (t4, p6)}
- Mo = {(4 * p1), (2 * p2)}

So, Petri Net 1 has six places, four transitions, and ten flows. As shown in Figure 14, some of the flows have numbers. This indicates the weight of the flow. As stated before, a transition can only fire when it is enabled. This means that the preset of the transition has enough tokens. The preset of t1 is p1 and p2. Furthermore, the flow between p1 and t1 has a weight of 2. This means that p1 needs two tokens *and* p2 needs one token before t1 can fire. In its current state, t1 will be able to fire twice. When fired, it produces tokens in its postset, here p3 and p4. Note that the weight of the flow (t1, p4) has a value of two. This means that two tokens will be produced in p4 after t1 fires. Mathematically, a transition is enabled if

$$\forall p \in \bullet t : m(p) \geq F((p,t))$$

which means *for all places of set preset t: tokens in place is equal or more to the weight of the flow (p,t)*.

To formally describe a Petri Net, it is important to look at the tokens. The distribution of tokens across a Petri net is called a *marking* (van der Aalst & Stahl, 2011**;** Reisig, 2013). The distribution of tokens is the only thing that can change in a Petri Net. The structure of a Petri Net cannot change (van der Aalst & Stahl, 2011). The marking of a Petri Net determines its state

(van der Aalst & Stahl, 2011). Firing transitions can change the number of tokens a Petri Net holds, as transitions can consume more tokens than they produce, or they could produce more tokens than consumed. The places that contain tokens from the start are called initial markings, denoted as *Mo*. The initial marking of Petri Net 1 is four tokens in p1 and two tokens in p2 (van der Aalst & Stahl, 2011).

D.2 Analysing with Petri Nets

As stated before, Petri Nets are strong in analysing and validating dynamic systems. A Petri Net can be analysed and validated with the use of a reachability graph (van der Aalst & Stahl, 2011). As the name suggests, a reachability graph shows which places of the Petri Net are reachable. A reachability graph contains nodes, which represent places, and edges, which represent transitions (van der Aalst & Stahl, 2011). Some of the properties that can be analysed with a reachability graph will be discussed in this section.

D.2.1 Boundedness

When a Petri Net is bounded, each place in the net has a maximum of tokens it can hold at one time. Formally written as $p \in P \geq k$, a net is k-bounded if no place in the entire net holds more than *k* tokens in any reachable marking. If this is not the case, then the Petri Net is unbounded. An unbounded net, even if not specified, is an indication of an incorrection in the net (van der Aalst & Stahl, 2011**;** Reisig, 2013). This property can only be analysed if the reachability graph is finite.

D.2.2 Terminating

A Petri Net can also be analysed for termination. This means that a net will always end in a place where no transition is enabled. Like the property boundedness, this property can only be analysed if the Petri Net is finite (van der Aalst & Stahl, 2011).

D.2.3 Deadlock Freedom

Following the property termination, the property deadlock-freedom analyses a net for a terminal marking, as a terminal marking is also known as a *deadlock***.** If a Petri Net is deadlock-free, then a transition is enabled at each reachable marking (van der Aalst & Stahl, 2011**;** Reisig, 2013).

D.2.4 Dead Transition

Analysing a Petri Net for dead transitions is important, as it shows if the net contains transitions that never fire during a run. This means that this transition is never enabled at any reachable marking (van der Aalst & Stahl, 2011).

D.2.5 Liveness

Contrasting the property of dead transitions is the property of liveness. Analysing this property means analysing the reachability graph for transitions that can be fired more than once. In other words, if a transition is fired, it can be fired again at a later stage. A Petri Net can only be called live if every transition in the net is live. The property of liveness excludes the properties of terminating and dead transitions, as if every transition is live, then there is not terminating marking, nor is there a dead transition in the net (van der Aalst & Stahl, 2011**;** Reisig, 2013).

D.2.6 Home-Marking

The sixth property is the property Home-Marking. Like the property liveness, a net is guaranteed the property home-marking if every marking can be reached after it is left. A net can be found *reversible* if the initial marking is a home marking (van der Aalst & Stahl, 2011**;** Reisig, 2013).

D.2.7 Invariants

The last and most important analysis method discussed here is invariants (Reisig, 2013). Invariants apply linear algebraic techniques to prove the properties of Petri Nets (van der Aalst & Stahl, 2011). There are two types of invariants, which are *place invariants* and *transition invariants.* These two variants will be discussed with the help of Figure 25. This figure shows the cycle of marriage and divorce. The place man and woman, both representing singles, go into the transition marriage, forming a couple. This couple goes into the transition divorce, which in turn goes to the place man and woman.

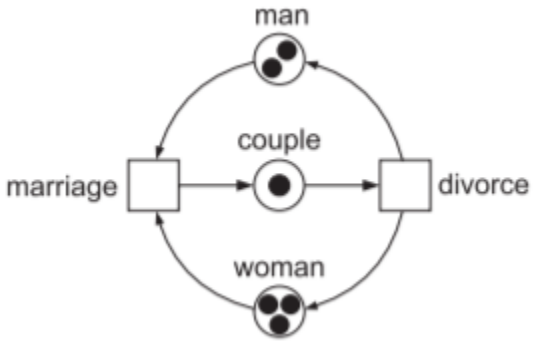


Figure 25: Petri Net showing marriage and divorce (van der Aalst & Stahl, 2011)

As the name suggests, place invariants focus on the places within the Petri Net. Place invariants are used to control the number of tokens in the Net. This is done by implementing a weight function on places. By implementing this weight, the following formal equation is made:

$$z1 \times m(p1) + \ldots + zk \times m(pk) = z0.$$

In this equation, *z* is the weight value of a place, *m* refers to the tokens, and *(pk)* refers to a place. The total of this equation, *z0*, is called the *weighted token sum* of marking m. Using this equation in Figure 25, the following is shown:

$$(2 \times 1) + (3 \times 1) + (1 \times 2) = 7$$

The weight of a place is assigned by the place invariant. In the context of Figure 25, marriage consists of one man and one woman. Therefore, the weight of place man and woman is one. These two form a couple, therefore the weight of place *couple* has a value of two. The total weighted token sum is in this model 7. To control that the tokens do not increase nor decrease,

the transitions within the net could be fired, after which the equation should lead to the same value (van der Aalst & Stahl, 2011).

The counterpart of place invariants, transition invariants are used to prove that a net can return to any marking after a series of fired transitions (van der Aalst & Stahl, 2011). A transition invariant assigns to each transition a weight. This weight, also shown as *z*, is always nonnegative. A net is a transition invariant if the number of tokens in a net does not change after firing the transitions a certain amount of times. Formally, this is similarly written as place invariants. The equation would be shown as:

$$z1 \times t1 + \cdots + zn \times tn.$$

In the equation, *zn* is the weight of the transition and *tn* is the transition. So, in the context of Figure 25, if both transitions are fired twice, it would result in 2 * marriage + 2 * divorce. This would not change the weighted sum of tokens in the net.

These invariants can be used to analyse the properties of a Petri Net by applying linear algebra and vertices. However, this information goes beyond the scope of this paper. For more information, see (van der Aalst & Stahl, 2011)**.**