Measuring and Reducing the Cognitive Load for the End Users of Complex Systems

James Oakes, Mark Johnson, James Xue and Scott Turner[1]

[1] University of Northampton, United Kingdom

Abstract. With the proliferation of complex computer systems, end users face a never-ending increase in the number of tasks, methods, inputs, passwords, usernames (and so on) when using online and standalone computer-based systems and applications. This paper examines a method and approach to measure how complex a system is to use, and how to reduce the complexity of such systems by minimising the requirement for human inputs as much as possible, in order to reduce the cognitive load for that user, or group of users. This paper addresses a study completed around using virtualised computer management systems interfaces of two well-known products AWS (Amazon Web Services), Oracle Cloud, and compares the complexity of the steps and interface for end users to a private cloud less well-known system called the IDE (Intelligent Design Engine). By using a set of derived formula, we examine how this can be applied to systems that have qualitative data feedback from the experiment process, and how to convert this effectively into quantitative data. This data is then analysed numerically using a unique approach to provide additional and meaningful results based of the original end user data.

Keywords: Cognitive Load, Virtualise, Cloud, Intelligent, Expert Systems

1 Introduction

1.1 Cognitive Load and End Users

Every task a human being undertakes requires a certain amount of cognitive power, or mental effort, in order to carry it out to completion [1], [2]. As an example, this could be from a singular simple task, such as clicking a mouse button, to a set of activities that need to be carried out in a specific sequence in order to complete an overall task successfully, such as cooking a meal using a set of ingredients and following a recipe. The question that rises from this, is how can the complexity of a task or set of tasks be measured, and is it possible to understand the cognitive load for a user or group of users?

The evidence from other studies and this work suggest it is possible to understand this, and other similar experiments in various experimental settings have provided evidence and results that demonstrate how to measure cognitive load [3], [4], [5].

Within the field of work, this method of study is generally referred to as Cognitive Load Theory (CLT) in relation to task orientated problem solving. One reason and requirement for making this feasible is because, like most processes, there is usually a start and an end, and a subsequent number of tasks in-between that are usually performed in a certain sequence. Once the process completes, this can result in a successful end and objective being met, or perhaps even in a full or partial failure. Understanding the sum of all the tasks in process is therefore essential to be able to measure the overall complexity load [6]. Some processes are simple, for example pressing a power on/off button on a Television (TV) remote control. You could consider that there are a few steps to this process, one locating the TV remote, two locating the correct button (power), and three physically pressing this button, to achieve the desired effect (e.g. switching the TV on/off). Conversely, other processes can be considered complex, such as the creation of a Virtual Machine (VM), due to the number of steps and the inherent know-how and technical expertise required to complete [7].

Further to existing studies, this paper examines how the cognitive load for a complex process (set of tasks) can be measured using a unique formula and method referred to as the Complexity Load Rating (CLR). The work examines the feasibility and challenges around recording qualitative feedback and results from end users and proposes a method to translate this into numerical or quantitative data [8], [9], [10]. The results are then calculated for each group of users, and are then evaluated to present evidence on how a complex process (such as VM provisioning) can be simplified as a result of the steps being developed with higher levels of automation and the use of pre-coded system intelligence [11], [12], [13].

1.2 Experiment Definitions

The following experiment was performed to measure the CLR of volunteer human participants using the Intelligent Design Engine (IDE), AWS and Oracle Cloud systems to provision a Virtual Machine (VM). In terms of participants, a total of ninety end users were split into three broad technical capability groups of expert, experienced and novice; all users were given access to a web-browser interface (Chrome); the user types are described in the Table 1 below:

Table 1. End User Evaluation Group Demographic.

User Group Type	Definition	Quantity of Users
Novice Users	A user with little or no formal training in computer science and no work experience in computing disciplines.	30
Experienced Users	A user with some training in computing disciplines, up to A-level standard, with some formal training or 1-3 years' work	30

	experience in the field	
Expert Users	A user with training in computing disciplines, with a bachelor's degree level or above, or with more than 5 years' work experience in the field	30

It should be noted, that end users were asked to categorise themselves into one of the three groups listed in Table 1. Once each category was filled (at a threshold of 30 users), no further end users of this type were included in the experiment to attempt to achieve a wider and equal spectrum of feedback based on general end user expertise levels. Table 2 below features definitions of how the process analysis was broken down into the tasks and sub-task components. For the purposes of this experiment and calculating the user feedback, we acknowledge sub-components of tasks, but never ask the users to provide their results at this level of granularity; instead, we only deal with the qualitative result given at the task level.

Table 2. Process, Task, Sub-component Definitions.

Categorisation	Definition
Process	A set of tasks which make up a complete process flow; for example the steps/tasks required for the building of a virtual machine
Task	An action, which is part of a process, such as creating an RSA public and private key pair for a user and then deploying it
Sub-Component	A task may be made up of sub-components, such as key generation, key distribution, and setting key permissions, and testing the private and public key handshake

Furthermore, for each task, we now consider the definitions associated with the task complexity rating as per Table 3:

Table 3. Task Complexity Definition.

Task Type	Definition	Weighting Score
Simple	Intuitive, no training required. An example of a simple tasks would be answering a question such as "What is your age?", accessing a URL via a browser to load a website, or sending a 10-20 worded SMS (Short Message Service) message	1

Moderate	Basic training required, some experience and know-how necessary to execute the task. An example of a moderate complex tasks would be following a recipe with 3-4 ingredients to prepare and make a meal, writing a BASIC computer program to calculate the Body Mass Index (BMI) value of a human being, or being able to describe and use Pythagoras theorem to calculate the length of the hypotenuse.	3
Difficult	Advanced training required, experience essential on how to implement and complete the task. An example of a difficult task would be completing a residential home extension architectural drawing to conform to local government planning and building regulations, or being able to write a computer program to graphically draw a chessboard, or being able to explain in a classroom the full implementation of Internet Protocol version 4 (IPv4), providing examples of network classes, subnets and network routing.	5

Finally, Table 4 defines the three possible Process Mechanism definitions we will allocate for each task:

Table 4. Process Mechanism Definition.

Task Mechanism	Definition
Manual	All sub-components of the task require manual user inputs
Semi-Automated	Some of the sub-components of the task require manual user inputs, some are automated
Automated	No sub-components of the task require any user inputs

It is worth pointing out the fact, that if a certain task is automated fully, no matter how complex it is (and irrespective of the number of subcomponent tasks), it can never be recorded as anything other than a simple task from an end user perspective. The reason for this being that the complex or intelligent system has been designed and coded in such a way as to alleviate or negate the cognitive requirement (or load) away from the end user.

1.3 Cognitive Load Rating

How to measure the cognitive load of a task is based upon the following general conditions, described by the qualitative (subjective) terms below:

1) The end user interpretation of the task as either simple, moderate or difficult
2) Is the task (or set of tasks) which make up the process automated, semi-automatic, or manual

It is important when collecting qualitative data, that not too many options are presented for the end user evaluation data outputs, based on their experience and the experiment process undertaken. For example, allowing human test subjects to input unstructured data such as free-text, or even handwritten text, makes the collation and analysis of data somewhat more difficult to interpret, simply because of the number of permutations and recognition of what the written data means [14].

Therefore, in the context of this study, when we refer to task complexity, this is defined or described (subjectively) by the end user as simple, moderate or difficult. Furthermore, each task undertaken has a process mechanism described as either automatic, semi-automatic, or manual. Of the three process outcomes, if a task is automated it requires no input, and is automatically set to simple; semi-automatic and manual task steps therefore require partial or full end user inputs and can receive a simple, moderate or difficult rating.

It is natural that humans prefer providing qualitative feedback for some activity they personally take part in [15]. Simple statements of whether something was good or bad is often typical of how people relate their experiences [16]. By capturing all the tasks for a process, it is possible to begin to measure the results from the experimentation method by converting qualitative data into quantitative data, thus, in effect performing a translation of words into numbers [17]. This leads us to the next phase of the experiment framework of how to use these sets of parameter variables, for Task Complexity (Table 3) and Process Mechanism (Table 4), by creating a unique method for measuring the Cognitive Load Rating (CLR) for a task or set of tasks; in this study we examine the complete process, of how an end user would deploy a VM within a computer based cloud environment.

1.3 Cognitive Load Rating Formula

The proposed formula for measuring the complexity of a singular task is as follows:

Where the Cognitive Load Rating (CLR) for one task stands for β
Where Task Complexity stands for Δ
Where Process Mechanism stands for $\emptyset$

$$\beta = \Delta \; x \; \emptyset \tag{1}$$

This general formula can be applied to any process type, or cumulatively to a set of processes, and is not just applicable to the field of computer science and VM provisioning. In order to apply this formula to a set of processes it is necessary to make this calculation able to measure the sum complexity of a set of tasks, represented as follows:

Where λ stands for the CLR for a set (sum) of tasks
Where n stands for the number of tasks
Where t stands for the task identifier

$$\lambda = \sum_{t=1}^{n} (\Delta x \emptyset) \tag{2}$$

Additionally, the formula can then be adjusted to work out the mean average of a processes task complexity by using the following method (divides by the total number of tasks represented by n):

Where *K* stands for the CLR mean average for a set of tasks

$$K = \frac{\sum_{t=1}^{n} (\Delta \, x \, \emptyset)}{n} \tag{3}$$

2 Measuring the CLR for the IDE, AWS and Oracle Cloud

The question of how to measure the CLR for complex systems is a challenging proposition. The primary reason for this is that systems designed for human end users, typically receive subjective feedback, in the form of qualitative data. For scientists, this is not a straightforward thing to measure, as it is easier for most to work directly with quantitative data output sets, rather than qualitative results [18]. This does not mean, however, that it is unfeasible or impossible to work with such data, as much work has already been completed in the field to address various approaches, for example converting words into numbers.

2.1 The 10-Step Sequence for Provisioning VM's

After building many basic VMs on various cloud platforms, using the standard interface offered through the IDE, AWS and Oracle web management portals, it was observed that the following sequence of steps were necessary to provide the required inputs to allow for each respective platform to provision a VM and make it accessible and therefore useable. By usable, we mean the point at which an end user can log in to the system and start using the VM. Table 5 explains the sequence in more detail:

Table 5. 10-Step VM Provisioning Process

Step No.	VM Provisioning Task	Description
1	Cloud Provisioning Access	This is the task needed to access and authenticate to be able to use the cloud platform, typically username/password
2	Configure Role	Setting up role access-based controls, such as administrator
3	Select compute as the option for VM deployment	Public cloud offerings prefer to allow manual choices for other offerings such as Database as a Service (DaaS), Platform as a Service (PaaS), or Software as a Service (SaaS). This experiment only deals with Infrastructure as a Service (IaaS).
4	Select the image you wish to use to install to the VM (usually the OS type/version)	Typically, the OS version and software packages, add-on's and any other supporting application software
5	Select the VM CPU, Memory, and Disk Parameters	VM Shell parameter definition phase
6	Define VM Parameters	Define, IP addresses, netmasks, OS version, packages and other such configurable parameters
7	Define VM Storage	Select type and amount of disk storage to use
8	Add SSH Key, create a Private/Public key and upload the pubic key	Generation of an appropriate SSH encryption key to secure communications and authentications
9	VM creation process	VM shell creation, install and boot process
10	Accessing the VM via the internet, or via a remote network	Access involves opening Firewall ports to access the system e.g. Transmission Control Protocol (TCP) 22 Secure Shell (Unix type), or TCP 3389 Remote Desktop Connection (Windows)

2.2 Provisioning System Values

For each platform included in the experiment, the following system values in Table 6 describe the observed level of automation for each step in the VM provisioning process. These are recorded as either manual, semi-automatic or automatic as defined in Table 4. The three system platforms for AWS, Oracle and IDE 10-step automation details are listed below.

Table 6. Provisioning Systems (AWS, Oracle and IDE)

Step No.	VM Provisioning Task	Oracle Step	AWS Step	IDE Step
1	Cloud Provisioning Access	Manual	Manual	Manual
2	Configure Role	Semi-Automatic	Automatic	Automatic
3	Select compute as the option for VM deployment	Semi-Automatic	Semi-Automatic	Semi-Automatic
4	Select the image you wish to use to install to the VM (usually the OS type/version)	Semi-Automatic	Semi-Automatic	Automatic
5	Select the VM CPU, Memory, and Disk Parameters	Semi-Automatic	Semi-Automatic	Semi-Automatic
6	Define VM Parameters	Semi-Automatic	Semi-Automatic	Automatic
7	Define VM Storage	Semi-Automatic	Semi-Automatic	Semi-Automatic
8	Add SSH Key, create a Private/Public key and upload the pubic key	Manual	Manual	Automatic
9	VM creation process	Automatic	Automatic	Automatic
10	Accessing the VM via the internet, or via a remote network	Manual	Manual	Automatic

The following weighting values described in Table 7 have been applied to the three types of automation method, manual, semi-automatic and automatic.

Table 7. Weighting Automation Values

Step Method	Weighting Value
Manual	10
Semi-Automatic	5
Automatic	1

2.3 Calculation of User Results

The CLR calculation for each user task results are derived in detail using the following formula.

Where R is the derived result
Where μ is the user input
Where s is simple
Where m is moderate
Where d is difficult
Where x is manual
Where y is semi-automatic
Where z is automatic

$$R = (\mu = (s \wedge x) \rightarrow (1 \times 10)) \vee (\mu = (m \wedge x) \rightarrow (3 \times 10)) \vee (\mu = (d \wedge x) \rightarrow (5 \times 10)) \vee (\mu = (s \wedge y) \rightarrow (1 \times 5)) \vee (\mu = (m \wedge y) \rightarrow (3 \times 5)) \vee (\mu = (d \wedge y) \rightarrow (5 \times 5)) \vee (\mu = (s \wedge z) \rightarrow (1 \times 1)) \vee (\mu = (m \wedge z) \rightarrow (3 \times 1)) \vee (\mu = (d \wedge z) \rightarrow (5 \times 1))$$

3 CLR Provisioning Results

As part of the experiments undertaken, three principle sets of results for the end user demographic were collected. These include expert, experienced and novice user groups (as defined previously in Table 1). Each user was observed, and the result for the 10-step VM Provisioning process are listed in Table 5 recorded; this provides the output for 3 sets of users listed in figure1, 2 and 3 respectively.

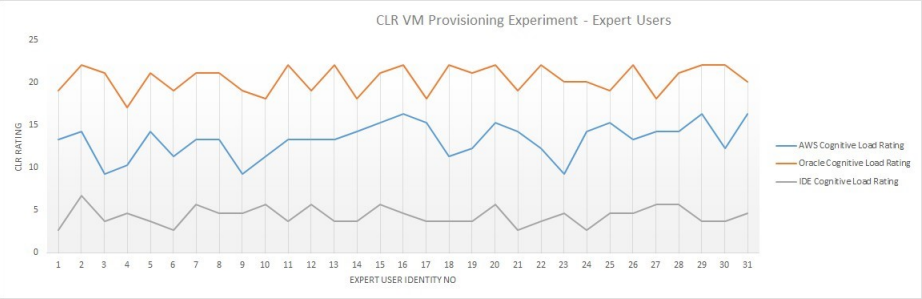

Fig. 1. CLR VM Provisioning Experiment – Expert Users.

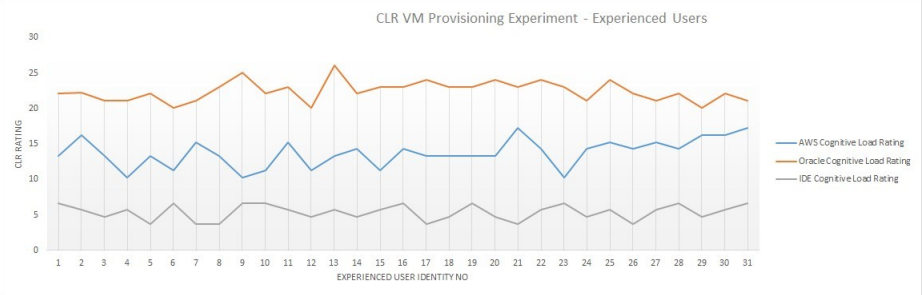

Fig. 2. CLR VM Provisioning Experiment – Experienced Users.

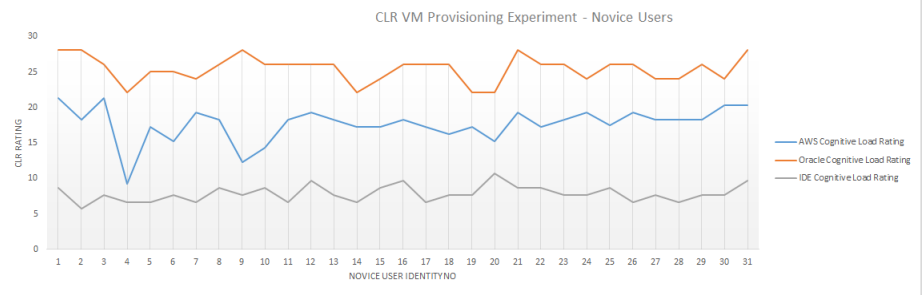

Fig. 3. CLR VM Provisioning Experiment – Novice Users.

As can be noted from the graph output results, the left-hand axis records the overall CLR for each user in their respective group, for provisioning a VM using the 10-Step process, for AWS, Oracle and the IDE. The bottom axis records the numerical identity of each user in that particular group.

4 Interpretation of CLR results

The following chart in Figure 4 provides guidance on how to interpret the CLR for users of a particular system. As discussed earlier, the idea within the field of CLT is to capture the mental energy or power required for a user to perform a certain process. The actual CLR experiment provides a unique method for doing this, and by design is agnostic to the system/process that it is used to measure. Therefore, the mechanism used as part of this experiment can be applied to any process or system.

Note, that the guide in Figure 4 provides insight on how much mental power is required for an end user, or group of users to use a system effectively. Interpreting the results is designed to be easy to understand, where a lower CLR score suggests that a system/process is easier for a user to perform or complete. The guidance categories fall into one of seven - either very-low (0-5), low (6-10), low-medium (11-15), medium (16-20), medium-high (21-25) and high (26-30) and very high (31+).

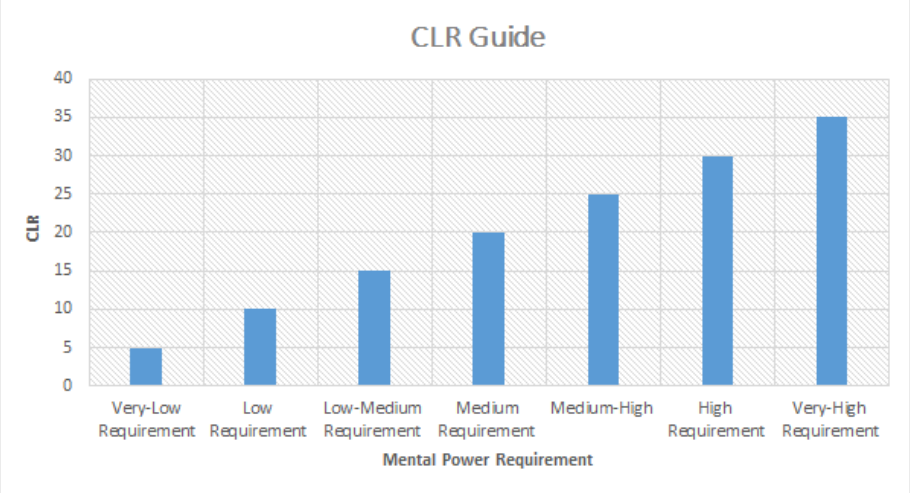

Fig. 4. CLR Guide – Mental Power Requirement

Based on the results in Figure 4, it consistently shows the same order of complexity for the Oracle, AWS and IDE systems. The order of complexity from high to low being Oracle, AWS, followed by the IDE as the simplest to use for end users to provision a VM using the 10-step sequence. Interestingly, it was observed that lowering the complexity for system usage is primarily based on the ability of making more steps in a process fully automated, therefore resulting the end to end process being considerably easier for the end users.

5 Conclusion

The data collected showed that of the three user groups, expert and experienced users had a similar set of results, despite the difference in technical/experience ability defined in Table 1; however, novice users showed an increased struggle to use the systems, and there was a noted increase in the CLR for this group. Lowering of the CLR for systems had a strong correlation to the number of tasks in the process that were fully automated. As highlighted in the study, any task in the process which was fully automated, had to be recorded as simple from the end user perspective, based on the fact that there is nothing for the user to actually do to progress that particular step towards the end objective. Therefore, it stands to reason that the CLR for users reduces dramatically as more steps are automated, and fewer human inputs are required. As a result of this study paper, it is the recommended that for those responsible for building any type of system consider carefully the following two principles to reduce the mental power/effort needed by end users to perform a process by:

1) Reducing the requirement for human inputs wherever possible
2) Enabling the automation of a process and its set of tasks

References

1. J. Sweller, Cognitive Load During Problem Solving: Effects on Learning, Cognitive Science, Vol 12, Issue2 (1988)
2. R. Yang, W. Wei, M. Cummins, Application of Cognitive Load Theory to the Design and Evaluation of Usability Study of mHealth applications: Opportunities and challenges IEEE International Conference on Healthcare Informatics (2017)
3. F. Paas, J. Merrienboer, Measurement of Cognitive Load in Instructional Research, Perceptual and Motor Skills, p79, 419-430, (1994)
4. F. Paas, J. Tuovinen, H. Tabbers, P. Gerven, Cognitive Load Measurement as a Means to Advance Cognitive Load Theory, Educational Psychologist, p.63–71, Lawrence Erlbaum Associates, Inc., (2003)
5. E. Kotova, Intellectual Support of the Learning Content Planning Considering the Cognitive Load, XIX IEEE International Conference on Soft Computing and Measurements, (2016)
6. S. Feinberg, M. Murphy, Applying Cognitive Load Theory to the Design of Web-Based Instruction, 18th Annual Conference on Computer Documentation Technology and Teamwork Proceedings, (2000)
7. S. Selvi, C. Valliyammai, V. Dhatchayani, International Conference on Recent Trends in Information Technology, Resource Allocation Issues and Challenges in Cloud Computing (2014)
8. E. Green, Can Qualitative Research Produce Reliable Quantitative Findings? Field Methods, Vol. 13, No. 1, February 2001 3–19, Sage Publications, Inc. (2001)
9. K. Srnka, S. Koeszegi, From Words to Numbers: How to Transform Qualitative Data into Meaningful Quantitative Results, Schmalenbach Business Review, Vol. 59, (2007)

10. S. Verdinelli, N. Scagnoli, Data Display in Qualitative Research, International Journal of Qualitative Methods, (2013)
11. J. Oakes, M. Johnson, J. Xue, S, Turner, Simplified Deployment of Virtual Machines using an Intelligent Design Engine, Proceedings of the SAI Computing Conference London IEEE, (2016)
12. I. Lokshina, R. Insinga, Expert Systems Supporting System Administrators Managing in a Distributed, Heterogeneous Environment, Joint IST Workshop on Mobile Future and the Symposium on Trends in Communications, (2004)
13. D. Menasce, M. Bennani, International Conference on Autonomic and Autonomous Systems ICAS06, Volume: 00, Issue: C, IEEE, (2006)
14. O. Rusu et al, Converting Unstructured and Semi-structured Data into Knowledge, 11th RoEduNet International Conference, IEEE, (2013)
15. X. Lui et al, Human Workload Monitoring Method Considering Qualitative and Quantitative Data fusion, Second International Conference on Reliability Systems Engineering, IEEE, (2017)
16. A. Austermann, S. Yamada, "Good Robot", "Bad Robot" – Analyzing Users' Feedback in a Human-Robot Teaching Task, Proceedings of the 17th IEEE International Symposium on Robot and Human Interactive Communication, Technische Universität München, Munich, Germany, August 1-3, (2008)
17. R. Franzosi, From Words to Numbers: Narrative, data, and social science, Cambridge University Press, (2004)
18. M. Penta, D. Tamburri, Combining Quantitative and Qualitative Studies in Empirical Software Engineering Research, IEEE/ACM 39th IEEE International Conference on Software Engineering Companion, (2017)