Application of A.I. Driven Cognitive User Modeling for Adaptive Serious Games

Bachelor Thesis

KIT – Karlsruher Institut für Technologie
Fraunhofer IOSB – Fraunhofer-Institut für Optronik,
Systemtechnik und Bildauswertung

Kolja Bauer

15.03.2021

Verantwortlicher Betreuer:	Prof. Dr.-Ing. J. Beyerer
	Prof. Dr.-Ing. T. Längle
Betreuender Mitarbeiter:	Dipl.-Inf. A. Streicher

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die Arbeit selbständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis in der gültigen Fassung beachtet habe.

Karlsruhe, den 15.03.2021

(Kolja Bauer)

Abstract

In the domain of adaptive serious games, accurate user models can be crucial for helpful adaptation. The goal is to automatically adapt such games to the user's current needs and knowledge level, thus increasing his learning flow and, ultimately, his learning outcome. In contrast to physiological measurements, a user's clicks can be captured without special equipment. Therefore, inferring a user's cognitive state solely from interaction data could make the development of adaptive serious games far more feasible.

In this thesis, the existing approach of the Cognitive Intelligent User Modeling tool (CogIUM) is enhanced and applied to the serious game Streamlined Lost Earth (SLE). The main research question is: How to apply CogIUM to serious games to provide adaptivity?

By extending CogIUM and applying it to SLE, it is investigated how exactly CogIUM can be utilized to compute quantitative measures of user performance called *adaptivity scores*. Subsequently, CogIUM is evaluated in a user study in which $n = 22$ participants played SLE and afterwards filled out self-assessments regarding various cognitive attributes.

Results show a moderate correlation between self-assessed cognitive load and CogIUM's inferences. CogIUM showcases promising potential to infer a user's cognitive load solely from his click data, enabling the implementation of adaptivity manifestation based on CogIUM's inferences in future work.

Kurzfassung

Im Bereich der adaptiven Lernspiele können akkurate Benutzermodelle entscheidend für eine hilfreiche Adaption sein. Ziel ist es, solche Spiele automatisch an die aktuellen Bedürfnisse und den Wissensstand des Nutzers anzupassen und so seinen Lernfluss sowie letztlich seinen Lernerfolg zu steigern. Im Gegensatz zu physiologischen Messungen können die Klicks eines Benutzers ohne spezielle Hardware erfasst werden. Die Bestimmung des kognitiven Zustands eines Benutzers allein aus Interaktionsdaten könnte somit die Entwicklung adaptiver Lernspiele weitaus praktikabler machen.

In dieser Arbeit wird der bestehende Ansatz des Cognitive Intelligent User Modeling tools (CogIUM) erweitert und auf das Lernspiel Streamlined Lost Earth (SLE) angewendet. Die Hauptforschungsfrage ist: Wie kann CogIUM auf Lernspiele angewendet werden, um Adaptivität zu ermöglichen?

Durch die Erweiterung von CogIUM und die Anwendung auf SLE wird untersucht, wie genau CogIUM genutzt werden kann, um quantitative Maße der Nutzerperformance, sogenannte *Adaptivitäts-Scores*, zu berechnen. Weiterhin werden CogIUM‘s Berechnungen mittels einer Nutzerstudie evaluiert, in der n=22 Teilnehmer SLE gespielt und anschließend Selbsteinschätzungen bezüglich verschiedener kognitiver Attribute ausgefüllt haben.

Die Ergebnisse zeigen eine moderate Korrelation zwischen selbst eingeschätzter kognitiver Belastung und CogIUM‘s Berechnungen. CogIUM zeigt vielversprechendes Potential, die kognitive Belastung eines Benutzers allein aus seinen Klickdaten zu ermitteln, was die Implementierung von Adaptivitätsmanifestationen basierend auf CogIUMs Berechnungen in zukünftigen Arbeiten ermöglicht.

Notation

List of Acronyms

CLT	Cognitive Load Theory
CogIUM	Cognitive Intelligent User Model
ECL	Extraneous Cognitive Load
ELAI Framework	E-Learning Artificial Intelligence Framework
GCL	Germane Cognitive Load
HBM	Hierarchical Bayesian Model
ICL	Intrinsic Cognitive Load
LA	Learning Analytics
LMS	Learning Management System
LRS	Learning Record Store
NASA TLX	NASA Task Load Index
RTLX	Raw Task Load Index
SCORM	Sharable Content Object Reference Model
SLE	Streamlined Lost Earth
xAPI	Experience Application Programming Interface

Contents

1 Introduction

As the education system is becoming ever more digital, new opportunities arise to convey learning objectives more efficiently. However, especially the current crisis caused by the Coronavirus showcases that there is still a lot to be done regarding the education system's digitalization.

E-Learning and, more specifically, game-based learning has been an active field of research for the past years. [Qian and Clark 2016] define game-based learning as an environment where game content and gameplay enhance knowledge and skills acquisition, and where game activities involve problem-solving spaces and challenges that provide players/learners with a sense of achievement. Games used for educational purposes are a subclass of serious games and are thus often referred to as *educational serious games*.

Educational serious games provide the possibility to learn playfully, catering to many teenagers' and young adults' preferences. It has been shown that this can lead to better performance, knowledge acquisition, and long-term concept retention [Boyle et al. 2016].

In an optimal scenario, the game's difficulty and the user's skill level are perfectly balanced, causing the user to be completely immersed in the game. This state is described in cognitive sciences as the state of *Flow* [Nakamura and Csikszentmihalyi 2009]. For this scenario to happen, serious games must automatically adapt to the user's specific needs. A visualization of the flow model combined with adaptive measures can be seen in figure 1.1.

[Pearson and EdSurge 2016] define adaptive learning tools as "education technologies that can respond to a student's interactions in real-time by automatically providing the student with individual support". The four-phased adaptivity cycle presented by [Shute and Zapata-Rivera 2012] formalizes the different stages of automatic adaptation of learning material to a user. It comprises the four phases capture, analyze, select and present, as shown in figure 1.2. In this thesis, the focus will be on the capture and analysis phase: The goal is to capture a user's interaction data with an educational serious game and subsequently infer the learner's cognitive state from the captured interaction data.

To provide an optimized learning experience, a serious game should automatically adapt to the learner's individual needs. But what are those individual needs, and how can they be measured or inferred? To make a well-founded decision on how and when to adapt, cognitive

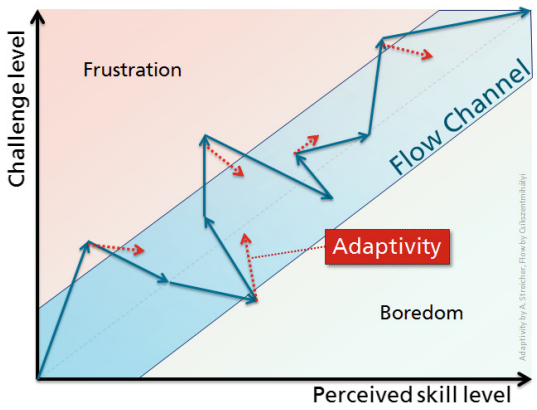


Figure 1.1: Staying in the flow channel with adaptive measures [Streicher and Smeddinck 2016].

variables like the user's skill level, motivation, prior knowledge, and cognitive load have to be captured. However, capturing brain activity or eye movement requires special equipment and limits the potential usage scenarios. In a realistic scenario, the only data that can easily be gathered is the user's interaction data with a serious game. Therefore, the previously mentioned cognitive variables have to be inferred from interaction data.

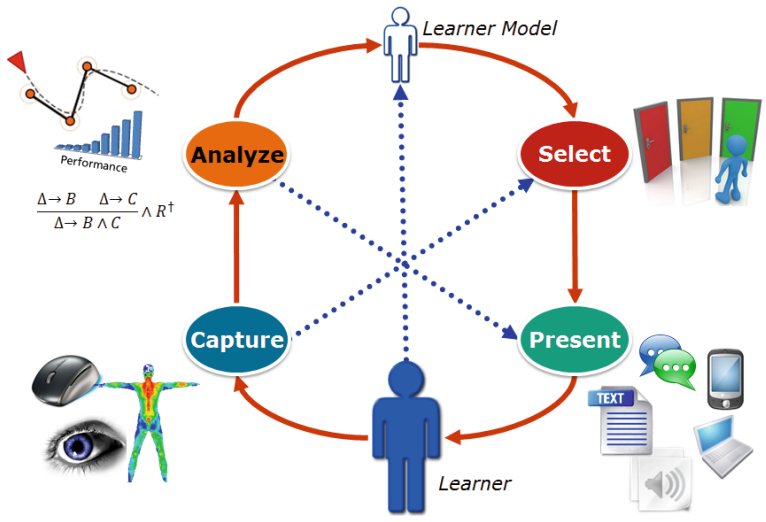


Figure 1.2: Four-phased adaptivity cycle from [Streicher and Smeddinck 2016], based on [Shute and Zapata-Rivera 2012].

1.1 Problem Statement

The specific problem statement of this thesis studies the application of the Cognitive Intelligent User Modeling tool (CogIUM) [Aydinbas 2019] to the serious game Streamlined Lost Earth (SLE), and the integration within the ELAI adaptivity framework [Streicher and Roller 2015].

CogIUM is a cognitive user modeling concept which demonstrates how to compute probability distributions of cognitive variables given synthetic user interaction data. At the start of this thesis, only a first concept and a general prototype of CogIUM were available, with no application to an existing serious game.

Input for the CogIUM model is an array of measured variables from the user's interaction with a game. In the prototypical implementation, those variables were *mission success*, *mission score* and *mission time*. From the observed variables, CogIUM infers probability distributions for latent cognitive variables like cognitive load, prior knowledge and motivation.

At its core, CogIUM is a hierarchical Bayesian model. For every cognitive variable, prior beliefs about its distributions are coded into the model. Using user interaction data as input, those beliefs are reallocated to account for the newly observed data. Outputs are new probability distributions for latent cognitive variables, called posterior distributions.

If assumptions about the variables' distributions are false, the model's inference might also differ from reality. Thus, prior distributions for newly introduced variables have to be chosen thoughtfully when expanding the model. Existing prior distributions within the CogIUM model have to be examined regarding their plausibility.

Prior to this thesis, the CogIUM model was only tested on synthetical data. Accordingly, there was no proof that the cognitive meaning that CogIUM ascribes to its latent variables corresponds to learners' actual cognitive attributes. Furthermore, the model's distribution assumptions might have proven false once the model is applied to real user interaction data.

Additionally, the initial CogIUM model only uses a small part of the interaction data's information. Interaction data is commonly represented as a series of Experience API (xAPI) statements that follow the format *subject, verb, object* and can additionally contain contextual information.

The variables *mission success*, *mission score*, and *mission time* can easily be extracted from this series of statements. However, they only represent part of the information that is present in the interaction data. The model needed to be extended to enable CogIUM to use all available information on the user.

To preserve general applicability, CogIUM must retrieve all of its input data from xAPI statements generated by events within the game. Especially, CogIUM should not be interconnected with the game to ensure its transferability to other serious games.

1.2 Research Objectives

This thesis' objective is to showcase CogIUM's applicability to an existing educational serious game. After each mission completed by the user, the goal is to answer the following question: How to adapt the game for the next mission to provide the user with an optimal learning experience? This question could potentially also be posed after every subtask within a mission. It is analyzed if CogIUM when applied to a serious game, can provide valuable insights for adaptivity. This goes hand in hand with evaluating the mathematical assumptions about the distribution of latent and observable variables that are made in the model.

To successfully apply CogIUM to an existing serious game, several supporting research questions have to be investigated:

- What specific features of the incoming xAPI data can be modeled by CogIUM (additional to the already modeled mission success, mission score, and mission time) in order to utilize all relevant information for adaptivity?
- Which assumptions about the relationships and distributions of latent and observable variables make sense?
- Can the CogIUM inferences provide valuable insights regarding adaptivity?
- Does the information about latent cognitive variables inferred by CogIUM correlate with users' self-assessment?
- How to compute adaptivity scores from the latent cognitive variables' posterior distributions (output of CogIUM)?

The thesis's main goal is to progress CogIUM from an experimental proof of concept to a validated technology demonstrated in a relevant environment. This corresponds to an advancement from Technology Readiness Level (TRL) 3 to 5. TRLs are a measure of a technology's maturity level originally introduced by NASA [Mankins 1995]. The scale ranges from *TRL 1: Basic principles observed* to *TRL 9: Actual system proven in operational environment.*

Furthermore, the goal is to evaluate CogIUM's inferences using real user interaction data by conducting a user study. Since CogIUM ascribes a cognitive meaning to its latent variables, it must be analyzed if those variables correspond to users' actual cognitive attributes.

1.3 Project Environment

The thesis was written in cooperation with the department for Interoperability and Assistance Systems (IAS) of the Fraunhofer Institute of Optronics, System Technologies, and Image Exploitation IOSB. It was supervised by Dipl.-Inf. Alexander Streicher, who is also the second reviewer of this thesis. Fraunhofer IOSB is one of the largest institutes for applied research in image exploitation and analysis across Europe. The IAS department focuses on innovative concepts and solutions for intelligent tutoring interfaces for technology-enhanced learning and adaptive learning systems. A main field of application is professional image exploitation. Learners of image exploitation tasks benefit from digital E-Learning methods and adaptive learning systems that automatically adapt to the user's needs.

The IAS department has already developed several solutions in the field of adaptive learning systems. Among them is the E-Learning Artificial Intelligence (ELAI) framework [Streicher and Roller 2015]. The ELAI framework is an interoperable tutoring agent that adjusts a simulation or game to a user's specific needs. It consists of a game engine adapter and an ELAI controller. Within the ELAI controller, the interpretation engine and influence engine compute optimal adaptations based on the collected user data.

Additionally, multiple serious games were developed at IAS: Seek and Find for Image Reconnaissance (SaFIR) plus adaptivity (SaFIRa) [Streicher, Roller, and Biegemeier 2017], Exercise Trainer (EXTRA) [Streicher, Szentes, and Gundermann 2016], and Lost Earth 2307. Lost Earth 2307 was developed to help image interpreters' training and support related learning objects like gaining basic knowledge in interpreting and analyzing aerial and satellite imagery and understanding processes of the Reconnaissance-Cycle.

1.4 Structure

The following section provides an overview of the thesis's structure: **Chapter 2** provides the theoretical foundations that this thesis is built upon. Most notably, cognitive modeling, the existing CogIUM prototype, and hierarchical Bayesian models are introduced to the reader.

In **chapter 3**, state of the art research on cognitive modeling, measurement of latent cognitive variables and hierarchical Bayesian modeling is presented.

Regarding the concept and development of *CogIUMa - CogIUM applied* that are discussed in **chapters 4 and 5**, my approach is structured as follows: As a first step, scenarios are developed that showcase how exactly CogIUM will work and interact with ELAI and SLE. Before the extended CogIUM model's iterative development could start, xAPI data had to be available. Several approaches for the collection of both synthetic and real-world xAPI statements are

explored. Using the collected data, the CogIUM prototype developed by [Aydinbas 2019] is iteratively extended and restructured. Additional observable variables are incorporated so that the model uses as much information from the xAPI statements as possible. The enhancement and extension of the model are guided by Cognitive Load Theory (CLT) that also forms the theoretical foundation for the CogIUM prototype by Aydinbas. Subsequently, it is analyzed how adaptivity scores can be computed from CogIUM's output. Outputs of the CogIUM inference are posterior distributions for latent cognitive variables like cognitive load, motivation, and prior knowledge.

Finally, the extended CogIUM model is evaluated in **chapter 6**. To obtain user interaction data as well as self-assessments regarding cognitive attributes, a user study is conducted. The study is described with regards to concept, hypotheses, planning, execution, and results. Self-assessment data and CogIUM inference data are compared and examined for correlations. Additionally, results and their implications for CogIUM's further usage are discussed.

Chapter 7 concludes the thesis. A summary of the work and its main contributions is given, and possible directions for future work are outlined.

2 Conceptual Background

In the following chapter, concepts that are fundamental for this thesis's theoretical background are presented. The first section will explain Bayesian data inference and hierarchical Bayesian modeling. Subsequently, the initial CogIUM prototype is described in detail. Since CogIUM is, at its core, a hierarchical Bayesian model, prior understanding of Bayesian modeling is helpful. Furthermore, Cognitive Load Theory (CLT), the serious game Streamlined Lost Earth (SLE), and the ELAI framework are presented.

2.1 Bayesian Modeling

Bayesian modeling is used in this thesis to reallocate beliefs about cognitive variables based on observed user interaction. As such, it is motivated by the functionality of the human brain: We humans receive sensory input through our eyes, ears, nose, mouth, and skin. We then combine this input with beliefs about our current situation to conclude our surroundings' current state. Aydinbas gives an intuitive example for this: *"We are interested in the state of the world, what we call a hypothesis, given certain observations. A typical everyday task could be to determine from our sensory input whether we know the person in front of us or not. Two alternative hypotheses would describe the fact that we either know the person or not"* [Aydinbas 2019].

Let us assume we have a set of hypotheses $\mathcal{H}$ and we observe a dataset $\mathcal{D}$. For each hypothesis $h \in \mathcal{H}$, the conditional probability $Pr(h|\mathcal{D})$ of h given observation $\mathcal{D}$ is calculated by Bayes' rule as follows:

$$\Pr(h|\mathcal{D}) = \frac{\Pr(\mathcal{D}|h) \cdot \Pr(h)}{\Pr(\mathcal{D}) = \sum_{i=1}^{n} \Pr(\mathcal{D}|h_i) \cdot Pr(h_i)} \tag{2.1}$$

$\Pr(h)$ is called the prior distribution and represents prior probabilities about the likelihood of a hypothesis. In the example given above, those prior probabilities could mean the following: Suppose we are on vacation in a foreign country on our own. When meeting a person, we assume that we probably do not know the person because we are far from our usual surroundings. Accordingly, the prior probability for the hypothesis that we know the person would be much lower than for the alternative hypothesis that we do not know the person. On

the other hand, suppose we are in a familiar environment like our school, university, or even our home. The prior probabilities would then be distributed the opposite way since we would assume that we probably know a person in our home or school.

$\Pr(\mathcal{D}|h)$ denotes the likelihood of dataset $\mathcal{D}$ given hypothesis h. The whole term is normalized by $\Pr(\mathcal{D}) = \sum_{i=1}^{n} \Pr(\mathcal{D}|h_i) \cdot Pr(h_i)$, which is the probability of data $\mathcal{D}$. Bayes' rule allows computing the conditional probability of hypothesis h given observation $\mathcal{D}$ from the likelihood, the prior probability of h, and the probability of $\mathcal{D}$. This conditional probability is also referred to as *posterior probability*. Often we are only interested in which hypothesis is the most likely, given an observation $\mathcal{D}$. To answer this question, the normalization term $\Pr(\mathcal{D})$ can be omitted since it is identical for all hypotheses.

For this thesis's scope, we use Bayesian models to represent the cognitive states of a learner. This means that the Bayesian models will be designed to allow the prescription of cognitive meaning to its parameters. The model's parameters will be latent cognitive variables like cognitive load, motivation, and prior knowledge. Those variables are then used to explain the observations from learners' interaction data with learning material.

2.1.1 Hierarchical Bayesian Models

Whenever the model is used to represent data from multiple learners or multiple learning materials, the use of Hierarchical Bayesian Models (HBMs) is appropriate. Gelman et al. emphasize that "simple non-hierarchical models are usually inappropriate for hierarchical data: with few parameters, they generally cannot fit large datasets accurately, whereas, with many parameters, they tend to 'overfit' such data in the sense of producing models that fit the existing data well but lead to inferior predictions for new data" [Gelman et al. 2013]. HBMs are especially useful when several parameters that influence a probabilistic process are related or connected in some way. To model this relation, HBMs allow the definition of multiple variables with prior distributions parameterized by values sampled from a joint parent distribution. A visualization of this can be seen in figure 2.1. Regarding the modeling approach in this thesis, this is used in the following way: Each user has his own motivation level. The motivation level of a single user can vary between missions. However, those variables should not be completely independent of each other. For a single user, motivation is expected to vary over time more steadily than rapidly.

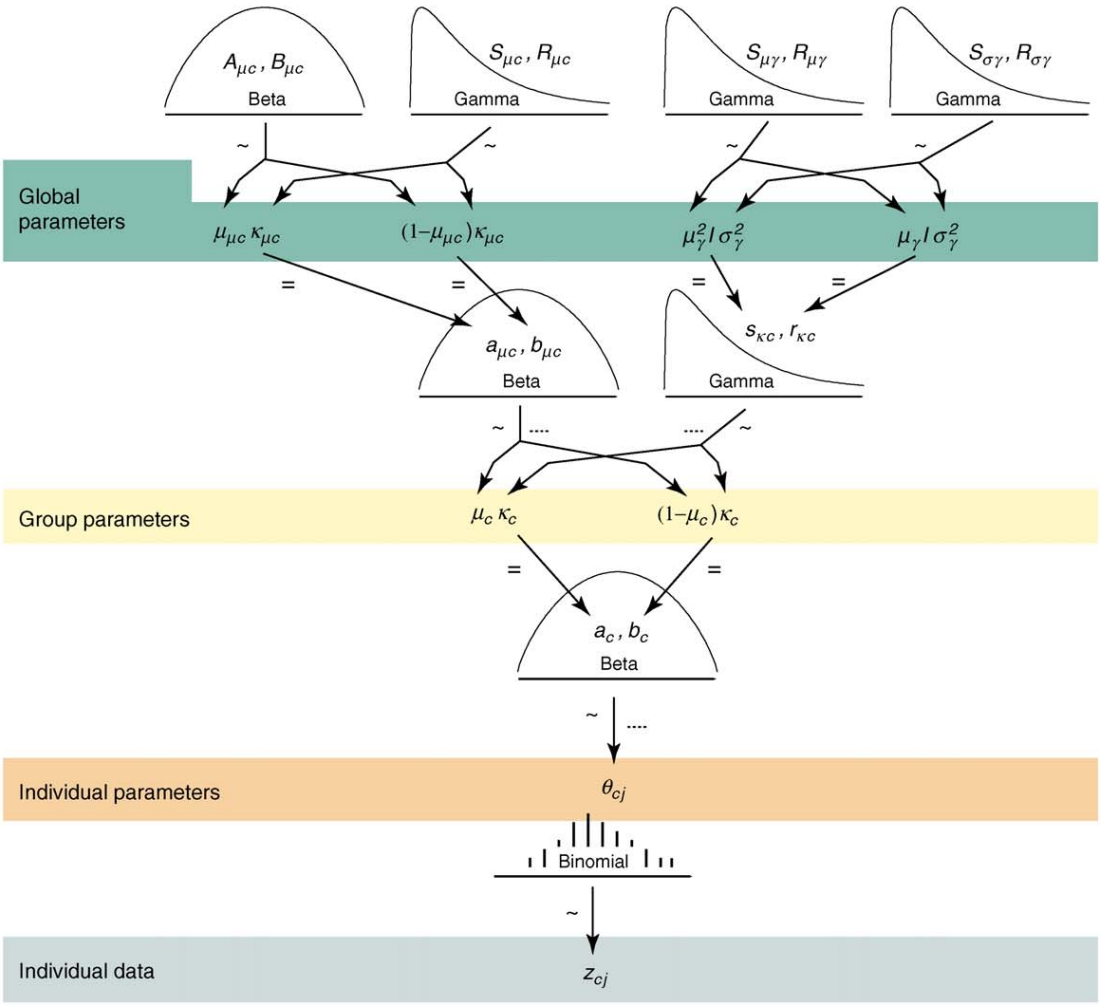
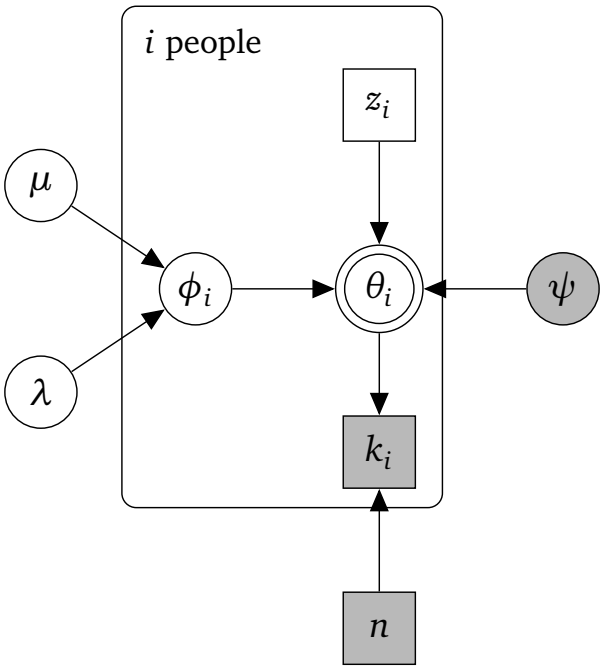


Figure 2.1: Exemplary hierarchical Bayesian model for an experiment about category learning (from [Kruschke 2010]). The group-level parameters come from global-level distributions, whereby data from one group can influence estimates for other groups.

When working with HBMs, the structure of a model can be visualized to facilitate an understanding of the model. [Lee and Wagenmakers 2014] present a visualization of HBMs that is also used as a standard in this thesis. An exemplary visualization by Lee and Wagenmakers can be seen in figure 2.2. Figure 2.3 explains the meaning of the nodes' shapes and colors in the visualization.



Priors

$z_i \sim \text{Bernoulli}(0.5)$
$\mu \sim \text{Uniform}(0.5, 1)$
$\lambda \sim \text{Gamma}(0.001, 0.001)$

Intermediate variables

$\phi_i \sim \text{Normal}(\mu, \lambda)_{I(0,1)}$

Deterministic variables

$\psi \leftarrow 0.5$

$$\theta_i \leftarrow \begin{cases} \phi_i \text{if } z_i = 1 \\ \psi \text{if } z_i = 0 \end{cases}$$

Observable variables

$k_i \sim \text{Binomial}(\theta_i, n)$

Figure 2.2: Graphical representation of HBMs by [Lee and Wagenmakers 2014].

	Type of Variable	
Status of Variable	Discrete	Continuous
Observed	■	●
Unobserved		
Stochastic	□	○
Deterministic	▣	◎

Figure 2.3: Notation for nodes in graphical representations of HBMs by [Aydinbas 2019] based on [Farrell and Lewandowsky 2018].

2.1.2 Bayesian Data Analysis

According to [Gelman et al. 2013], Bayesian data analysis consists of three steps:

1. **Setting up a full probability model:** This means to postulate a joint probability distribution for all observable and non-observable variables in a problem. The model should incorporate all available domain knowledge on the problem's underlying processes and the data collection process.

2. **Calculate posterior distribution:** Given observed data, calculate the *posterior distribution* - the conditional probability distribution of the non-observable variables. Those posterior distributions are of final interest.

3. **Evaluate the fit of the model:** Examine how well the model fits the data. Are the conclusions that can be drawn from the posterior distribution reasonable? How sensitive are the results to the modeling assumptions in step 1?

Kruschke divides the process of Bayesian data analysis into five steps, giving a more detailed description [Kruschke 2014]. Aydinbas vividly visualizes the following steps in figure 2.4.

1. Identify the data relevant to the research question. This involves the measurement scales of the data and the definition of variables that are to be predicted and variables that are predictors.

2. Define a descriptive model for the relevant data. The mathematical form and its parameters should be meaningful. The model should be appropriate for the theoretical purposes of the analysis.

3. Specify a prior distribution of the parameters. The prior distribution should capture the analyst's assumptions and be agreeable to other skeptical scientists.

4. Use Bayesian inference to reallocate credibility across parameter values. Interpret the posterior distribution with respect to the research questions.

5. Check that the posterior predictions mimic the data with reasonable accuracy, which is called "posterior predictive check". If not, consider a different descriptive model and repeat the analysis.

Outputs of the first step are a set of variables divided into observable and latent variables. In the application scenario of a serious game, observable variables can include mission success,

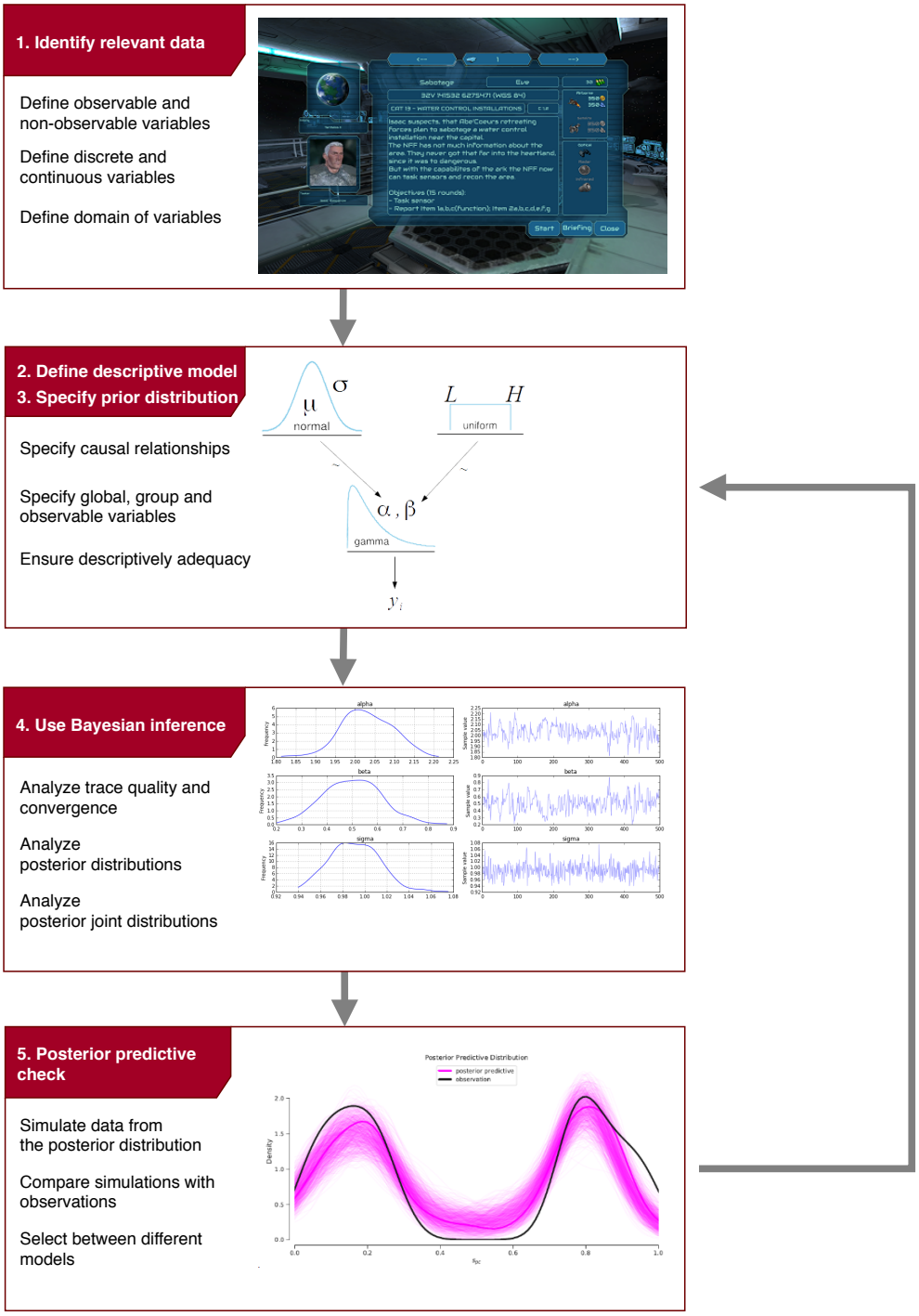


Figure 2.4: The five steps of Bayesian data analysis with exemplary output by [Aydinbas 2019]

mission score, mission time, and others. Latent variables are designed to represent cognitive variables like cognitive load, motivation, and prior knowledge. In the second and third step, causal relationships are specified, and a joint probability distribution over all observable and latent variables is postulated. By specifying the causal relationships between the variables, the hierarchic structure of the HBM is also implicitly defined. Variables are categorized as global, group, individual, and observable variables. This process includes assigning prior distributions to all variables. For the design choices made in steps 2 and 3, domain knowledge of the underlying processes is crucial. False causal relationships or unsuitable prior distributions can lead to false inferences by the model. In the fourth step, called Bayesian Inference, the posterior distributions for all model variables are computed. Because the posterior distribution cannot be computed analytically, this is done by sampling values from it: Drawing a large number of samples from the distribution yields an approximation of the distribution's shape. The fifth and final step comprises the posterior predictive check: From the posterior distributions that were the output of step 4, values are sampled. Those values are compared to the originally observed data. The simulated data should resemble the observed data.

When designing the initial CogIUM prototype, Aydinbas closely followed the presented steps above. This design process is described in detail in section 2.4.

2.2 User Tracking - Experience API (xAPI)

In all Learning Analytics (LA) applications, data on a user's interaction with learning materials needs to be tracked, stored, and retrieved. To achieve interoperability, the data's format must be standardized, allowing an adaptivity framework like ELAI to work with the interaction data regardless of whether the data originated from user interaction with a Learning Management System (LMS) or with an educational serious game.

Several approaches to specifying such a standard exist. Traditionally, SCORM has been the most commonly used format to store this type of data. SCORM stands for Sharable Content Object Reference Model. It was developed by the Advanced Distributed Learning Initiative (ADL) and consists of a set of technical standards for LMSs and E-Learning content. The basic idea is that any SCORM conformant LMS can work with any SCORM conformant E-Learning content. An analogy for this is a DVD player: It can play DVDs by every manufacturer, and a DVD can be played by DVD players from every manufacturer as well [Rustici 2021a]. This works because there is a defined set of technical standards, just like what SCORM is for E-Learning environments.

In 2011, ADL contracted Rustici Software to develop the next evolution of SCORM. This resulted in the Tin Can API, which is called that way because the development process was

"meant to be a two-way conversation between [Rustici Software] and the E-Learning industry" [Rustici 2021b]. In 2013, the third generation of Tin Can was published under the name Experience API (xAPI).

xAPI is a standard that captures user interaction as a stream of activities. Every activity is represented as an xAPI statement, which is based on the JavaScript Object Notation (JSON) format. An xAPI statement is mainly based on the structure *subject, verb, object*. Figure 2.5 by [Vidal, Rabelo, and Lama 2015] shows a detailed description of an xAPI statement's semantic structure. In addition to the *subject, verb, object* format, a statement includes a timestamp and can also store information about the context and the result of an activity. This leads to a machine- and human-readable stream of activities. To send, store and retrieve statements, the xAPI standard uses RESTful HTTP requests, meaning it can be used with any programming language. Statements are sent to, stored in, and retrieved from a Learning Record Store (LRS) using a RESTful web-service [ADL 2021]. An LRS stores all statements and can exist on its own or inside an LMS [Rustici 2021b].

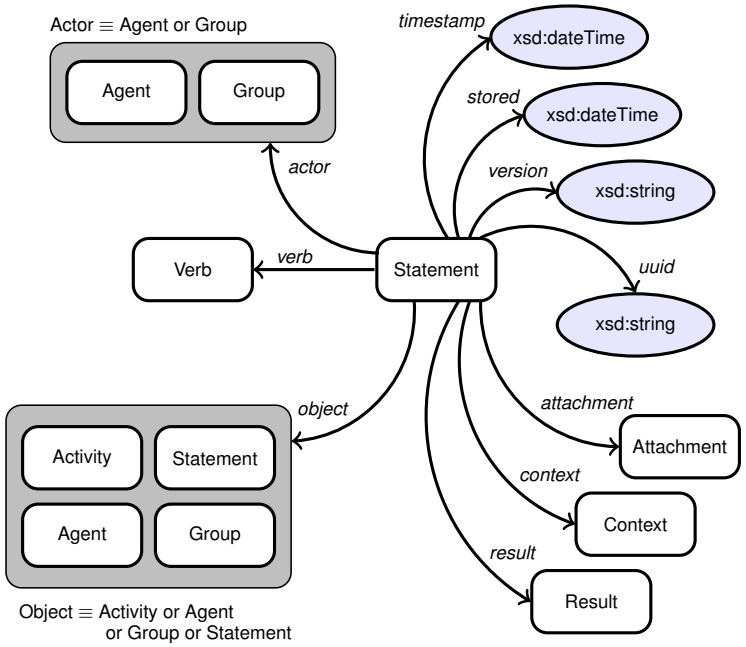


Figure 2.5: The semantic network of the xAPI statement model (from [Vidal, Rabelo, and Lama 2015])

2.3 Cognitive Load Theory (CLT)

During the design process of CogIUM, Cognitive Load Theory (CLT) acted as a guideline for the model's structure. Therefore, understanding CogIUM also requires understanding CLT, CogIUM's theoretical foundation.

While researching the difference in problem-solving skills between experts and novices, Sweller found evidence suggesting that *schema acquisition* and *automation* are the primary mechanisms of learning [Sweller 1988]. Built on his findings, he presents the Cognitive Load Theory (CLT), intending to showcase how information can be structured in order to focus the learner's cognitive activities on schema acquisition.

One central assumption of the CLT is that a learner only has limited working memory capacity [Van Merrienboer and Sweller 2005]. According to CLT, a learner can hold about seven elements in his short-term working memory. Additionally, he can only operate on two to four of those elements simultaneously. Furthermore, CLT assumes that a learner's total cognitive load is the sum of intrinsic cognitive load (ICL) and extraneous cognitive load (ECL).

ICL represents the intrinsic difficulty of the learning material. More precisely, it reflects the number of interactive elements that the learning material comprises. Therefore, it cannot be altered for a given learning material. However, ICL is dependent on the knowledge level of the learner. If the learner is an expert, ICL will use much less of his working memory because the relevant schemes are already stored in his long-term memory. For a novice, all information is new, and accordingly, ICL for him is very high. For a fixed knowledge level, ICL can only be changed by changing what is learned or by the act of learning itself [Sweller 2010].

ECL represents the cognitive load caused by instructional design that is more complex than necessary. Thus, ECL is load that is not necessary for learning and can be reduced by optimizing the instructional design. If the learning material features high element interactivity and ECL is high, this can lead to a scenario where the total cognitive load is greater than the working memory capacity. In this case, high ECL can interfere with the learning and negatively affect the learning outcome [Sweller 1994]. This is because working memory resources that deal with ECL do not contribute to learning but must be allocated if the instructional procedures demand those resources [Aydinbas 2019].

The third component of CLT is germane cognitive load GCL. GCL describes the working memory resources dedicated to dealing with ICL of the learning material or, in other words, the proportion of a user's cognitive load that is *relevant* for the current learning goal. Hence, GCL does not contribute to the total cognitive load and is independent of the presented information.

A vivid visualization by Aydinbas showing the relation between the three types of cognitive load can be found in figure 2.6 [Aydinbas 2019].

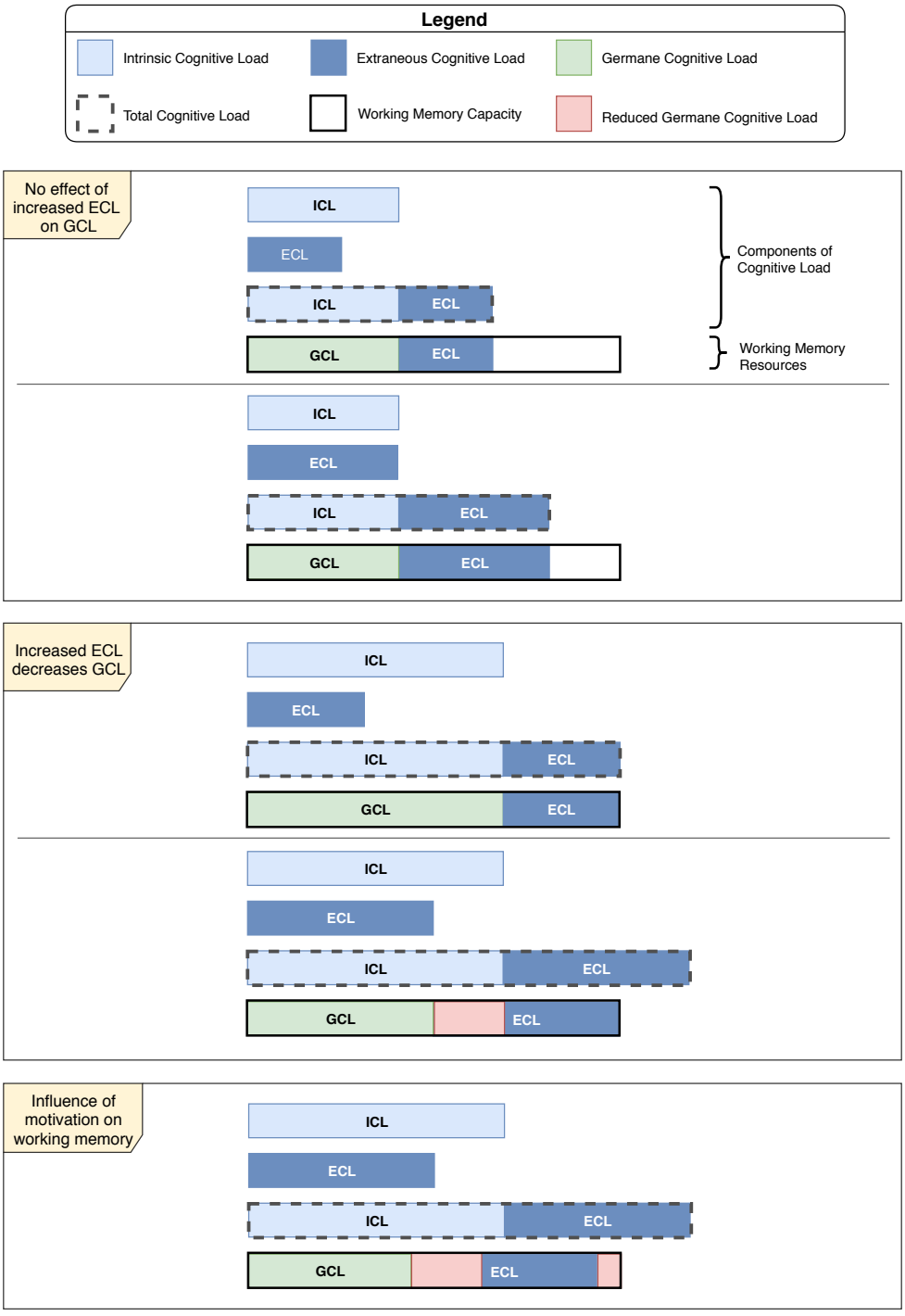


Figure 2.6: Influence of ECL and motivation on GCL for three different cases by [Aydinbas 2019] based on [Sweller 2010].

When both ICL and ECL are much lower than the working memory capacity, an increase in ECL does not influence the learning outcome because there are still enough working memory resources to deal with both ICL and ECL. In this case, ICL and GCL are equal since all of the ICL can be dealt with. This scenario is shown in the upper box of figure 2.6.

Let us assume that ICL is high, ECL is low, and together they roughly equal the working memory capacity. Again, ICL and GCL are equal because all of ICL can be dealt with. However, if ECL now increases, the sum of ICL and ECL becomes greater than the working memory capacity. ECL's required resources still have to be allocated, since they are necessary to understand and follow the instructions. This leaves fewer resources to deal with ICL than what would be needed. Therefore, GCL decreases and is now lower than ICL. This can be seen in the middle box of figure 2.6.

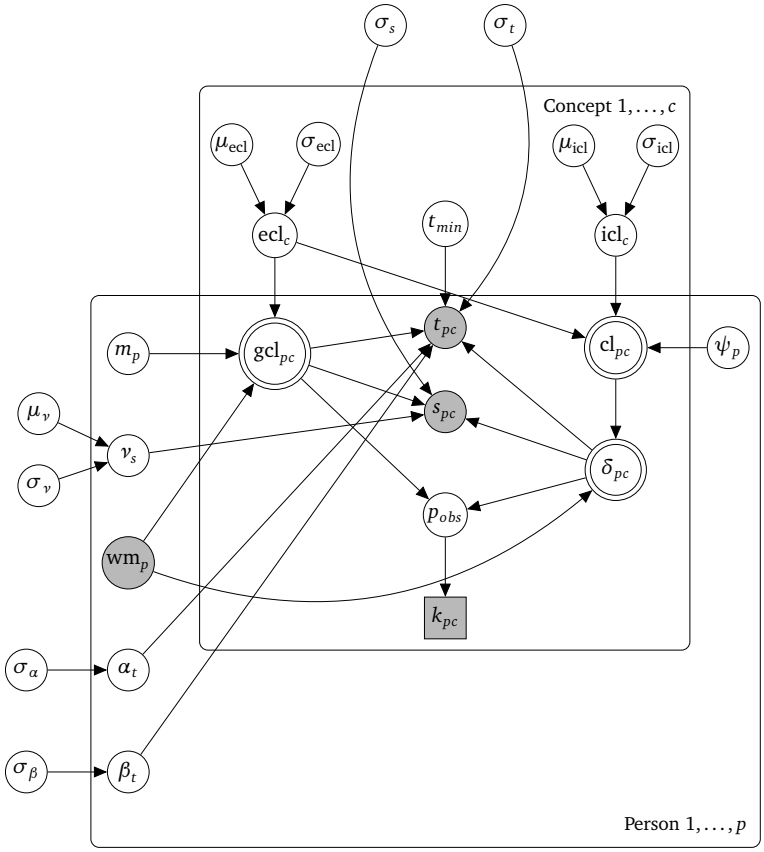
The bottom box of figure 2.6 depicts a scenario with the same initial situation as in the middle scenario: The sum of ICL and ECL equals the working memory capacity. If the learner's motivation decreases, he can use less of his total working memory capacity. This also leads to a decline of GCL.

2.4 CogIUM Framework

Aydinbas developed an initial prototype of the Cognitive Intelligent User Modeling Framework (CogIUM) as the product of his master's thesis [Aydinbas 2019]. Since it also forms the foundation of this work, it will be described in-depth in the following section.

CogIUM is - at its core - a Hierarchical Bayesian Model (HBM) (see section 2.1.1). Simply put, it receives observed data as input, combines this data with prior beliefs about the distribution of its latent variables, and outputs posterior distributions for all of its latent variables. Additionally, CogIUM ascribes a mechanistic meaning to its variables: The latent variables of the model were designed to represent cognitive variables like *cognitive load*, *motivation* and *prior knowledge*. Therefore, CogIUM's latent variables are also referred to as latent *cognitive* variables. Those variables are used to explain the observable variables, variables that model the observed input data. In the final CogIUM model by Aydinbas, the observable variables were **mission success**, **mission score** and **mission time**. A graphical representation of CogIUM's structure is shown in figure 2.7. The model's large structure and complexity include numerous design choices and mathematical assumptions. An overview of those is given in the following section:

CogIUM models the three observable variables *mission success*, *mission score* and *mission time*. Before explaining how the observable variables are modeled, an introduction to CogIUM's most prominent latent cognitive variables and their meaning is required:



Priors

$$
\begin{aligned}
m_p &\sim \text{Beta}(3,1) \\
\psi_p &\sim \text{Beta}(2,2) \\
t_{min} &\sim \text{Gamma}(2,0.5) \\
\mu_{\text{icl}} &\sim \text{Gamma}(2,0.5) \\
\mu_{\text{ecl}} &\sim \text{Gamma}(2,0.5) \\
\mu_\nu &\sim \text{Normal}(0.5,0.01) \\
\sigma_{\text{icl}} &\sim \text{HalfCauchy}(3) \\
\sigma_{\text{ecl}} &\sim \text{HalfCauchy}(3) \\
\sigma_s &\sim \text{HalfCauchy}(1) \\
\sigma_t &\sim \text{HalfCauchy}(1) \\
\sigma_\nu &\sim \text{HalfCauchy}(1) \\
\sigma_\alpha &\sim \text{HalfCauchy}(1) \\
\sigma_\beta &\sim \text{HalfCauchy}(1)
\end{aligned}
$$

Intermediate variables

$$
\begin{aligned}
\text{icl}_c &\sim \text{Normal}(\mu_{\text{icl}}, \sigma_{\text{icl}})_{\mathbb{I}(1,\infty)} \\
\text{ecl}_c &\sim \text{Normal}(\mu_{\text{ecl}}, \sigma_{\text{ecl}})_{\mathbb{I}(0,\infty)} \\
p_{obs} &\sim \text{Beta}\left(1 + \text{gcl}_{pc}, 1 - \delta_{pc}\right) \\
\nu_s &\sim \text{Normal}(\mu_\nu, \sigma_\nu)_{\mathbb{I}(0,1)} \\
\alpha_t &\sim \text{HalfNormal}(\sigma_\alpha) \\
\beta_t &\sim \text{HalfNormal}\left(\sigma_\beta\right)
\end{aligned}
$$

Deterministic variables

$$
\begin{aligned}
\text{wm}_p &= 7 \\
\text{cl}_{pc} &\leftarrow (1 - \psi_p) \cdot \text{icl}_c + \text{ecl}_c \\
\text{gcl}_{pc} &\leftarrow \begin{cases} 0, \text{ if } m_p \cdot \text{wm}_p - \text{ecl}_c < 0 \\ \dfrac{m_p \cdot \text{wm}_p - \text{ecl}_c}{\text{wm}_p}, \text{ else} \end{cases} \\
\delta_{pc} &\leftarrow \frac{\text{wm}_p - cl_{pc}}{\text{wm}_p}
\end{aligned}
$$

Observable variables

$$
\begin{aligned}
s_{pc} &\sim \text{Normal}\left(\text{gcl}_{pc} + \nu_s \cdot \delta_{pc},\ \sigma_s\right) \\
k_{pc} &\sim \text{Bern}\left(p_{obs}\right) \\
t_{pc} &\sim \text{Normal}(t_{min} + \alpha_t \cdot (1 - \text{gcl}_{pc}) - \\
&\qquad \beta_t \cdot \delta_{pc},\ \sigma_t)
\end{aligned}
$$

Figure 2.7: Final model of the CogIUM protoype by [Aydinbas 2019].

- The variable ψ_{pc} represents the prior knowledge of a user p regarding concept c.
- The variable m_{pc} models the motivation of a user p while playing concept c.
- Conforming to Cognitive Load Theory (CLT)(see section 2.3), CogIUM includes three types of cognitive load: icl_c describes the intrinsic complexity of concept c; ecl_c describes the cognitive load caused by the instructional design of concept c; gcl_{pc} describes the germane cognitive load specific to a user p and a concept c. A user's total cognitive load is calculated by the formula $cl_{pc} = (1 - \psi_{pc}) \cdot icl_c + ecl_c$. The more prior knowledge a user has in the domain of concept c, the less influence icl_c has on his total cognitive load.
- δ_{pc} represents a user's free working memory capacity. It is calculated by subtracting a user's total cognitive load cl_{pc} from his working memory capacity wm_p and then normalizing to the range [0,1]: $\delta_{pc} = \frac{wm_p - cl_p}{wm_p}$

Using the latent cognitive variables outlined above, CogIUM accounts for the three observable variables *mission success* k_{pc}, *mission score* s_{pc} and *mission time* t_{pc}. For each observable variable, the exact underlying model structure and the corresponding design choices are described in the following listing:

- Mission success k_{pc} is modeled as a Bernoulli distribution with success probability p_{obs}. The main advantage of the Bernoulli distribution is that it produces only binary outcomes, either success or failure, or 1 and 0, which is exactly the domain of the variable task success. The success probability p_{obs} is again modeled as a distribution, more specifically a beta distribution. Germane cognitive load gcl_{pc} influences the a parameter and free working memory capacity δ_{pc} influences the b parameter of the beta distribution. The equations $a = 1 + gcl_{pc}$ and $b = 1 - \delta_{pc}$ were chosen to ensure that k_{pc} has a chance of 50% to be either 1 or 0 when $gcl_{pc} = 0$ and $\delta_{pc} = 0$, that is, when the learner dedicates no resources to the task and the learning task demands all available resources. A positive gcl_{pc} increases the success probability as well as a positive δ_{pc} . A negative gcl_{pc} decreases the success probability as well as a negative δ_{pc}.
- Mission score s_{pc} is modeled as normally distributed with a mean value determined by germane cognitive load gcl_{pc}, free working memory capacity δ_{pc}, and a global standard deviation σ_s. The distribution is defined by the following formula:

$$s_{pc} \sim \mathcal{N}(gcl_{pc} + v_s \cdot \delta_{pc}, \sigma_s) .$$

  - v_s is a multiplicative factor that governs the degree to which free working memory capacity δ_{pc} influences gcl_{pc}. Each subject has its own value for v_s. Additionally, v_s is modeled hierarchically in the way that the values for all subjects are drawn from a normal distribution with global parameters μ_v and σ_v.
  - The standard deviation σ_s is a global variable, thus it is assumed that the score deviates from its distribution center the same for all subjects and all missions.

- Mission time t_{pc} is influenced by three distinct parts: a minimal time t_{min}, a contribution from germane cognitive load gcl_{pc}, and a contribution from free working memory capacity δ_{pc}. Minimal time t_{min} is modeled as a concept group variable. The underlying assumption is that each mission has a minimal time required to complete it. Mission time t_{pc} is postulated to have the following distribution:

$$t_{pc} \sim \mathcal{N}(t_{min} + \alpha_t \cdot (1 - gcl_{pc}) - \beta_t \cdot \delta_{pc}, \sigma_t).$$

Now that all involved variables - latent and observable - have been introduced, I will present the latent variable's modeling in more detail.

As outlined above, CogIUM includes the three different types of cognitive load defined by CLT. Intrinsic Cognitive Load (ICL) is represented in CogIUM by the variable icl_c. icl_c is modeled hierarchically with a mean value and standard deviation that are also latent cognitive variables. It is a conceptual variable, thus only varies between missions, not between subjects. Furthermore, icl_c is always positive and at least 1 because each mission must have at least one learning element. Extrinsic Cognitive Load (ECL) is also a conceptual variable within CogIUM denoted ecl_c. Like icl_c, it is modeled hierarchically by a Normal distribution with a mean value and standard deviation that are also latent cognitive variables. ECL represents the cognitive effort necessary to process a mission or task's instructions, but not the learning material itself. In an ideal learning environment, ECL should be as low as possible so that the subject can fully focus on the actual learning process. ECL is assumed to have the possibility to be 0, but not negative. The third component of CLT is Germane Cognitive Load (GCL). It represents the amount of a user's working memory dedicated to handling the ICL. The total available amount of working memory capacity is retrieved by multiplying the motivation m_p with working memory capacity wm_p. Since motivation m_p is normalized between 0 and 1, it determines the *proportion* of working memory capacity available for the current learning task. From this capacity, the value of ecl_c is subtracted because the learner has to deal with ECL to understand the instructions. It is limited to positive values because negative values are neither plausible nor interpretable. To allow a comparison between different users, gcl_{pc} is normalized

by the working memory capacity wm_p because this variable can differ between subjects. This gives the following formula:

$$gcl_{pc} = \begin{cases} 0, & \text{if } m_p \cdot wm_p - ecl_c < 0 \\ \frac{m_p \cdot wm_p - ecl_c}{wm_p}, & \text{else} \end{cases}$$

Due to the normalization, gcl_{pc} is always between 0 and 1. A value for gcl_{pc} of 1 means that there is no ecl_c and all of the working memory capacity is dedicated to learning. A value for gcl_{pc} of 0 can mean one of two things, or a combination: Either the motivation is so small that there was not enough initial working memory capacity, or the value for ECL is higher than the working memory capacity available to deal with ICL. The total cognitive load cl_{pc} is the sum of icl_c and ecl_c.

In addition to the components of cognitive load, another essential variable within CogIUM is prior knowledge ψ_p. It represents a user p's level of domain knowledge. Prior knowledge ψ_p is assumed to determine the influence of ICL on a user's total cognitive load. The more domain knowledge a user has, the less ICL should be imposed on him by the learning material. Therefore, ψ_p is modeled as a multiplicative factor between 0 and 1 that determines how much the ICL icl_c contributes to the total cognitive load cl_{pc}.

Each user p has a personal variable working memory capacity wm_p that represents how many learning elements a subject can process in parallel. Initially, it was modeled by Aydinbas as normally distributed around a mean value of 7 with a standard deviation of 1 to reflect the knowledge about the general limitations of the working memory [Chong 2005]. However, Aydinbas found during his validation that the model used the variable to explain individual differences by setting it close to 0 or high above 7. Since this did not correspond to the variable's designed range, it was set to a fixed value of 7 [Aydinbas 2019].

This concludes the description of CogIUM's complex structure. When observed data is provided as input, CogIUM computes posterior probability distributions for all latent cognitive variables. This is referred to as CogIUM inference. Because the analytical calculation of the posterior distributions is intractable in such a large model, an approximation of the distributions is obtained by drawing a large number of samples. CogIUM's sampling process can take up to 10 minutes on a modern laptop's CPU when using a reasonable sample size of at least 1000 samples.

2.5 Educational Serious Games - Lost Earth

To showcase and evaluate CogIUM's functionality, an educational serious game was required as a demonstrator. For this purpose, *Streamlined Lost Earth* (SLE) was chosen. SLE is a version of *Lost Earth*, an educational serious game in the domain of image exploitation developed at Fraunhofer IOSB. The following section provides an overview of the *Lost Earth* project and its different versions.

2.5.1 Lost Earth 2307

Lost Earth 2307 is an educational serious game designed to support training of image interpreters. It is a turn-based 4X strategy game. 4X stands for explore, expand, exploit and exterminate. By playing the game, users should learn how to analyze aerial and satellite images and formally describe the identified objects. The image data can originate from optical, radar, or infrared sensors. It was developed by the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB) to enhance the education of professional image interpreters at the Air Force Training Center for Image Reconnaissance (AZAALw) of the German Armed Forces [IOSB 2016]. Interpretation of aerial imagery is an important tool for reconnaissance and can be used to gather information in inaccessible locations. Radar image interpretation, for example, is used in search and rescue operations to find missing earthquake victims [Streicher and Smeddinck 2016].

The game is set in a science-fiction scenario in the future, where a vicious cult took over the galaxy and oppressed humankind. As a rebel organization member, the user must free the galaxy that consists of multiple colonies by solving missions that include image interpretation tasks. As the game progresses, the missions become more and more challenging. To immerse the user in the game while also achieving a maximum learning outcome, the learning objectives and game objectives were designed to correlate very closely [Atorf, Kannegieser, and Roller 2019]. The game features two different kinds of missions: reconnaissance missions and deployment missions.

In a reconnaissance mission, the user first receives a briefing on that mission's objectives. Next, he has to task a sensor and a platform to record the imagery. Depending on the weather conditions at the target location, the user must choose the right sensor and platform in order to succeed. Once the imagery recorded by the sensor becomes available, he has to analyze, annotate and align the imagery and finally fill out a report with the results. If the report's quality is sufficient, the user successfully completed the mission.

The second type of missions are deployment missions. They focus on illustrating the

advantages and disadvantages of specific sensors and platforms [Atorf, Kannegieser, and Roller 2019]. Each sensor or platform in the game has some strengths and weaknesses that can be analogously found in real-world sensors and platforms [Streicher and Smeddinck 2016].

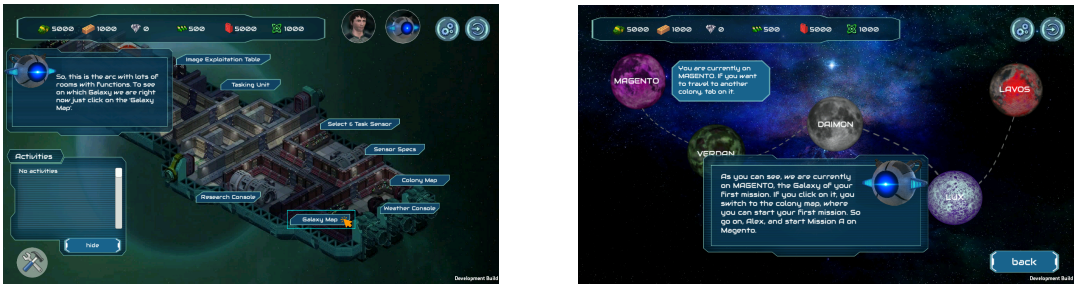
2.5.2 Streamlined Lost Earth

Streamlined Lost Earth is a heavily adapted version of Lost Earth 2307. Extensive evaluations of Lost Earth 2307 showcased the need to improve the game's usability, provide shorter missions that can be completed in a few minutes and improve accessibility by evolving it from a desktop application to a web-based application [Atorf, Kannegieser, and Dillig 2020].

The streamlined version only provides a few missions to showcase the functionality. Its primary advantage is that it is accessible in a browser. This also means that it can be used for online user studies, which is not possible with LE 2307 since it is not web-based. In the game, the user receives help from the assistant L.I.S.A., who introduces the game's functionality and walks the user through his first mission. The L.I.S.A. assistant can give the user explanations in a separate dialogue or highlight the section of the screen that is relevant for the next necessary step, as shown in figure 2.8. The image exploitation tasks, as well as the general game structure, are a lot more simplified than in LE 2307 (see figure 2.10).

(a) (b)

Figure 2.8: L.I.S.A. dialogue and highlighted screen section in SLE



(a) (b)

Figure 2.9: Main Menu and Galaxy Map in SLE

(a) (b)

Figure 2.10: Image Exploitation Tasks in SLE

2.6 ELAI Framework

To facilitate CogIUM's applicability and interoperability, this thesis's goal is to integrate CogIUM within the E-Learning Artificial Intelligence (ELAI) framework.

ELAI is an interoperable intelligent tutoring framework developed at Fraunhofer IOSB, with the objective to enable interoperable adaptivity for educational serious games. By adapting the game to the user, the goal is to optimize his learning experience and ultimately maximize his learning outcome. The framework's architecture decouples the game engine and the adaptive entity, allowing for better interoperability [Streicher and Roller 2017].

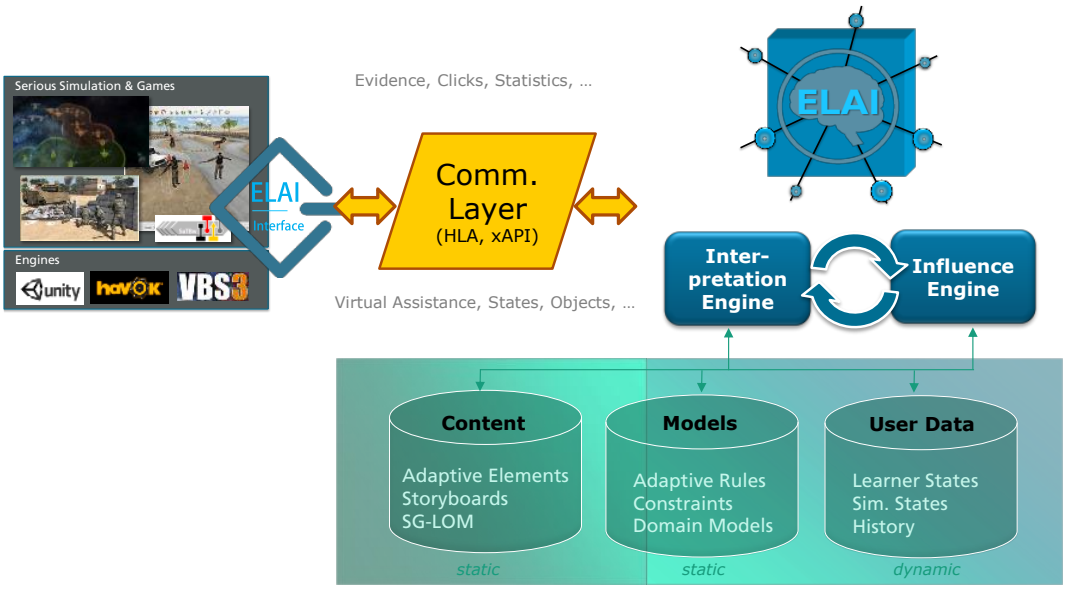


Figure 2.11: The E-Learning Artificial Intelligence framework [Streicher and Roller 2015]

It consists of a game engine adapter and an ELAI controller (see figure 2.11). The game engine adapter is responsible for capturing usage data and presenting the adapted content. For tracking a user's interaction with a game, the framework makes use of the Experience API (xAPI) protocol introduced in section 2.2.

The ELAI controller is the architecture's central "intelligent" element, also referred to as the tutoring component. Within the ELAI controller, the interpretation engine and influence engine compute optimal adaptations based on the collected user data: Captured usage data is analyzed by the interpretation engine. Subsequently, adequate adaptive measures are selected by the influence engine. Internally, the interpretation engine can use different microservices to analyze usage data. CogIUM is implemented as such a microservice.

3 State of the Art

This chapter presents state-of-the-art research on relevant topics for this thesis. The first part focuses on methods to gather data on latent cognitive variables. Regarding Cognitive Load Theory (CLT), it is discussed how a user's cognitive load can be measured or inferred. As a benchmark for self-assessment surveys, the NASA Task Load Index (NASA TLX) is introduced to the reader [Hart and Staveland 1988]. Modern adaptations and enhancements are presented as well as entirely new approaches. In the second part, research on the evaluation and comparison of Bayesian models is presented.

3.1 User Modeling - Measuring Latent Cognitive Variables

As the name suggests, latent cognitive variables are *latent* or *hidden*, which means that they cannot be easily measured. However, to model a learner's cognitive state, knowledge about those variables is crucial. In this section, different approaches to the collection of data on latent cognitive variables are presented. After a general overview, the measurement of cognitive load is covered in detail.

[Conati et al. 2020] investigated the usage of interaction data as an information source to predict cognitive abilities. In a user study, they compared the predictive performance for cognitive abilities using only interaction data, only eye-tracking data and both interaction and eye-tracking data. The researched cognitive abilities were perceptual speed, visual working memory, spatial memory, visual scanning and visualization literacy. To measure the cognitive abilities, the participants had to take a series of tests after completing the actual task. While eye-tracking data generated the most accurate predictions, results showed that interaction data can still outperform a majority-class baseline. Additionally, it was found that interaction data can predict several cognitive abilities with better accuracy at the very beginning of the task than eye-tracking data, which is valuable for delivering adaptation early in the task. *Left click rate* and *time to first click* were the top two predictors for all cognitive abilities, suggesting the importance of those two features for predicting cognitive abilities. [Conati et al. 2020] concluded that adaptation for interactive visualizations tasks could be enabled using solely interaction data. This is also the approach taken in this thesis.

3.1.1 Measuring Cognitive Load

The methods used for measuring cognitive load can be divided into four broad categories [Brunken, Plass, and Leutner 2003]: subjective direct (self-reported stress), subjective indirect (self-reported mental effort), objective direct (brain signals and dual task performance), and objective indirect (physiological).

A widely used subjective measure of cognitive load is perceived task difficulty, which typically consists of rating the perceived difficulty of a task on a 7- or 9-point Likert scale. Paas et al. found that people are quite capable of giving a numerical indication of their perceived mental burden [F. Paas et al. 2003]. Studies have shown that reliable and non-intrusive measures of cognitive load can be obtained with uni-dimensional scales [F. G. Paas and Van Merriënboer 1994]. [Jovanović et al. 2019] provided study participants with a 2D-Canvas (see figure 3.1) to rate their perceived task difficulty. In their study, they found significant association between trace-based measures of examined learning constructs - cognitive load and self-efficacy - with some indicators of the students' engagement with learning activities as well as with the students' final exam score.

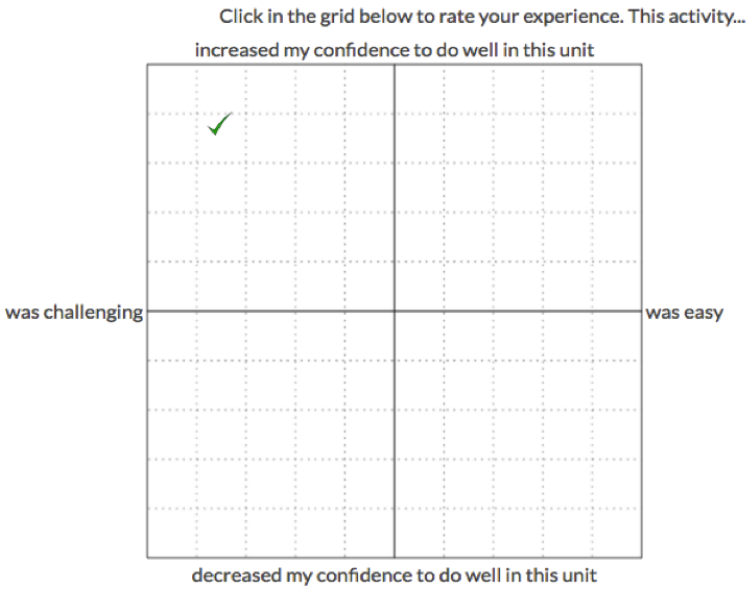


Figure 3.1: 2D canvas tool for self-reporting of perceived task difficulty by [Jovanović et al. 2019].

In conclusion, rating scales have been a frequently used method to obtain cognitive load measures and have been widely accepted by the research community [Jovanović et al. 2019]. However, there are two feasible approaches when measuring cognitive load with rating scales:

Students can be asked to rate their cognitive load immediately after each subtask and eventually the average rating can be used as the cognitive load measure. Alternatively, students can be asked to rate their cognitive load only once, after they finished the whole task. Schmeck et al. investigated the differences of those two approaches by asking study participants to rate their perceived effort and task difficulty six times during problem solving and once after they finished the problem. They found that the delayed ratings of both effort and difficulty were significantly higher than the average of the six ratings made during problem solving [Schmeck et al. 2015]. However, they also found that for ratings of affective variables such as interest and motivation, the delayed rating did not differ from the average of immediate ratings. It is still subject to debate which of the two methods is better. For more abstract concepts like learning strategies and learning behaviour, research has shown that students' survey-based self-reports about properties of learning and traces of their actual learning activities are not well aligned [Jamieson-Noel and P. Winne 2003]. In particular, students tended to overestimate their use of study tactics. Zhou and Winne attribute this to the fact that students may have incomplete and reconstructed memories as well as subjective and implicit theories of the mental processes involved [Zhou and P. H. Winne 2012].

Another popular approach to infer cognitive load is through eye-tracking, which falls into the category of objective indirect measures. Buettner proposes the inference of cognitive load based on four eye-tracking measurements [Buettner 2013]: Pupillary diameter mean, pupillary diameter standard deviation, number of gaze fixation and saccade speed. This method is also used by Sharma et al. in a lab study to determine the causal relationship between cognitive load and information flow [Sharma et al. 2021]. It must be stated that for this type of inference, specialized eye-tracking equipment is required to precisely measure pupil diameter, fixation and saccades. This limits the practical use-cases to lab settings.

One attempt to infer cognitive load with low cost gear was made by [Gjoreski, Luštrek, and Pejović 2018]: With the help of a cheap off-the-shelf wearable device physiological data was recorded. The data included pulse rate, galvanic skin response and skin temperature. During the recording, the users were exposed to cognitive tasks of varying difficulty. Subsequently, the authors use Machine Learning algorithms to map the recorded physiological data to collected reference data. This approach has the benefit that it is non-intrusive in contrast to rating scales and does not require expensive equipment. However, Gjoreski et al. reported poor predictive performance of the trained ML algorithm. As reference data, the user's cognitive load was collected through a self-assessment of the user.

For the self-assessment, Gjoreski et al. use the NASA Task Load Index that is explained in the following section 3.1.2.

3.1.2 NASA Task Load Index

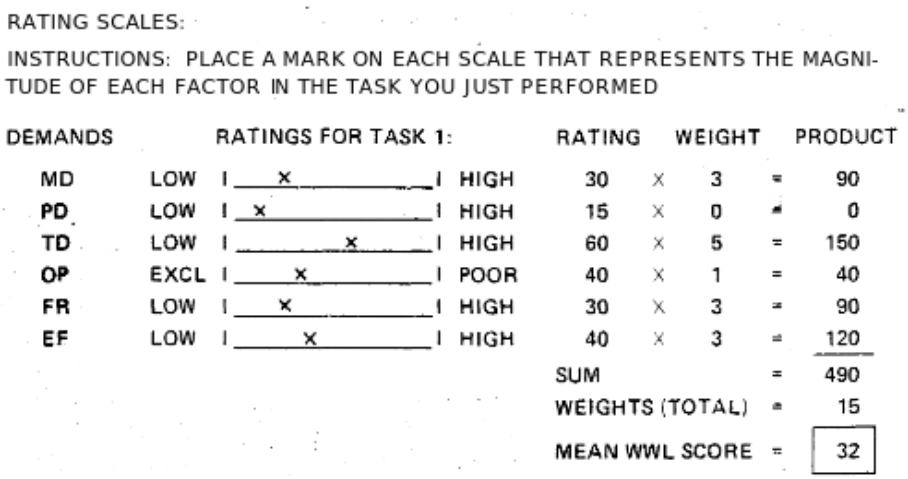
Since the NASA Task Load Index (NASA TLX) was presented in 1988 by [Hart and Staveland 1988], it has become a benchmark against which the efficacy of other measures, theories, or models are judged [Hart 2006]. Its use has spread far beyond its original application (aviation), focus (crew complement), and language (English). It has been cited in over 550 studies and has been transferred into several modern formats like an iOS app and a web app [Sharek 2011].

The NASA TLX comprises two parts: In the first part, users are asked to rate six workload-related factors on a multi-dimensional rating scale (see figure 3.2). The second part consists of pair-wise comparisons of those workload-related factors: For every pair, the user has to choose which factor provided the most significant source of workload variation in the completed task (see figure 3.3). Eventually, the ratings from the first part are weighted based on how often they were picked in the second part. Those weighted ratings are then summed up and divided by a normalization term. This yields a total workload score in the range of 0 to 100.

The six involved workload-related factors are defined as follows:

- MENTAL DEMAND (MD): How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)? Was the task easy or demanding, simple or complex, exacting or forgiving?
- PHYSICAL DEMAND (PD): How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
- TEMPORAL DEMAND (TD): How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?
- OWN PERFORMANCE (OP): How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?
- FRUSTRATION LEVEL (FR): How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?
- EFFORT (EF): How hard did you have to work (mentally and physically) to accomplish your level of performance?

Over the last 20 years, the most common modification made to NASA-TLX has been to eliminate the weighting process altogether or weighting the subscales and then analyzing them

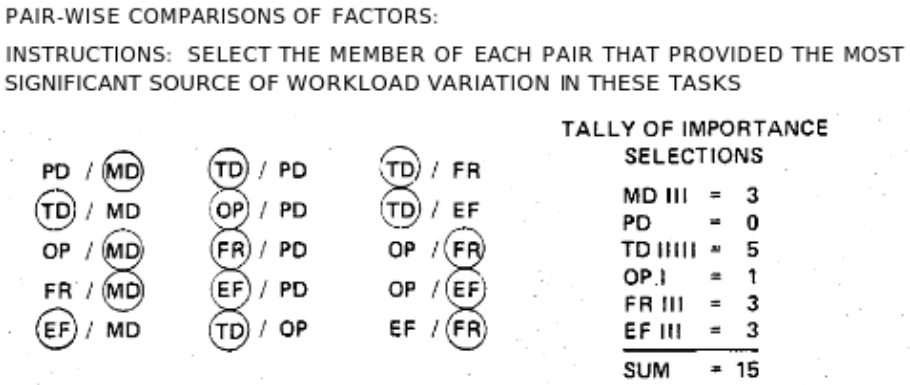


RATING SCALES:

INSTRUCTIONS: PLACE A MARK ON EACH SCALE THAT REPRESENTS THE MAGNITUDE OF EACH FACTOR IN THE TASK YOU JUST PERFORMED

DEMANDS	RATINGS FOR TASK 1:			RATING		WEIGHT		PRODUCT
MD	LOW	I ×———— I	HIGH	30	×	3	=	90
PD	LOW	I ×———— I	HIGH	15	×	0	=	0
TD	LOW	I ———— × I	HIGH	60	×	5	=	150
OP	EXCL	I ——×——— I	POOR	40	×	1	=	40
FR	LOW	I —×——— I	HIGH	30	×	3	=	90
EF	LOW	I ——×——— I	HIGH	40	×	3	=	120
				SUM			=	490
				WEIGHTS (TOTAL)			=	15
				MEAN WWL SCORE			=	32

Figure 3.2: NASA TLX scales. The acronyms are explained in section 3.1.2.



PAIR-WISE COMPARISONS OF FACTORS:

INSTRUCTIONS: SELECT THE MEMBER OF EACH PAIR THAT PROVIDED THE MOST SIGNIFICANT SOURCE OF WORKLOAD VARIATION IN THESE TASKS

PD / (MD)	(TD) / PD	(TD) / FR
(TD) / MD	(OP) / PD	(TD) / EF
OP / (MD)	(FR) / PD	OP / (FR)
FR / (MD)	(EF) / PD	OP / (EF)
(EF) / MD	(TD) / OP	EF / (FR)

TALLY OF IMPORTANCE SELECTIONS

MD III	=	3
PD	=	0
TD IIIII	=	5
OP I	=	1
FR III	=	3
EF III	=	3
SUM	=	15

Figure 3.3: NASA TLX weighting of contributing factors. The pair-wise comparisons determine the influence of the ratings from figure 3.2 on the total cognitive load.

individually [Hart 2006]. This is referred to as Raw TLX (RTLX) and has gained popularity because it is easier to apply. Studies that compare NASA-TLX and Raw TLX have shown mixed results: In some, RTLX was reported to be more sensitive than NASA TLX, some reported them to be equally sensitive, and some reported NASA TLX to be more sensitive [Hart 2006]. Therefore, it remains unclear how the weighting scale's omission influences the ratings' accuracy.

For this thesis, the NASA TLX is used to collect users' self-assessment about their perceived cognitive load. Since the users are only asked to fill out the assessment twice, it was decided that the weighting process can be included without over-straining the user study participants. A detailed description of the design choices made for the user study in this thesis is given in section 6.1.

3.2 Model Evaluation and Model Comparison

When creating or extending Bayesian models, there is a need for methods to compare those models with regard to their quality. The primary objective of such a model is predictive accuracy: The model should be able to predict new data as well as possible.

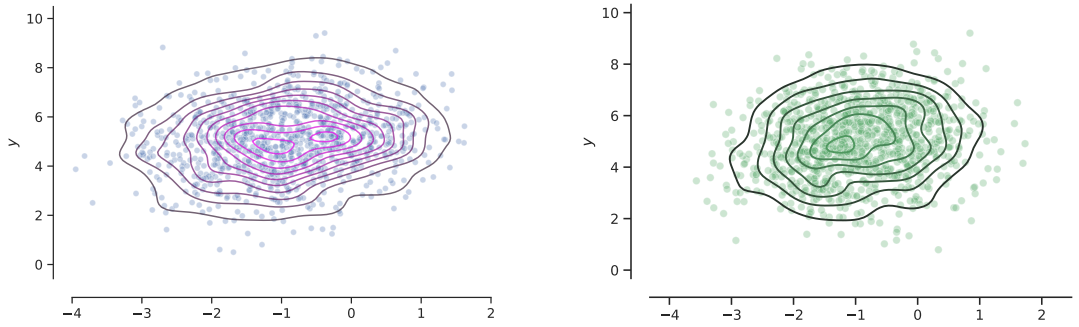
Gelman et al. state that checking the model is crucial to statistical analysis. Bayesian prior-to-posterior inferences assume the whole structure of a probability model and can yield misleading inferences when the model is poor [Gelman et al. 2013]. Therefore, good Bayesian analysis should include at least some check of the adequacy of the model's fit to the data and the model's plausibility for the purposes for which the model will be used.

The most common way to check the fit of a model is by performing a posterior predictive check. Posterior predictive distribution denotes a probability distribution over possible values of future data $\tilde{y}$. The posterior predictive distribution is analytically intractable in a realistically sized model because of the high-dimensional intervals involved. Therefore, sampling is required to obtain an approximation of the posterior predictive distribution. The process of drawing a sample consists of two steps [Lambert 2018]:

1. Sample $\theta^s \sim p(\theta|y)$, that is, sample a parameter value from the posterior distribution.
2. Sample $\tilde{y}_i \sim p(\tilde{y}|\theta^s)$, that is, sample a data value from the sampling distribution conditional on the parameter value from the previous step.

Those two steps are iterated. The drawn samples approximate the shape of the posterior predictive distribution. Increasing the number of drawn samples will also increase the approximation quality. In Bayesian data analysis, posterior predictive distributions are used to check

whether the model can reproduce the observed data's most important features. Gelman et al. state that if the model fits, then replicated data generated under the model should look similar to observed data [Gelman et al. 2013]. A minimal but vivid example for this is given in figure 3.4 by [Aydinbas 2019]. Figure 3.4a shows the observed data that the model will try to fit. The observed data shall be modeled with a multivariate normal distribution. For the sake of simplicity, it is assumed that the covariance matrix Σ is known. Hence, only the mean values μ_1 and μ_2 are the unknown latent variables. Using sampling, a posterior distribution for both unknown variables can be obtained. Drawing from those posterior distributions and then drawing from the Normal distributions parameterized by the two unknown variables yields a sample of the posterior predictive function. Each of those samples is represented as a green dot in figure 3.4b on the right. Visually comparing the observed data to the simulated data from the posterior predictive function, it is evident that the model is able to reproduce the key features of the observed data. This procedure is called posterior predictive checking and is usually a lot more complex than in this toy example.



(a) Two-dimensional observations obtained from a two-dimensional multivariate normal distribution.

(b) 1000 samples drawn from the posterior predictive distribution of the model. The density is approximated by a kernel density estimation.

Figure 3.4: Example of Bayesian data analysis fitting a multivariate normal distribution with unknown mean and known co-variance matrix to two-dimensional data by [Aydinbas 2019].

Comparing models is especially important since it is typically the case that more than one reasonable probability model can provide an adequate fit to the data in a scientific problem. When comparing models, two scenarios are distinguished:

First, the scenario of an existing model that is extended. By expanding the model, new complexity is added. The benefits that come with this extra complexity must be assessed to decide if they are worth the added complexity. While the larger model typically has the advantage of making more sense and fitting the data better, it also has the disadvantage of

being more difficult to understand and compute [Gelman et al. 2013]. The key questions to answer are: (1) is the improvement in fit significant enough to justify the additional difficulty in fitting, and (2) is the prior distribution on the additional parameters reasonable?

The second scenario involves two unrelated models, meaning that neither of the models is a generalization of the other. In this case, the goal is usually not to pick one over the other. Instead, it can be useful to combine them into a larger model as special cases [Gelman et al. 2013].

To quantify predictive performance, the most widely used metric in literature is the mean squared error (MSE) [Pelánek 2015]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - E(y_i|\theta))^2$$

It uses a single value as a point prediction for all future data and computes the sum of squared differences from this point prediction. The MSE has the advantage of being easily computed and directly interpretable, but the disadvantage of being less appropriate for models that are far from the normal distribution [Gelman et al. 2013]. Point predictions do not represent the full uncertainty over the unobserved data $\tilde{y}$ and are thus not optimal for quantifying a model's posterior predictive performance.

Several methods exist to estimate the out-of-sample predictive accuracy using available data. One of the most popular approximation methods is the *Widely Available Information Criterion* (WAIC). WAIC is a fully Bayesian approach for estimating the out-of-sample expectation, referred to as the *expected log pointwise predictive density* elppd (see 3.2). Starting with the *computed log pointwise predictive density* (computed lppd, see 3.1) a correction is added for the effective number of parameters to adjust for overfitting [Gelman et al. 2013].

$$\text{computed llpd} = \sum_{i=1}^{n} log(\frac{1}{S}\sum_{s=1}^{S} p(y_i|\theta^s)) \tag{3.1}$$

$$\text{elppd} = \sum_{i=1}^{n} E_f(logp_{post}(\tilde{y}_i)) \tag{3.2}$$

Gelman et al. describe the characteristics of WAIC as more appealing than other approximation methods [Gelman et al. 2013]. During the development process of the original CogIUM prototype, Aydinbas also utilized WAIC to compare different models [Aydinbas 2019].

4 Concept - Application of Cognitive Modeling to Serious Games

This chapter presents the concept and design choices for the expansion and remodeling of the initial CogIUM model. For the remainder of this thesis, **CogIUMa** (CogIUM applied) will denote the enhanced CogIUM model that is the result of this thesis. In the first section, an identified usage scenario will be described step-by-step to clarify the expanded model's requirements. Since the overall concept's goal is to apply CogIUM to an existing serious game, a chain of several aspects has to be covered: First, input data needs to be made available, and the format and content of the input data need to be defined. This process is described in section 4.2. Using more input data also requires structural changes within the CogIUM model since new observable variables are added. Additionally, some design flaws of the initial CogIUM model are tackled, some of which were already identified in the conclusion of Aydinbas' thesis [Aydinbas 2019]. The expansion and remodeling of CogIUM is covered in section 4.3. Output of the CogIUM model are posterior distributions for all latent cognitive variables, as described in section 2.4. From those posterior distributions, an Adaptivity Response (AR) needs to be computed. Furthermore, an approach is explored to distinguish *online* and *offline* inference, allowing fast and slightly less accurate inference if there is no time for the whole sampling process. In section 4.4, this will be described in detail. Finally, CogIUM is to be integrated into the ELAI framework, which is discussed in section 4.5.

4.1 Usage Scenario

Before the concept for the extension of CogIUM was developed, requirements were investigated by creating a step-by-step usage scenario. The identified usage scenario is displayed in figure 4.1 and features the following steps:

Initially, the user opens up SLE and is asked to log in with a unique user id. SLE requests an Adaptivity Response (AR) from the ELAI framework. The ELAI framework forwards this request to CogIUMa. CogIUMa checks if there is a user model for that user id in the database. If there is, an AR is computed from the current context and stored in the user model.

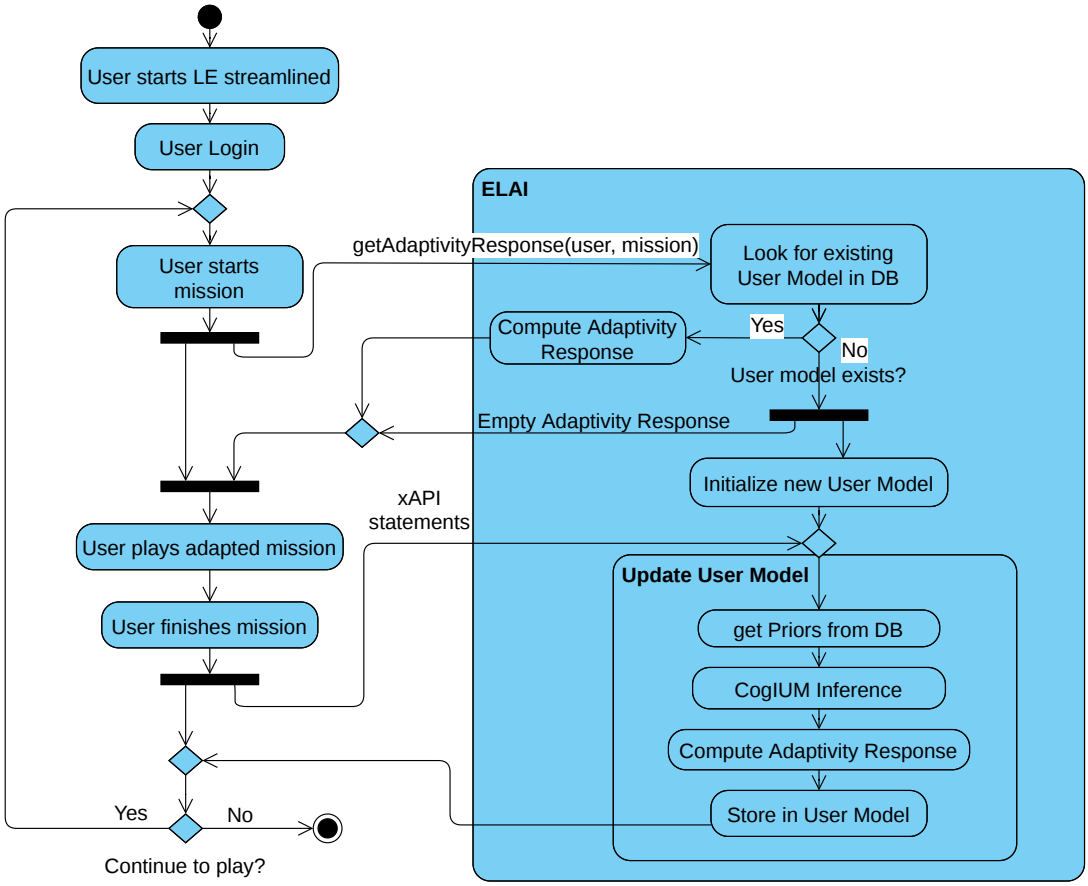


Figure 4.1: Usage Scenario for CogIUMa.

Subsequently, the AR is sent back to ELAI and from there to SLE, where the next mission is to be adapted accordingly. If no user model is available, no Adaptivity Response is computed, and a new user model is initialized. Next, the user starts the (potentially adapted) mission and plays through it. For every action the user performs, SLE sends an xAPI statement to the Learning Record Store (LRS).

Once he finishes the mission, xAPI data from that mission becomes available in the LRS. CogIUMa polls this data from the LRS. For the current user and mission, prior distributions for CogIUM's HBM are retrieved from the user database. If an inference for that user has been computed before, those prior distributions will come from the past inference. If not, the prior distributions will be uninformative. The newly available xAPI data is parsed to obtain the observed variables. With the prior distributions and the observed variables, CogIUM Inference is performed. This means sampling the posterior distribution of all latent cognitive variables to

obtain an approximation of the posterior distribution's shape. For a detailed explanation, see section 2.1.2. Since it is not feasible to make the user wait for the sampling process while he is playing the game, a faster and slightly less accurate inference method is explored in section 4.4. Once the posterior distributions become available, an AR, also referred to as *score*, must be computed. This score is then stored in the user model and sent back to ELAI and from there to SLE, where appropriate adaptation can take place for the next mission.

4.2 Feature Extraction

As input, CogIUMa polls xAPI data from an LRS. As can be seen in figure 4.2, this xAPI data is a stream of xAPI statements. In the figure, a minimal example of an xAPI stream from SLE is visualized by the software tool ELAISim [Streicher, Bach, and Roller 2019].

From this stream of xAPI statements, the observable variables of the CogIUM model are extracted. For the original CogIUM model, those observable variables were **mission success**, **mission score** and **mission time**. Since those values represent only part of the information present in the xAPI statements, the goal was to extract additional information from the xAPI statements as further input for CogIUM. This meant introducing new observable variables. One of the main guidelines during the CogIUM model development was to ensure interoperability, meaning that CogIUM can work with arbitrary educational serious games. Hence, the variables it uses as input must not be specific to SLE but must also be present in comparable games. Analyzing LE 2308, [Aydinbas 2019] identified potential candidates for additional observable variables and categorized them by the level of their transferability (see table 4.1).

Name	Level	Variable	Type	Domain
task success	performance	k	binary	{0,1}
mission score	performance	s	discrete	$[0,...,max_{items}$
mission time	performance	t	continuous	$\mathbb{R}_{\geq 0}$
required rounds	domain	n_{rnd}	discrete	$[1,...,max_{rounds}$
required hints	domain	n_{hnt}	discrete	[0,...(
location changes	domain	n_{loc}	discrete	[0,...(
dialogues	domain	n_{dia}	discrete	$[min_{dialogues},...($
detours	game	n_{det}	discrete	[0,...(

Table 4.1: Observable variables in LE 2308, as identified by [Aydinbas 2019]

The top section of table 4.1 shows the three variables that were implemented as observable variables for the original CogIUM model by Aydinbas. For this thesis's scope, the goal was to utilize more data from the xAPI statements, which meant introducing new observable variables. Because Aydinbas analyzed LE2308 and this work is focused on the application of CogIUM to SLE, there are slight differences in what variables can be used. Some described variables from the table are not present in SLE since it is a *streamlined* and hence minimal version of Lost Earth. Required hints and required rounds are not available in SLE because the user is not given the possibility to request hints, and the game is not round-based.

After investigating what variables are available in SLE and what informational value they can add to the input data, it was decided to utilize the following two observable variables:

1. **required attempts:** In SLE missions, the user has to answer multiple-choice questions about images or find hidden objects within an image. Thus, the number of attempts the user required until he completed the mission appears to be valuable information about the interaction. Furthermore, *required attempts* as a variable is transferable to all other games that are mission-based.

2. **detours:** SLE requires the user to understand a procedure described in the mission briefing: (1) Choose a sensor to take images, (2) Activate that sensor, (3) View the images taken by the sensor, (4) Answer multiple-choice questions about the images. If the user does not fully understand the instructions, he can be disoriented during the mission and perform actions that are unnecessary for the mission's progress. Those actions are identified as *detours*. Although Aydinbas listed *detours* as a game-specific variable, it is still easy to transfer to other games: An ideal path from mission start to mission end has to be defined, e.g. by letting an expert play the mission and recording the xAPI statements. Detours can then easily be calculated by comparing a user's xAPI statements to those of the expert.

Figure 4.2: Minimal example of an xAPI stream from SLE, visualized by ELAISim [Streicher, Bach, and Roller 2019].

4.3 CogIUM Enhancements and Extensions

To account for the newly added observable variables, the CogIUM model had to be expanded. Additionally, some flaws in the model that were already discovered by [Aydinbas 2019] required remodeling and a revision of the latent variables' prior distributions. Other flaws were only discovered during this thesis and guided new design choices for parts of the model. While the iterative development process is the topic of section 5.2, the following section will cover the concept that guided the development and design of the enhanced CogIUM model.

As described in section 4.2, it was decided to add the observable variables **required attempts** and **detours** to the model. Adding an observable variable to an HBM always entails the design choice for the variable's distribution and how this distribution is parameterized. Since the model is hierarchical, those parameters themselves will also originate from a parameterized distribution. Additionally, some existing structures are remodeled due to identified flaws in the model's ability to explain plausible data. In the following sections, those design choices are presented along with the reasoning that motivated them.

4.3.1 Remodeling of existing structures

In the discussion of his final CogIUM model, Aydinbas establishes that the model works best for one concept and can successfully explain three continuous observed variables. When there is more than one concept, the model struggles to explain the data, and the model's performance decreases with an increased number of concepts. Furthermore, Aydinbas states that this was to be expected since some important variables are not given the flexibility to vary between missions [Aydinbas 2019]. This was also identified as a flaw in the initial analysis at the start of this thesis. Mainly, two latent cognitive variables are concerned: Motivation m_p and prior knowledge ψ_p. Both were modeled as personal variables in Aydinbas' final model, meaning the model could learn an individual manifestation of the variable for each user p. However, for a given user, motivation and prior knowledge could not vary between missions. This does not appear plausible and hampers the model's ability to explain differences in a given user's performance in two equally complex missions. Therefore, it was decided to change those variables to be dependent on both user p and concept c.

Prior knowledge ψ_p was subsequently renamed to prior knowledge ψ_{pc} and will be referred to as such from now on. Semantically, the change means that a user is now assumed to potentially have different levels of domain knowledge for different missions. This appears very plausible, especially if missions cover a variety of knowledge domains.

Accordingly, the same is done for motivation m_p, which will be denoted as motivation m_{pc}.

Thus, a user's assumed motivation is now allowed to vary across missions.

Another issue concerned the modeling of the variable mission time t_pc. During the iterative development process, it became evident that the model's posterior predictive function was not able to represent the observed data. Therefore, several changes were introduced to the modeling of the variable. How those modifications were motivated is covered in detail in section 5.2. In the final model of Aydinbas, mission time t_{pc} was modeled with the following distribution: $t_{pc} \sim \mathcal{N}(t_{min} + \alpha_t \cdot (1 - gcl_{pc}) - \beta_t \cdot \delta_{pc}, \sigma_t)$. The multiplicative factors α_t and β_t are changed from personal variables to personal and conceptual variables to provide the model with more flexibility. Additionally, the term for the distribution's mean value is changed to also include prior knowledge ψ_{pc} as an inversely proportional component. Hence, the final formula for mission time's distribution is given by $t_{pc} \sim \mathcal{N}(t_{min} + \alpha_{tpc} \cdot (1 - gcl_{pc}) - \beta_{tpc} \cdot \delta_{pc} + \rho_{tpc} \cdot (1 - \psi_{pc}), \sigma_t)$. As can be seen in section 5.2, this modification yielded a significant improvement in the model's predictive performance.

4.3.2 Required Attempts as observable variable

The purpose of adding *required attempts* as an observable variable is to feed additional information into the CogIUM model, allowing for a better-informed classification. For the model to explain the new observable variable, two things have to be decided: First, what distribution is assumed for the observable variable. The observable variable is thought of as a probability distribution, and future observed values are assumed to originate from this distribution. Hence, the distribution must be designed in a way that future observed values could plausibly stem from it. Since the model is hierarchical, this does not mean that the future data already has to be known when designing the model. Instead, the observable variable's distribution parameters are variables of the model themselves and can be learned when observing new data. Specific to CogIUM is that the latent variables are ascribed a cognitive meaning.

Accordingly, it has to be determined what latent cognitive variables should influence the observed variable required attempts. For the following section, required attempts will be designated as a_{pc}, which is also the notation in the expanded CogIUM model. In SLE, users are asked to answer multiple-choice questions. If they know the answer, they can obviously pick the right solution at first try, otherwise, they have to guess. If they do not know the answer but can eliminate one possibility, they are still likely to need fewer attempts than if they had no prior knowledge at all. This line of reasoning is transferable to all other knowledge-based games, not only with multiple-choice questions: The more prior knowledge a user has, the fewer attempts he will probably require to complete the task successfully. Therefore it was concluded that the latent cognitive variable prior knowledge, denoted as ψ_{pc} in CogIUM (see

section 2.4), should influence the expected distribution of required attempts a_{pc}. For the distribution itself, a normal distribution was chosen. This is the most common choice for an observable variable since it simply means that the mean value parameterized by the model is regarded to be the most likely observation, and the further observations divert from this expected mean, the less likely they are. The mean value of required attempts' distribution is postulated to have the formula $\mu_a = (1 - \psi_{pc}) \cdot \gamma_c$. Variable ψ_{pc} denotes the prior knowledge of a specific user p for a specific concept c and is assumed to be inversely proportional to the expected mean value μ_a. Because ψ_{pc} is normalized to the range [0,1], it is multiplied by a factor γ_c to obtain the expected mean value for required attempts. γ_c is specific to a concept c but equal for all users across one concept. This design choice was motivated by the differences in mission length and complexity. In a long, complex mission, it is assumed that users without prior knowledge can require a large number of attempts. In contrast, in a short mission, they will find a solution after a limited number of attempts, even without any prior knowledge. To summarize, the following distribution is postulated for required attempts a_{pc}: $a_{pc} \sim \mathcal{N}((1 - \psi_{pc}) \cdot \gamma_c, 0.5)$

4.3.3 Detours as observable variable

As a fifth observable variable, detours d_{pc} was added to the model. In contrast to the variable required attempts a_{pc}, the idea was to capture how direct the user's path to completing the mission is. Several things can cause a detour: If the user is lost or overstrained, he will involuntarily take detours since he does not know the direct path. This matter is reflected in the model through an inversely proportional influence of prior knowledge ψ_{pc} and free working memory capacity δ_{pc}. Another possible explanation for a detour might be an explorative playing style of the user: Even though the user knows the direct path to completion, he might decide to discover other dialogues and sections that the game has to offer. This is modeled by an explorative factor ef_p, which is designed to represent how curious the player's approach to the game is. As for required attempts a_{pc}, a normal distribution was also chosen to model detours d_{pc}. The distribution is parameterized as follows: $d_{pc} \sim \mathcal{N}((1-\psi_{pc}) \cdot md_c + (1-\delta_{pc}) \cdot fb_p + ef_p, 0.5)$. md_c is a concept-specific factor that maps the normalized range of [0,1] for ψ_{pc} to the range of required detours. It is allowed to vary between missions for a similar reason as γ_c in the formula for required attempts' distribution (see 4.3.2): Depending on the complexity of a mission, the level of prior knowledge can influence the number of detours to a varying extent.

A visualization of the final model's structure is presented in figure 4.3. To enable comparison with Aydinbas's final CogIUM model presented in section 2.4, all newly introduced variables are marked by a thick border. Additionally, the layout is consistent with figure 2.7.

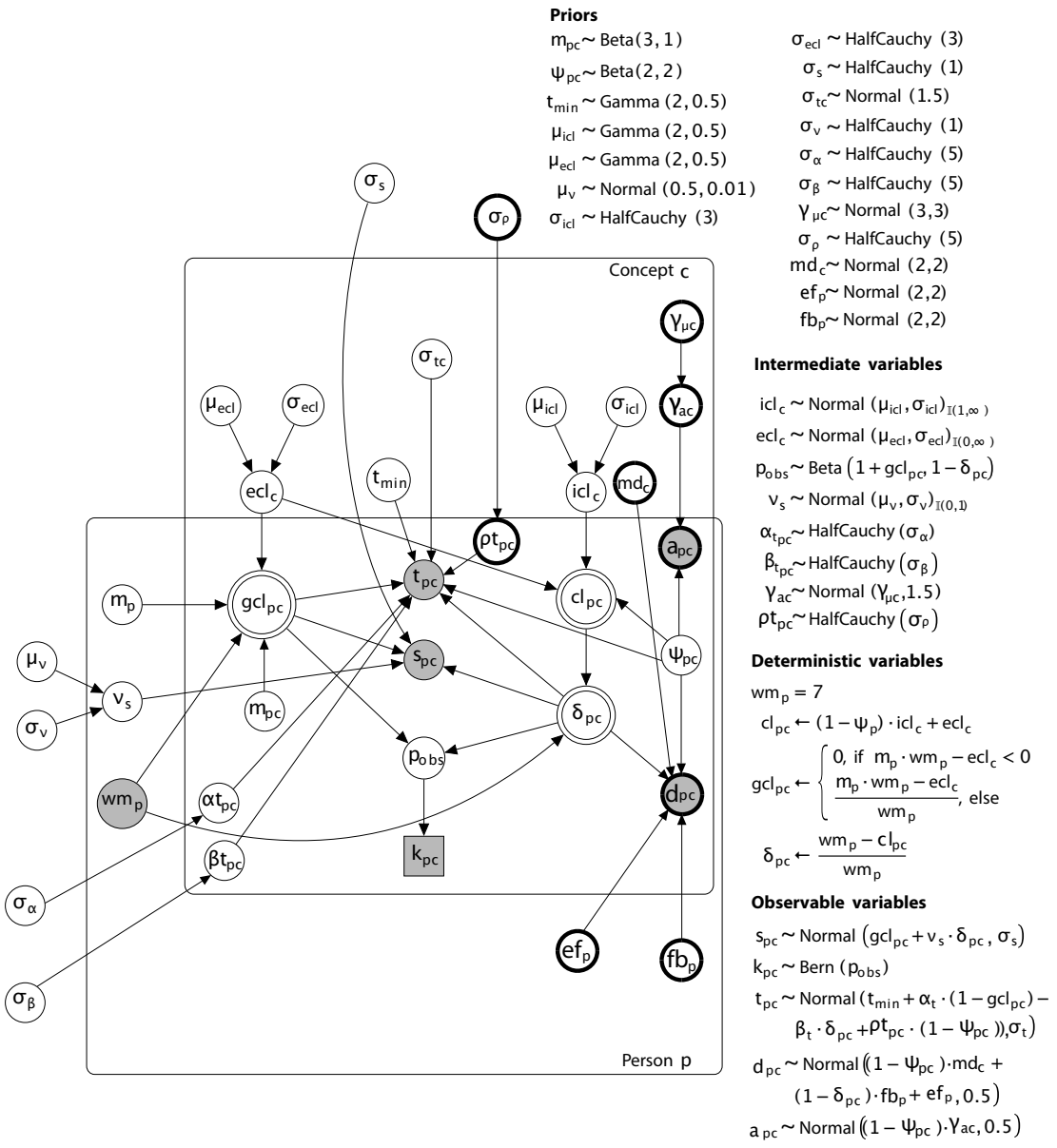


Figure 4.3: Final CogIUM5 model after expansion and restructuring, as it is used within the CogIUMa microservice. All newly introduced variables are marked by a thick border. The layout is consistent with the visualization of Aydinbas' final model in figure 2.7 to facilitate comparison. This is the model used during the user study.

4.4 Score Computation

Score computation bridges the gap between the CogIUM model's output and what the ELAI framework expects as an Adaptivity Response (AR). To reiterate, the output of CogIUM are posterior distributions for all latent cognitive variables. Those latent cognitive variables include motivation m_{pc}, prior knowledge ψ_{pc}, total cognitive load cl_{pc}, intrinsic cognitive load icl_c, extrinsic cognitive load ecl_c and germane cognitive load gcl_{pc}. An explanation of Cognitive Load Theory (CLT) and how its different components are defined can be found in 2.3.

From those posterior distributions, an Adaptivity Response must be computed. Since CogIUM is integrated as a micro-service within the ELAI framework, the AR needs to conform to the ELAI specifications of an AR.

An AR is defined in ELAI as $AR(Context\ C, Time\ t) = (P^C(t), A^C(t), S^C(t))$ [Streicher and Roller 2015]. Accordingly, it comprises three components:

- Performance Score $P^C(t)$: Scalar value in the range [0,1] that encodes the user's overall progress within context C at time t.
- Assistance Level $A^C(t)$: Scalar value in the range [0,1] that represents a user's need for assistance within context C at time t.
- Skill Level $S^C(t)$: Scalar value in the range [0,1] that models the estimated competence level of a user for context C at time t. This value can be independent of $P^C(t)$ and $A^C(t)$, since a beginner might coincidentally score high in a mission, even though he has a low skill level.

Regarding *Skill Level* $S^C(t)$, the computation from the posterior distributions is very straight forward: *Prior knowledge* ψ_{pc} is the cognitive latent variable in CogIUM that represents exactly that: The skill level of a user p for a concept or context c. Additionally, ψ_{pc} is already normalized to the range [0,1]. Hence, the mean value of prior knowledge's posterior distribution can be used as the score for skill level $S^C(t)$.

For *Assistance Level* $A^C(t)$, retrieving a meaningful score from the posterior distributions is not as straightforward. However, some plausible relationships can be postulated: If the user's germane cognitive load is lower than his intrinsic cognitive load, he does not dedicate as many resources to learning as required. Accordingly, he probably needs assistance. A detailed explanation of this scenario can be found in section 2.3. Generally, if the user's motivation is very low, this might also imply a need for assistance. Taking those considerations into account, it was decided to model *Assistance Level* $A^C(t)$ as a weighted sum of the latent cognitive variable's predicted mean values. As part of the user study (see section 6.1), users were asked

for a self-assessment of their need for assistance. Rather than postulating the weights of the weighted sum, it was decided to learn them from the gathered user data: For every user, the latent cognitive variables' predictive mean values were inferred. Additionally, the user's self-assessment regarding their need for assistance was collected. The problem of finding optimal weights for the linear sum to map the predicted mean values to the self-assessments can be formulated as an overdetermined system of linear equations. Optimal weights that minimize the squared distance between the weighted sum and the self-assessments can be found through linear compensation. This is covered in-depth in section 5.3.

As for *Performance Score* $P^C(t)$, it was determined that there is no obvious way to compute a meaningful value from the latent cognitive variable's posterior distributions. Therefore, it was decided to stick with the first two scores presented and potentially leave the computation of the AR's third component to a different micro-service within ELAI.

4.5 ELAI Integration

From the offset, the goal of this thesis was to integrate CogIUMa into the ELAI framework as a micro-service. As an adaptivity framework, ELAI receives adaptivity requests from SLE and sends back an adaptivity response. Internally, it can use several different micro-services for the computation of the adaptivity response. For the explanations in this section, we assume that it uses CogIUMa as the micro-service. Figure 4.4 visualizes the interaction of all components involved.

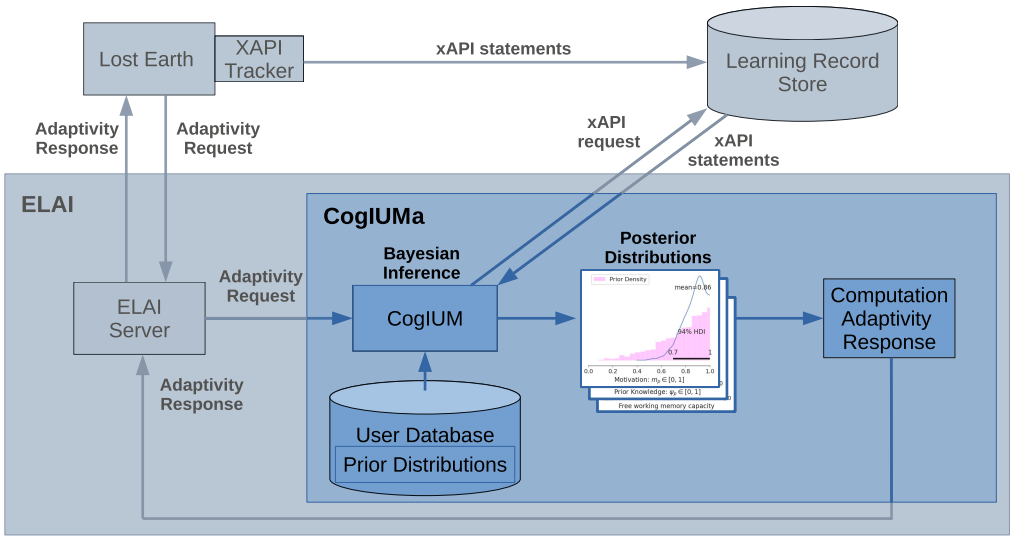


Figure 4.4: Architecture diagram - CogIUMa integration into ELAI adaptivity framework

During user interaction with SLE, the xAPI tracker integrated within SLE sends xAPI statements to the LRS. Adaptivity requests can be sent to ELAI by SLE on various occasions: If functionality to request a hint is added to SLE, such a user request will trigger an adaptivity request. Additionally, when the user is about to start a mission, SLE can request an adaptivity response and adapt the mission accordingly. Once ELAI receives an adaptivity request, it forwards the adaptivity request to CogIUMa. CogIUMa provides a REST interface to receive adaptivity requests, meaning adaptivity requests can be sent as HTTP GET requests. Adaptivity requests are parameterized with a context that includes mission and user id. On receiving the adaptivity request, CogIUMa polls the corresponding xAPI data from the LRS. Subsequently, it performs the inference, which consists of the steps described in the previous sections: (1) Extract features from the xAPI statements (see section 4.2), (2) Perform sampling to approximate posterior distributions, (3) Compute adaptivity response from the posterior distributions (see section 4.4). Next, CogIUMa sends the adaptivity response back to ELAI in the form of an HTTP response. ELAI forwards the adaptivity response to SLE, where adequate manifestation of adaptivity can take place.

5 Implementation

The following chapter will present the implementation process of CogIUMa, based on the concept described in chapter 4. Design choices that were made due to findings during the implementation process will be motivated and described in the following sections.

Basis for all implementation in this thesis was the cogium Python package created by [Aydinbas 2019]. It includes Aydinbas' final CogIUM model and is based on the Python library PyMC3 (see section 2.4). This model was used as a starting point and was then iteratively expanded and enhanced. Throughout the implementation of CogIUMa, the model was validated using two exemplary datasets. In line with the structure of chapter 4, the following sections will describe the implementation of required components, ordered according to the CogIUMa processing chain: First, generation and parsing of xAPI input data is discussed. Subsequently, the enhancement of CogIUM itself is presented. Finally, the adaptivity response's computation is also presented from an implementation perspective.

5.1 Input Data - Generating and Parsing xAPI Statements

On receiving an adaptivity request by the ELAI framework (see section 4.5), CogIUMa polls the corresponding xAPI data from an LRS and uses it as input. Therefore, functionality needed to be implemented to extract all observable variables from the xAPI statements. To reiterate, those observable variables are *mission success*, *mission score*, *mission time*, *required attempts* and *detours*.

In order to implement parsing functionality for xAPI statements, xAPI data was needed for testing. Additionally, xAPI data was later required to validate the CogIUMa processing chain. To quickly obtain xAPI data at scale as input for CogIUMa, the idea was to generate it synthetically. This task is nontrivial since the statements have to be coherent and reflect patterns of xAPI statements that could come out of real user interaction with a serious game. Several approaches to the generation of xAPI data were explored and will now be presented briefly.

DATASIM is an open-source tool developed by Yet Analytics and funded by the Advanced Distributed Learning Initiative at US DoD. The project's objective is to provide a flexible model

that can be customized to a specific use case and then automatically generate xAPI data that is coherent with that usage scenario's specifications. DATASIM is an acronym for Data and Training Analytics Simulated Input Modeler. To allow for this customization, the tool provides several configuration files in JSON format. The main configuration file is the so-called *profile*. A DATASIM profile describes what verbs and objects are available for xAPI statements, defines statement templates and patterns in which the statements are allowed to occur. Since the tool is currently still in its alpha phase, the documentation is not very comprehensive, and the functionality is also not fully available yet. Still, it was possible to create a DATASIM profile that describes vocabulary and basic patterns of statements resembling user interaction data with a serious game like Lost Earth. However, configuring the xAPI statements to be generated exactly as required for the testing of CogIUMa proved to be too time-consuming and tedious. Therefore, this approach was no longer followed, and different options were explored.

To simulate user behaviour and systematically test external adaptivity software, the ElaiSim tool was developed by [Streicher, Bach, and Roller 2019]. It allows the visualization of an xAPI statement stream, defining statement templates and adding them to the statement stream. While this process is less automated than the approach with DATASIM, it is intuitive to use and allows the creation of xAPI statements that strictly conform to the requirements for the input of CogIUMa. Furthermore, the dataset for testing was not required to be very large. Aydinbas states that the number of observations the model needs to make accurate predictions is relatively low. Already with five subjects, predictions improve up to the point where they are spot on [Aydinbas 2019]. Two datasets were generated using ElaiSim, both containing ten users and two missions: The first dataset $\mathcal{D}_1$ features five users from a *high-performing* group and five users from a *low-performing* group. For the sake of simplicity, they will be referred to as high-performing users and low-performing users in the following explanation. High-performing users completed both missions successfully with a score of over 50 percent, low-performing users failed both missions with a score of less than 50 percent. In the second dataset $\mathcal{D}_2$ there are again five high-performing users, identical to the high-performing users in dataset 1. However, instead of low-performing users, dataset $\mathcal{D}_2$ features five users that all succeeded in the first mission but failed in the second mission. The intention behind the design of $\mathcal{D}_2$ was to investigate how well the model handles a user's change in performance across missions.

A third approach that was explored was to generate the xAPI data manually by simply playing SLE. For the user study setup (see section 6.2), an xAPI tracker had to be built into SLE anyway. Therefore, we decided to move its implementation up in time so that it could already be used to generate xAPI data for testing. This allowed for easy generation of xAPI data by simply playing through the missions repeatedly. By doing that with intentional variation in

performance, dataset $\mathcal{D}_3$ was created. It features ten users with varying performance.

With datasets $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$ as testing data, the parsing functionality was implemented. Given the xAPI data, it extracts the observable variables *mission success*, *mission score*, *mission time*, *required attempts* and *detours* for every user and every mission: Mission success and mission score can be directly retrieved from the xAPI statement that is sent on completion of the mission. It contains a boolean for success and a numerical value for the mission score. Mission time can easily be calculated with the timestamps of the *started mission* statement and the *completed mission* statement. Required attempts can be extracted by counting the *failed mission* statements per user and mission. For obtaining the number of detours, some additional effort is required: For both SLE missions, it was defined how many necessary statements the shortest path from mission start to mission completion includes. To retrieve the number of detours, statements between mission start and mission completion are counted, and the length of the shortest path is subtracted. While the transfer of this calculation to other games does require some effort, it is still manageable. A shortest path for each mission can be obtained by letting an expert play all missions and recording the xAPI data.

The xAPI parser is included as a Python script within the cogium package.

5.2 CogIUMa - Experimental Iterative Development

After generating sufficient user interaction data as input for testing, the iterative development of CogIUMa started. Aydinbas' final CogIUM model formed the starting point of the development. To reiterate, this model accounts for the three observable variables *mission success*, *mission score* and *mission time*. As a first measure, the posterior predictive functions for those three variables are analyzed using the datasets $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$ as input.

Looking at the posterior predictive functions for the three observable variables regarding dataset $\mathcal{D}_1$, the model seems to be able to reproduce the main trends in the observed data. However, the model's posterior predictive functions regarding dataset $\mathcal{D}_2$ differ significantly from the observed data, especially for the variable mission time. Figure 5.1 showcases the described problem. The estimated mean value of mission time is roughly the same for all users, causing the spike of the posterior predictive mean function in the plot. Observations and predicted mean values are both shown as a kernel density estimate, observations in black and predicted mean in blue. The green line represents the average prediction over all users for this mission. It is evident that there is is a large discrepancy between observations (black) and posterior predictive mean function (blue). This does not come as a surprise, since the discovered flaws described in section 4 already indicated that the model will struggle to explain a user's variance in performance. Intuitive explanations for individual performance variation

could either be the change of a users motivation over time, or different levels of prior domain knowledge regarding the different missions.

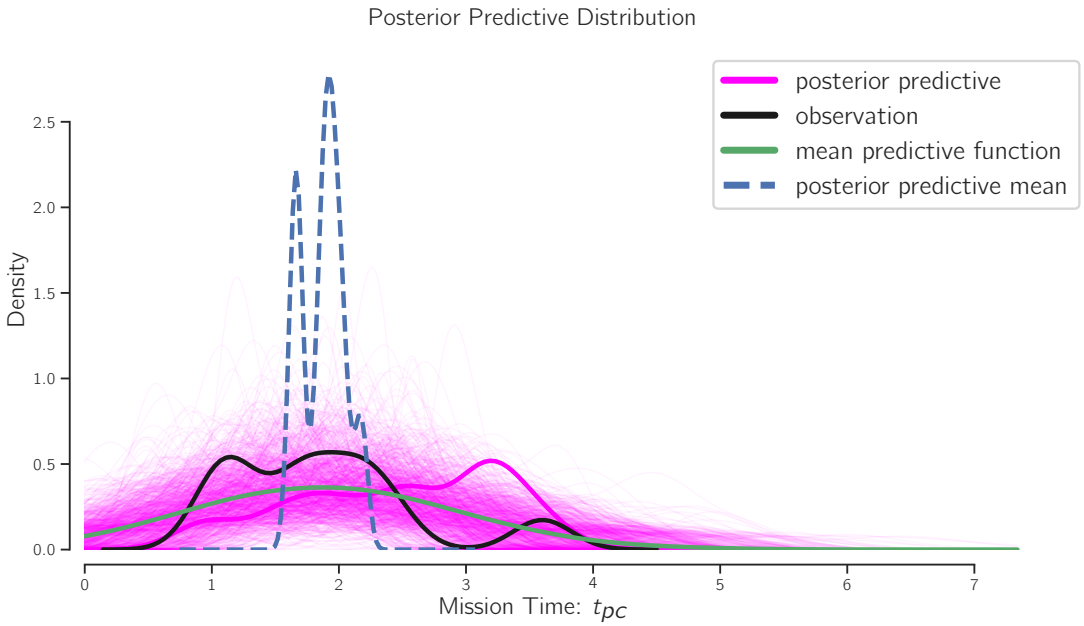


Figure 5.1: Posterior predictive function of mission time t_{pc} for mission 1 in dataset $\mathcal{D}_2$

Accordingly, the development approach's starting point were the flaws identified in section 4.3: The latent cognitive variables *motivation* m_p and *prior knowledge* ψ_p are both modeled as personal variables and are thus not allowed to change across missions. As described in section 4.3, it was decided to remodel those variables to be dependent on both user p and mission c. Therefore, they are denoted *motivation* m_{pc} and *prior knowledge* ψ_{pc} from now on.

In Aydinbas' final CogIUM model, *mission time* t_{pc} was modeled by the following distribution: $t_{pc} \sim \mathcal{N}(t_{min} + \alpha_t \cdot (1 - gcl_{pc}) - \beta \cdot \delta_{pc}, \sigma_t)$. Expanding this formula, it was decided that prior knowledge ψ_{pc} should also have an influence on mission time t_{pc}: The higher the level of prior knowledge, the less time the user is expected to take for completing the mission. To provide the model with even more flexibility for the modeling of mission time, the multiplicative factors α_t and β_t were expanded from personal variables to personal and conceptual variable α_{tpc} and β_{tpc}. Furthermore, less informative prior distributions were assumed for α_{tpc} and β_{tpc} to allow for more deviation in mission times between different users. The final formula for mission time's distribution is given by $t_{pc} \sim \mathcal{N}(t_{min} + \alpha_{tpc} \cdot (1 - gcl_{pc}) - \beta_{tpc} \cdot \delta_{pc} + \rho_{tpc} \cdot (1 - \psi_{pc}), \sigma_t)$.

After this extensive remodeling process, the model was able to capture the observed data's structure significantly better. While the accuracy for the other observable variables remained

the same, the prediction's accuracy for mission time of mission 1 in dataset $\mathcal{D}_2$ improved a lot. This can be seen by comparing figure 5.2 to figure 5.1: Figure 5.2 shows the posterior predictive function for mission time after the extensive remodeling process described above. In both figures, the observed data are the values of mission time for mission 1 in dataset $\mathcal{D}_2$, only the y-axis is scaled differently. While the prediction is not perfectly aligned with the observation, the remodeling was still deemed successful due to the large improvement.

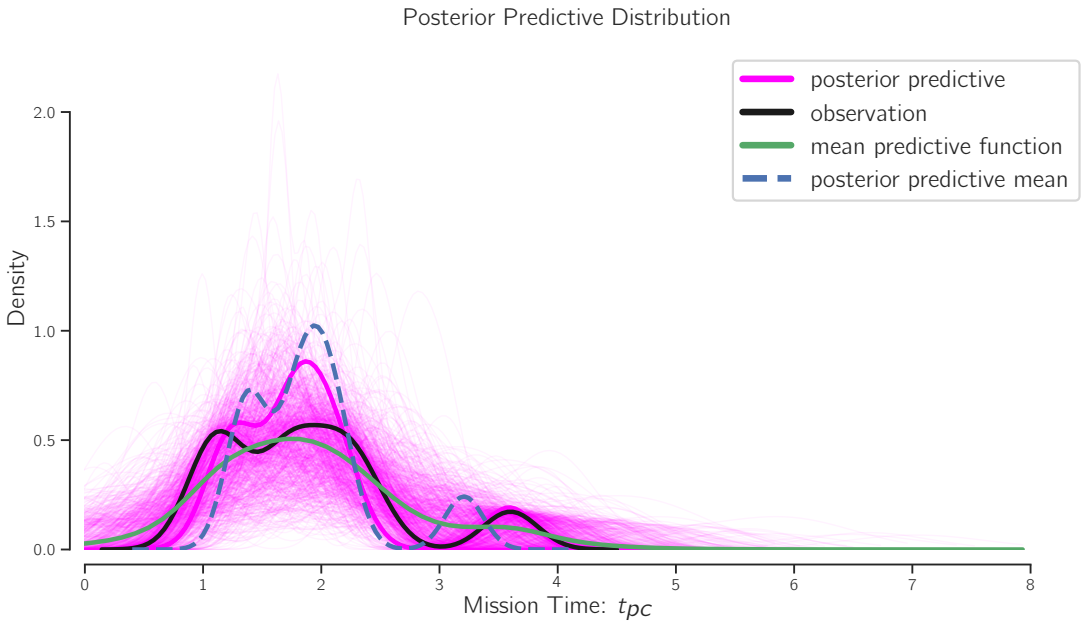


Figure 5.2: Posterior predictive function of mission time t_{pc} for mission 1 in dataset $\mathcal{D}_2$, after extensive remodeling.

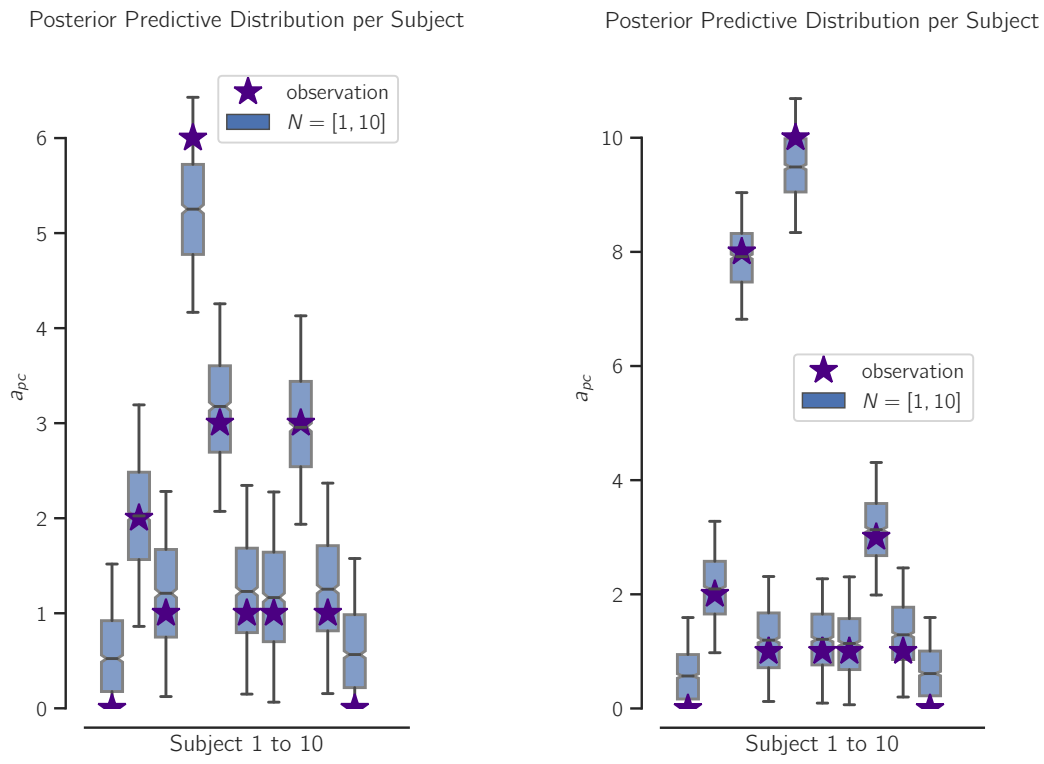
Furthermore, the standard deviation of mission score s_{pc} is represented in the original CogIUM model as a global variable. Thus, the score is assumed to vary the same for all subjects and all missions. This assumption seems implausible since some missions might suit all users roughly the same, while other missions are easy for some users but difficult for others. It is also reasonable to assume that the performance throughout several missions might vary more for some users than for other users. Accordingly, the standard deviation σ_s of the mission score was also remodeled to be dependent on both user p and mission c.

Once the described remodeling of existing structures finished, the model was able to reproduce all three observable variables (*mission success*, *mission score* and *mission time*) reasonably well for all three datasets $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$.

Subsequently, the implementation of the two newly introduced observable variables began:

Required attempts and *detours*.

As motivated in section 4.3, required attempts a_{pc} is modeled by the following distribution: $a_{pc} \sim \mathcal{N}((1 - \psi_{pc}) \cdot \gamma_c, 0.5)$. Initially, factor γ_c was designed as a global variable independent of a specific concept c. However, in a first experimental sampling run it became evident that a global multiplicative factor γ is not adequate: For the experimental run, the SLE user interaction data from dataset $\mathcal{D}_3$ was used as input. In $\mathcal{D}_3$, the values of required attempts for mission 1 are in a lower range than those for mission 2. This is likely caused by the fact that mission 2 is longer and consists of more subtasks. Thus, the multiplicative factor was modeled to be dependent on concept c and is therefore denoted γ_c. Overall, the model's predictive performance for required attempts regarding dataset $\mathcal{D}_3$ is reasonably accurate, as can be seen in figure 5.3: Observations are marked by the dark blue stars. The blue box represents the area in which 50 percent of the posterior predictive samples lie, and the horizontal line marks the sample median. Apart from some minor deviations, the predictions are all very close to the observation. Even the outliers in mission two are represented correctly.



(a) mission 1

(b) mission 2

Figure 5.3: Box plots of posterior predictive samples for required attempts a_{pc} in dataset $\mathcal{D}_3$.

After successfully implementing *required attempts*, the next step was to expand the model further to also account for *detours* d_{pc}. As motivated in section 4.3, detours d_{pc} is modeled by a normal distribution with a mean value influenced inversely proportional by both *prior knowledge* ψ_{pc} and *free working memory capacity* δ_{pc}. The distribution is defined by the following formula: $d_{pc} \sim \mathcal{N}((1-\psi_{pc}) \cdot md_c + (1-\delta_{pc}) \cdot fb_p + ef_p, 0.5)$. The predictive accuracy for detours is not as good as for required attempts, but the observed data's main features are still captured. To avoid overfitting, no new latent variables were introduced only to improve predictive accuracy for the observed data.

This concluded the iterative development of CogIUM. The final model is called CogIUM$_5$ because it accounts for five observable variables, and is used for the CogIUM inference within the CogIUMa microservice. Along with all model iterations by Aydinbas, it is included in the cogium Python package.

5.3 Score Computation

Score computation is required to map the latent cognitive variables' posterior distributions to an adaptivity response. As explained in section 4.4, the adaptivity response's format is defined by the ELAI framework: It contains the three components *Performance Score* $P^C(t)$, *Assistance Level* $A^C(t)$ and *Skill Level* $S^C(t)$. In the analysis performed in section 4.4, it was decided to only model the second and third component *Assistance Level* $A^C(t)$ and *Skill Level* $S^C(t)$.

Skill Level $S^C(t)$ is represented within CogIUMa as prior knowledge ψ_{pc}'s posterior predictive mean. Therefore, it can be obtained from the sampling trace summary and was straightforward to implement.

For *Assistance Level* $A^C(t)$, the computation is not as trivial. As described in section 4.4, it was decided to model a user's assistance level as the weighted linear sum of the latent cognitive variables' posterior distributions' mean values. Since *motivation* and *germane cognitive load* vary only marginally between users, it was decided to exclude them from the weighted sum. For the computation of a user i's assistance level, the four cognitive variables *cognitive load* cl_i, *intrinsic cognitive load* icl_i, *extrinsic cognitive load* ecl_i, and *prior knowledge* ψ_i are used. Given inferred mean values for the cognitive variables and ground truth assistance levels al_i, the weights $\alpha_1,\ldots,\alpha_5$ can be determined by minimizing the residual of an overdetermined equation system:

$$\arg\min_{\alpha_1,\ldots,\alpha_5} \left\| \left(\begin{bmatrix} cl_1 & icl_1 & ecl_1 & \psi_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ cl_n & icl_n & ecl_n & \psi_n & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_5 \end{bmatrix} \right) - \begin{bmatrix} al_1 \\ al_2 \\ \vdots \\ al_m \end{bmatrix} \right\|$$

Optimal weights $\alpha = (\alpha_1,\ldots,\alpha_5)$ can be found by solving the normal equation: $A^T \cdot A \cdot \alpha = A^T \cdot al$, where A denotes the matrix of cognitive variables' mean values and al denotes the vector of assistance levels. After finding optimal $\alpha_1,\ldots,\alpha_5$ that minimize the term above, we obtain the following formula for our sought-after assistance score X:

$$X_i = \alpha_1 \cdot cl_i + \alpha_2 \cdot icl_i + \alpha_3 \cdot ecl_i + \alpha_4 \cdot \psi_i + \alpha_5$$

Since X_i can also take on values less than 0 or greater than 1, it has to be clipped to the range of [0,1] to obtain $A^C(t)$:

$$A^C(t) = \begin{cases} 0, & \text{if } X_i < 0 \\ X_i, & \text{if } 0 \leq X_i \leq 1 \\ 1, & \text{else} \end{cases}$$

From the user study, we get 44 linear equations: Each of the 22 users played two missions. For each of those missions, CogIUM calculated posterior mean values for the cognitive variables *cognitive load cl*, *intrinsic cognitive load icl*, *extrinsic cognitive load ecl*, and *prior knowledge* ψ. Additionally, the user's reported a perceived need for assistance in the self-assessment survey for each mission. This gives 44 equations, where the cognitive variables have to be mapped to the self-assessed assistance level with a weighted sum. Accordingly, the matrix in the term above has 44 rows. Performing linear compensation to retrieve optimal values $\alpha_1,\ldots,\alpha_5$ yields the following results: $\alpha_1 = 0.45$, $\alpha_2 = 0.93$, $\alpha_3 = -0.93$, $\alpha_4 = -0.96$, and $\alpha_5 = 0.35$.

Therefore, the following formula is postulated for the assistance level $A^C(t)$ of user i:

$$X_i = 0.45 \cdot cl_i + 0.93 \cdot icl_i - 0.93 \cdot ecl_i - 0.96 \cdot \psi_i + 0.35$$

$$A^C(t) = \begin{cases} 0, & \text{if } X_i < 0 \\ X_i, & \text{if } 0 \leq X_i \leq 1 \\ 1, & \text{else} \end{cases}$$

While the values $\alpha_1,\ldots,\alpha_5$ have been calculated mathematically to minimize the residual of the overdetermined equation system, they can also be interpreted with regards to their

semantic meaning: α_2 is the largest positive α value and is the weight that corresponds to intrinsic cognitive load *icl*. Accordingly, intrinsic cognitive load has a positive linear influence on the assistance level: The higher the intrinsic complexity of a mission, the higher the user's need for assistance. α_1 is the second-highest positive α value and determines the influence of total cognitive load *cl* on the user's assistance level. Again, the relationship is positive and linear, meaning that higher values for a user's total cognitive load lead to a higher assistance level. Prior knowledge ψ has the largest negative linear influence on the assistance level: The corresponding weight, α_4, was calculated to be -0.96. Accordingly, a higher level of prior knowledge results in a lower predicted assistance level. Extrinsic cognitive load *ecl* also has a negative linear on the assistance level, it is weighted with $\alpha_3 = -0.93$.

Although the weights were results of the above term's mathematical minimization, their meaning is still largely in line with the expected semantic relationships: Users with high prior knowledge will likely not need any assistance. When a task is highly complex (high *icl*), users are more likely to need assistance. A highly complex task often causes users to experience a high total cognitive load. Therefore, it is to be expected that a high cognitive load also leads to a higher assistance level. Since the calculated weights α_1,...,α_5 were derived from the user study data, additional data is required to validate them.

5.4 CogIUMa as a micro-service - REST API

To integrate CogIUMa within the ELAI framework, a RESTful webservice was built using the Python library FastAPI. FastAPI is a modern, high-performance framework that allows to quickly build REST APIs [Tiangolo 2021]. Figure 5.4 shows an overview of possible requests that can be sent to CogIUMa.

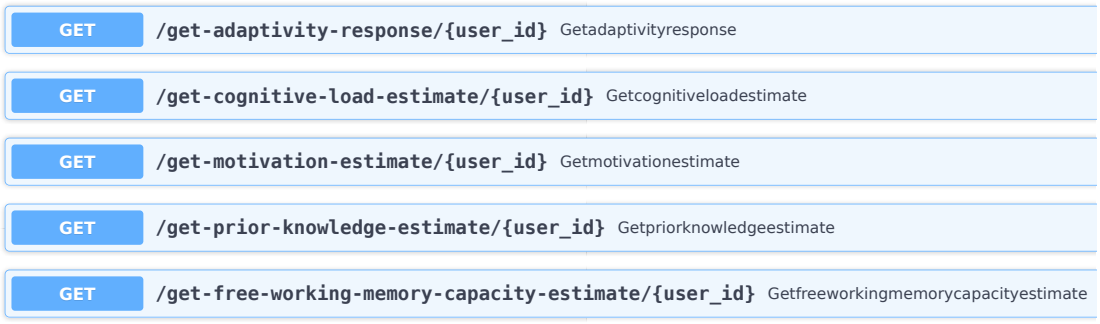


Figure 5.4: REST API for CogIUMa microservice

The most important request for CogIUMa's functionality as a microservice is the *get-adaptivity-response* request. It triggers the whole CogIUMa pipeline, as described in sections 4.2, 4.3 and 4.4: Given the user id, CogIUMa polls the corresponding xAPI statements from the LRS. Those statements are then parsed and used as input for the inference. Eventually, the computed scores are returned as an adaptivity response.

When a game sends an adaptivity request to ELAI and thus to CogIUM, waiting over a minute until the CogIUM sampling process finishes is not an option. Therefore, a concept was introduced to distinguish between *offline* and *online* inference. For offline inference, the normal CogIUM sampling is performed since waiting a couple of minutes for the result is feasible. For the online inference, a new concept is introduced: The 5-dimensional CogIUM input vector per user and mission consists of *mission success*, *mission score*, *mission time*, *required attempts* and *detours*. To enable fast inference, a large number of input vectors is taken from this 5-dimensional vector space. For each input-vector, the CogIUM sampling is performed, and thus, inference values for those inputs are *precomputed*. Those values are all stored in a *precomputation database*. Performing this computation for roughly 6000 input vectors took about three days on a modern laptop's CPU. However, this only has to be computed once. For online inference, the xAPI statements are parsed the same as for offline inference to obtain the input vector. Next, instead of performing CogIUM sampling, the *nearest neighbour* in the precomputation database is searched for: Since all input vectors are 5-dimensional vectors with real-valued components, the euclidian distance can be used as a metric for distance. After looking up the nearest neighbour, CogIUMa returns the nearest neighbor's precomputed adaptivity response. Obviously, this online inference becomes more accurate the larger the precomputation database is. The inference mode can be passed to CogIUMa as a query parameter of the adaptivity response request, default mode is *online*.

6 Evaluation - User Study

After the iterative development of the CogIUM model had finished, the coherent next step was to find a way to validate the model. Since no real-world ground truth data existed for the metrics that the model tries to infer, a user study appeared to be the logical choice. Supporting this approach, [Aydinbas 2019] stated in his master's thesis that all of his results "must be examined by a detailed model evaluation in the form of a user study still to be carried out".

The main idea for the user study can be described as such: Have users interact with a serious game and collect all their click data. After their interaction, ask the users about their self-assessment regarding the latent cognitive variables cognitive load, motivation, and prior knowledge. Let the CogIUM model compute its inference with the collected click data as input and compare the inference output to the users' self-assessments.

How the user study and the questionnaire were designed, what the hypotheses were, and what results the user study yielded will be discussed in the following sections.

6.1 Concept and Hypotheses

In order to carry out the user study, a serious game that users could interact with was required. It had to be web-based since the current situation with Coronavirus did not allow for a user study to be carried out in person. Additionally, the choice to make the whole user study online came with the benefit of potentially more users. Streamlined Lost Earth (SLE) was chosen as a minimal serious game prototype that runs in a browser. It consists of two playable missions that are both relatively small. To collect data on the cognitive latent variables, the users are asked to fill out a questionnaire after each of the missions.

The main latent cognitive variables that the CogIUM model tries to infer are *cognitive load*, *motivation*, and *prior knowledge*. Without being able to directly measure those variables, the only feasible way to gather real-world data on those variables in a realistic setting is through a self-assessment of the users. Therefore, a questionnaire was built with the goal to receive a realistic self-assessment of the users for the mentioned variables.

For assessing cognitive load in a questionnaire, the gold standard is the NASA Task Load Index (NASA TLX) [Hart and Staveland 1988]. It consists of two separate parts, as was explained

in section 3.1.2: In the first part, the users are asked to rate several aspects that can contribute to cognitive load on a Likert scale. In the second part, the users compare those aspects pair-wise, each time selecting which aspect had the greater contribution to their cognitive load. Those pair-wise comparisons determine the weight with which the Likert scales from the first part influence the total cognitive load computation. In addition to the NASA TLX, the questionnaire was also designed to include a question that directly asks the users to rate their cognitive load on a 7 point Likert scale. This was done to examine the difference between the results of the NASA TLX and the direct question about cognitive load. However, the NASA TLX was chosen to be the primary metric for cognitive load since it is assumed to be more accurate than straight-up asking the users about their cognitive load.

Regarding the variables *motivation* and *prior knowledge*, the questionnaire directly asks the user for a rating on a 7-point Likert scale. This design choice was made for two reasons: For both those variables, there is no gold standard to assess them through a questionnaire like the NASA TLX for cognitive load. Furthermore, it was assumed that motivation and prior knowledge are more tangible than cognitive load, and thus users might find it easier to rate those variables directly.

Another point of interest was how the users would react to the model's inferences about them. More precisely, as how accurate would the users rate the model's inferences if they saw them visualized in an understandable way? To examine this question, it was decided to visualize the inferred mean values for the latent cognitive variables *motivation*, *prior knowledge*, *cognitive load*, and *free working memory capacity*, which is the inverse of *cognitive load*. This was done separately for mission one and mission two. An example of this visualization can be seen in figure 6.1. After seeing their inference visualization, the users were asked to rate the results on a 5-point Likert scale with 1 being "very accurate" and 5 being "very inaccurate". To ensure that users did not rate the results as very accurate only to support my user study and my thesis, it was decided to split the users into two groups: The experimental group would see their real inference data visualization. The second group was designed as a control group. Users in the control group would see the visualization of randomly sampled values. To make sure the values were somewhat plausible, they were sampled from a Normal distribution with a mean value of 0.5. Additionally, it was made sure that the random sample values did not coincidentally match the actual inference values. If the difference between them was less than 0.1, a new sample was drawn from the distribution.

Preceding the user study's execution, two hypotheses were postulated:

Hypothesis (1): The inferred values of the CogIUM model and values from the users' self-assessment for the latent cognitive variables *cognitive load*, *motivation*, and *prior knowledge* correlate.

Hypothesis (2): Users from the experimental group rate their inference results as more accurate than users from the control group.

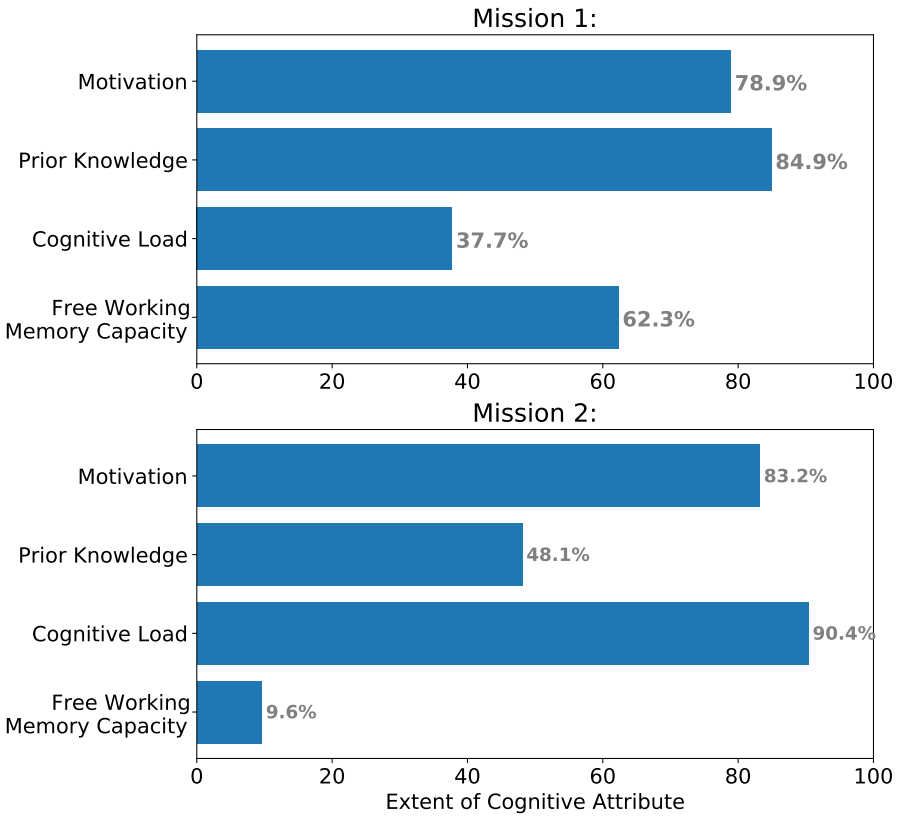


Figure 6.1: Visualization of the CogIUM inference. Users received such a bar chart after completing the user study and had to rate its accuracy on a 5-point Likert scale.

6.2 Planning and Setup

When planning the user study's execution, it quickly became apparent that it had to be carried out as a guided study for several reasons: The user study design required the user to switch between one tab with the questionnaire and another tab with SLE multiple times. Furthermore, the game proved to be very counterintuitive in initial trial runs, leading to confusion of the user. It was decided that it could prove helpful to have recordings of the users while they participate, to identify problems in the user study.

Since the expected duration of each study run was around 30 minutes, it was evident that only a very limited amount of users could realistically participate in the study. Because the users were split in two, the goal was postulated to reach $n = 20$ participants. This would mean that each group consists of at least $\frac{n}{2} = 10$ users.

The recruitment of users mainly consisted of asking my friends and family to participate. Users were aged 13 to 52 and came from various academic and non-academic backgrounds.

Before the start of the user study, the questionnaire had to be set up, and SLE had to be built as a Web-GL build and deployed on a server.

For the questionnaire, the survey tool LimeSurvey was chosen. It allows for easy creation of questionnaires, managing participants, and exporting the answers. The NASA TLX and the other custom questions described in section 6.1 could easily be implemented.

The preparation of SLE for the user study proved more challenging than expected. There were two main problems that needed to be solved before the user study could start.

Firstly, an xAPI tracker had to be built into SLE so that every click by the user would trigger the transmission of an xAPI statement to an LRS. This required extensive studying of the existing game's code to figure out where which button click is handled and to send out an according statement. Since in Unity, buttons can be assigned methods of several scripts via the GUI, this can be quite time-consuming and confusing. Additionally, this led to two errors that were hard to localize and solve: When playing the game in the preview tab of the Unity editor, everything worked fine and xAPI statements were sent to the LRS correctly. However, when the game was built as a Web-GL build and run in the browser, the game worked fine, but no statements were sent to the LRS. After extensive trial and error, it was concluded that the outgoing statements had to be blocked by Unity internally. The Unity manual says that "Due to security implications, JavaScript code does not have direct access to IP Sockets to implement network connectivity. As a result, the .NET networking classes [...] are non-functional in Web-GL" [Unity 2020]. When dealing with xAPI statements, the standard practice is to use the TinCan library for functionality like starting authenticated communication with an LRS to send xAPI statements. However, since the TinCan library internally uses HTTP Post requests

to send the statements, those requests were blocked internally by the Unity Web-GL build. After figuring this out, the logical next step was to try and use the TinCan library to construct and return the proper HTTP requests. From this, another problem arose: TinCan internally utilizes another library called NewtonSoft in its method to construct an HTTP request. This library is not compatible with Web-GL and led to a runtime error when playing the Web-GL build in a browser. As a workaround, the HTTP requests had to be manually constructed to match the requests that the TinCan library would have sent, to then send them using the UnityWebRequest class. This process was very time-consuming.

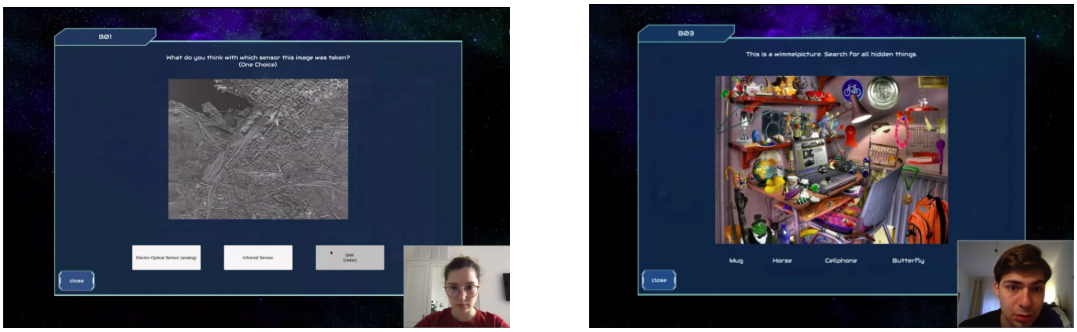
The second aspect of the game that had to be adapted was the game's functionality. In the first mission, an in-game assistant called LISA provided a click-by-click walk-through of the mission. Thus, there was no challenge for the user. This assistant had to be disabled for some situations to create a more challenging task. Additionally, the GUI included several buttons without functionality that were removed to avoid confusion.

In conclusion, adapting the game SLE for the user study turned out to be far more challenging and time-consuming than expected.

6.3 Execution

Once the planning and setup process was complete, the user study's execution could finally begin.

The procedure for a study run was as follows: At a previously appointed time, the user received an invitation e-mail. This e-mail contained a link to a video call and a link to the survey. Once they entered the video call, they were instructed on how to proceed. The video calls during the study run were recorded. The recordings show SLE with the mouse cursor movement of the user and a video feed of the user's webcam as shown in figure 6.2. At the start of the recording, the user was informed about the recording and its usage and was asked for his consent. Subsequently, he received a brief introduction to the user study's topic and the thesis in general. Next, the user was asked to start the game and play the first mission. Once he completed the mission, he was instructed to fill out the self-assessment questionnaire concerning mission one. After completing the questionnaire, the user was asked to play mission two and then fill out the same questionnaire again, this time with regard to mission two. This concluded the guided part of the study run. About 30 minutes after the study run, the user received an e-mail that included the visualization of his inference results. He was asked to rate the results on a 5-point Likert scale from 1 to 5, 1 being "very accurate" and 5 being "very inaccurate".



(a) (b)

Figure 6.2: Screenshots from the user study.

A study run's average duration was around 25 minutes. Overall, conducting the user study online and guided through a simultaneous video call proved very successful. Technical difficulties could be resolved via the video call, and the users could be supported when they were confused by the game or the self-assessment survey. Additionally, it was interesting to see and hear users interact with SLE: One common pattern was that most users struggled with the last subtask, the hidden object challenge. In particular, a majority of users found 3 out of 4 objects relatively fast and then struggled to find the fourth object, a yellow wooden horse. The hidden object challenge and the horse are displayed in the appendix in figure 1.

6.4 Results and Discussion

After the user study's execution finished, the collected data was analyzed with regard to the two hypotheses postulated prior to the execution (see section 6.1). In this section, both hypotheses are evaluated, and possible explanations for the findings are discussed.

To reiterate, the first hypothesis was that CogIUM's inferred values for its latent cognitive variables and the corresponding self-assessments by the users correlate. Thus, the hypothesis has to be evaluated distinctly for the three cognitive variables that CogIUM tries to predict: *Cognitive load*, *motivation*, and *prior knowledge*.

The range of values for cognitive load is displayed in figure 6.3. For missions one and two, the range of users' self-assessments (orange box) and the range of CogIUM's inferences (blue box) are visualized by separate boxes. In the first mission, CogIUM's cognitive load inferences, on average, are higher than the users' self-assessments. However, there is still a weak to moderate correlation between inferences and self-assessments: The Pearson correlation coefficient, also referred to as Pearson's r-value, is 0.382. However, the correlation is not characterized

as significant because Pearson's p-value is above the commonly used significance level of $\alpha = 0.05$. Pearson's p-value describes the probability that assuming the hypothesis is not true (meaning the null hypothesis is true), the observed data or even more strongly correlated data could have been produced on coincidence. In our scenario, this can be described as the following probability: Assuming there is no relation between CogIUM's inferences and the users' self-assessments, what is the probability to still observe data with a correlation coefficient of 0.382 or higher. Therefore, the p-value is a valuable metric for assessing a correlation's meaningfulness.

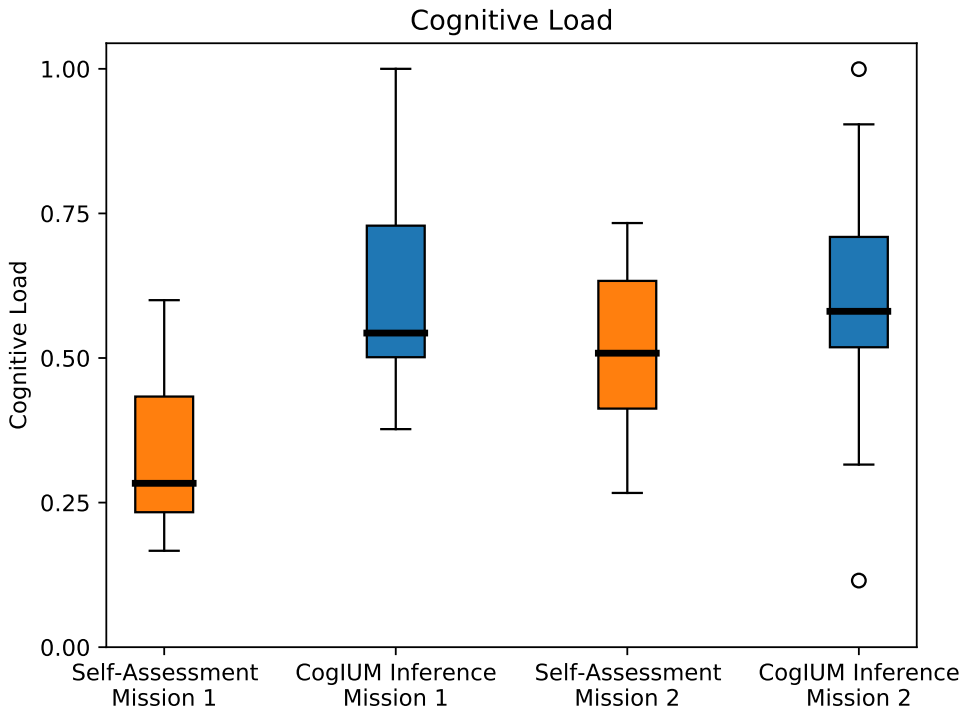


Figure 6.3: Self-Assessment and CogIUM inference for cognitive load in mission one and mission two. Pearson correlation coefficient is 0.382 for mission one and 0.516 for mission two.

For the second mission, the values of CogIUM's cognitive load inferences and the users' self-assessments lie roughly in the same range. The Pearson correlation coefficient is 0.516, and the p-value is below $\alpha = 0.05$. Accordingly, the correlation is characterized as a significant moderate to strong correlation.

Those results are very promising and confirm CogIUM's capability to infer a user's cognitive load solely from his click data. One possible explanation for the weaker correlation in mission

one is a *cold-start problem*: Users do not know what to expect and are unfamiliar with the game's interface. During mission one, several participants reported being rather irritated by the game's counterintuitive interface than being cognitively challenged. CogIUM might have falsely classified this irritation as cognitive load, causing the inference values to lay in a higher range than the users' self-assessments. The stronger correlation and similar data range in mission two support this theory.

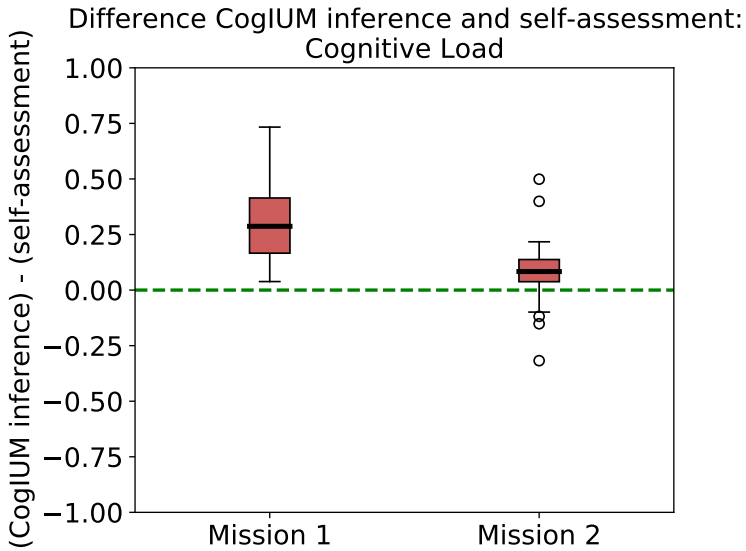


Figure 6.4: Differences between prediction and self-assessment for cognitive load. Values greater than 0 mean the prediction was higher than the self-assessment, values less than 0 mean the opposite.

While figure 6.3 only shows the *range* of the inference and self-assessment data, figure 6.4 visualizes CogIUM's predictive performance for cognitive load in mission one and two. For each user, his self-assessed value is subtracted from CogIUM's inferences. Those differences are summarized for each mission in the boxplot. The dotted green line at level 0 marks the sweet spot: If the difference between inference and self-assessment is 0, they are equal. Values greater than 0 mean the inference is higher than the self-assessment; values less than 0 mean the inference is lower than the self-assessment. Figure 6.4 clearly showcases the improved prediction accuracy in mission two compared to mission one: Values are much closer to 0, and the standard deviation is much smaller. A side-by-side comparison of the individual inference and self-assessment values for each user is displayed in figure 6.5 for mission one and in figure 6.6 for mission two.

To evaluate hypothesis (1) with regards to the other two inferred cognitive variables *motiva-*

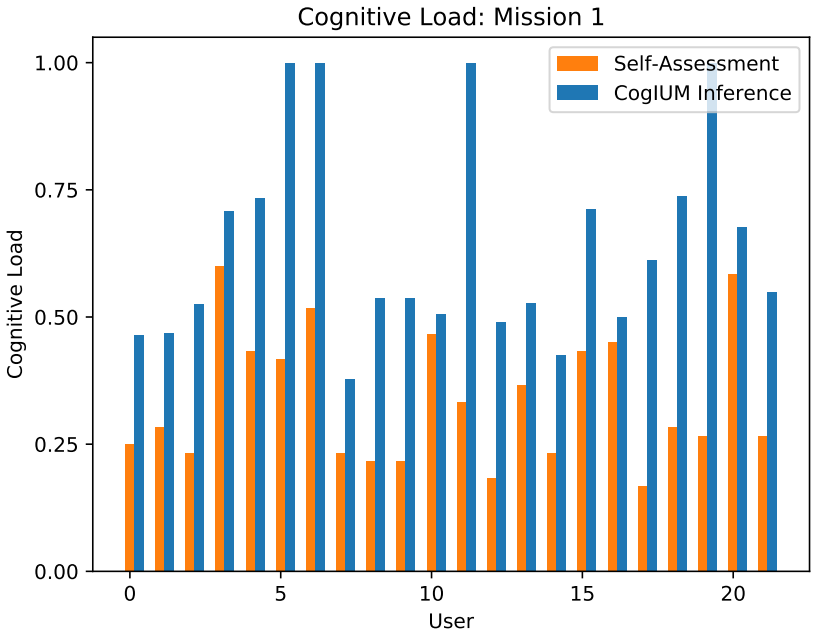


Figure 6.5: Self-Assessment and CogIUM Inference for cognitive load in mission one. Pearson correlation coefficient is 0.382.

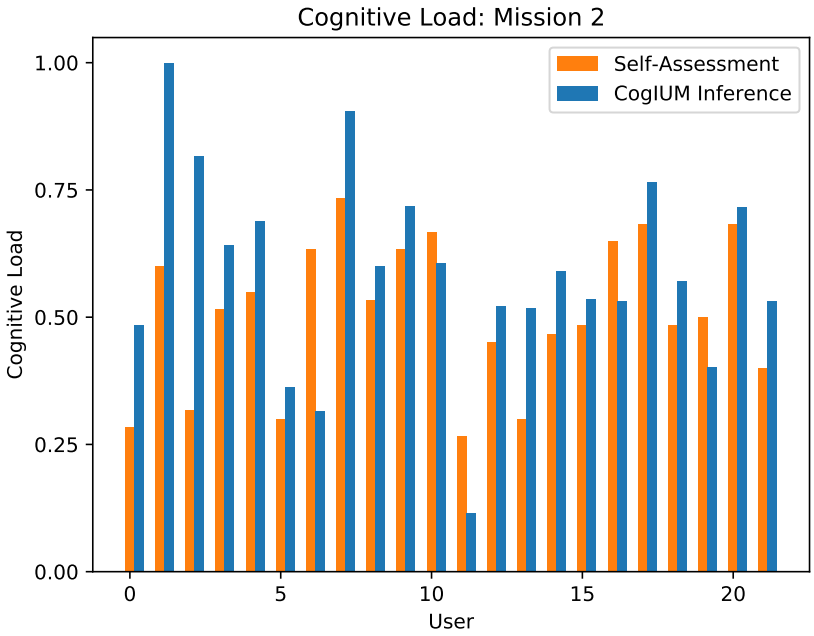


Figure 6.6: Self-Assessment and CogIUM Inference for cognitive load in mission two. Pearson correlation coefficient is 0.516.

tion and *prior knowledge*, I will follow the same approach as for cognitive load.

When analyzing the data for motivation, it is evident that CogIUM inferred almost the same level of motivation for every user (see figure 6.7): All inference values lie between 0.75 and 0.8. Accordingly, the boxes for CogIUM's inference values are so narrow that they are barely visible. While most users' self-assessments for motivation also lie in a small range, there is still significantly more variation than in the inference values: The mean 50% of self-assessments lie between 0.7 and 0.8. However, some users rated their motivation as low as 0.2.

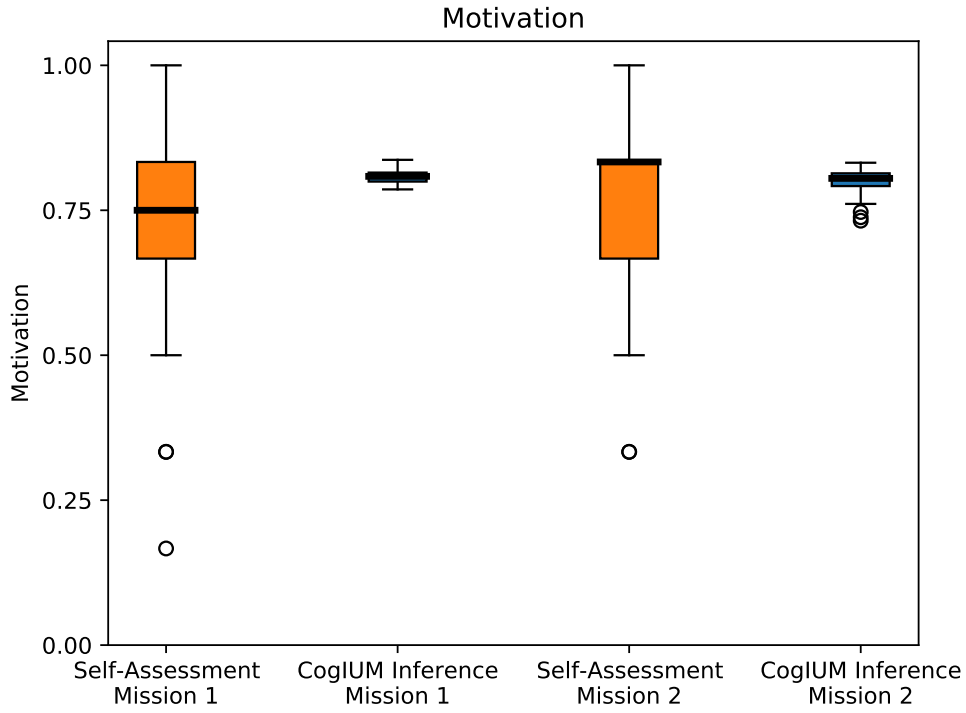


Figure 6.7: Self-Assessment and CogIUM inference for motivation in mission one and mission two. Pearson correlation coefficient is 0.132 for mission one and −0.143 for mission two.

Similar to the evaluation for cognitive load, figure 6.8 displays CogIUM's posterior predictive performance for motivation. Because the data ranges for inferences and self-assessments are both small and centered around similar values, the prediction error is relatively low: For both missions, the differences between inference and self-assessment are centered around 0, and the lower and upper quartiles are also close to 0. However, this result must be interpreted carefully: The model predicted almost the same motivation for all users, thus not taking into account individual differences. Because the value that was predicted for all users is similar to

the mean value of self-assessments, the difference between inference and self-assessment is relatively small. Looking at the correlation coefficients for both missions, it becomes evident that the model's inferences do not correspond to individual differences in perceived motivation: Pearson's r-value is 0.132 for mission one and −0.143 for mission two.

Therefore, it has to be concluded that CogIUM's prediction of a user's motivation is not working. The very low variance in prediction for motivation implies that the model's structure is inadequate. Motivation is modeled by the variable m_{pc} in CogIUM. Its prior distribution is a beta distribution with a mean value of 0.75. Since the predicted values are all very close to the prior distribution's mean value, the prior distribution might have been too informative to allow greater variation in the predictions. Going forward, it has to be investigated in additional experiments why exactly the inferences for motivation are all so close together. Once the cause is located, the variable m_{pc} needs to be remodeled accordingly.

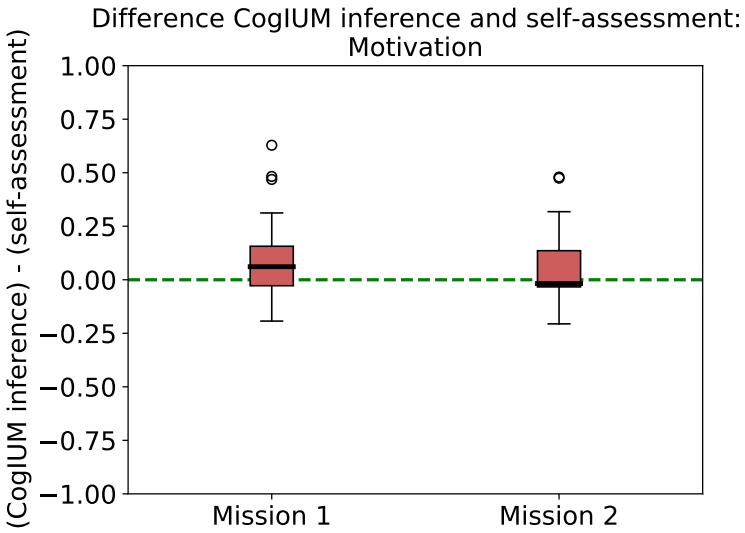


Figure 6.8: Differences between prediction and self-assessment for motivation. Values greater than 0 mean the prediction was higher than the self-assessment, values less than 0 mean the opposite.

Next, CogIUM's inference of a user's prior knowledge is evaluated similarly to the previous two cognitive variables. Looking at the data for inferences and users' self-assessments, it is apparent that CogIUM's inferences are in a much higher range than the users' self-assessments for both missions: A majority of the users' self-assessments for prior knowledge are in a low range around 0.25. In contrast to that, the model predicted much higher values in a range of around 0.75 for both missions.

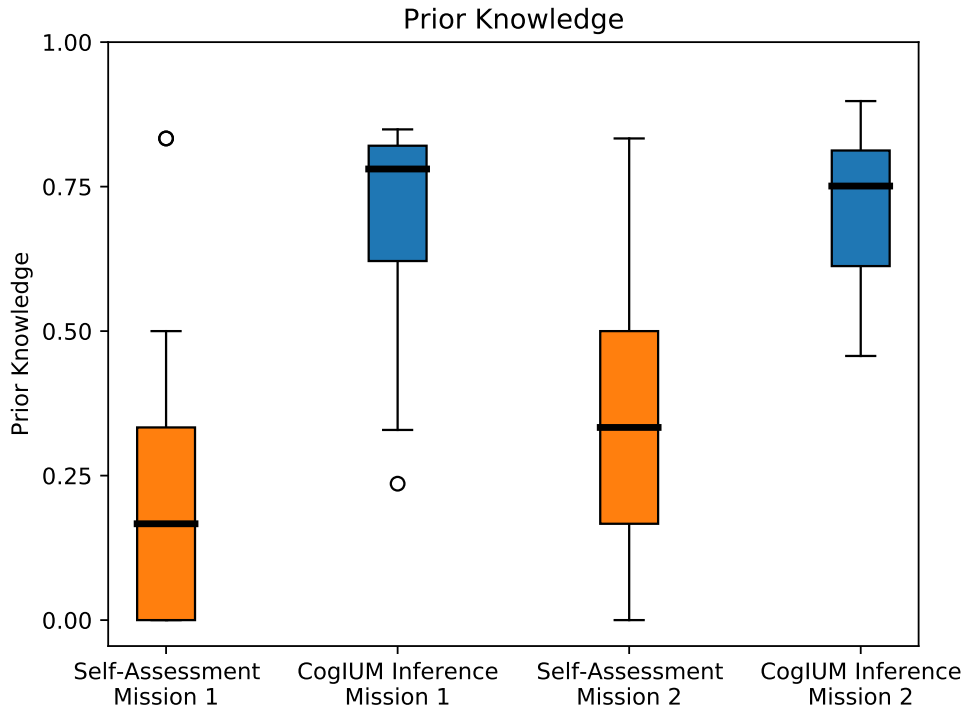


Figure 6.9: Self-Assessment and CogIUM inference for prior knowledge in mission one and mission two. Pearson correlation coefficient is 0.324 for mission one and 0.151 for mission two.

The difference between inference and self-assessment is showcased by figure 6.10: The values for both missions are largely above the dotted green line, indicating that the inference is much higher than the self-assessment for a majority of users. A potential explanation for this discrepancy is the varying interpretation of the term *prior knowledge*: During the study, several participants reported being unsure of the exact meaning of prior knowledge in the user study's context. Most users rated their prior knowledge low because they had no initial knowledge of the game Streamlined Lost Earth. However, what was meant by the question was the prior knowledge regarding the image exploitation tasks. Additionally, 3 out of 4 image interpretation tasks could be solved without special technical knowledge. Instead, general knowledge of images and aerial imagery was required, which most users considered self-evident. Therefore, it has to be concluded that due to the previously described flaws in the study design, the evaluation results for prior knowledge are not meaningful.

Hypothesis (2) states that users will rate CogIUM's inferences on them as more accurate than they will rate randomly generated Placebo-values. To investigate this hypothesis, users were split into two groups: Users from the experimental group were shown a visualization of their

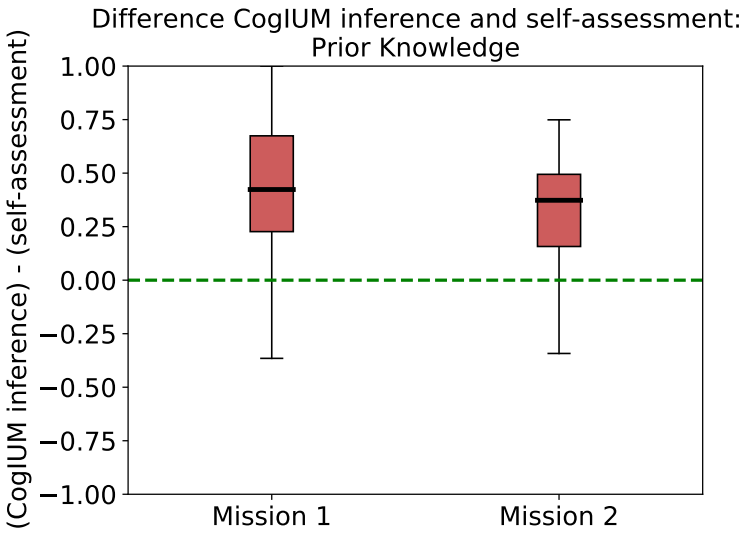


Figure 6.10: Differences between prediction and self-assessment for prior knowledge. Values greater than 0 mean the prediction was higher than the self-assessment, values less than 0 mean the opposite.

actual inference results. Users from the control group saw randomly generated Placebo-values. For both groups, the visualization was sent to the users via e-mail once CogIUM's calculation finished. The visualization plot contained four bars corresponding to four cognitive variables: *Motivation*, *prior knowledge*, *cognitive load* and *free working memory capacity*. Users were asked to rate the values' accuracy on a 5-point Likert scale. The scale ranged from 1*: very accurate* to 5*: very inaccurate*.

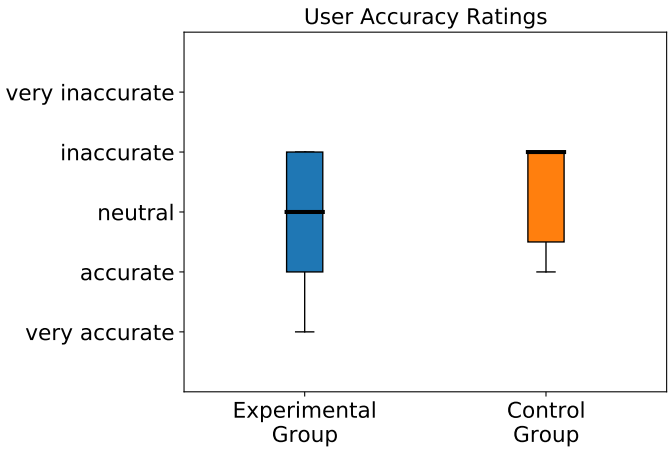


Figure 6.11: Users' ratings of the inferences' accuracy for experimental group and control group.

Figure 6.11 visualizes a summary of the users' ratings from the experimental group and the control group. Both groups' answers are distributed in a similar range. The median rating from the experimental group's users is *neutral*, the median rating from the control group is *inaccurate*. On average, the experimental group's users did rate their results as more accurate than users from the control group: On the scale from 1 to 5, the average answer for the experimental group is 2.82, the average answer for the control group is 3.36. However, the difference is not significant enough to conclude a causality behind the slightly lower mean value for the experimental group.

Hypothesis	**Result**
CogIUM's inferences and user's self-assessments are correlated...	
...for variable *cognitive load*.	confirmed
...for variable *motivation*.	refuted
...for variable *prior knowledge*.	unclear
Users from the experimental group rate their inference results as more accurate than users from the control group.	refuted

Table 6.1: The user study's hypotheses and their results.

To summarize, the results confirmed hypothesis (1) for the latent variable *cognitive load*. For motivation, hypothesis (1) was refuted since there was no correlation, and the model predicted almost the same motivation level for all users. Regarding prior knowledge, no clear conclusions could be drawn. Users' self-assessments were in a much lower range than CogIUM's inferences, but there was still a low to moderate correlation. Several users reported feeling unsure about the exact definition of prior knowledge in the user study's context. Hence, hypothesis (1) could neither be confirmed nor refuted for prior knowledge due to imprecise question design. Similarly, the evaluation of hypothesis (2) also yielded ambiguous results: While users from the experimental group, on average, did rate their results as more accurate than users from the control group, the difference in ratings was very small. Therefore, hypothesis (2) was refuted.

6.5 Posterior Predictive Check

During CogIUM's iterative development process, posterior predictive checks were used to assess the model's capability to reproduce the input data's key features. However, only synthetical input data was available at the time of development to perform posterior predictive checks. Subsequent to the user study, the check can now be performed with real-world user interaction data. By evaluating the hypotheses in section 6.4, it was analyzed if CogIUM's latent variables actually correspond to users' cognitive attributes. In addition to that, posterior predictive checks also play an essential role in the model's quality assessment: If the model was unable to reproduce the observable variables' main features, this could be an additional explanation for false inference about the users' cognitive attributes. The theory behind posterior predictive checks is explained in section 3.2.

In the initial analysis of this thesis, CogIUM revealed structural flaws that caused an inadequate reproduction of observable variable mission time t_{pc}. After the extensive remodeling process described in section 5.2, CogIUM was able to reproduce the main aspects of mission time t_{pc}'s distribution. Figure 6.12 visualizes the posterior predictive mean's distribution (blue dotted line) alongside the observed values (black line) for mission time t_{pc} in mission one.

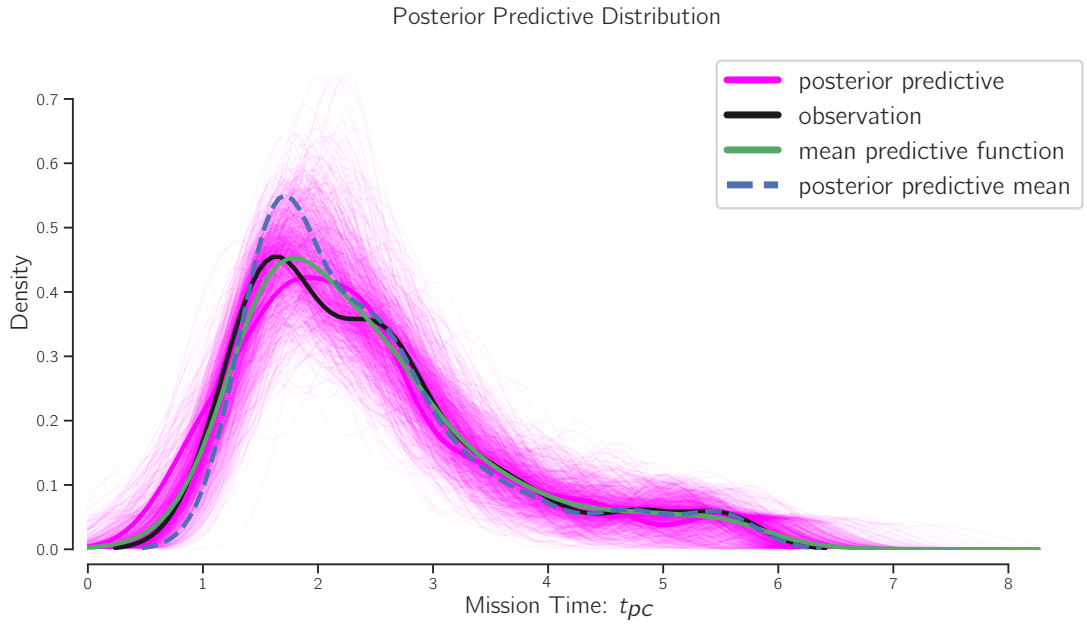


Figure 6.12: Posterior predictive check for mission time t_{pc} in mission one of the user study dataset. Observations are displayed as a kernel density estimate over the $n = 22$ participants.

As showcased by figure 6.12, CogIUM's posterior predictive function is very close to the observations for mission time t_{pc} in mission one of the user study dataset. Thus, the model is able to reproduce the observed data satisfactorily. For mission two, both the posterior predictive function and the observed data's distribution are very similar to mission one.

With regards to mission score s_{pc}, the posterior predictive function and the observations are displayed in figure 6.13. Strikingly, all observed scores lie in a high range, and 1.0 is the most common score. This can be attributed to low task difficulty and the multiple-choice question style within SLE. Despite the uneven distribution of scores, the model is able to reproduce the observed data for mission scores s_{pc} very well: All spikes in the observed data (black line) are also represented in the posterior predictive mean distribution (dotted blue line).

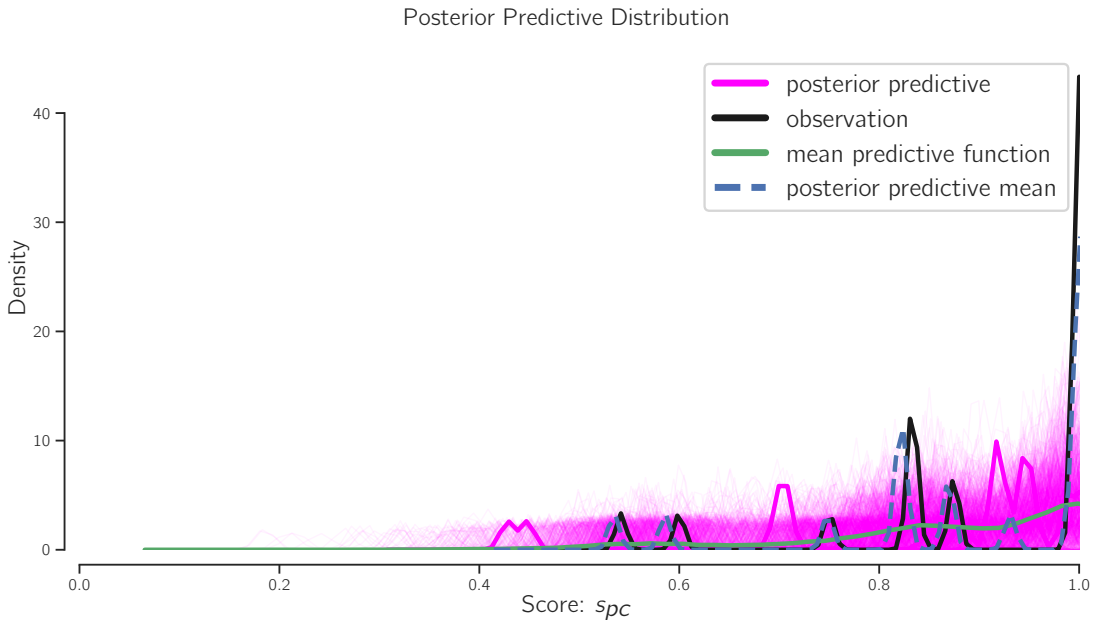


Figure 6.13: Posterior predictive check for mission score s_{pc} in mission one of the user study dataset. Observations are displayed as a kernel density estimate over the $n = 22$ participants.

Concerning the newly added observable variable *required attempts* a_{pc}, the model's posterior predictive mean function diverts more from the observations than for the previously analyzed observable variables. This is illustrated by figure 6.14: The posterior predictive mean values are not distributed equally or at least very closely to the observed values. Instead, more predicted mean values than observed values lie around 0.5, causing the spike of the dotted blue line. However, the observed data's key features are still represented by the posterior predictive mean distribution, albeit slightly shifted.

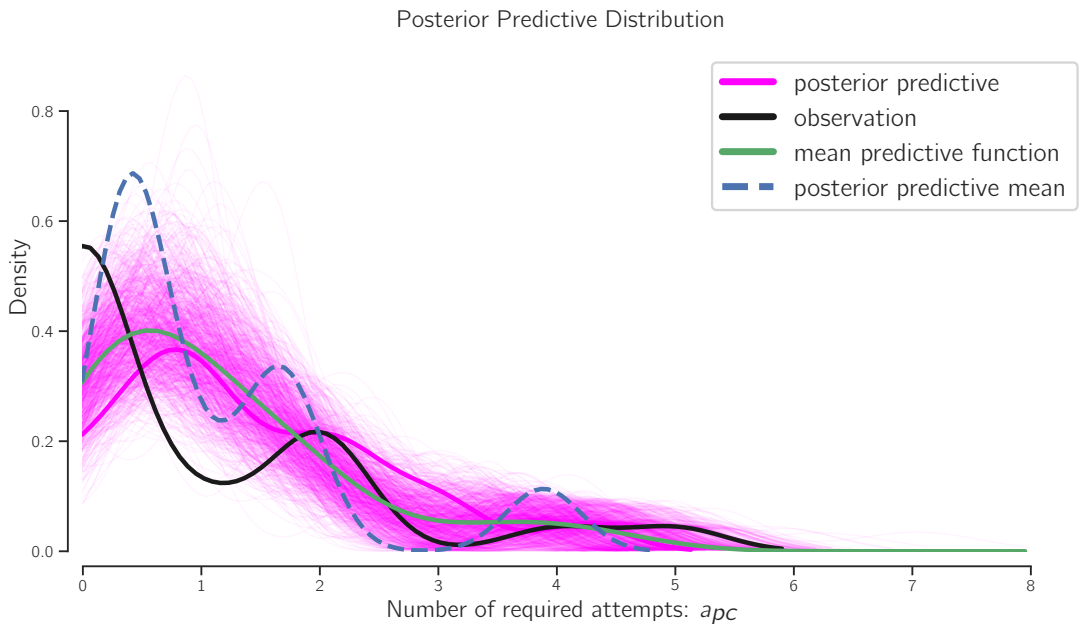


Figure 6.14: Posterior predictive check for required attempts a_{pc} in mission one of the user study dataset. Observations are displayed as a kernel density estimate over the $n = 22$ participants.

For the other newly introduced variable *detours* d_{pc}, CogIUM's posterior predictive performance is similar to the performance for *required attempts* a_{pc}: While there are some deviations of the posterior predictive mean distribution from the observed data, the observed data's key features are still reproduced. To avoid redundancy, the posterior predictive plots for detours d_{pc} in missions one and two are omitted in this section. They can be found in figures 6 and 7 in the appendix.

To summarize, the CogIUM model is able to reproduce the key features for all five observable variables. Especially accurate posterior predictive mean distributions were found for *mission time* t_{pc} and *mission score* s_{pc}. Therefore, it is concluded that the model's mathematical structure is adequate in the sense that it enables a reasonably good fitting of the observed data.

7 Conclusion and Outlook

The thesis's main goal was to investigate how CogIUM can be applied to an existing serious game to enable adaptivity. CogIUM's applicability was showcased by enhancing, expanding, and finally applying CogIUM to the serious game Streamlined Lost Earth (SLE). Subsequently, the model was evaluated in a user study in which users played SLE missions and filled out self-assessments regarding their cognitive attributes. By comparing CogIUM's inferences to users' self-assessments, it was investigated if CogIUM's latent variables actually correspond to users' cognitive attributes.

Based on the findings, it can be concluded that CogIUM proved the potential to infer a user's cognitive load only from his interaction data: There is a significant moderate correlation between CogIUM's inferences and users' self-assessments of total cognitive load in the user study. Additionally, users rated visualizations of CogIUM's inference on them as more accurate than they rated visualizations of Placebo values, albeit only slightly. For the other two predicted variables *motivation* and *prior knowledge*, the results showed no significant correlation. For motivation, CogIUM inferred almost the same value for all users and was thus unable to predict individual differences. The lack of correlation for the variable *prior knowledge* can be attributed to flaws in the user study's questionnaire design.

During this thesis's development, the whole processing chain from the input of user interaction data to the output of adaptivity scores was covered. Parsing functionality was implemented to extract the observable variables from the xAPI statements containing the user interaction data. By extending the CogIUM model with the two observable variables *required attempts* and *detours*, more information on the user's interaction is now utilized for CogIUM's inference. Therefore, better-informed adaptivity scores can be computed. Furthermore, structural issues that hampered the model's ability to fit certain patterns of input data were resolved. Thus, the model was able to reproduce the key data features from the user study data for all five observable variables *mission success*, *mission score*, *mission time*, *required attempts*, and *detours*.

CogIUM's sampling process yields posterior distributions for latent cognitive variables. While the CogIUM's sampling duration of several minutes limits practical usage, an *online inference* approach was implemented that utilizes a precomputed database to return posterior distributions for very similar input instantly. Furthermore, methods were developed to compute

the adaptivity response's components *Assistance Level* and *Skill Level* from the posterior distributions.

In conclusion, a complete processing chain that takes user interaction data as input and outputs adaptivity scores was implemented. The CogIUM model was enhanced significantly by introducing two additional observable variables. Thus, CogIUM was applied to the serious game SLE, fulfilling this thesis's primary research objective. In the evaluation, CogIUM showed promising capability to infer a user's cognitive load solely based on interaction data.

Future Work

The overall objective should be to integrate CogIUM into the complete adaptivity cycle: By implementing adaptivity manifestations based on CogIUM's adaptivity scores, it can be investigated if the enabled adaptivity actually improves learning outcomes.

Prior to that, several aspects should be addressed: It must be investigated why CogIUM struggled to predict the latent cognitive variables *motivation* and *prior knowledge*. For *motivation*, further investigation of the model's structure is required since it failed to predict individual differences. Possible explanations and suggestions for remodeling were presented in section 6.4. Regarding *prior knowledge*, the ambiguous results can be attributed to imprecise questions in the user study. Therefore, it remains to be investigated if CogIUMa can accurately predict a user's prior knowledge level.

Regarding the adaptivity response defined by ELAI, CogIUMa can compute scores for two of three components: *Skill Level* and *Assistance Level*. One obvious starting point for future work is to research how *Performance Score*, the adaptivity response's third component, can be inferred. Possible approaches could either directly use xAPI statements as input or use the latent cognitive variables' posterior distributions computed by CogIUMa. Additionally, the computed score for *Assistance Level* requires further evaluation: It is calculated as a weighted sum of the cognitive variables' predicted mean values. The weights were chosen to represent best the self-assessed need for assistance from the user study data. A feasible way to evaluate the weights' validity would be by conducting another user study. Due to time restrictions, this was not possible within this thesis.

To enable real-time processing of adaptivity requests, this thesis introduced the *online inference* mode that returns precomputed adaptivity responses for similar input data. Since CogIUM's sampling process takes too long for real-time applications, future work should investigate the online inference's accuracy and explore further approaches to accelerate the adaptivity response's computation.

Bibliography

ADL (2021). *xAPI Specification by ADL*. https://adlnet.gov/projects/xapi/. Accessed: 2021-03-02.

Atorf, Daniel, Ehm Kannegieser, and Marlene Dillig (2020). "Towards a concept for streamlining game design of an existing serious game and preliminary evaluation." In: *19th International Conference on WWW/INTERNET 2020*. International Association for Development of the Information Society, pp. 155–160.

Atorf, Daniel, Ehm Kannegieser, and Wolfgang Roller (2019). "Balancing Realism and Engagement for a Serious Game in the Domain of Remote Sensing." In: *Games and Learning Alliance*. Ed. by Manuel Gentile, Mario Allegra, and Heinrich Söbke. Springer International Publishing, pp. 146–156.

Aydinbas, Paul Michael (2019). "Realizing Cognitive User Models for Adaptive Serious Games." Master's Thesis. TU Darmstadt.

Boyle, Elizabeth A. et al. (2016). "An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games." In: *Computers and Education* 94, pp. 178–192. URL: https://www.sciencedirect.com/science/article/pii/S0360131515300750.

Brunken, Roland, Jan L. Plass, and Detlev Leutner (2003). "Direct Measurement of Cognitive Load in Multimedia Learning." In: *Educational Psychologist* 38.1, pp. 53–61. eprint: https://doi.org/10.1207/S15326985EP3801_7. URL: https://doi.org/10.1207/S15326985EP3801_7.

Buettner, Ricardo (2013). "Cognitive Workload of Humans Using Artificial Intelligence Systems: Towards Objective Measurement Applying Eye-Tracking Technology." In: *KI 2013: Advances in Artificial Intelligence*. Ed. by Ingo J. Thimm and Matthias Thimm. Springer Berlin Heidelberg, pp. 37–48.

Chong, Toh Seong (2005). "Recent Advances Cognitive Load Theory Research: Implications for the Instructional Designers." In: *Recent Advances in Cognitive Load Theory Research*.

Conati, Cristina et al. (May 2020). "Comparing and Combining Interaction Data and Eye-tracking Data for the Real-time Prediction of User Cognitive Abilities in Visualization

Tasks." In: *ACM Transactions on Interactive Intelligent Systems* 10.2, 12:1–12:41. URL: https://doi.org/10.1145/3301400 (visited on 11/29/2020).

Farrell, Simon and Stephan Lewandowsky (2018). *Computational modeling of cognition and behavior*. Cambridge University Press.

Gelman, Andrew et al. (2013). *Bayesian Data Analysis*. CRC Press.

Gjoreski, Martin, Mitja Luštrek, and Veljko Pejović (Oct. 2018). "My Watch Says I'm Busy: Inferring Cognitive Load with Low-Cost Wearables." In: *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. UbiComp '18. New York, NY, USA: Association for Computing Machinery, pp. 1234–1240. URL: https://doi.org/10.1145/3267305.3274113 (visited on 11/24/2020).

Hart, Sandra G. (2006). "Nasa-Task Load Index (NASA-TLX); 20 Years Later." In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50.9, pp. 904–908. eprint: https://doi.org/10.1177/154193120605000909. URL: https://doi.org/10.1177/154193120605000909.

Hart, Sandra G. and Lowell E. Staveland (Jan. 1988). "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research." en. In: *Advances in Psychology*. Ed. by Peter A. Hancock and Najmedin Meshkati. Vol. 52. Human Mental Workload. North-Holland, pp. 139–183. URL: http://www.sciencedirect.com/science/article/pii/S0166411508623869 (visited on 11/26/2020).

IOSB, Fraunhofer (2016). *Lost Earth 2307*. https://www.iosb.fraunhofer.de/de/projekte-produkte/lost-earth-2307.html. Accessed: 2021-03-11.

Jamieson-Noel, Dianne and Philip Winne (Nov. 2003). "Comparing Self-Reports to Traces of Studying Behavior as Representations of Students' Studying and Achievement." In: *German Journal of Educational Psychology* 17, pp. 159–171.

Jovanović, Jelena et al. (Mar. 2019). "Introducing meaning to clicks: Towards traced-measures of self-efficacy and cognitive load." In: *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*. LAK19. New York, NY, USA: Association for Computing Machinery, pp. 511–520. URL: https://doi.org/10.1145/3303772.3303782 (visited on 11/24/2020).

Kruschke, John K. (2010). "What to believe: Bayesian methods for data analysis." In: *Trends in cognitive sciences* 14.7, pp. 293–300.

– (2014). *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press.

Lambert, Ben (2018). *A student's guide to Bayesian statistics*. Sage.

Lee, Michael D and Eric-Jan Wagenmakers (2014). *Bayesian cognitive modeling: A practical course*. Cambridge university press.

Mankins, John C. (1995). *Technology readiness levels: A white paper*. http://www.hq.nasa.gov/office/codeq/trl/trl.pdf. Accessed: 2021-03-08.

Nakamura, Jeanne and Mihaly Csikszentmihalyi (2009). "Flow theory and research." In: *Handbook of positive psychology*, pp. 195–206.

Paas, Fred et al. (2003). "Cognitive load measurement as a means to advance cognitive load theory." In: *Educational psychologist* 38.1, pp. 63–71.

Paas, Fred GWC and Jeroen JG Van Merriënboer (1994). "Instructional control of cognitive load in the training of complex cognitive tasks." In: *Educational psychology review* 6.4, pp. 351–371.

Pearson and EdSurge (2016). *Decoding Adaptive*. https://www.pearson.com/uk/educators/schools/making-an-impact/research-summaries/decoding-adaptive.html. Accessed: 2021-03-08.

Pelánek, Radek (2015). "Metrics for Evaluation of Student Models." In: *Journal of Educational Data Mining* 7.2, pp. 1–19.

Qian, Meihua and Karen R Clark (2016). "Game-based Learning and 21st century skills: A review of recent research." In: *Computers in human behavior* 63, pp. 50–58.

Rustici (2021a). *SCORM overview by Rustici Software*. https://scorm.com/scorm-explained/one-minute-scorm-overview/. Accessed: 2021-03-02.

– (2021b). *xAPI Overview by Rustici Software*. https://xapi.com/overview/. Accessed: 2021-03-02.

Schmeck, Annett et al. (2015). "Measuring cognitive load with subjective rating scales during problem solving: differences between immediate and delayed ratings." In: *Instructional Science* 43.1, pp. 93–114.

Sharek, David (2011). "A Useable, Online NASA-TLX Tool." In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 55.1, pp. 1375–1379. URL: https://doi.org/10.1177/1071181311551286.

Sharma, K. et al. (2021). "Information flow and cognition affect each other: Evidence from digital learning." In: *International Journal of Human Computer Studies* 146.

Shute, Valerie and Diego Zapata-Rivera (Jan. 2012). "Adaptive Educational Systems." In: *Adaptive Technologies for Training and Education*, pp. 7–27.

Streicher, Alexander, Lukas Bach, and Wolfgang Roller (2019). "Usage Simulation and Testing with xAPI for Adaptive E-Learning." In: *Transforming Learning with Meaningful Technologies*. Ed. by Maren Scheffel et al. Springer International Publishing, pp. 692–695.

Streicher, Alexander and Wolfgang Roller (2015). "Towards an interoperable adaptive tutoring agent for simulations and serious games." In: *International Conference on Theory and Practice in Modern Computing, MCCSIS*, pp. 194–197.

Streicher, Alexander and Wolfgang Roller (2017). "Interoperable Adaptivity and Learning Analytics for Serious Games in Image Interpretation." In: *Data Driven Approaches in Digital Education*. Ed. by Élise Lavoué et al. Springer International Publishing, pp. 598–601.

Streicher, Alexander, Wolfgang Roller, and Christian Biegemeier (2017). "Application of Adaptive Game-Based Learning in Image Interpretation." In: *European Conference on Games Based Learning*. Academic Conferences International Limited, pp. 975–978.

Streicher, Alexander and Jan D. Smeddinck (2016). "Personalized and Adaptive Serious Games." In: *Entertainment Computing and Serious Games*. Ed. by Ralf Dörner et al. Springer International Publishing, pp. 332–377. URL: https://doi.org/10.1007/978-3-319-46152-6_14.

Streicher, Alexander, Daniel Szentes, and Alexander Gundermann (2016). "Game-Based Training for Complex Multi-institutional Exercises of Joint Forces." In: *European Conference on Technology Enhanced Learning*. Springer, pp. 497–502.

Sweller, John (1988). "Cognitive load during problem solving: Effects on learning." In: *Cognitive Science* 12.2, pp. 257–285. URL: https://www.sciencedirect.com/science/article/pii/0364021388900237.

– (1994). "Cognitive load theory, learning difficulty, and instructional design." In: *Learning and Instruction* 4.4, pp. 295–312. URL: https://www.sciencedirect.com/science/article/pii/0959475294900035.

– (2010). "Element interactivity and intrinsic, extraneous, and germane cognitive load." In: *Educational psychology review* 22.2, pp. 123–138.

Tiangolo (2021). *FastAPI documentation*. https://fastapi.tiangolo.com/. Accessed: 2021-03-07.

Unity (2020). *Unity Web-GL Documentation*. https://docs.unity3d.com/Manual/webgl-networking.html. Accessed: 2021-03-11.

Van Merrienboer, Jeroen JG and John Sweller (2005). "Cognitive load theory and complex learning: Recent developments and future directions." In: *Educational psychology review* 17.2, pp. 147–177.

Vidal, J. C., T. Rabelo, and M. Lama (July 2015). "Semantic Description of the Experience API Specification." In: *2015 IEEE 15th International Conference on Advanced Learning Technologies*. ISSN: 2161-377X, pp. 268–269.

Zhou, Mingming and Philip H Winne (2012). "Modeling academic achievement by self-reported versus traced goal orientation." In: *Learning and Instruction* 22.6, pp. 413–419.

List of Figures

Appendix

Figure 1: A screenshot from the user study. The user is playing the hidden object challenge, the horse is highlighted by the red circle for visualization. Users did not see the circle.

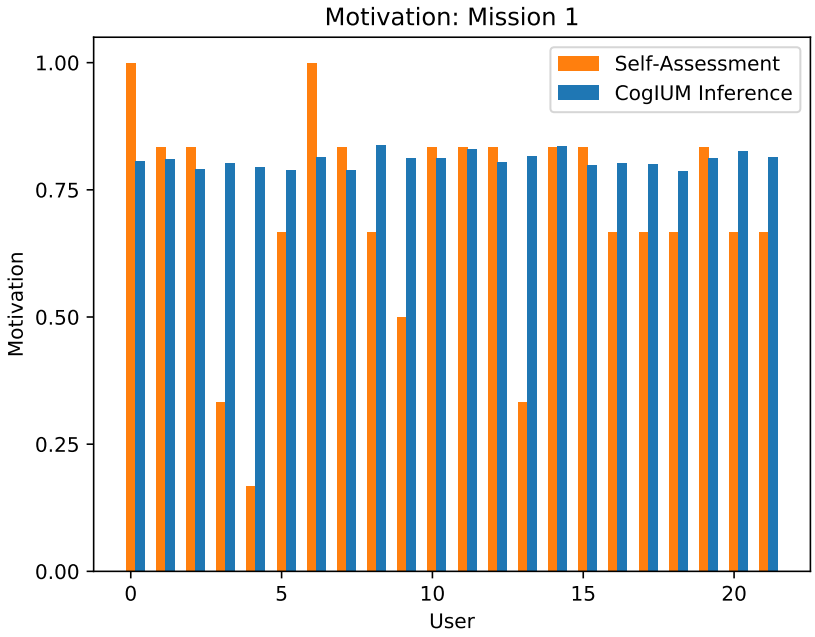


Figure 2: Self-Assessment and CogIUM Inference for motivation in mission one. Pearson correlation coefficient is 0.13.

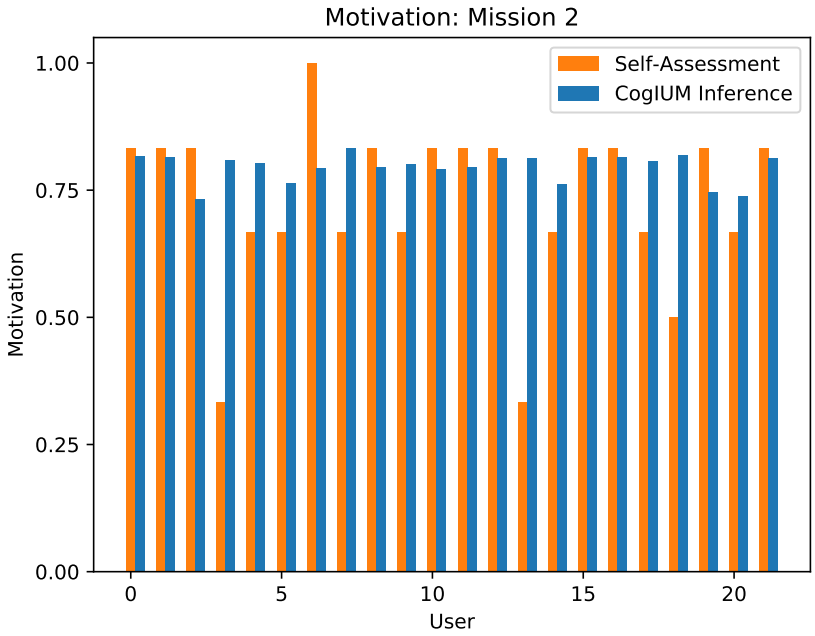


Figure 3: Self-Assessment and CogIUM Inference for motivation in mission two. Pearson correlation coefficient is −0.14.

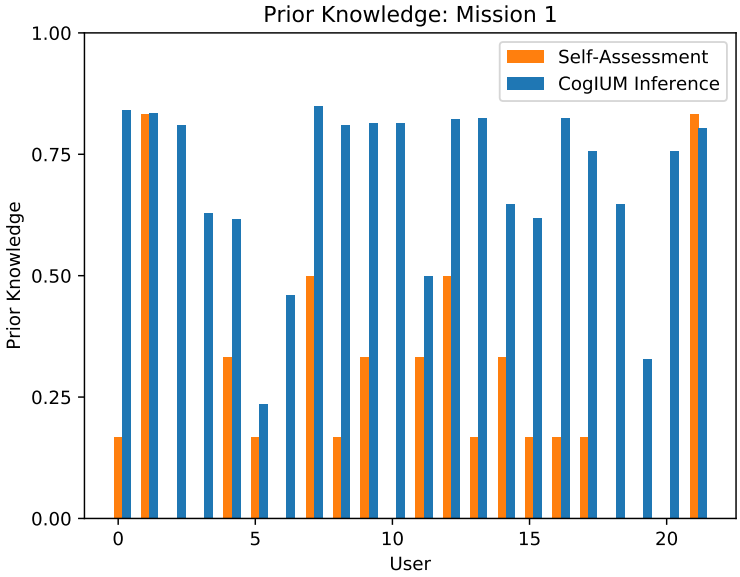


Figure 4: Self-Assessment and CogIUM Inference for prior knowledge in mission one. Pearson correlation coefficient is 0.32.

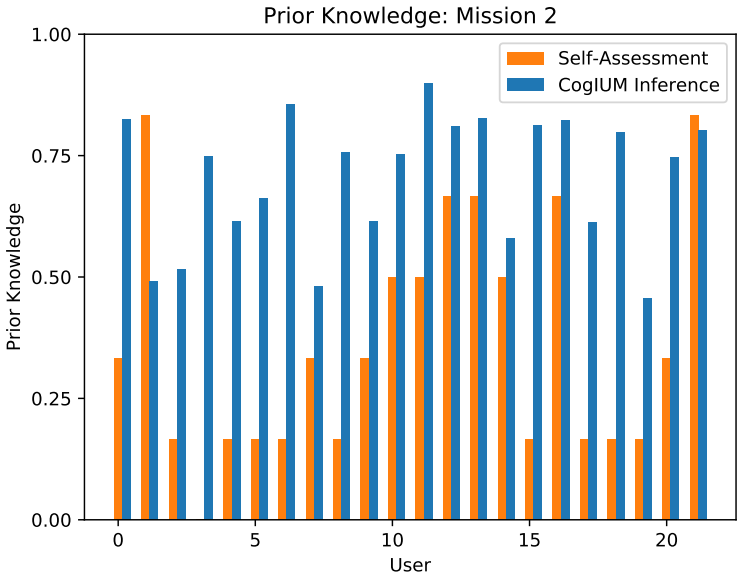


Figure 5: Self-Assessment and CogIUM Inference for prior knowledge in mission two. Pearson correlation coefficient is 0.15

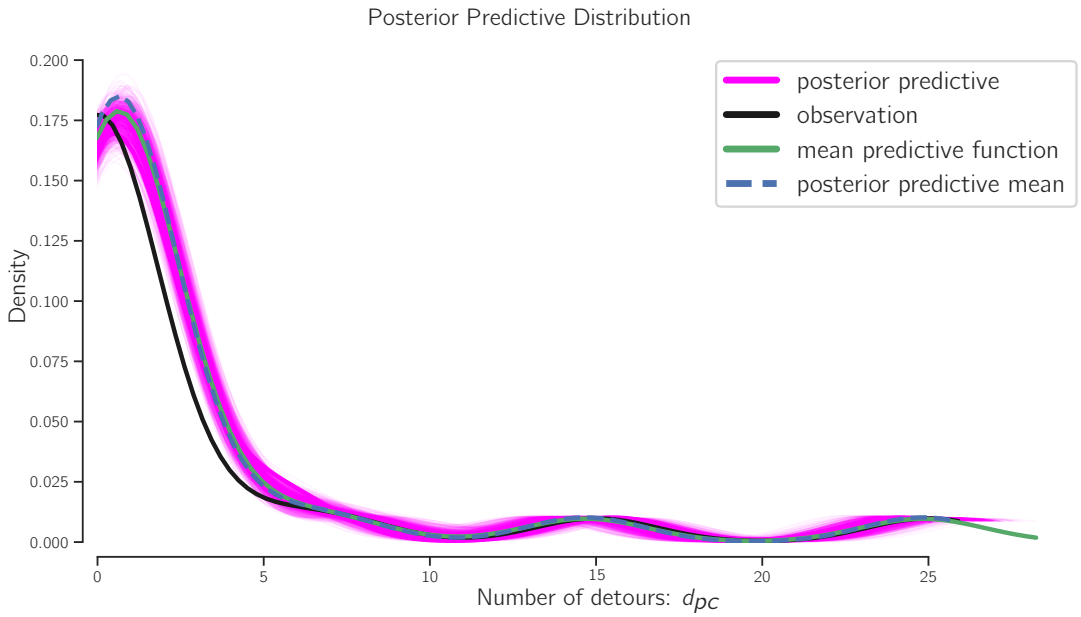


Figure 6: Posterior predictive check for detours d_{pc} in mission one of the user study dataset. Observations are displayed as a kernel density estimate over the $n = 22$ participants.

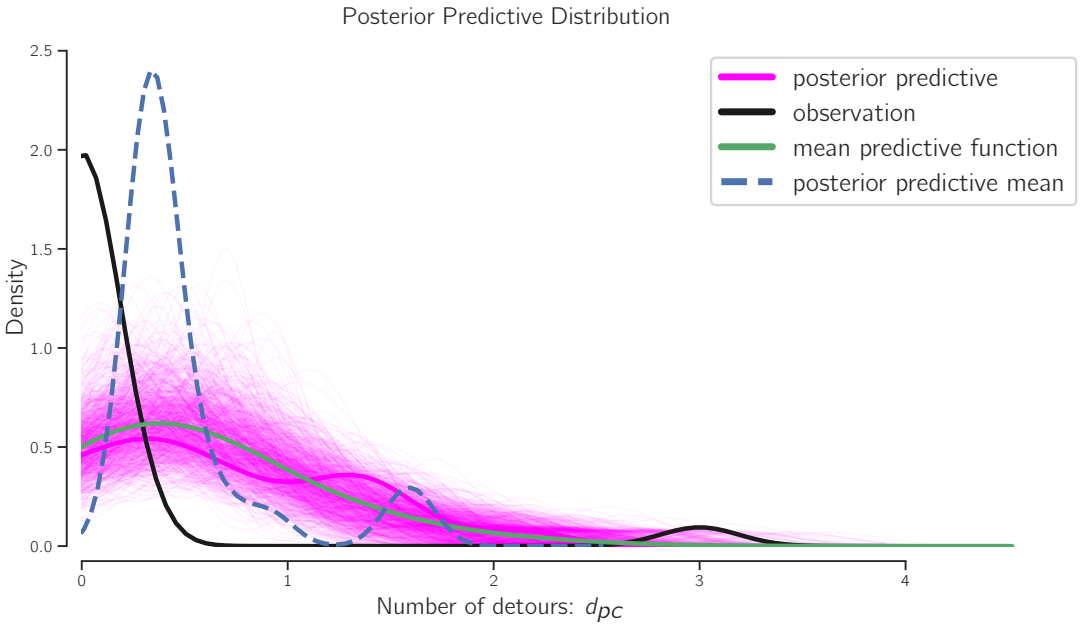


Figure 7: Posterior predictive check for detours d_{pc} in mission two of the user study dataset. Observations are displayed as a kernel density estimate over the $n = 22$ participants.