# A Human-centric and Environment-aware Testing Framework for Providing Safe and Reliable Cyber-Physical System Services

In-Young Ko, KyeongDeok Baek*, Jung-Hyun Kwon, Hernan Lira and HyeongCheol Moon

*School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea*
*E-mail: iko@kaist.ac.kr; kyeongdeok.baek@kaist.ac.kr; junghyun.kwon@kaist.ac.kr; lirahernan@kaist.ac.kr; hc.moon@kaist.ac.kr*
**Corresponding Author*

Received 01 December 2019; Accepted 14 April 2020; Publication 03 June 2020

## Abstract

The functions, capabilities, and effects produced by the application services of cyber physical systems (CPS) are usually consumed by users performing their daily activities in a variety of environmental conditions. Thus, it is critical to ensure that those systems neither interfere with human activities nor harm the users involved. In this paper, we propose a framework for testing and verifying the safety and reliability of CPS services from the perspectives of CPS environments and users. The framework provides an environment-aware testing method by which the efficiency of testing CPS services can be improved by prioritizing CPS environments and by applying machine-learning techniques. The framework also includes a metric by which we can automate the test of the most effective services that deliver effects from physical devices to users. Additionally, the framework provides a computational model that assesses mental workloads to test whether a CPS service can cause cognitive depletion or contention problems for users. We conducted a series of experiments to show the effectiveness of the proposed approaches for ensuring the safety and reliability of CPS application services during the development and operation phases.

**Keywords:** Service-oriented systems, Cyber-physical systems, Environment-aware testing, Service effects, Human cognitive resources.

## 1 Introduction

In recent years, the world wide web has evolved into a platform where various types of cyber and physical computing resources can be linked and utilized to provide user-centric services to help accomplish their goals. A *cyber-physical system (CPS)* is a system that connects and coordinates computational and physical resources to produce useful functionalities[23]. Various types of CPSs such as smart homes, autonomous vehicles, and smart factories are already have been assimilated into our daily lives. Essentially, the functionalities and actuations produced by a CPS application are physically employed by users during their daily activities. Therefore, such CPS applications must be safe and reliable so that they neither interfere nor harm humans.

Traditional software-testing techniques evaluate software by verifying source code using a set of test cases, which usually comprise inputs affecting the functional behavior of software and expected outputs. However, unlike traditional software, CPS applications may produce physical effects and actuation as outputs that can affect and can be affected by the status and conditions of users and surrounding environments. Therefore, to ensure the safety and reliability of a CPS application, new means of software testing are required that consider the contexts of users and the conditions of the environments.

Figure 1 shows an example CPS application that may result in severe accidents due to various environmental conditions. This application runs on

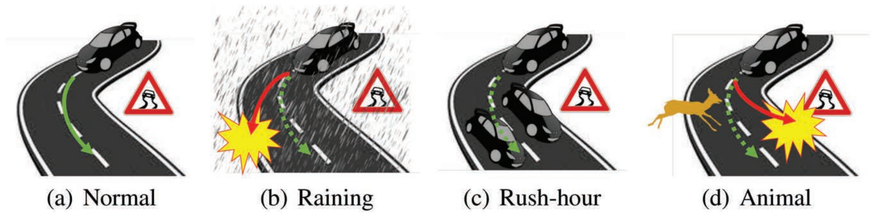


**Figure 1** Various Environmental Conditions for Auto-driving CPS Application.

an autonomous vehicle and provides CPS services such as autopilot and collision avoidance. Under normal road conditions (Figure 1(a)), the vehicle can make a turn successfully on a curved road by reducing its speed. However, in wet conditions (Figure 1(b)), the vehicle may fail to slow down enough and cause an accident if the application behaves the same as at the normal conditions. Furthermore, in a rush-hour (Figure 1(c)) or other exceptional circumstances such as animal crossings (Figure 1(d)), the collision avoidance service may not work properly.

In this work, as an extension to our previous work in [14], we propose a human-centric and environment-aware testing framework that tests safety and reliability of CPS applications according to the contexts of their users and environments that closely interact with the application. Our framework is built upon a service-oriented software architecture that we defined in a previous work [15], which allows developers to build CPS applications by defining and composing services as building blocks. We also consider DevOps practices [13] used to define testing processes and verification standards.

The main contribution of this work is the proposition of a framework for testing and verifying the safety and reliability of CPS services from the perspectives of CPS environments and users. The framework provides an environment-aware testing method by which the efficiency of testing CPS services can be improved by prioritizing CPS environments and by applying machine-learning techniques. The framework also includes a metric and an algorithm by which we can automate the test and selection of the most effective services that deliver effects from physical devices to users. Additionally, the framework provides a computational model that assesses mental workloads to test whether a CPS service can cause cognitive depletion or contention problems for users. We conducted a series of experiments to show the effectiveness of the proposed approaches for ensuring the safety and reliability of CPS application services during the development and operation phases.

In Section 2, we describe the service-oriented CPS framework. We explain the core elements of the environment-aware and human-centric CPS software testing framework, and required activities of developing CPS applications during DevOps phases. In Section 3, we explain the core testing techniques of our framework. The experiment results derived from the approaches developed for the testing framework are shown and discussed in Section 4. We explain related works in Section 5, and we conclude the paper in Section 6.

## 2 Service-oriented CPS Application Framework

CPS is a system of systems associated with diverse cyber and physical subsystems. Extant software engineering efforts related to CPS have been focused mostly on developing platforms or domain-specific applications. However, such platforms or domain-specific applications are lack of testing safety and reliability of CPS services in various real-world environments and users. In this work, we develop a service-oriented framework for development and operation of CPS applications, by taking advantages of service-oriented computing to solve the problems of improving safety and reliability of CPS services. Specifically, our framework consists of an architecture and a development process that are developed based on a common CPS architecture called the 5C architecture and the DevOps practices.

### 2.1 CPS Framework Architecture

The left side of Figure 2 shows the 5C architecture used to design CPS systems, providing connection, conversion, cyber, cognition, and configuration layers [17]. The connection layer includes various types of physical things (e.g., sensors and actuators) that are connected to network devices. At the conversion layer, the data obtained from the physical things are aggregated and converted into useful information that can be used to mine valuable

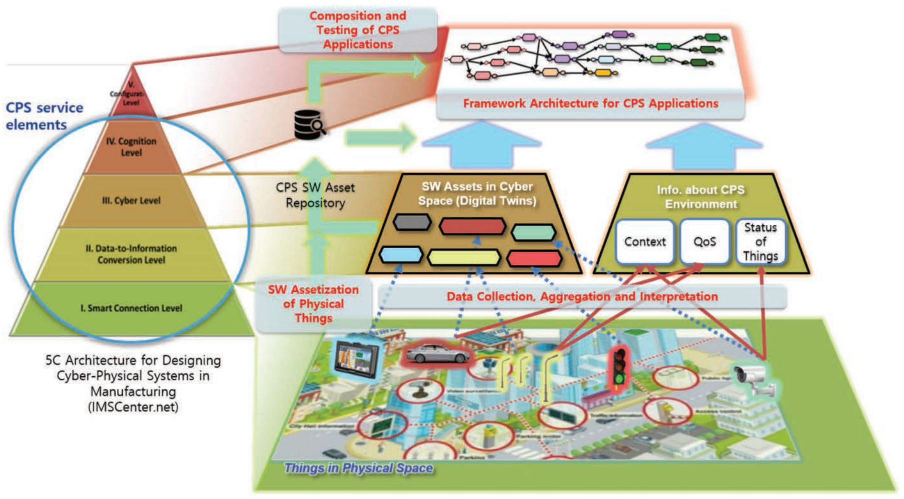

Composition and Testing of CPS Applications
Framework Architecture for CPS Applications
CPS service elements
V. Configuration Level
IV. Cognition Level
III. Cyber Level
II. Data-to-Information Conversion Level
I. Smart Connection Level
CPS SW Asset Repository
SW Assets in Cyber Space (Digital Twins)
Info. about CPS Environment
Context
QoS
Status of Things
SW Assetization of Physical Things
Data Collection, Aggregation and Interpretation
5C Architecture for Designing Cyber-Physical Systems in Manufacturing (IMSCenter.net)
Things in Physical Space

**Figure 2** Service-oriented CPS application architecture.

patterns and predict future trends and potential problems. This is done by maintaining and analyzing virtual representations (i.e., digital twins) of physical things at the cyber layer. The crucial patterns, trends, and problems can be visualized for human consumption at the cognition layer, and the system can be reconfigured to improve its capabilities and solve problems based on human decisions at the configuration layer. The CPS elements in the 5C architecture are usually heterogeneous in terms of their interfaces and functional/data semantics, and their availability and functional/data quality can change dynamically over time. To address these problems effectively, we developed a service-oriented CPS application framework based on the 5C CPS architecture.

In this framework, all physical and computational elements of the 5C architecture are servicized to make them accessible through standard service interfaces. Figure 2 shows the servicization process for the CPS elements. This enables the heterogeneous CPS elements to inter-operate effectively for an application. Usually, a service is modeled and defined from the user perspective in terms of achieving a task goal. In other words, the capabilities of CPS services are represented using common semantics that are closely related to user activities for achieving a task goal in a specific application domain. Each service provides its service capabilities by utilizing one or more physical devices. This helps application developers effectively choose and integrate relevant CPS services for a user task while overcoming the semantic heterogeneity of service functions and data. In addition, the framework provides the basis for ensuring the reliability and safety of CPS applications.

As shown in the right side of Figure 2, a CPS environment comprises various devices scattered across a physical space. Understanding and predicting the characteristics of physical environments and users is essential for generating and delivering high-quality service effects. In particular, the context or perception of a user in a CPS environment as well as the status or QoS of physical devices are critical to enabling the user to consume service effects from a physical environment.

## 2.2 DevOps Cycle of CPS Applications

Figure 3 shows the activities of composing, testing, monitoring, and reconfiguring CPS applications using our framework, which should be performed for each phase of DevOps. The left side of the Figure 3 shows the activities of development phases, and the right side of the Figure 3 shows the activities of the operation phases.

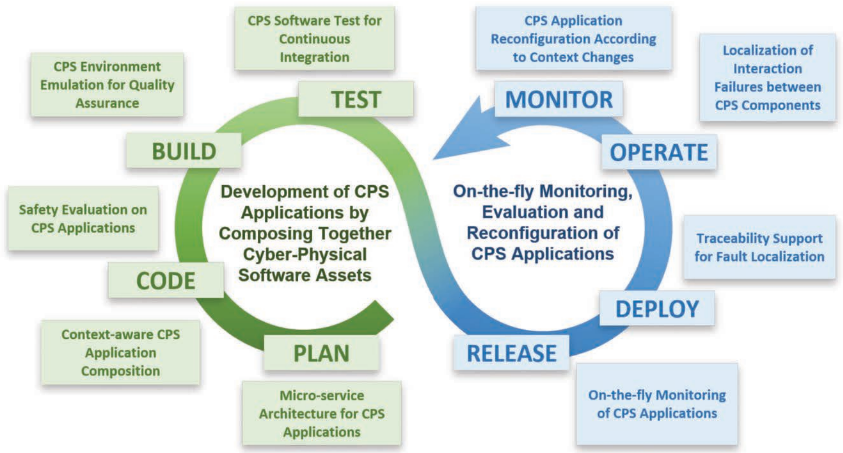

CPS Software Test for Continuous Integration
CPS Environment Emulation for Quality Assurance
TEST
BUILD
Safety Evaluation on CPS Applications
CODE
Context-aware CPS Application Composition
PLAN
Micro-service Architecture for CPS Applications
Development of CPS Applications by Composing Together Cyber-Physical Software Assets
CPS Application Reconfiguration According to Context Changes
Localization of Interaction Failures between CPS Components
MONITOR
OPERATE
On-the-fly Monitoring, Evaluation and Reconfiguration of CPS Applications
Traceability Support for Fault Localization
DEPLOY
RELEASE
On-the-fly Monitoring of CPS Applications

**Figure 3** DevOps cycle for developing reliable and safe CPS applications.

### 2.2.1 Activities of Development Phases

First, at the planning phase of the development of a CPS application, architecture based on micro-services should be designed. Micro-service architecture is a concept of dividing a software into a set of small sized services, so enables efficient reuse and effective team-based development.

Second, at the coding phase, context-aware CPS applications can be composed from the micro-services, a CPS application may be considered as a composition of micro-services. In here, context-awareness is necessary for CPS applications because they can be highly affected by physical contexts of surrounding environments. Furthermore, the safety of the CPS applications should be evaluated not to harm any humans, especially for the CPS applications that frequently interact with the environments and human users.

Third, at the building phase, the CPS environment should be emulated for the quality assurance of the CPS applications. Evaluating the quality of CPS applications under various environmental conditions is highly cost-expensive, so emulation of CPS environments should be done to evaluate the CPS applications efficiently before real-world evaluation.

Fourth, at the testing phase, the CPS applications and their internal micro-services should be tested in an environmental-aware manner, under continuous integration practice. Since continuous integration has been adopted as a common development principle in modern software developments, CPS software should be also continuously integrated and be tested. However, testing and assuring quality of CPS applications frequently is cost-expensive because there are usually too many environmental conditions to test the CPS applications.

### 2.2.2 Activities of Operation Phases

At the beginning of the operation phases, a developed CPS application is released and deployed. Because there are too many uncertainties that the CPS application may face over various environments, the CPS application should be monitored in an on-the-fly manner to collect unexpected failures and report those to the developers.

Second, for a reported failure of the CPS application, fault localization should be done to pinpoint the location of fault of the CPS applications. In detail, faults may be embedded not only in software but also in interaction design, such as human cognition for human-computer interactions and failure of service-service interactions.

Finally, a CPS application should be monitored continuously and being reconfigured when some components of the CPS application becomes unavailable according to fluctuating context of CPS environments. Such reconfiguration becomes more critical for CPS applications, because of highly dynamic nature of physical spaces.

### 2.2.3 Current Research Focuses

In this DevOps process, we initially focus on the efficient testing, dynamic reconfiguration of services, and predicting cognitive bugs.

First, we perform environment-aware regression testing for CPS applications, extending traditional regression testing techniques to the CPS domain. A CPS environment comprises many types of physical resources, such as IoT sensors and actuators from which service capabilities can be produced. Thus, a CPS application might need to interact with a different set of physical resources based on the location of the user and/or the availability of their quality-of-service. It is time-consuming to test CPS applications while considering the many combinations of physical resources available. Environment-aware CPS application testing is one major component of our framework. We aim to improve the efficiency of testing CPS applications by prioritizing their use cases while applying machine-learning techniques [16].

Second, we design an evaluation metric for quality of physical effects that CPS services generate. During operation phase, physical environments can affect the delivery of service effects produced from CPS resources. By using the metric we designed, quality of CPS services can be tested and the most effective CPS services may be selected by the user.

Third, the framework provides a computational model for assessing the types and numbers of cognitive user interferences based on a set of services from a CPS application applied to daily human activities. We define a *cognitive bug* as an issue of high-intensity cognitive interference that causes the interruption of the ongoing work or drastically decreased performance. Theoretically, our model is based on the multiple resources model [27] and human-processing system economy [21] from the cognitive psychology field. Using the models, we can test whether a CPS application might cause a cognitive bug, which could lead to an unstable or dangerous situation. In particular, we present the results of a theoretical cognitive model application using the ACT-R cognitive architecture: a framework built on cognitive psychology theories to simulate human behavior. Additionally, we apply threaded cognition theory [24] to explicitly model multitasking behaviors in the architecture. Furthermore, using this model, we estimate cognitive load and the amount of expected interference as a proxy for the estimation of cognitive bugs.

## 3 Human-centric and Environment-aware CPS Software Testing

The runtime environments of a CPS application can vary depending on the combinations of available elements in cyber and physical spaces. In addition, because the cognitive abilities of CPS users are diverse, leveraging their cognitive characteristics to select and integrate CPS services for user tasks is necessary. Thus, to ensure the safety and reliability of a CPS application, validating its functionalities and identifying runtime defects is necessary.

Figure 4 shows the overall process of providing CPS services to a user. First, the user sends a request to a CPS service that can provide a necessary content to the user. Second, the CPS service sends the content to a physical device, which is a medium for delivering the content to the user. Third, the device generates physical effects such as lights and/or sounds that correspond to the requested content. Fourth, the physical effects from the device are delivered to the user through the space. Fifth, the user perceives the physical effects and recognizes the delivered contents.

In the physical effect generation phase, various environmental conditions such as the types and characteristics of the service medium and other devices that may interfere with the medium can affect the quality of generating the physical effects. Therefore, prioritization of the test environments need to be done for efficient testing of CPS services with considering numerous

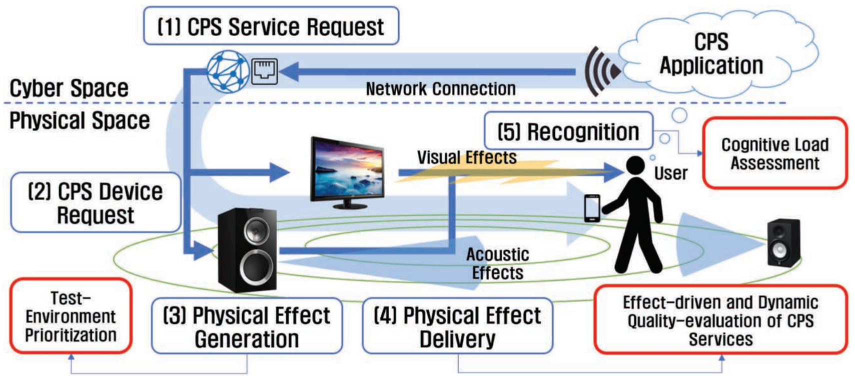

(1) CPS Service Request
CPS Application
Cyber Space
Network Connection
Physical Space
(5) Recognition
Cognitive Load Assessment
Visual Effects
User
(2) CPS Device Request
Acoustic Effects
Test-Environment Prioritization
(3) Physical Effect Generation
(4) Physical Effect Delivery
Effect-driven and Dynamic Quality-evaluation of CPS Services

**Figure 4** Process of CPS service provisions.

environmental configurations. Furthermore, in the physical effect delivery phase, the quality of delivering the service effects to the user needs to be evaluated to ensure the successful delivery of the service contents. In the recognition phase, the cognitive load of the user must be assessed and considered to avoid potential cognitive failures. We describe these testing techniques in detail in the following sub-sections.

## 3.1 Environment-aware Regression Testing for CPS Applications

Although testing a CPS application in a variety of environments is critical, a CPS application requires more testing effort than does traditional software. This is because CPS software interacts with physical devices, which can lead to abnormal operations, causing not only monetary losses but also user inconvenience and personal injury. Moreover, because a CPS application is operated via combination of multiple services, failure can occur because of service and application malfunction. In various CPS environments, CPS services can behave differently than indicated by specified requirements, which can lead to even more defects. In a DevOps environment where tests are frequently performed, the longer the testing duration, the longer the time required for information delivery to the developer. This results in delays fixing defects and in the release of new versions.

One manner with which to deal with the aforementioned problems is to perform regular regression tests. These tests should be cost-effective in a DevOps environment. Existing cost-effective regression testing techniques may not be appropriate for a DevOps environment, because the time between

**Table 1** Technique and code

| Technique | Code | Technique | Code |
|---|---|---|---|
| No ordering | B1 | Failure frequency-based | P3 |
| Random ordering | B2 | Machine learning-based | P4 |
| Exact match-based | P1 | P1 and P3 | H1 |
| Similar match-based | P2 | P1 and P4 | H2 |

regression tests is shorter. Furthermore, existing techniques require code analysis for each case, including code instrumentation and code coverage, which can consume considerable time. Thus, the preparation time for applying existing techniques may be longer than when conducting regression tests in a DevOps environment.

For cost-effective regression testing for CPS applications, test prioritization techniques are often used. These techniques involve prioritizing test cases, test suites, and test environments. In a DevOps environment, where regular integration practices are employed, prioritizing test cases and suites is not always cost-effective. Therefore, we adopt test-environment prioritization techniques [16]. These techniques utilize test-history data. In a prior study, we suggested the following five prioritization techniques: exact matching-based, similarity matching-based, failure-frequency-based, machine-learning-based, and hybrid. Table 1 lists the name of each technique and its code.

## 3.2 Effect-driven and Dynamic Quality-evaluation of CPS Services

As shown in Figure 4, a service can produce physical effects such as lights and sounds, in a CPS environment. When evaluating the quality of such a CPS service, the quality of the perceived effects from the users' perspective becomes more important. To address this issue, we suggested a new class of metric, named *service effectiveness*, which measures the quality of physical effects generated by physical devices in the user's perspective, such that the physical effects can be delivered to a user through a physical space and perceived appropriately by the user [3].

In our previous work, we proposed a computational model of visual service effectiveness that could measure how visual contents generated by rectangular display devices could be perceived by a user, especially in case of textual contents [3]. As shown in Figure 4, the visual CPS services that we consider in this work are the CPS services that utilize display devices to generate visual effects and deliver textual contents to users. We designed the visual service effectiveness model based on the domain knowledge of human vision systems, such as Snellen [26] and Bailey-Lovie charts [4], which are the most popular tests applied to human visual acuity. Our visual service effectiveness model was defined as a rule-based model as follows:

$$e = \begin{cases} 0 & distance(user, device) \geq distance_{max}(device) \\ 0 & device \notin FoV(user) \\ 0 & angle(user, device) < 90^{\circ} \\ 1 & otherwise \end{cases}, \quad (1)$$

where $distance(user, device)$ is the physical distance between the user and the display device, $FoV(user)$ is the field-of-view area of the user according to the orientation of the user's eyes, and $angle(user, device)$ is the angle between two orientation vectors of the user and the display device [3]. The intent of the first rule is to ensure that the distance between a user and a device should be close enough to read textual contents, and the threshold of the distance can be estimated based on the domain knowledge of human visual acuity measurement. The intent of the second rule is to ensure that the device should be located in the user's field of view, so that the user can perceive the light effect generated by the device. The field-of-view area can be estimated from the orientation of the user's eyes and their location. The intent of the third rule is to ensure that the device faces the user, such that the light generated by the device can be delivered to the user. When those three rules are met, the effectiveness is 1. An effectiveness of 0 indicates that one of the rules has been violated.

By using the visual service effectiveness model, we can evaluate the quality of visual CPS application during the operate and monitor phases of the DevOps cycle (Figure 3) to detect the problems that users might experience while interacting with CPS applications via service effects. When detecting a problem of delivering physical effects to the user, the CPS application can be reconfigured to use an alternative display device in the operation time.

### 3.3 Finding Cognitive Bugs in CPS Applications

In CPS environments, the manner in which the functionalities and effects produced by a CPS application affect user task accomplishment is relevant. We assume that, in these environments, users perform multiple tasks at the same time (i.e., multitasking). In particular, we can deepen the concept of user performance by studying the human cognitive demands required during human computer interaction (HCI). Although it seems that people use applications effortlessly while performing other activities during their daily lives, the cognitive demands required in CPS environments can hinder their correct use because of the limited human cognitive capacity [25]. Cognitive interference arises because the user must divide his/her attention and distribute cognitive resources to each task [24]. Eventually, high-intensity interferences lead to task errors. Otherwise, they render it necessary to stop ongoing tasks. This is a *cognitive bug* problem.

We developed an approach for analyzing the combination and/or sequence of application features that can minimize the number of cognitive bugs during runtime and development time. Thus, we turned theories from cognitive psychology into a computational model. In particular, we focused on the study of cognitive demands with respect to app usage and performing physical activities, including walking and conversing. For this type of testing, applying standard software testing techniques is not possible. Instead, models must be developed that emulate application usage during the early stages of the software process. The use of cognitive resources must be assessed, and defining guidelines to improve the safety of the application design must be produced. In addition, using this computational model, we can create tools to help developers design apps considering combinations of use cases and physical activities that can be produced in a real-life scenario. Then, by understanding cognitive demands we can raise new software requirements.

In our previous research [18], we developed an experimental design in which users performed six mobile application tasks on a smartphone using Gmail and Tripadvisor applications. Users were simultaneously required to interact with their environments by performing physical activities, such as walking and talking. This experimental design was based on the multiple resources model (MRM) of C. Wickens [27], which established the mechanisms of cognition for employing cognitive resources in different scenarios (i.e., during multitasking). Using the MRM model enables us to theoretically determine the numbers and intensity of cognitive interference and to establish different levels of mental demand.

Towards the development of a model of cognitive bug prediction, we simulate the execution of app tasks and physical activities. Cognitive architectures are used to simulate human behavior during different types of tasks [2]. In particular, the ACT-R architecture has proven effective at estimating cognitive loads during the early stages of system design [2]. For this study, we adopted a fusion of the approach proposed by Salvucci et al. [24] (i.e., ACT-R in multitasking) and that of Park et al. [22] (i.e., estimation of cognitive load).

The simulation is performed using the same tasks described in the previous section. The simulation proceeded as follows. First, we decomposed the tasks into several task units with the aim of increasingly modeling the specific goals of each task. Each task unit was then decomposed into a number of functional-level goals comprising a number of keystroke-level goals, including the sequence of attention shifts across a screen, the encoding of the information on the screen, movement of the finger to the desired location, and pressing the screen with the finger. For instance, if we want to model a reading activity, such as reading an e-mail, we need to express it in terms of functional-level goals in which the minimal chunk was reading a single word. Then, it was necessary to iterate the reading of the next word, modeled for the rest of the tasks with variations in the types of the ACT-R architecture modules involved. In addition, we simulated arithmetic tasks following the model proposed by [22]. To simulate the experiment scenario, we simplified the complete flow of tasks, assuming that all participants performed the same actions in the same order.

The assumptions of the simulation are as follow: cognitive load is defined as the ratio of the required resources to the available resources; each module of ACT-R causes a different amount of cognitive load, because it represents a different cognitive process; and errors in each module cause a higher cognitive load [22]. We modeled multitasking while aggregating both simultaneous tasks using the threaded cognition approach, in which a task must wait for a specific cognitive resource before its execution if the same cognitive resource is currently being utilized [24]. In case the task waits for time $t > threshold$, a cognitive interference is produced. This cognitive bug prediction method can be used during the code, build and test phases of the DevOps cycle (Figure 3) to predict and prevent the cognitive conflict and depletion problems that users may face with when they receive CPS application services while performing activities in physical environments.

## 4 Evaluation

### 4.1 Testing Environment Prioritization

To evaluate the proposed environment-aware testing techniques, we conducted an experiment using an application called JQuery, which is a widely used JavaScript library, while it can be installed on CPS devices that support JavaScript engines. We chose JQuery because many of the Web applications that use JQuery are deployed and run on various environments that can be characterized by the types and versions of Web browsers, operating systems, and underlying physical devices. There is usually a large number of combinations of these environmental elements, and as we discussed earlier, it is crucial to test these applications in a cost-effective manner. In addition, JQuery can be installed directly on various CPS devices that support JavaScript engines.

We collected the build history of JQuery from JQuery's TestSwarm[1]. 45 builds that included the dates of the build history (ranging from 2017/12/13 to 2018/09/07) were contained in the history data. For each build history, regression tests were performed for an average of 16 environments. Each test environment consisted of a web browser and an operating system, both based on name and version. 42 builds included at least one failed test environment and at least one passed test environment, meaning that at least one environment-specific failure in a build was possible. We used these 42 builds in the experiment.

We measured the cost effectiveness of each test-environment prioritization technique by using $APFD_C$, which is a widely used measure in test-case prioritization studies [6]. We regarded a test environment as a test case when a test-environment prioritization technique was evaluated. The $APFD_C$ equation is as follows:

$$APFD_C = \frac{\sum_{i=1}^{m}(f_i \times (\sum_{j=C_i}^{n} t_j - \frac{1}{2}t_{C_i}))}{\sum_{i=1}^{n} t_i \times \sum_{i=1}^{m} f_i}, \tag{2}$$

where $n$ is the number of test environments, $m$ is the number of failed test environments in a build, $f_i$ is the severity of a failed test environment, $t_i$ is the runtime when performing a test for an environment, and $C_i$ is the position at which failure $i$ is detected in several test environments. $APFD_C$ values range from 0 to 100, where a higher $APFD_C$ means improved cost effectiveness. We assigned the same value (e.g., 1) to the severity, because we did not have information about it.

Figure 5 shows an experimental result. The X axis refers to the codes of the techniques, described in Table 1. The Y axis refers to $APFD_C$. Our techniques outperformed the baseline techniques with no ordering or random ordering. The $APFD_C$ of our test-environment techniques ranged from 85.05% to 91.70%, whereas the $APFD_C$ of the baseline techniques ranged from 47.18% to 49.76%. Compared to no ordering and random ordering, improvement rates ranged from 37.87% to 44.52% and from 35.29% to 41.94%, respectively.

[1]http://swarm.jquery.org/

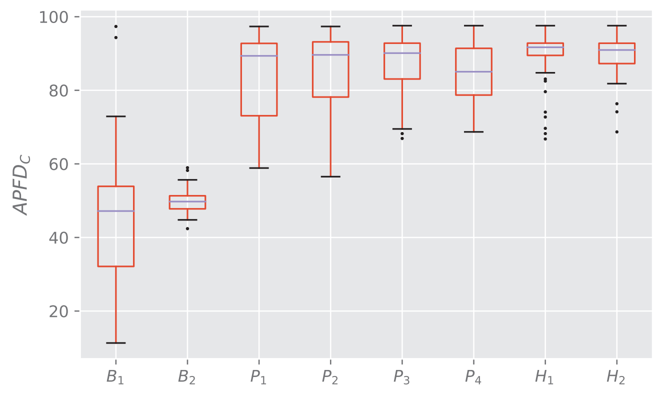


**Figure 5** Cost-effectiveness of the environment-aware testing techniques.

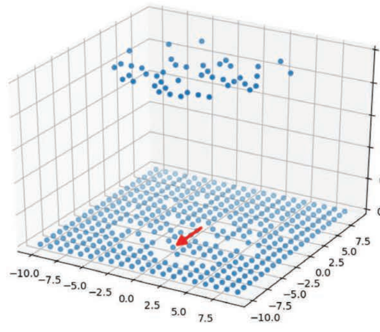
**Table 2** Parameters of visual service effectiveness model

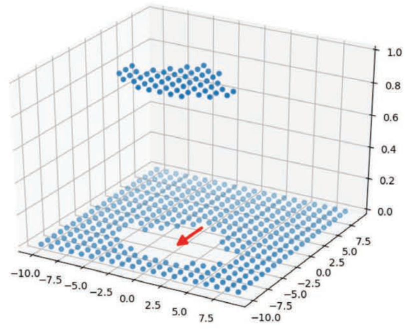
| Parameter | Value |
|---|---|
| Resolution of Display Device | 1080p (1920 x 1080) |
| Size of Display Device | 1 m |
| Minimum Size of Text | 12 pixel |
| Field of View | 105° |
| Orientation Threshold | 60° |
| Visual Acuity of User | 6/6 (standard) |

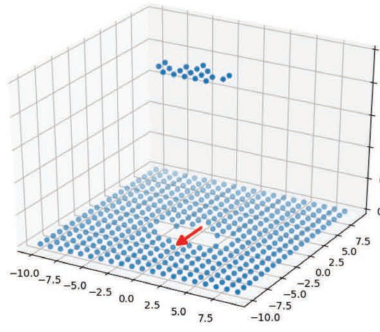
## 4.2 Visual Service Effectiveness Metric

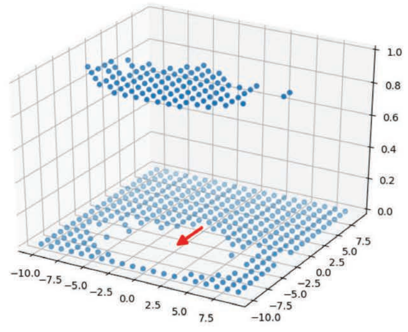
We defined the visual service effectiveness model to evaluate how well the physical effect that is generated by a visual CPS service via a rectangular display device can be delivered to a user. Figure 6 shows the plots of service effectiveness of a user in front of a display device, calculated by using our visual service effectiveness model. A display device is located at the center (i.e., (0, 0, 1) coordinate) and virtual users at positions in a range from $(-10, -10, 1.7)$ to (10, 10, 1.7). We set the z-axis coordinate of users as the common height of human. Table 2 shows the values we used for parameters of the calculation of visual service effectiveness.

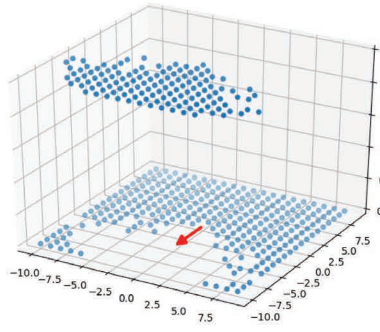
Figure 6(a) shows the calculated values of visual service effectiveness based on our model when we set the orientation of users randomly. The red arrow shows the orientation of the display device, and the result shows that the delivery of visual effects is likely to be effective in front of the display device, matching our experiences. As Figure 6(a) shows, the value of service effectiveness is almost one when we located virtual users within a random orientation. Figure 6(b) shows the calculated values of visual service


(a) Normal


(b) Face Toward Display


(c) Small Text


(d) Higher Visual Acuity


(e) Larger Display

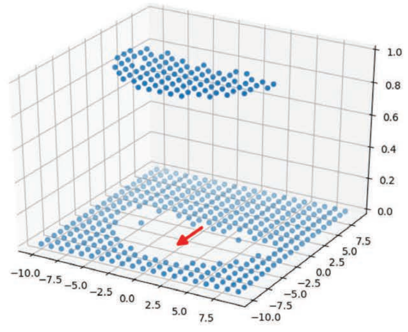

(f) Higher Orientation Threshold

**Figure 6** Calculated visual service effectiveness in various conditions.

effectiveness when we set the orientation of each virtual user toward the device, making visualization more successful to understand the behavior of our visual service effectiveness model.

Compare to Figure 6(a), Figure 6(b) shows the effective area of the visual service more clearly, assuming that the user pays attention to the display device. Figure 6(b) indicates that a user at 5m away from the display device can recognize the textual content. When the size of textual content reaches half, the effective area of the visual service gets smaller, as shown in Figure 6(c). In contrast, when the visual acuity of a user was set higher, the effective area of the visual service becomes larger as a user with a higher visual acuity can recognize the letter at a farther distance, as shown in Figure 6(d). Similarly, when the size of the display device was set larger, the effective area of the visual service becomes larger as shown in Figure 6(e), which matches our intuition from experiences. We also could adjust threshold values of each rule in our visual service effectiveness model. For instance, when we enlarged the threshold value of orientation (3rd) rule of the model, the effective area of the visual service became wider, as shown in Figure 6(f). This enabled us to adjust threshold values based on real-world data for future works. The visualized results show that calculation of service effectiveness based on our visual service effectiveness model generally matches our intuition from experiences.

## 4.3 Cognitive Load Assessment

In human-centric CPS applications, accomplishing tasks is highly dependent on user performance [8]. In our previous work [18], we addressed one of the main causes of low user performance (i.e., multitasking). From cognitive psychology, we know that during multitasking, a user's cognitive resources (i.e., core assets used by cognition to perceive, think, remember, make decisions, and respond to the environment [21]) are intensive, and this causes these resources to interfere with each another. Several studies have shown that weak cognitive interferences can degrade user performance by 30%, and strong interferences can stop an ongoing task. To avoid this type of inconvenience, in [18] we proposed a model to assess and classify mental workloads: explicit mental efforts required to employ cognitive resources.

To collect data, we employed physiological sensors, surveys, and a video recorded by each user. In our experiment, we asked total 50 participants to perform five tasks that are about using a messaging app and a web search engine: (1) read a simple e-mail that requests for simple mathematical

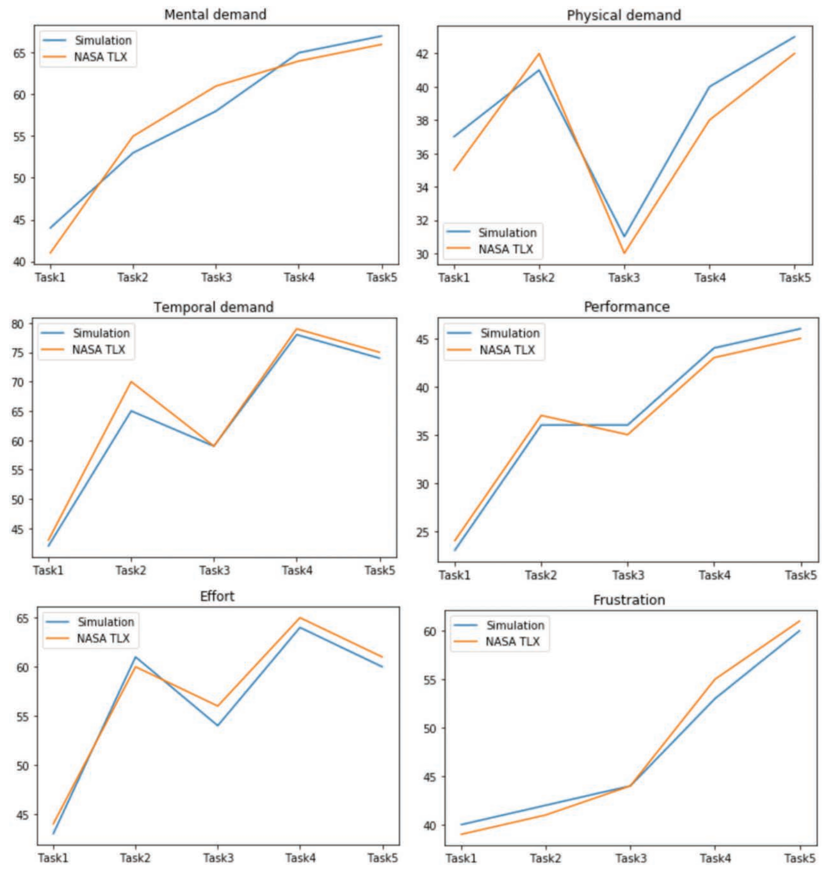


**Figure 7** Comparison of simulation results and NASA TLX responses.

calculations, (2) reply to the simple e-mail with the calculation results, (3) read an information request e-mail and search the information, (4) memorize the search result, and (5) reply to the request e-mail. For the first experiment, we asked the participants to perform the tasks without doing anything else. For the second experiment, we asked the participants to perform the same set of tasks while answering some questions and listening to music.

In addition to data collection, we simulated the mental workload required by each task using the cognitive framework ACT-R [2]. We validated that the data simulated by ACT-R could be correlated with the real data collected by the sensors. This represented a major result, because it showed that we could test CPS application features in the early stages of the developmental process when no prototype of the system existed.

**Table 3** Statistics of simulation results for each source of cognitive load

| **Sources** | $R^2$ | RMS | p-value |
|---|---|---|---|
| Mental Demand (MD) | 0.899 | 4.514 | 0.0335 |
| Physical Demand (PD) | 0.947 | 2.854 | 0.0364 |
| Temporal Demand (TD) | 0.908 | 4.905 | 0.0412 |
| Performance(PE) | 0.939 | 3.944 | 0.0223 |
| Effort (EF) | 0.951 | 2.986 | 0.0402 |
| Frustration (FR) | 0.943 | 3.261 | 0.0118 |

The results of the simulation are shown in Figure 7. First, we compared ACT-R results with NASA TLX [10] experimental results following the methodology proposed in [22]. NASA TLX is a widely used questionnaire for mental workload estimation. As indicated in Table 3, Pearson correlation analysis indicates that the simulation was highly correlated with the NASA TLX dimensions. Additionally, we checked that the number of interferences increased when the both tasks were modeled.

Second, we utilized the data from physiological sensors and ACT-R to create a machine-learning model (i.e., support vector machine) to classify levels of mental workload. We placed special emphasis on the feature selection process, when we developed an algorithm that used both the aforementioned sources of data to improve the selection of features (CFS-SVM). The purpose of this algorithm was to increase the chances of selecting features from signal values that approximated simulation values. With this model, we were able to classify two levels of mental workload having a 93.1% of accuracy on average for all tasks tested. We compared this result with a standard classification model that simply uses recursive feature elimination (RFE-SVM) to select features. Figure 8 shows the results of both models against five different tasks.

Although results were good for this scenario, and by using this approach we could solve our problem of simulating cognitive bugs during early stages of design, in practice, ACT-R modeling is highly dependent on the task, in which a manual configuration is needed for task representation and parameter fixation. Therefore, this approach is suitable for a customized type of scenario.

## 5 Related Work

Various studies have focused on testing mobile applications [1, 12, 11]. Because mobile applications can be run on a variety of devices, device prioritization techniques have been proposed. Additionally, "energy-bug" detection, referring to code that abnormally consumes device battery power,

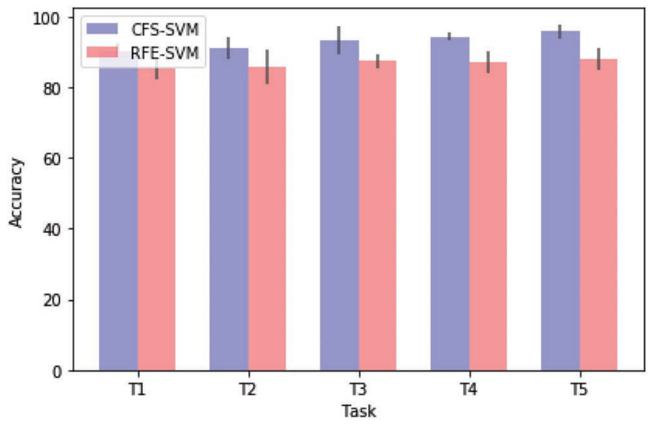


**Figure 8** Comparing Models for different tasks.

has been studied. Our study considers other environments, such as web browsers, operating systems, and devices for testing CPS applications. Service-related regression testing techniques have also been studied [28, 20]. They have proposed regression-test selections and test-case prioritization techniques for unit services. Other studies have considered test-case prioritization for a combination of services (e.g., business processes). Our work focuses on testing services or a composite thereof under various test environments. We use test-environment prioritization techniques instead of test-case prioritization to identify environment-specific faults as early as possible.

Studies focusing on human cognition processes can be divided into empirical estimation methods of mental workload, which collect data from several types of sensors, and analytical estimation methods, which simulate mental processes based on computational cognition frameworks. Concerning models using physiological signals for mental workload estimation, Haapalainen et al. [9] assessed tasks of visual perception and cognitive speed. Fritz et al. [7] studied tasks performed by professional software developers using an eye tracker, electrodermal activity sensor and an electroencephalogram sensor. Regarding analytical models, Salvucci et al. [24] used the ACT-R cognitive architecture [2] to model several tasks performed at the same time. Park et al. [22] used this architecture to model human performance, obtaining highly correlated values among the simulation and data collected using the widespread NASA TLX [10].

## 6 Conclusion

A CPS application helps users accomplish tasks successfully by providing a set of necessary functionalities and/or by producing physical effects required to assist user activities. The behaviors of such CPS applications are highly affected by surrounding environments and users. Therefore, it is critical to improve reliability and safety of CPS applications from development through operation phases. In this study, we developed a software-testing framework that ensures the reliability and safety of CPS applications that coordinate the various cyber-physical elements interacting in the users' environments and activities.

Our contributions are summarized as follows. First, to the best of our knowledge, this is the first study to propose a service-oriented CPS application framework by which CPS resources in different layers of the 5C architecture can be integrated while overcoming heterogeneity problems. Second, to improve the efficiency of testing CPS applications, we applied a regression testing technique that prioritizes various combinations of CPS environments. Third, we investigated a means of evaluating and selecting CPS services based on the effectiveness of delivering service effects to users through physical spaces. Finally, we developed an approach for analyzing different types and amounts of cognitive resources required by CPS applications and human activities, and we proposed a method of identifying cognitive bugs that can cause cognitive interference and overload.

We are currently developing a service-oriented CPS application framework that employs micro-service technologies and a set of tools that can be used for the DevOps activities. Our environment-aware software testing technique can be extended to practical CPS application domains and then further evaluated. We have plans to perform evaluation by using simulations in virtual reality, which can imitate real-world environments while providing realistic experiences in terms of vision and audio [5, 19]. For instance, the validity of our visual service effectiveness model can be evaluated by simulating visual-service provision scenarios in virtual reality under various environmental conditions. We could then compare the calculated effectiveness via user feedback. Additionally, the models and algorithms for human-centric CPS testing will be enhanced by identifying and incorporating diverse situations and service effects that are directly and closely related to CPS applications.

## Acknowledgment

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science and ICT (NRF-2017M3C4A7066210)

## References

[1] Domenico Amalfitano, Anna Rita Fasolino, and Porfirio Tramontana. A gui crawling-based technique for android mobile application testing. In *Software testing, verification and validation workshops (icstw), 2011 ieee fourth international conference on*, pages 252–261. IEEE, 2011.

[2] John R Anderson, Michael Matessa, and Christian Lebiere. Act-r: A theory of higher level cognition and its relation to visual attention. *Human-Computer Interaction*, 12(4):439–462, 1997.

[3] KyeongDeok Baek and In-Young Ko. Effect-driven dynamic selection of physical media for visual iot services using reinforcement learning. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 41–49. IEEE, 2019.

[4] Ian L Bailey and Jan E Lovie. New design principles for visual acuity letter charts. *American journal of optometry and physiological optics*, 53(11):740–745, 1976.

[5] Jim Blascovich, Jack Loomis, Andrew C Beall, Kimberly R Swinth, Crystal L Hoyt, and Jeremy N Bailenson. Immersive virtual environment technology as a methodological tool for social psychology. *Psychological Inquiry*, 13(2):103–124, 2002.

[6] Sebastian Elbaum, Alexey Malishevsky, and Gregg Rothermel. Incorporating varying test costs and fault severities into test case prioritization. In *Proceedings of the 23rd International Conference on Software Engineering*, pages 329–338. IEEE Computer Society, 2001.

[7] Thomas Fritz, Andrew Begel, Sebastian C Müller, Serap Yigit-Elliott, and Manuela Züger. Using psycho-physiological measures to assess task difficulty in software development. In *Proceedings of the 36th international conference on software engineering*, pages 402–413. ACM, 2014.

[8] Levent Gurgen, Ozan Gunalp, Yazid Benazzouz, and Mathieu Gallissot. Self-aware cyber-physical systems and applications in smart buildings

and cities. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*, pages 1149–1154. IEEE, 2013.

[9] Eija Haapalainen, SeungJun Kim, Jodi F Forlizzi, and Anind K Dey. Psycho-physiological measures for assessing cognitive load. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 301–310. ACM, 2010.

[10] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[11] Reyhaneh Jabbarvand, Alireza Sadeghi, Hamid Bagheri, and Sam Malek. Energy-aware test-suite minimization for android apps. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*, pages 425–436. ACM, 2016.

[12] Hammad Khalid, Meiyappan Nagappan, Emad Shihab, and Ahmed E Hassan. Prioritizing the devices to test your app on: A case study of android game apps. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 610–620. ACM, 2014.

[13] Gene Kim, Jez Humble, Patrick Debois, and John Willis. *The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution, 2016.

[14] In-Young Ko, KyeongDeok Baek, Jung-Hyun Kwon, Hernan Lira, and HyeongCehol Moon. Environment-aware and human-centric software testing framework for cyber-physical systems. In *2nd International Workshop on Maturity of Web Engineering Practices (MATWEP 2019*. International Conference on Web Engineering, 2019.

[15] In-Young Ko, Han-Gyu Ko, Angel Jimenez Molina, and Jung-Hyun Kwon. Soiot: Toward a user-centric iot-based service framework. *ACM Transactions on Internet Technology (TOIT)*, 16(2):8, 2016.

[16] Jung-Hyun Kwon, In-Young Ko, and Gregg Rothermel. Prioritizing browser environments for web application test execution. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 468–479. IEEE, 2018.

[17] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18–23, 2015.

[18] Hernan Lira, In-Young Ko, and Angel Jimenez-Molina. Mental workload assessment in smartphone multitasking users: A feature selection approach using physiological and simulated data. In *2018*

*IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 639–642. IEEE, 2018.

[19] Xiao Ma, Megan Cackett, Leslie Park, Eric Chien, and Mor Naaman. Web-based vr experiments powered by the crowd. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 33–43. International World Wide Web Conferences Steering Committee, 2018.

[20] Lijun Mei, Zhenyu Zhang, WK Chan, and TH Tse. Test case prioritization for regression testing of service-oriented business applications. In *Proceedings of the 18th international conference on World wide web*, pages 901–910. ACM, 2009.

[21] David Navon and Daniel Gopher. On the economy of the human-processing system. *Psychological review*, 86(3):214, 1979.

[22] Sungjin Park, Sungoo Jeong, and Rohae Myung. Modeling of multiple sources of workload and time pressure effect with act-r. *International Journal of Industrial Ergonomics*, 63:37–48, 2018.

[23] Cyber-physical systems. https://www.nsf.gov/pubs/2008/nsf08611/nsf08611.htm. accessed January 25, 2019.

[24] Dario D Salvucci and Niels A Taatgen. *The multitasking mind*. Oxford University Press, 2010.

[25] Amitai Shenhav, Sebastian Musslick, Falk Lieder, Wouter Kool, Thomas L Griffiths, Jonathan D Cohen, and Matthew M Botvinick. Toward a rational and mechanistic account of mental effort. *Annual review of neuroscience*, 40:99–124, 2017.

[26] Herman Snellen. *Probebuchstaben zur bestimmung der sehschärfe*, volume 1. H. Peters, 1873.

[27] Christopher D Wickens. Multiple resources and mental workload. *Human factors*, 50(3):449–455, 2008.

[28] Ke Zhai, Bo Jiang, and WK Chan. Prioritizing test cases for regression testing of location-based services: Metrics, techniques, and case study. *IEEE Transactions on Services Computing*, 7(1):54–67, 2014.

## Biographies

**In-Young Ko** is an associate professor in the School of Computing at the Korea Advanced Institute of Science and Technology (KAIST) in Daejeon, Korea. He received his Ph.D. in computer science from the University of Southern California (USC) in 2003. Prof. Ko's research interests include services computing, web engineering, and software engineering.

**KyeongDeok Baek** is currently a Ph.D. candidate in the School of Computing at the Korea Advanced Institute of Science and Technology (KAIST) in Daejeon, Korea. He also got his bachelor's degree in the School of Computing, KAIST, Korea. His research interests is on solving service-oriented computing problems in Internet of Things (IoT) domain, by using reinforcement learning techniques.

**Jung-Hyun Kwon** received his Ph.D. degree in the School of Computing at the Korea Advanced Institute of Science and Technology (KAIST) in Daejeon, Korea, in 2019. He is currently a researcher in Korea Telecom, South Korea. His research interests include software engineering, AI-based enterprise applications, and network management systems.

**Hernan Lira** received his Master's degree in the School of Computing at the Korea Advanced Institute of Science and Technology (KAIST) in Daejeon, Korea. He got his bachelor's degree at University of Chile. His research interests include cognitive engineering and AI systems.

**HyeongCheol Moon** received his Master's degree in the School of Computing at the Korea Advanced Institute of Science and Technology (KAIST) in Daejeon, Korea. He also got his bachelor's degree in the School of Computing, KAIST, Korea. His research interests is on Vehicle-to-Everything, Internet of Things, and services computing.