WP1 Report

Draft Version 1.0
19-03-2018

UNIVERSITY OF SURREY

Document Info

Author: Haiyue Yuan
Patrice Rusconi
Shujun Li

Release Date: 19/03/2018
Issue: Draft version 1.0

1. Introduction

This work package is led by UoS, and focus on design descriptive languages for human and HCI modelling, create a human behaviour database and gather a collection of reusable behaviour templates.

2. Tasks

2.1 Selected/adapted human cognitive models

- **CogTool vs. SANLab-CM**

➢ **Psychological theory underlying CogTool (KLM)**

The psychological theory underlying CogTool draws from scores of results in the psychological literature and has been validated through HCI research and real world use over the decades. The human information processing (HIP) theory underlying CogTool is specific to skilled human behaviour with computer-based tasks with certain time parameters. KLM version of HIP theory of human behaviour, which consists of three modules:

Perception module: take information from the world and passes it to a cognitive module. Perception takes **100ms** because signals in computer-based tasks are assumed to be relatively easy to perceive (For example, perception would take longer if the tasks took place in extremely low light for visual signals or in high noise for auditory signals)

Cognitive module: process the information from the perception model, possibly combining it with information from other sources (such as a long-term memory), and sends commands to a motor module. The mental operator (M) that represents a composite of all unobservable activities, including cognitive and perceptual activities such as comprehending perceptual input and recognising the situation, command recall, and decisions. Card, Moran, and Newell empirically estimated M to be 1350ms. In CogTool, we estimate M to be **1200ms** to accommodate the ACT-R architecture, which normally requires a 150ms look-at operator to accompany each typical mental operation.

Motor module: manipulate the prototype to take action in the world. Motor module in KLMs for desktop applications includes operators for pressing keys (K), pointing (P), and homing between the keyboard and the mouse. The motor operators' duration is set based on psychology literature with K dependent on a person's typing speed, P related by Fitts's Law to the distance moved and the target's size.

- **Psychological theory underlying SANLab-CM (CPM-GOMS)**

SANLab-CM is specialised to model CPM-GOMS. CPM-GOMS combines the task decomposition of a GOMS analysis with a model of human resource usage at the level of cognitive, perceptual, and motor operations. Decomposition produces a representation of the task as a set of nested goal states that include an initial state and a final state.

CPM-GOMS:

Human performance predictions are constructed from primitives that include estimates of the time for the elementary cognitive, motor, and perceptual operations. These primitives are hypothesised to underlie actions such as typing a key or moving a mouse. Much of the power of CPM-GOMS to predict skilled behaviour comes from its ability to model overlapping actions by interleaving cognitive, perceptual and motor operators.

SANLab-CM:

Can model parallel process and output results in distribution format, but to be notice that the distribution produced by SANLab-CM does not represent different methods for accomplishing a task, but individual differences within a method. That is, it does not predict, for example, the difference in time attributed to using menu vs. keyboard shortcuts, but the variation inherent in human behaviour even when using a single method.

Comparison between KLM and CPM-GOMS

KLM and CPM-GOMS are both family of GOMS modelling, which produce the same sequence of observable operators. However, KLM offers more higher-level modelling, and CPM-GOMS can produce in a more detailed level.

Prediction: Normally KLM predicts the performance by an average expert user, and CPM-GOMS predicts performance by an extreme expert user, which can produce maximum operator over-lapping, finer-grain time estimates for the individual operators, and the minimum of cognitive activity allowed by the MHP.

Unobservable operators: both models make some assumptions for unobservable operators:

KLM: makes the simplest assumption, putting all such operations (perceiving information, eye movements, comparisons, decisions, mental calculations, etc.) into one operator, M, 1.35 seconds in length, always put at the beginning of a cognitive unit.

CPM-GOMS: represents the underlying unobserved operations in terms of the cycle times of the MHP processors, such as the cognitive cycle time, perceptual cycle time, and eye movement time. However, the other operators needed to accomplish a task and their dependencies make every critical path different, and no one estimate of 'mental time' is meaningful in CPM-GOMS (might need multiple operators either in serial or parallel processing).

Summary: KLM is easy to apply, but predicts only execution time, and only from analyst-supplied methods. CPM-GOMS predicts execution time for subtle, overlapping patterns of activities, but also requires analyst-supplied methods. In current stage, we will focus use KLM modelling technique. The UoS team decides to develop a software CogTool+ based on CogTool.

2.2 Suitable descriptive languages

The starting step of using CogTool to model an interactive system is to define its UI layout and user interaction tasks via CogTool GUI, and the outcome of this process is known as a CogTool project. To make CogTool more flexible with more compatibility, it allows users to import and export CogTool project in XML format thanks to the built-in XML schema. Having in mind that the proposed CogTool+ needs to be compatible with CogTool, we decided to take the advantage of existing XML format support of CogTool, and designed a XML based machine-readable language for defining high level modelling of an interactive system in a scalable and dynamic way to achieve parameterization and automation of cognitive tasks modelling.

2.3 A model describing HCI and its descriptive language

Based on the descriptive language, we developed a meta model to define high level User Interface (UI)s and user interactions. It consists of two sub-models: one is descriptive model, and another one is algorithmic model. The descriptive model is responsible to define Human Computer Interface (HCI), and combing use of descriptive model and algorithmic model is dealing with all user interactions (more details can be found in our paper to be submitted to UIST2018 [1])

2.4 Human behaviour database

We have done some user studies to observe human behaviours while interacting with user authentication systems. The first user study used an eye tracker to observe how human visually interact with a relative complex user authentication system ‘Undercover’ [2]. The results can be found in our HAS, HCII2017 paper [3]. In the second user study, we used an eye tracker to observe how human visually interact with a simpler user authentication system, which expect to produce more general understanding of human visual search process, and the results can be found in our pass picture user study paper to be submitted to IJHCS [3].

2.5 A collection of reusable behaviour templates

Based on the findings from section 2.4, we have developed a reusable behaviour templates implemented using JavaScript, this can be used as algorithmic model with descriptive model in the meta model to define complex and dynamic user interaction. Please see ‘behaviourTemplate.js’ for more information.

3. References

[1] Working Paper from UoS to be submitted to UIST2018

[2] Hirokazu Sasamoto, Nicolas Christin, and Eiji Hayashi. 2008. Undercover: authentication usable in front of prying eyes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). ACM, New York, NY, USA, 183-192.

[3] Yuan H., Li S., Rusconi P., Aljaffan N. (2017) When Eye-Tracking Meets Cognitive Modeling: Applications to Cyber Security Systems. In: Tryfonas T. (eds) Human Aspects of Information Security, Privacy and Trust. HAS 2017. Lecture Notes in Computer Science, vol 10292. Springer, Cham