Personal and Ubiquitous Computing (2022) 26:1413–1444
https://doi.org/10.1007/s00779-020-01508-x

ORIGINAL PAPER

Modeling and "smart" prototyping human-in-the-loop interactions for AmI environments

Miriam Gil[1] · Manoli Albert[2] · Joan Fons[2] · Vicente Pelechano[2]

Received: 13 May 2020 / Accepted: 15 December 2020 / Published online: 8 January 2021

Abstract
Autonomous capabilities are required in AmI environments in order to adapt systems to new environmental conditions and situations. However, keeping the human in the loop and in control of such systems is still necessary because of the diversity of systems, domains, environments, context situations, and social and legal constraints, which makes full autonomy a utopia within the short or medium term. Human-system integration introduces an important number of challenges and problems that have to be solved. On the one hand, humans should interact with systems even in those situations where their attentional, cognitive, and physical resources are limited in order to perform the interaction. On the other hand, systems must avoid overwhelming the user with unnecessary actions. Therefore, appropriate user-centered methods for AmI development should be used to help designers analyze and design human-in-the-loop interactions in AmI environments. This paper presents a user-centered design method that defines a process with a set of tools and techniques that supports the process steps in order to systematically design, prototype, and validate human-in-the-loop (HiL) solutions. The process starts with the definition of the HiL design, which defines how the system cooperates with the human. This HiL design is built using a conceptual framework that focuses on achieving human-system interactions that get human attention and avoid obtrusiveness. Then, we provide a software infrastructure to generate a prototype based on the HiL design and validate it by having end-users use a web simulator. The feedback data generated during the prototype user validation is gathered and used by a machine learning tool that infers the user's needs and preferences. Finally, these inferences are used to automatically enhance the human-in-the-loop designs and prototypes. We have validated the proposed method through a twofold perspective: an experiment to analyze the perception of interaction designers regarding their acceptance of the design method and another experiment to evaluate the usefulness of the "smart" prototyping technique. The results obtained point out the acceptability of the proposed method by designers and the useful adaptations provided by the "smart" prototyping technique to achieve a HiL design that adapts well to users' preferences and needs.

Keywords Human in the loop · Human-system interactions · Context-aware interactions · Human-centered design · Smart prototyping · Machine learning

1 Introduction

Autonomous capabilities are required in the next generation of systems in order to adapt themselves to new environmental conditions and situations. This is especially relevant in Ambient Intelligence (AmI) environments (e.g., from smart objects to smart nations), where "smart" systems should adapt at runtime to changing user needs and intentions, new types of devices, new technologies, and new services [1]. However, the diversity of systems, domains, environments, context situations, and social and legal constraints all point to the need for going beyond "smart-only," technology-driven ubiquitous instrumentations and installations and keeping the human in the loop and in control of such systems [2].

As discussed in [2], the term "ambience" is an important aspect of the concept of "ambient intelligence," which stresses the environmental character of these systems and the experiences that they evoke. The concept of ambience in AmI

✉ Miriam Gil
miriam.gil@uv.es

[1] Escola Tècnica Superior d'Enginyeria, Departament d'Informàtica, Universitat de València, Avenida de la Universidad, s/n, 46100 Burjassot, Valencia, Spain

[2] Centro de Investigación en Métodos de Producción de Software, Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Valencia, Spain

implies that computation becomes unobtrusively integrated into objects and spaces and that the integration of humans with systems should not be disturbing. This human-system integration introduces an important number of challenges and problems to be solved [3]. Some of the challenges are inherent to the complexity of human beings. Humans should interact with systems even in those situations where their attentional, cognitive, and physical resources are limited in order to perform the interaction. This leads to the challenge of managing the user's attention in order to get the human involved when required and help him/her to maintain a suitable level of attention. At the same time, the system must avoid overwhelming the user with unnecessary actions or information that can require too much attention or cause undesired results or bad user experiences. However, good interaction designs are very difficult to develop especially when properties related to quality, such as obtrusiveness, must be taken into account together with the need for acquiring the correct level of human attention. A recent accident involving an Uber self-driving test car in Arizona illustrates these challenges. The accident was due to an improper collaboration between the human and the system. The car setup required that the human not be distracted while the car was driving autonomously, and it did not achieve getting the human's attention when his participation was required [4].

To confront the aforementioned challenge, appropriate user-centered methods and strategies for AmI development should be used to help designers analyze and design human-in-the-loop (HiL) interactions. Research in the field of Ambient Intelligence has highlighted the relationship between Artificial Intelligence (AI) and AmI and has studied their role in the advancement of the AmI vision [5, 6]. Furthermore, it is well known that AI is leading a revolution in software. The rise of AI may significantly transform the practice of software engineering (SE), helping us to build better software faster. AI can accelerate the lifecycle of software development, facilitate rapid prototyping, build intelligent programming wizards, analyze and manage errors automatically, improve code quality (through automatic refactoring), etc. In summary, AI can be used to support more efficient software development and construction processes.

Taking these aspects into consideration, this work provides a user-centered solution for helping designers to design, prototype, and validate human-in-the-loop interactions in AmI systems at the early stages of the software lifecycle. The proposal starts with a technique for modeling user engagement in AmI systems that defines both the control strategies for human-system integration and the human-system interactions that are required to specify how the system operates with the human. This technique is built under the premise of managing human attention in a way that achieves human-in-the-loop interactions that get the human's attention and avoid obtrusiveness. In [7, 8], the authors introduced a conceptual framework to model human-in-the-loop interactions, which is consolidated in this paper. Based on this conceptual framework, the proposal is extended with a software infrastructure that helps designers to easily and automatically obtain a "smart" prototype from the human-system collaboration design. The prototype is characterized as "smart" since it uses a machine learning technique that leverages the feedback that users provide during the prototype validation to infer the users' preferences. Moreover, the prototypes are endowed with autonomous reconfiguration capabilities that allow them to adapt themselves automatically to the inferred users' preferences and to refine the human-system collaboration design accordingly. Among the most important properties of AmI systems, we find that they are often tailored to the user's needs [6]. Therefore, the human-in-the-loop interactions are enhanced to fit the needs and preferences that users exhibit through the use of the prototypes. The improvements in the designs are related to changes in the attention level of interactions, which range from using more intrusive interactions to using unobtrusive interactions. To tie the different techniques that we introduce in this approach together, we propose a method that guides the entire process.

We illustrate our approach by applying it to the domain of autonomous cars in which there are some situations that require human-in-the-loop interactions. Recent advances towards autonomous driving have been hindered by failures (e.g., not recognizing speed-limit signs or being fooled by so-called scam stickers); therefore, humans are necessary to be able to achieve safe driving. The design of human-in-the-loop interactions in an autonomous car requires a user-centered approach to better suit the drivers and achieve trustworthy systems and a good user experience.

The remainder of the paper is organized as follows. Section 2 presents the state of the art. Section 3 introduces the method overview, which provides insight about how we face the challenge of designing, prototyping, and validating human-in-the-loop solutions. Section 4 presents the conceptual framework to model the user engagement in AmI systems. Section 5 describes a "smart" prototyping technique for designers to validate human-system collaboration designs with users in order to use their feedback to improve the designs. Section 6 introduces an experiment to analyze the perception of interaction designers regarding their acceptance of the design method. Section 7 presents an experiment performed to evaluate the usefulness of the "smart" prototyping technique. Finally, Section 8 presents conclusions and further work.

2 State of the art

A lot of work has been done on characterizing the interaction that should be established between humans and systems that

automatize tasks [9]. These automation capabilities shift the role that humans play in task control, from the role of planning, executing, and monitoring to a supervisory role [10]. The roadmap to achieving fully autonomous systems involves a progressive shift in the control and supervision by humans and machines [10]. Other works have tried to identify problems that are related to human automation interactions. For example, [11, 12] deal with the human mistrust problem caused by delegating the control responsibilities to the systems. These works encourage the development of autonomous systems that keep the human informed, that provide him/her with feedback, and that reduce cognitive responsibilities and stress as well as keeping the human with "situational conscience" of the automated processes. They argue the need for designing solutions that satisfy these requirements in order to achieve wide acceptance of autonomous systems by humans.

The problems that can arise from considering the human as an active participant in a shared or switched control loop have also been studied in the literature. Some proposals such as [13–15] provide a partial solution (focusing on execution tasks) to reason about the integration of the user in the self-adaptation processes. However, none of them provides mechanisms or techniques to specify how human integration and participation should be. In [16], Cranor proposes a framework for reasoning about human-in-the-loop in secure systems. It provides a systematic approach to identify potential causes for human failure. However, it is domain dependent (only applicable on secure systems). Nothwang et al. in [17] investigate the contributors to success/failure in current human-autonomy integration frameworks and propose guidelines for the safe and resilient use of humans and autonomy with regard to performance, consequence, and the stability of human-machine switching. This is still an unresolved issue in the work on the human-machine collaboration.

As stated in [18], the characteristics of the new software systems force a change in the way of understanding the human-system interaction. In these new software systems, humans and systems collaborate together in the performance of certain tasks. Thus, it is necessary to change the focus in the design of the interactions towards a holistic conception of the collaboration between the human and the system in order to achieve systems that integrate the human in a natural way. The natural integration of humans with systems is especially relevant in AmI environments, where the computation becomes non-obtrusively integrated into everyday objects and spaces [2]. Consequently, the physical environment becomes a medium for supporting interaction between the user and the AmI functionality; hence, the interface becomes spatialized [19]. The problem with these interfaces is that users are often not fully aware of the interaction options provided in an AmI environment. From this problem, some authors introduced the concept of "affordances" as the set of possible actions [20, 21]. Wrong designs can lead to having a lack of information (hidden affordances) or wrong information (false affordances) [22]. To face this challenge in developing human-in-the-loop interactions, it is important to do some kind of rapid prototyping in a way that designers can early find out the users' preferences [23]. In this work, we propose a "smart" prototyping technique that allows users to validate the interactions designed and learn from their feedback to self-adapt the designs and the prototypes automatically.

Furthermore, AI and AmI are thoroughly connected as stated in [5, 6]. The authors argue that we humans are satisfied if the advanced technologies take care of us and do not want our attention or even additional activities besides giving occasional guidelines and orders. This leads to an outstanding problem when dealing with humans participating in AmI environments, which is the need for adapting the human-system communication to what the user needs or prefers. This is a key aspect to be achieved when designing human-system interactions in AmI systems. AI can help AmI to face this challenge. More than two decades ago, in [24], Langley explored the potential of machine learning in helping to build personalized user interfaces. The author concluded that machine learning plays a central role in personalizing interfaces to the user's needs. Different AI techniques have been studied throughout the following two decades to build human-interactive systems specifically for each particular user. In [25], the authors build user-adaptive models for activity and emotion recognition. These models are predictive models that adapt to each user's characteristics and behaviors with reduced training data. The authors applied deep transfer learning and data augmentation. In this work, we use machine learning techniques to infer the users' feedback and adapt the prototypes to the users' needs and preferences.

3 Method overview

One of the main challenges of "keeping the human in the loop and in control" is: How much attention is required from users to do some tasks? How much data can users process? Is all the information relevant for users? [2, 26]. In some cases, it might be useful for a system not to ask for the user's feedback and confirmation for every single step in an analysis, diagnosis, and action chain because this would result in an information and processing overload. However, finding the trade-off between getting human attention and not overwhelming users is a tricky issue. This leads to the challenge of *managing the user's attention* when designing the human in the loop in order to:

- Get human attention when required and maintain a suitable level of attention

- Avoid overwhelming the user with unnecessary actions or information

Furthermore, there is a clear need for integrating humans into systems in a way that takes into account their preferences and needs [6]. Users will likely have different preferences in the use of the interaction mechanisms or interaction devices than the designers. It is necessary to adapt the designs to the users' preferences and needs in terms of the interaction mechanisms used and interaction devices.

Our proposal aims to cover the lack of engineering approaches to design and develop the human-system collaboration required to achieve the system's goals [27] in AmI environments. To do so, we provide an agile method to design, prototype, and validate human-in-the-loop solutions in the early stages of development. This method allows designers to model the human-in-the-loop interactions focusing on managing the user attention and then validate the modeled interactions by means of "smart" prototyping. The "smart" prototyping is applied to automatically and dynamically adapt HiL solutions to the users' preferences and needs. Specifically, the method proposes the use of machine learning and autonomous reconfiguration capabilities in order to extract knowledge about the users' preferences and needs from the use of the prototypes and based on them to adapt the prototypes and the human-system collaboration designs. Performing this kind of "smart" prototype validation with users provides valuable feedback that can be used early in the design process to achieve designs that offer good user experience. A key point of our design method is the consideration of obtrusiveness as a key factor together with the need for acquiring the correct human attention to involve the humans.

Figure 1 shows the method overview of the proposal. The lifecycle of a human-in-the-loop design starts with the following steps:

1) Design how the human should participate in the system (Design HiL Interactions of Fig. 1)

This is usually a time-consuming step, which in practical settings often relies on the knowledge of domain experts. In order to help designers with this specification, our approach provides a conceptual framework with the main concepts required to specify how the human collaborates with the AmI system. An important aspect of the conceptual framework is the consideration of human attention as a first-class concept. In this way, designers and domain experts only have to take these concepts into account to define the human-system collaboration.

Even though we provide designers with a technique to specify human-system collaboration, additional techniques for rapid prototyping, validating, and then enhancing the specifications in an iterative way are needed. To fill this gap, we provide a software infrastructure that allows designers to:

2) Automatically generate prototypes from the designs (obtain HiL prototype of Fig. 1)

These generated prototypes are early samples of HiL solutions that focus on representing how humans and the system work in a collaborative way. We also provide a simulation infrastructure to execute the prototypes and validate the designed human-in-the-loop interactions. By means of this simulation infrastructure, users can:

3) Validate the HiL prototype (validate HiL prototype of Fig. 1)

Furthermore, the software infrastructure is endowed with a machine learning technique and reconfiguration capabilities that allow the system to:

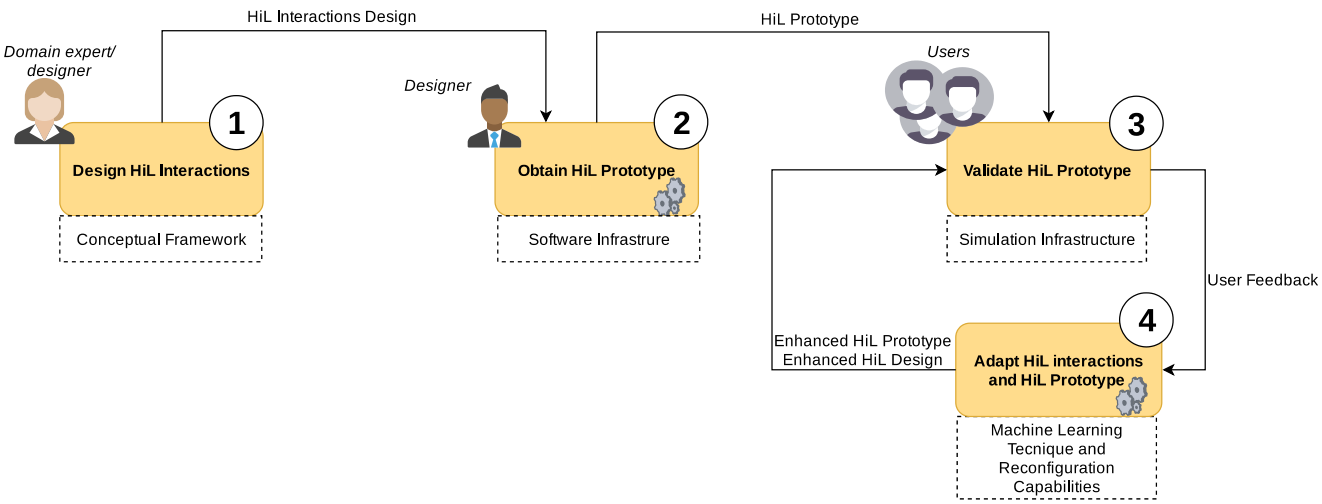


Fig. 1 Method overview of the proposal

4) Leverage this feedback and adapt the human-system collaboration design and the prototype according to the users' feedback (adapt HiL interactions and HiL prototype of Fig. 1)

In this way, the design and the prototype are enhanced automatically until they fit the users' preferences. The smart prototyping validation allows designers to know the users' preferences. After the validation process, designers should evaluate the designs that have been obtained for each user and build the final HiL design.

Finally, it is also important to highlight that the obtained prototype can be used as input for the development phase to evolve it into the target human-in-the-loop system.

4 Conceptual framework to design human-in-the-loop interactions

This section introduces the conceptual framework that we propose to deal with the design of human-in-the-loop interactions for AmI systems. The conceptual framework (1) identifies and defines the concepts that are required to describe the main aspects of a HiL solution and (2) defines the behavior semantics that underpins the operationalization of a human-system collaboration solution. This framework is an evolution of the work presented in [7].

4.1 Main concepts

4.1.1 The root concept: collaborative task

What a human and a system are required to do in a collaborative way in order to complete a particular functionality is represented by the concept of **collaborative task**. For example, in the case of the autonomous car, a collaborative task is the work done by the system and the human when the system transfers control of the car to the human in a situation that does not require hurry (e.g., the car is approaching a city, and it cannot continue driving autonomously). This transition of control from the autonomous system to the driver is called *takeover* according to [28]. We are using the *Takeover* task as a collaborative task example throughout the section.

4.1.2 Who can participate in the task?

A collaborative task requires a human with specific capabilities, knowledge, and background in order to be executed. These human features are represented by the concept of **human profile**. The humans that fit the profile/s associated with the task are the only ones that can assist in performing that task. For example, the *Takeover* task is associated with the *driver* profile, meaning that only a person with driving capabilities (he/she has a driving license) can help in performing that task. Humans can participate in tasks by playing different roles. They can perform actions to provide information or identify situations in the context of the running task (acting as sensors), or they can incorporate input into the decision-making process to provide better insight about the best way of performing that task, or they can even be responsible for managing, executing, and leading the task execution and control (totally by themselves or assisted by the system).

4.1.3 When can the task be executed?

An appropriate context situation is required to execute a task. This context situation can be considered as the **precondition** of the task. The precondition represents the context in which the system, its environment, and the human are prepared to execute the task under appropriate conditions in order to achieve proper task performance. For example, to maximize the success of a safe takeover, the volume on the radio should not be high, the human should pay attention to the driving task, he/she should be in the driver seat, and his/her hands should take the wheel. These conditions constitute the precondition of the *Takeover* task.

At this point, the question about what this context means arises. We adopt the semantics of the context concept introduced by Dey in [29]: "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is relevant to the interaction between a user and an application, including the user and applications themselves." From this definition, it is possible to derive the relevant properties of the context as the state of the physical environment (e.g., temperature, noise, etc.), the state of the resources regarding the system surroundings, the state of the system itself, and the user's situation (e.g., its location, activity, etc.). However, in this approach, the situation or context of the user acquires special importance in determining whether the task can be executed under appropriate conditions. Therefore, the "opportunity-willingness-capability" (OWC) model [30] is used to define the required characteristics of the human factors. The OWC model classifies human factors into three categories: *Opportunity*, which identifies the set of variables that humans need to fulfill in order to attempt a task (e.g., human.location or human.hands); *Willingness*, which indicates the human predisposition to perform the task (e.g., human.attention or human.stressLevel); and *Capacity*, which defines the human skills and abilities that are necessary to successfully execute the task (e.g., human.experience or human.hasSmartPhone).

4.1.4 Decomposing tasks

Collaborative tasks can be decomposed into more fine-grained units called **HiL actions**. Each HiL action represents a

perceivable interaction that is performed either by the system or by the human, or an internal function carried out by the system. Actions are inter-related to determine a partial order in which they must be processed. For example, the *Takeover* task can be decomposed into three actions that are executed sequentially:

- *Notify takeover*: the system notifies the human that he/she is required to take over control of the car.
- *Confirm takeover*: the human confirms the takeover.
- *Transfer control*: the system transfers the control to the human.

These three actions are the **core actions** that allow the human and the system to collaborate and to perform the task.

In order to build trust in AmI systems, the systems must include a number of essential actions that are oriented towards enhancing the human's confidence. This can be done by including explanations about what and why the system operates as it does. Therefore, we propose collaborative tasks that include actions that are oriented to providing explanations to humans so that they properly receive knowledge to collaborate. These actions are called **feedback actions**. For example, for the *Takeover* task, we include the feedback action:

- *Inform about driving mode*: the system informs the driver about the new driving mode.

This action should be executed once the human takes control of the car.

4.1.5 Setting a favorable context for human participation

In addition to the actions that perform the task functionality (the core and feedback actions), collaborative tasks include actions that aim at preparing the system, the environment, and the human to maximize the success of the task. These actions are oriented to achieving the fulfillment of the precondition. We call these actions **preparatory actions**. Thus, when a task initiates, this task will execute the preparatory action/s to achieve a proper context for executing the task. Consider again the *Takeover* task. When the system runs that task, if the human has no situational awareness, or he/she is in the passenger seat, or his/her hands do not take the wheel, or the volume on the radio is too loud, the three actions defined above for the task are insufficient to guarantee a safe takeover (the human is not likely to be aware of the notification about taking control of the car). The context situation requires additional actions in the task to correct it. Therefore, the task may include four preparatory actions:

- *Pay attention*: the system warns the human to pay attention.
- *Move to the driver's seat*: the system asks the human to move to the driver's seat.
- *Take the wheel*: the system asks the human to take the wheel.
- *Turn down the radio volume*: the system turns down the radio volume.

Preparatory actions are executed before core actions.

4.1.6 Context-dependent interactions

Actions can be dependent on a context condition that establishes whether it is necessary to execute that action. We call this type of context condition an **execution condition**. For example, consider the feedback action we have defined for the *Takeover* task: *Inform about driving mode*. This action is likely not required if the human is an expert in autonomous driving; therefore, this action should not be always executed. The execution condition is defined upon context factors. By taking into account the context to set the actions to be executed within a task, we avoid annoying the human with notifications that are not necessary and thereby avoid obtrusiveness.

4.1.7 Managing errors, exceptions, or abnormal situations

A task can have a **constraint** associated to it. Constraints are conditions that must always be fulfilled during the execution of tasks. They describe a required system state or a limit on a task execution (e.g., an execution time limit). For example, in the case of the *Takeover* task, it has the constraint *before 10 s* associated to it because if the human does not take control before 10 s, the system should cancel the task.

Having human-in-the-loop poses challenges that are related to the unexpected behavior of humans. Therefore, the design of tasks with human integration requires a safeguard, or a **fallback plan**, to be executed when the task is not working as planned. When executing a task, a fallback plan can be launched due to:

- The non-presence of a human with the specific human profile.
- The incapacity to achieve the appropriate context situation (in fact, this occurs when the appropriate context situation is not accomplished within the limits forced by the constraint).
- The execution of actions is not as expected (i.e., it does not fulfill some constraint).

Fallback plans are, in turn, tasks. Some of the fallback plans are also autonomous tasks that may or may not require human integration. In the example of the *Takeover* task, the

fallback plan *Drive the car to a minimal risk condition* is specified with a constraint of 10 s, which indicates that this fallback plan must be executed if the task is not performed within that time limit.

4.1.8 Managing user attention

Actions usually entail an interaction between the human and the system. The actions that involve an interaction should be designed to manage human attention resources in order to get the human's attention while at the same time avoiding overwhelming the human. The specification of human attention resources required for the actions serves as a guide for interaction designers in the selection of the most suitable concrete interaction mechanisms. Actions that are associated with a high attention demand will require concrete interaction mechanisms that are more noticeable or more intrusive, while a low attention demand will require mechanisms that are less noticeable or less intrusive.

The HCI field has proposed solutions to manage human attention based on the levels of automation and the workload [31–33]. We analyzed and designed the actions from two points of view:

- Who initiates the interaction (the system or the human)
- The attentional resources needed to assist the interaction (the attentional demand imposed on the human)

We consider this specification of **initiative** and **attention** (attentional demand) to be appropriate since these are factors that vary independently. For example, an interaction of feedback (initiated by the system) may require more attention or less depending on the importance/criticality of the information to be offered. In the same way, a preliminary interaction could be initiated by the system (if it tries to capture the user's attention) or by the human (if the system waits for an answer from the human to confirm that he/she is prepared). Therefore, establishing an initiative and an attention level helps designers in selecting the most suitable interaction mechanisms. The intersection of the initiative and attention level constitutes what we call the **obtrusiveness level**, which determines the level of human involvement required by an action. Each HiL action is associated to an obtrusiveness level, for example:

- ObtrusivenessLevel(*NotifyTakeover*) = (system, high attention)
- ObtrusivenessLevel(*ConfirmTakeover*) = (human, slight attention)
- ObtrusivenessLevel(*TransferControl*) = (system, low attention)
- ObtrusivenessLevel(*InformAboutDrivingMode*) = (system, slight attention)

4.1.9 Selection of concrete interaction mechanisms

Defining the obtrusiveness level of an action helps interaction designers to select the most appropriate interaction mechanism to support that action; the interaction designer should choose the concrete interaction mechanisms for each HiL action based on this obtrusiveness level. Obtrusiveness levels that require more attention must be associated with interaction mechanisms that are more obtrusive or that draw more attention. Conversely, obtrusiveness levels that require less attention must be associated with discreet interaction mechanisms. It is important to note that actions initiated by the system (system initiative) will be supported by output interaction mechanisms and that actions initiated by the human (human initiative) will use input interaction mechanisms.

Due to the nature of AmI environments, interaction is not confined to one device, but it should encompass multiple physical devices. Therefore, interaction mechanisms that are domain dependent are described by means of:

- The interface element that supports the interaction modality (e.g., a button, a vibration, a screen, etc.)
- The computing device that offers the interaction mechanism (e.g., a smartphone, a smart car, etc.)

For example, concrete interaction mechanisms can be a visual message on the head-up display of a smart car or a text message on the screen of a smartphone. It is worth noting that we include the computing/physical device that offers the interaction mechanisms in the specification of the interaction mechanisms. This is useful in AmI systems where the interaction mechanisms are in different computing devices. In our example, we only use interaction mechanisms of the car, but we could use interaction mechanisms of a smartphone such as mobile speakers or notification messages.

Continuing with the example of the autonomous car domain, several systems under research have been described in the literature that identify methods that warn drivers in order to improve situational awareness. These methods include vibration on the steering wheel [34] or the driver's seat [35], audio alerts, and visuals on the head-up display [36, 37]. From these studies, Table 1 defines the possible interaction mechanisms of the car device that can be used for each obtrusiveness level.

In order to specify which interaction mechanisms support a certain action for a given obtrusiveness level, we use the Superimposition operator ⊙. This operator takes an action and an obtrusiveness level and returns the set of concrete interaction mechanisms required for that action. Some examples of the relationship between the obtrusiveness levels and the actions for the *Takeover* task are the following:

⊙NotifyTakeover(system; high attention) = {Car.speakers.voice_feedback, Car.head-up_display.textual}

Table 1 Concrete interaction mechanisms for each obtrusiveness level

Obtrusiveness level	Interaction mechanisms
(System, high attention)	Car.speakers.voice_feedback, Car.head_up_display.textual
(System, slight attention)	Car.head_up_display.image, Car.speakers.beep, Car.steering_wheel.vibration
(System, low attention)	Car.head_up_display.icon Car.driver_seat.vibration
(Human, high attention)	Car.microphone.voice_recognition, Car.head_up_display.textual_input
(Human, slight attention)	Car.steering_wheel.pressure,
(Human, low attention)	Car.cam.gesture, Car.cam.body_recognition

⊙ConfirmTakeover(human; slight attention) = {Car.steering_wheel.pressure}

⊙InformAboutDrivingMode(system; slight attention) = {Car.head-up_display.image, Car.Speakers.beep}

The selection of the concrete interaction mechanisms that best suit each obtrusiveness level can be based on existing multimodal design taxonomies and frameworks that depend on the cognitive characteristics of interaction modalities, i.e., how information carried by different modalities is perceived by the human perceptual-sensory system [39–41, 42]. For example, an image uses less mental workload than a text, and visual-auditory combinations impose less cognitive load than visual combinations [39]. Auditory mechanisms are useful for attention alerting [43], and vibration or lighting feedback does no interrupt other activities [44].

Figure 2 presents a metamodel that describes the concepts used to specify HiL designs, which have just been introduced. The MOF standard[1] is used to build the metamodel, which describes the concepts and their relationships.

4.2 Operationalization of human-system collaboration

To run a human-in-the-loop solution that is designed using the proposed concepts, it is necessary to know the behavior semantics that is associated to these concepts. This subsection introduces how to operationalize the abstractions we have proposed. To do this, we define the sequence of steps that must be performed to execute a collaborative task and that indicates how to operationalize the concepts introduced. The sequence of steps is the following:

Step 1. Trigger the collaborative task. The system is continuously monitoring its environment to identify situations in which it should take an action. If it detects that a collaborative task needs to be executed, it raises an event that triggers the need for human help to carry out that task and goes to Step 2.

Step 2. Check the availability of the human profile. Before executing a collaborative task, the presence of a human that fits the human profile required for that task must be verified:
• If the human exists, the system continues the execution of the collaborative task and goes to Step 3.
• If the human does not exist, the system must perform a fallback plan and goes to Step 5.

Step 3. Check the fulfillment of the precondition. The context conditions that determine the suitability of the context for proper human collaboration are verified:
• If the context conditions are fulfilled, it means that the context is suitable for continuing the execution of the collaborative task, and the system goes to Step 4.
• If some context conditions are not fulfilled, it means that the context is not optimal to perform the task execution. The corresponding preparatory action is executed, and the system returns to Step 3.

Step 4. Execute the actions. The system must perform the actions according to its specification.
• The core and feedback actions are executed according to the established order. When all of the actions are executed, the task is completed, and the system returns to the autonomous mode.

Step 5. Launch the fallback plan if something is not working as planned. The system executes a fallback plan or alternative actions that leave the system in a safe state when a human profile does not exist or a constraint is violated. Then, the system returns to the autonomous mode.

The system is continuously checking the fulfillment of the constraints during Steps 2–4. These steps must be

[1] https://www.omg.org/mof/

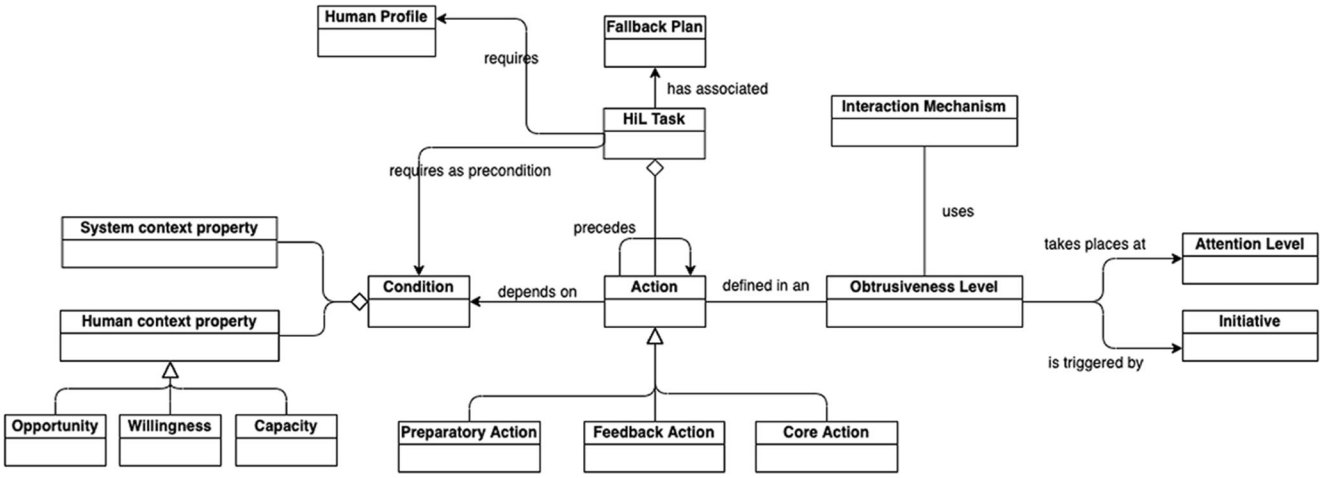


Fig. 2 Metamodel that represents the concepts proposed to build HiL models

carried out without violating the task constraints (temporary, contextual, etc.). The system is responsible for checking them at any moment during the execution of the HiL actions. If these constraints are not satisfied at any time, the fallback plan must be executed, and the system goes to Step 5.

These steps constitute the execution model that specifies the behavior of the elements introduced in the metamodel. This execution model is used as the basis to implement the behavior of the collaborative tasks of HiL solutions.

5 Human-in-the-loop "smart" prototyping

Gathering feedback from users through prototype validation is a crucial element in all human-centered design processes. On that basis, our approach proposes the use of prototypes in order for designers to validate human-system collaboration designs with users and to use their feedback to achieve improved designs.

As introduced in the above section, HiL designs focus on the specification of the shared control and interactions required to perform the collaborative work between humans and the system. Therefore, our prototypes, which we call HiL prototypes, are representations of those elements. Specifically, a HiL prototype is a way for designers to validate the flow of interactions, the attentional resources assigned to these interactions, and the feedback/information provided to the human. These prototypes also allow different context settings to be simulated to check the different behavior of these aspects according to each context. This kind of validation does not require an in situ evaluation in a real usage context with a fully functional prototype. A minimum infrastructure for simulating the execution environment is enough to represent the aspects that enable designers to check whether the design is suitable before other parts of the system are defined.

User validation of the prototypes provides valuable user feedback that must be analyzed. Designers must pay specific attention to how users interact with the prototype in order to figure out the users' preferences and needs and adapt the designs according to them. Thus, for example, if designers identify that users are not aware of some actions when participating in a specific task, this may lead to a change in the design of the task that entails increasing the attentional resources demanded by that task.

In traditional prototype user validation, designers observe the users interacting with the prototype and collect the users' feedback. Then, based on this feedback, they change both the design and the prototype (and begin the validation again). Our approach proposes doing this cycle "smartly" using (1) supervised machine learning, which allows the users' perception of the obtrusiveness from the HiL prototype usage to be extracted automatically, and (2) autonomous reconfiguration capabilities, which allow the prototype to be adapted automatically at runtime from the knowledge extracted using machine learning. Smartly doing this cycle allows (1) performing the iterations of the prototype validation faster and (2) changing the prototype implementation automatically according to the user's feedback. To do this, we propose a "smart" prototype validation process that is composed of four steps and a prototyping infrastructure that contains a set of components to support them (see Fig. 3):

- **Step 1: Generating the HiL prototype.** We provide a **software framework** (aka the HiL framework) and a code generator that takes a HiL design as input and automatically generates the HiL prototype based on the HiL framework.
- **Step 2: Validating the HiL prototype.** Since executing a HiL prototype in a real environment is complex, we provide a **simulation infrastructure** (aka HiL simulator) to visually reproduce the execution of a HiL prototype. The

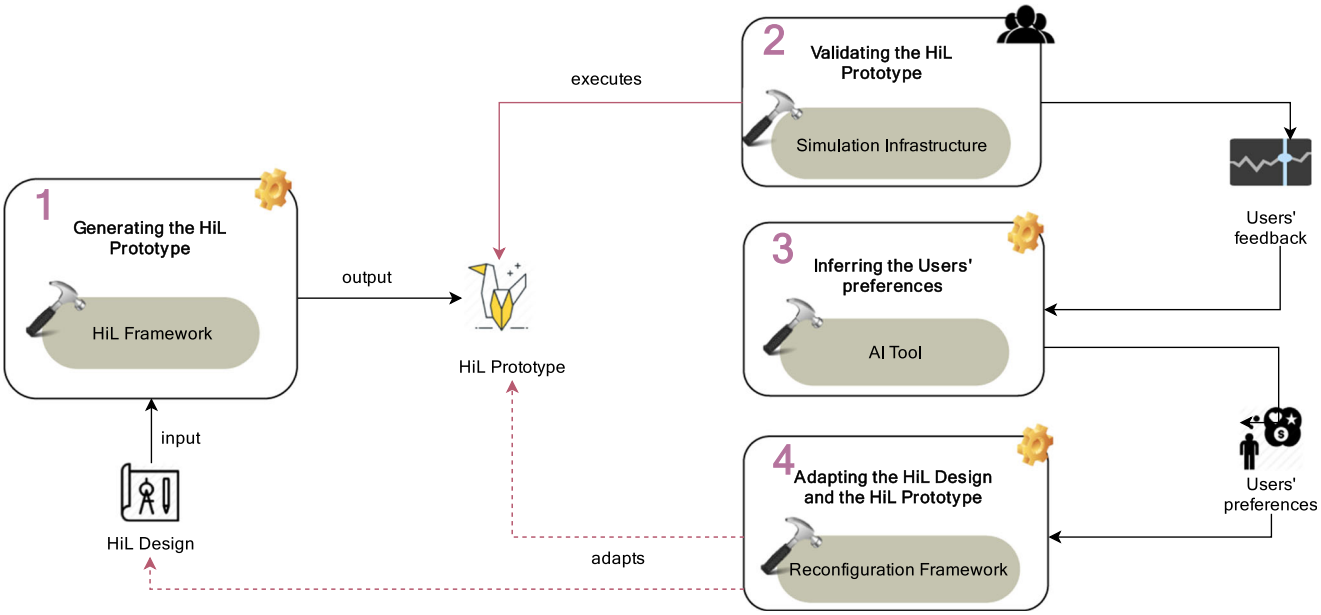


Fig. 3 Process overview of the HIL prototyping

HiL simulator allows users to perform the validation of the HiL prototype at the same time that it gathers the user's implicit feedback, which feeds into an AI tool that infers knowledge from this feedback.

- **Step 3: Inferring the users' preferences.** We have built an **AI tool** that continuously receives the user's implicit feedback from the HiL simulator to infer the user's preferences in terms of obtrusiveness. This inference is based on a prediction model that is built using machine learning techniques.
- **Step 4: Adapting the HiL prototype.** The inference of the AI tool is used to self-adapt the HiL prototype to the user's obtrusiveness preferences at runtime. This self-adaptation is done by using a reconfiguration framework (aka the FADA framework) that assists designers in engineering, building, and deploying solutions with autonomic computing capabilities. We have used the FADA framework to endow the HiL prototype with autonomous reconfiguration capabilities.

The "smart" prototyping process begins with the automatic generation of the HiL prototype (Step 1). Once the HiL prototype has been created, designers initiate the prototype validation with users (Step 2). During the prototype validation, the simulator is continuously providing the user's usage data to the AI tool. From this data, the AI tool infers the user's preferences (Step 3). These inferences are reported to the reconfiguration framework, which adapts the HiL prototype at runtime according to these preferences (Step 4). Then, the HiL design is adapted accordingly. For example, while the user is validating the HiL prototype through the HiL simulator, the simulator provides the user's usage data to the AI tool, which interprets that the human is feeling disturbed by a HiL action and infers that less attentional resources are required to perform this HiL action. Then, the reconfiguration framework adapts the prototype by changing the interaction mechanism associated with the action to a less intrusive interaction mechanism, and the HiL design is refined to decrease the attention level of the action. Note that Step 1 is performed only once at the beginning of the process, while Steps 2 to 4 are performed iteratively throughout the validation process. The whole process is automatically carried out with just the designer's supervision. At the end of the prototype validation, the designer gets the HiL designs refined according to each user's preferences. This allows designers to uncover insights about the user's preferences and to figure out a user's mental model. Based on this knowledge, the designer builds the final HiL design.

In the following subsections, we provide details about the four steps of the proposal and describe the software component that we provide to support each step.

5.1 Automatic generation of HiL prototypes: the HiL framework

The first step of the prototyping process is to build a HiL prototype. A HiL prototype is an early sample of a HiL solution that focuses on representing how humans and the system work in a collaborative way. This prototype is not required to provide a complete representation of the system functionality but rather the part that is related to how the collaborative work is shared between the human and the system and the interactions that are required to establish the human-system

communication. According to our proposal, this collaborative work is specified by means of HiL tasks. Therefore, a HiL prototype is an implementation of the HiL tasks that have been specified for a specific system.

We provide a software infrastructure (aka HiL Framework) and a code generator that are capable of generating a HiL prototype automatically taking the task specifications (HiL design) as input (see Fig. 4). In this way, we allow developers to focus on designing the characteristics of the human-system interaction, without worrying about operational aspects. This HiL prototype can evolve into the final solution during the development phase. The following subsections describe the HiL framework and how the HiL prototypes are automatically generated using it.

5.1.1 The HiL framework

The HiL framework is a common reusable software infrastructure that provides support to the conceptual framework proposed in Section 4. It has been developed to be applied for different solutions in any domain.

Figure 5 illustrates the HiL framework architecture. It is composed of two layers: the infrastructure layer, which is made up of a *common infrastructure module* with libraries and utilities to support communications between modules and an *interaction mechanism module* that provides components to implement the interaction with the user (within our prototyping solution, the interaction mechanisms are implemented in the simulation infrastructure), and the task management layer, which is built on top of the infrastructure layer and provides support to the implementation of HiL tasks. Specifically, the implementation of the *task management layer* is composed of the following elements (see Fig. 6):

- **Event monitor.** It defines an abstract component (*HiL_EventMonitor*) that is responsible for receiving events triggered from an external component (that may be the autonomous system or the HiL simulator). It launches the execution of the corresponding collaborative task.

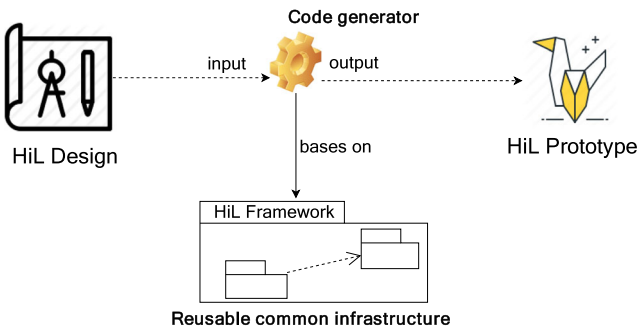


Fig. 4 Step 1 of the "smart" prototyping process: generating the HiL prototype

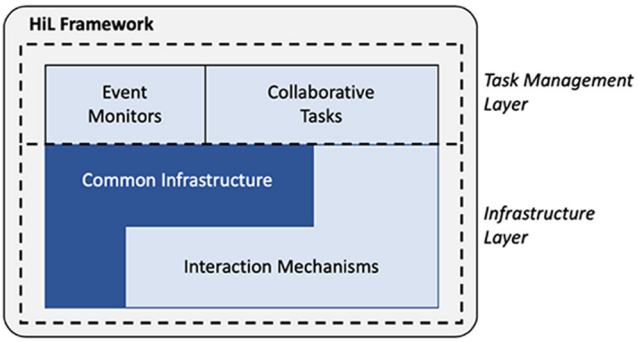


Fig. 5 HiL framework architecture

- **Execution strategy.** It is a concrete component (*HiL_ExecutionStrategy*) that implements the strategy for executing the collaborative tasks (through the *executeTask()* method). This strategy is defined according to the operationalization of collaborative tasks that was introduced in Subsection 4.2. This component encapsulates a strategy that is common for every collaborative task. Encapsulating the strategy in a separate component allows better separation of concerns to be achieved and lets developers vary the execution strategy independently of the tasks, making it easier to understand and extend.
- **Collaborative task components**. These components implement the concepts of the conceptual framework (e.g., collaborative task, human profile, precondition, action, constraint, fallback plan, etc.) according to its execution semantics. The main component is the abstract component *HiL_CollaborativeTask*, which is instantiated to create new collaborative tasks. This component maintains a reference to the execution strategy component. Then, the *execute()* method delegates the task execution to the execution strategy component. The collaborative task component maintains a reference to the components that implement the elements involved in a collaborative task and defines a method to execute the core and feedback actions (*executeCoreActions()*). A *HiL_CollaborativeTask* is composed by a *HiL_FallbackPlan*, several *HiL_Action* (which in turn are composed of several *HiL_InteractionMechanism)*, a *HiLConstraint*, a *HiL_Precondition* (which in turn is composed of several *HiL_PreparatoryAction)*, and a *HiL_HumanProfile*.

The HiL framework has been implemented in Java/OSGi.[2] OSGi provides a service-oriented architecture that enables components (i.e., bundles) to dynamically discover each other for collaboration. An installed component can register services by publishing their interfaces using the service registry of the OSGi framework. Thus, when a component queries the registry, it obtains references to actual service objects that are registered under the desired service interface. Therefore, each

[2] http://www.osgi.org/

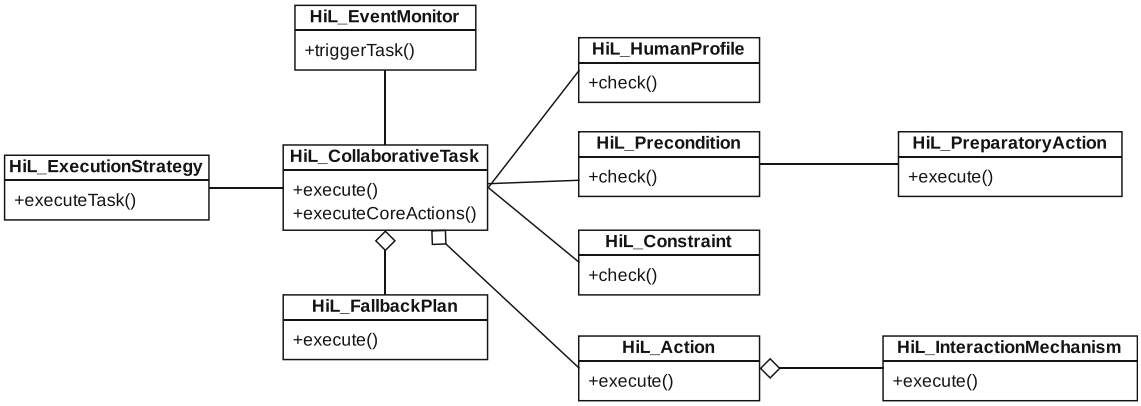


Fig. 6 Task management layer implementation

HiL_CollaborativeTask is encapsulated in an OSGi bundle. In this way, we can add as many collaborative tasks as we need without having to stop the system. This provides flexibility for adding new collaborative tasks.

5.1.2 Code generation

We have also built a code generator for transforming the HiL designs into operational HiL prototypes based on the HiL framework. This generator is implemented by means of model-to-code transformations. These transformations are implemented using a template strategy as defined in the Model-to-Text (M2T) project,[3] which is part of the Eclipse Modeling Project.

Specifically, model-to-code transformations are provided for three different resources: (1) the interaction mechanisms, (2) the HiL actions, and (3) the collaborative tasks. Note that other components (such as components of the common infrastructure module) do not need to be generated since they are reused (because they are not specific for a HiL solution). The code-generation process instantiates the HiL framework, taking a HiL specification as input, according to the following mappings:

- For each concrete interaction mechanism, a new instance of the *HiL_InteractionMechanism* class is generated and assigned to its supported obtrusiveness levels.
- For each action, a new component extending the *HiL_Action* class is created. This component is bound to the available interaction mechanisms according to its obtrusiveness level.
- For each collaborative task, a *HiL_CollaborativeTask* class is instantiated as well as all of its components: a human profile (*HiL_HumanProfile*), fallback plan (*HiL_FallbackPlan*), preconditions (*HiL_Precondition*), its related preparatory actions (*HiL_PreparatoryAction*), and the constraints (*HiL_Constraint*). The task is assigned to its previously defined *HiL_Actions*.

[3] https://www.eclipse.org/modeling/m2t/

5.1.3 Code-generation example: *Takeover* task.

To better exemplify the code-generation process, we illustrate the generation of the *Takeover* collaborative task (which has been specified throughout Section 4). Figure 7 shows an excerpt of the code generated. Below, we provide some details about this code:

- The interaction mechanisms that are involved in the system (the smart car) are generated and configured by creating instances of the *HiL_InteractionMechanism* class (lines 1 to 19). In this case, the interaction mechanisms that are included in Table 1 are generated.
- The actions of the task are generated as classes that inherit from the *HiL_Action class*. Lines 20 to 29 show the definition of the notify takeover action. Three more actions (*Confirm takeover*, *Transfer control*, and *Inform about driving mode*) are also generated for the *Takeover* task; however, they are not shown due to space limitations. The *Notify takeover* action is assigned to the *system initiative* and *high attention level* (line 25). According to this obtrusiveness level, this action is bound to the *Speakers.VoiceFeedback* and *HeadUpDisplay.Textual* interaction mechanisms (lines 26 and 27, respectively).
- The *Takeover* task is defined as a class and is initialized (lines 30 to 55), which includes the declaration of:

  – The *Takeover* task,
  – An event monitor (‘*hil/Takeover*’, line 35)
  – A human profile (‘*human.driver* = *true*’, lines 37 to 39)
  – A set of preconditions with their corresponding preparatory actions (lines 41 to 44)
  – A fallback plan (line 46)
  – A constraint (line 47)
  – The core and feedback actions (lines 49 to 52)

```
1  public class InteractionMechanismsGeneration {
2
3      public void createInteractionMechanisms() {
4
5          InteractionMechanism speakers_VoiceFeedback = new HiL_InteractionMechanism("Speakers.VoiceFeedback");
6          speakers_VoiceFeedback.addSupportedObtrusiveness(EInitiative.SYSTEM, EObtrusivenessLevel.HIGH);
7
8          InteractionMechanism headUpDisplay_Textual = new HiL_InteractionMechanism("HeadUpDisplay.Textual");
9          headUpDisplay_Textual.addSupportedObtrusiveness(EInitiative.SYSTEM, EObtrusivenessLevel.HIGH);
10
11         InteractionMechanism headUpDisplay_Image = new HiL_InteractionMechanism("HeadUpDisplay.Image");
12         headUpDisplay_Image.addSupportedObtrusiveness(EInitiative.SYSTEM, EObtrusivenessLevel.SLIGHT);
13
14         ...
15
16         InteractionMechanism cam_BodyRecognition = new HiL_InteractionMechanism("Cam.BodyRecognition");
17         cam_BodyRecognition.addSupportedObtrusiveness(EInitiative.HUMAN, EObtrusivenessLevel.LOW);
18
19     }
```

```
20 public class NotifyTakeoverAction extends HiL_Action implements IHiLAction {
21
22     public NotifyTakeoverAction(HiL_BundleContext context) {
23         super(context, "NotifyTakeover");
24
25         setObtrusiveness(EInitiative.SYSTEM, EObtrusivenessLevel.HIGH);
26         addInteractionMechanism(context.searchInteractionMechanism("Speakers.VoiceFeedback"));
27         addInteractionMechanism(context.searchInteractionMechanism("HeadUpDisplay.Textual"));
28
29     }
```

```
30 public class TakeoverTask {
31
32     public void initialize(HiL_BundleContext context) {
33
34         ICollaborativeTask ct = new HiL_CollaborativeTask(context, "Takeover Task", "Perform the takeover");
35         ct.setEventMonitor(new HiL_EventMonitor("hil/Takeover"));
36
37         IHumanProfile profile = new HiL_HumanProfile();
38         profile.addCapability("human.driver", true);
39         ct.setHumanProfile(profile);
40
41         ct.addPrecondition(new RadioVolumePrecondition(context),     new RadioVolumePrepAction(context));
42         ct.addPrecondition(new AttentionLevelPrecondition(context), new AttentionLevelPrepAction(context));
43         ct.addPrecondition(new DriverSeatPrecondition(context),      new DriverSeatPrepAction(context));
44         ct.addPrecondition(new HandsOnWheelPrecondition(context),    new HandsOnWheelPrepAction(context));
45
46         ct.setFallbackPlan(new TakeoverFallbackPlan(context));
47         ct.addConstraint(new TakeoverConstraint(context));
48
49         ct.addHiLAction(new NotifyTakeoverAction(context));
50         ct.addHiLAction(new ConfirmTakeoverAction(context));
51         ct.addHiLAction(new TransferControlAction(context));
52         ct.addHiLAction(new InformAboutDrivingModeAction(context));
53
54         context.registerCollaborativeTask(ct);
55     }
```

Fig. 7 Code generated for the takeover collaborative task using the HiL framework

- All of these components are declared and assigned to the *Takeover* task.
- Finally, the task is registered (line 54) to make it available and executable.

5.2 Validating the HiL prototype: the HiL simulator

Once the HiL prototype is built, it must be validated. This validation requires executing the HiL prototype; however, since these prototypes are complex to execute in real situations, we have built a HiL simulator that allows the execution of the HiL prototypes. The HiL simulator visually recreates the execution of the collaborative tasks that are implemented in a HiL prototype (see Fig. 8) allowing users to experience an artificial recreation of the interactions that occur during their execution. Also, the HiL simulator gathers data from its usage, what we call the users' feedback. Specifically, it captures the following at each time step:

- The user context situation (such as the user's face position, the user's location, or if the user is accompanied)
- The HiL action being performed by the system or by the human
- The human response time, which is the number of seconds that it takes for the human to respond from the moment the system requests him/her to act

During the execution of the HiL simulator, the users' feedback is provided to the AI tool at each time step. From this

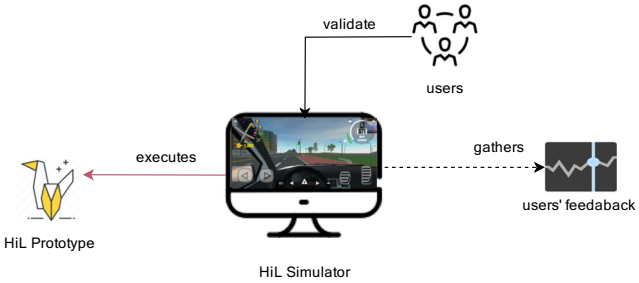


Fig. 8 Step 2 of the "smart" prototyping process: validating the HiL prototype

feedback, the AI tool infers the users' preferences and enhances the HiL prototype according to them at runtime.

5.2.1 The HiL simulator

The HiL simulator has been implemented using the Unity platform and is provided as a web application. Unity allows the generation of graphics and sounds that represent the system actions and interactable elements that support human actions. The HiL simulator is connected to a HiL prototype and to the AI tool by means of MQTT messaging. In this way, the HiL simulator executes the collaborative tasks of the HiL prototype through its interaction mechanisms, and at the same time, it continuously provides the data captured of the users' usage to the AI tool.

The HiL simulator provides a display from the user's perspective to allow the user's participation in a simulation of the execution of the HiL prototype. It is intended to graphically represent:

- The shared control flow of actions
- The interactions required to perform the collaborative work between humans and the system

Thus, the HiL simulator allows designers to validate the flow of actions between the human and the system that is designed for a collaborative task and the attention level associated with each action (which is determined by the interaction mechanisms used for each action).

Specifically, the HiL simulator is a web application that consists of a *main panel* that shows a scene where the actions that occur during the task simulation are visually represented. In addition, the panel displays a graphical representation of all of the available interaction mechanisms. The current implementation of the HiL simulator contains a basic set of interaction mechanisms that are commonly used in most systems. The interaction devices that the simulator uses are the following: speakers, microphone, screen, steering wheel, and light bulbs. These devices provide support to the following interaction mechanisms: beep from the speakers, synthetic voice from the speakers, voice recognition using the microphone, icons on the screen, visual text on the screen, touch button on the screen, haptic vibration on the steering wheel, haptic touch on the steering wheel, and light signals on the light bulbs. The interaction mechanisms are interactable and allow users to interact with the system through the selected interaction mechanism. For example, by clicking on the wheel element, the user performs the interaction to take the wheel.

Also, the simulator includes a representation of the context variables that may impact the behavior of the HiL prototype. The users can modify the context variables to simulate changes in the human and system context. This allows users to figure out how the AmI system works depending on the context since it leads to a different behavior of the human-system cooperation. Currently, the simulator includes the following context variables: user's face position, user's location, user's hand position, speaker's volume, user's activity, and users accompanied. The state of the context variables can be changed by clicking on the interactable elements or by means of shortcut keys. The HiL simulator gathers the value of these context variables together with the HiL action that is being executed in the system and the human response time at each time step to feed the AI tool.

The HiL simulator allows designers to configure it to include the appropriate visual representation of the scene, the interaction mechanisms used in the prototype, and the context variables needed. Also, to initiate a simulation, some data is required to configure a task execution. This data is:

- The timing of system actions.
- The notification text that should be shown to the user in case of notification actions.

This data is specified as properties of each action when modeling the collaborative tasks.

5.2.2 HiL simulator example: the autonomous car simulator

In the case of the autonomous car, the main panel of the HiL simulator shows a car dashboard with the in-vehicle interaction mechanisms available: the steering wheel, a head-up display, and the speakers, as shown in Fig. 9. The context variables that are shown and can be modified in the simulator are the following:

- *User's face position*: A graphical metaphor is shown in the upper-right corner of the window to indicate the value of this context variable. This element can change its value by means of the arrow keys and can take the values: looking forward or not looking forward.
- *User's hands on wheel*: An image of hands on the wheel is shown when the human has his/her hands on the wheel and no image of hands when the human does not have his/

Fig. 9 Main panel of HiL simulator for the autonomous car

her hands on the wheel. The user can switch this value by clicking on the wheel.
- *User's location*: An image of legs on the car's seat is shown to indicate where the human is located. The user can switch this value by clicking on one seat.
- *Radio volume*: A sound icon is shown in the upper-right corner of the window to indicate the value of this context variable. The user can turn the radio volume up or down by clicking on the radio element.
- *User's activity*: A finger near the head-up display is shown when the human is busy. The user can switch this value by clicking on the head-up display.

This smart car simulator is used in the evaluation section (Section 7), where we show an example of the use of the simulator.

5.3 Gathering user feedback and inferring user needs: the AI tool

The HiL simulator gathers feedback from users as mentioned above. Traditionally, this feedback is analyzed by designers who extract knowledge by hand and then enhance their designs and prototypes. However, in our proposal, we apply AI techniques to smartly extract this knowledge. In the last few decades, AI techniques have been extensively studied seeking new ways to automate Software Engineering tasks and help developers make better decisions. Specifically, there is growing interest in exploring how to use techniques from machine learning to personalize interfaces based on the observation of user activity [24]. We have brought this practice to the validation phase and propose "smart" prototyping to self-adapt HiL solutions to the user's preferences at runtime. Specifically, our solution proposes the use of machine learning techniques to learn from the user's experience which attentional demand best fits their preferences and needs. Then, using this knowledge, the human-system collaboration designs can be enhanced before developing the system.

5.3.1 The AI tool

We have built an AI tool that uses the data that the simulator provides to infer the user's obtrusiveness perception (see Fig. 10). The AI tool is continuously receiving usage data from the HiL simulator. This data contains information about the user context situation (which includes six context variables, such as the user's face position and the user's location), the HiL action being performed at a given time, and the human response time. From this data, the AI tool infers the user's obtrusiveness perception for the HiL action executed in the HiL simulator. Specifically, for each HiL action, the AI tool infers

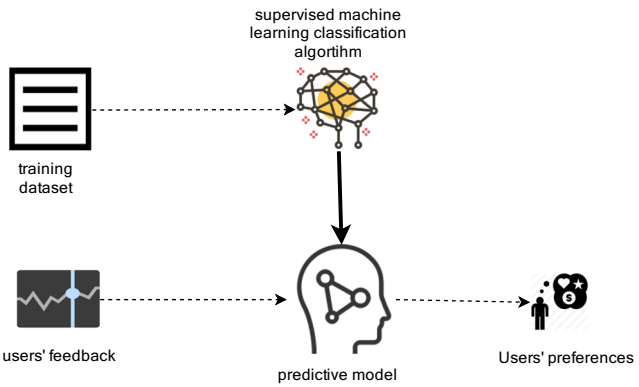


Fig. 10 Step 3 of the "smart" prototyping process: inferring users' preferences

whether the action is annoying the user, if it has not been perceived, or whether it is suitable. As mentioned above, the communication between the AI tool and the HiL simulator is performed using messaging queues.

The knowledge inferred by the AI tool allows the HiL prototype and the HiL design to be adapted using the reconfiguration framework. The output of the AI tool is analyzed by the reconfiguration framework that, when necessary, adapts the HiL prototype by changing the interaction mechanisms used to perform the action. Then, the HiL design is adapted to refine the level of attention defined for the action.

The behavior of the AI tool is executing the following sequence in an endless loop, while the simulator is being executed:

1. It reads the data about the user's context and the system activity that the HiL simulator provides to the queue.
2. It processes this data to fit it to the predictive model that we have built.
3. It calls the predictive model passing the input data as argument, which classifies this input into three classes:

- Is annoying the user (0: DISTURBING)
- Has not been perceived (1: NOT-AWARE)
- Is suitable (2: OK)

4. It builds and returns a tuple made up of the HiL action that is involved in the input and the user perceived obtrusiveness that is inferred by the predictive model, for example, (*NotifyTakeover*, DISTURBING).

In order for the AI tool to be able to infer the perceived obtrusiveness preferences, a predictive model should be built. The following section describes the predictive model that the AI tool uses in Step 3.

5.3.2 Building the predictive model

The predictive model has been built by applying a supervised machine learning classification algorithm [45], which maps an input to an output based on example input-output pairs. The classification algorithm is in charge of identifying the relationship between the inputs and outputs from these example pairs and producing a predictive model that can be used for mapping new examples.

We have implemented this algorithm as a neuronal network with TensorFlow.[4] The algorithm builds a prediction model that classifies the input data into three classes: (0) DISTURBING, (1) NOT-AWARE, or (2) OK. The data include eight input variables, such as the user's face position, the user's hand position, the user's location, and the user's response time. The algorithm has been implemented as a neural network with five fully connected layers since this number of layers has resulted in the best classification accuracy. The activation functions that are used to define the behavior of each layer and that determine the output of a node given an input or set of inputs are the following: the "ReLU" function [46] for the four hidden layers and the "softmax" function [47] for the output layer. The activation function "softmax" is often found in the output layer of a neural network and returns the probability distribution over the mutually exclusive output classes. The "ReLU" trigger function has proven to work in many different situations and is now widely used. As a loss function, one of the parameters necessary to quantify how close a particular neural network is to the ideal weight during the training process, "sparse_categorical_crosentropy," has been used since our output must be in categorical format. Choosing the best loss function requires understanding what type of error is or is not acceptable for the particular problem. As an optimizer, another of the elements required to perform the inference, "Adam" has been used which implements a stochastic gradient descent method [48]. Of the different existing optimizers (SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam), "Adam" is the one that best minimizes the loss function in our classification problem.

In order to build the predictive model, the algorithm must be trained with a labeled dataset with input-output pairs. The training data has special importance in a machine learning tool, and it must be specific for each HiL prototype. Therefore, designers can use our AI tool, but they must build their own dataset to train the algorithm, and they must check the accuracy of the prediction in order to validate it by means of test datasets. In the following section, we detail how we have built the dataset and have obtained the prediction model for the autonomous car.

5.3.3 The example of the autonomous car

To illustrate the process for building the predictive model by means of the supervised machine learning classification algorithm, we have implemented an example for the case of the autonomous car.

We did not find any public dataset to train our machine learning algorithm. Therefore, we built our own dataset using the web simulator. The dataset was made up of 400 samples. The data was obtained by the web simulator from 10 different users with different levels of driving expertise. The users participated in different collaborative tasks in different situations. As mentioned in Section 5.2, the web simulator registered the user's context, the HiL action performed by the system or the human, and the human response time at each instance.

The 400 registers of the obtained data were hand-labeled by designers based on the observed user's perception of obtrusiveness in each situation: (0) DISTURBING, (1) NOT-AWARE, or (2) OK. Table 2 shows a few examples of the

[4] https://www.tensorflow.org/

Table 2 Examples from the training dataset

Input								Output
HiL action	User's face position	User in driver's seat	User's hands on wheel	User's activity	Fallback plan activated	Speaker volume action	User response time	Class
Notify takeover	Looking forward	True	True	notBusy	False	Volume down	3.0 s	DISTURBING
Confirm takeover	Looking forward	True	True	notBusy	True	No action	2.8 s	DISTURBING
Take the wheel	Not looking forward	True	False	notBusy	False	Volume up	4.1 s	NOT-AWARE
Transfer control	Looking forward	True	True	notBusy	False	No action	2.6 s	OK

dataset that was used for training the data. The dataset was composed of 400 registers (90% was for the training dataset, and 10% was for the test dataset).

As Table 2 shows, register 1 corresponds to a situation where the *Notify Takeover* action was being executed by the system. At that moment, the user was looking forward, was seated in the driver's seat, had their hands on the wheel, and was not performing any activity other than driving. Moreover, the fallback plan was not activated, and the user turned down the radio volume. The user answered the system's demand in 3 s. At that moment, we observed that the user was feeling annoyed by the system. Therefore, we labeled this row as DISTURBING. In contrast, register 3 corresponds to a situation that was labeled as NOT-AWARE. In this situation, the *Take the wheel* action was being executed by the system. At that moment, the user was not looking forward, was seated in the driver's seat, did not have their hands on the wheel, and was not performing any activity other than driving. Moreover, the fallback plan was not activated, and the user turned up the radio volume. The user answered the system's demand in 4.1 s. At that moment, we observed that the user was not aware of the system action.

To build the predictive model, we used the neural network specified in Section 5.3.2. Once the algorithm was trained, we evaluated the obtained prediction model by using the test dataset. This evaluation confirms (or rejects) that the predictive model infers the output that actually corresponds to the input by checking the input-output pairs of the test dataset. Figure 11 shows the confusion matrix that indicates how the algorithm behaved for the three output classes. The matrix indicates the percentage of the predicted classes with regard to the true class. DISTURBING was occasionally confused with NOT-AWARE (13%), and NOT-AWARE was confused with OK (2%) a few times. This picture gives us insight into the prediction accuracy (the percentage of correct predictions divided by the total number of predictions), which was 96.75% in our algorithm.

Note that our proposal currently focuses on refining the level of attention defined for each action and thus adapting the interaction mechanisms used to perform the action. However, the proposed infrastructure could be used to enhance other aspects of HiL designs. Using AI to infer further knowledge about users' preferences and needs can help to obtain the HiL designs that are best adapted to the users, achieving better user experiences. For example, AI can infer knowledge about the level of a user's understanding of the system, which can help to adjust the number of feedback actions that the system provides to the user. Likewise, AI can extract knowledge about how specific context situations impact the success or failure of the task, which would help to refine the preconditions of the tasks. Machine learning algorithms can also be used to extract the way users work and achieve HiL designs that fit it.

5.4 Run-time refinement of HiL designs and prototypes: the FADA framework

An inference of the *AI tool* could imply that the HiL prototype and the HiL design need to be refined. This refinement should be performed automatically at runtime, that is, while the users

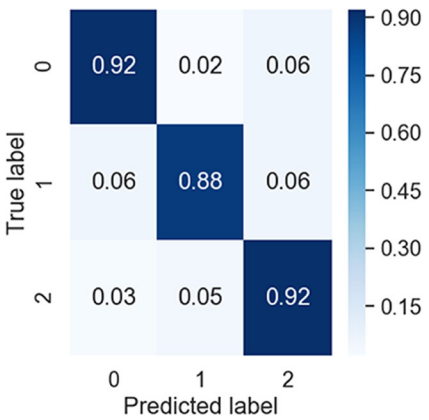


Fig. 11 Confusion matrix table

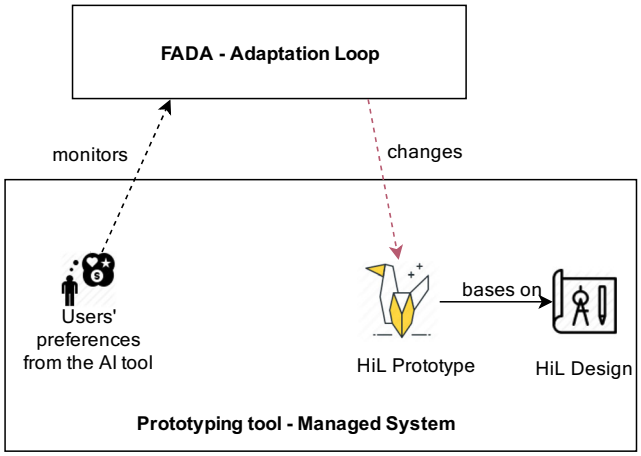


Fig. 12 Step 4 of the "smart" prototyping process: run-time refinement of HiL prototypes

are interacting with the HiL simulator. For each inference, the prototype should be adjusted by increasing or decreasing the obtrusiveness level of the HiL action referred by the inference. Then, the HiL design is refined according to the changes of the HiL prototype.

To cope with this requirement, we included a new component to the HiL prototypes, the *FADA - Adaptation Loop* (see Fig. 12). It endows the HiL prototypes with autonomic reconfiguration capabilities, which allow them to automatically adapt themselves at runtime. This component is responsible for (1) monitoring the outputs from the AI tool, (2) analyzing and deciding what changes are required, and (3) performing those changes to both the HiL prototype and then the HiL design.

We used the FADA framework[5] to implement this component. The FADA framework assists designers in engineering, building, and deploying systems with autonomic computing capabilities. This framework was built taking the self-adaptive engineering principles as basis, which promotes the usage of control loops [49] (*Control Layer*) on top of the system being managed (*Managed System*). The following subsections describe the FADA framework and how we use this framework to endow HiL prototypes with self-adaptive capabilities.

5.4.1 The FADA framework

The FADA framework provides us with tools and code generators to define and implement the elements of a MAPE-K control loop for both the managed system and the adaptation loop. Specifically, the FADA framework is based on the MAPE-K design pattern [50] as the specific conceptualization to build the adaptation control loop. Following this pattern, the FADA framework is composed of the following components (see Fig. 13):

[5] http://fada.tatami.webs.upv.es

Managed system:

- *Sensors*: They take system measurements and send those measurements to an adaptation loop monitor.
- *Effectors*: They receive inquiries to change the system configuration and are responsible for performing those change actions.

MAPE-K adaptation loop:

- *Monitors*: They collect, filter, and process system measurements and correlate this information into symptoms that can be analyzed. The monitor stores these symptoms as adaptation properties. These monitors are part of the monitoring module.
- *Adaptation properties*: They are used to support adaptation processes and represent relevant knowledge to perform an adaptation. These properties are stored in the knowledge module.
- *Adaptation rules*: They are tied to changes in adaptation properties and define a set of changes to the system. These rules are part of the analyzing module.
- *Planner*: It takes a set of changes proposed by the adaptation rules to calculate an adaptation plan. It delivers this plan to the executor module. This plan is made up of actions that describe changes in the infrastructure of the managed system. The planner is part of the planning module.
- *Executor*: It provides the mechanism to schedule and perform the necessary changes to the system defined by the adaptation plan using effectors. The executor is part of the executor module.

The FADA framework provides a set of pre-implemented components (such as a planner, an executor, or a set of effectors) to speed up the development stage.

5.4.2 Implementing the self-adaptive capabilities for the HiL prototypes

We used the FADA framework to implement the following components (see Fig. 13):

- A *sensor* that handles the outputs of the AI tool. These outputs refer to tuples made up of a HiL action and a value that determines the user perceived obtrusiveness (as explained in Subsection 5.3.1). An example of a tuple is: (*NotifyTakeover*, DISTURBING). The sensor reports this information to the monitor.
- A *monitor* that collects data from the sensor. It detects if there is a potential need to adapt (aka symptom) and then upgrades the knowledge module by updating an adaptation property. For example, if the monitor receives the tuple

(*NotifyTakeover*, DISTURBING), the monitor updates the adaptation property *NotifyTakeoverObtrusiveness* to *disturbing*.

- A set of *adaptation properties* that establishes the perception of the obtrusiveness level for each action. It is stored in the knowledge module. For example: *NotifyTakeoverObtrusiveness*, *disturbing*.
- An *adaptation rule* that listens to changes in the set of adaptation properties and requests a set of changes to the system. For example, when the adaptation property *NotifyTakeover Obtrusiveness* changes, a rule is triggered to adjust the current HiL prototype by increasing or decreasing the obtrusiveness level related to this action. The rule requires changing the current interaction mechanisms that this action is using (that no longer satisfy the desired obtrusiveness level) by other interaction mechanisms that support the new required obtrusiveness level.

5.4.3 Adaptation example: decreasing the obtrusiveness level of the *Notify Takeover* action

Figure 13 shows an execution example of how these components work when a user perceived obtrusiveness request is made. For instance, the AI tool reports that the *NotifyTakeover* action of the *Takeover* task is "disturbing" the user (1). Then, the *user obtrusiveness preference* sensor sends this measurement to the monitor in the control loop (2). This monitor updates the *NotifyTakeoverObtrusiveness* adaptation property in the knowledge module, setting it to *disturbing* (3). The defined adaptation rule is then triggered to analyze and evaluate how to handle this situation (4). In this case, the adaptation rule decides to lower the attention level of this action by changing its connected interaction mechanisms.

Let us assume that the *NotifyTakeover* action was initially configured in a *high attention* obtrusiveness level. As Fig. 14 shows, at time t_n, the *Speakers.VoiceFeedback* and *HeadUpDisplay.Text* interaction mechanisms are assigned to this action. In order to lower the obtrusiveness level of this action, the adaptation rule performs a *Change Request* to unbind these mechanisms (red-dashed lines) and to connect the *HeadUpDisplay.Image*, *Speakers.Beep*, and *SteeringWheel.Vibration* (black line) to this action (according to Table 1). Finally, the rule notifies the planner of this change request (5).

The planner compares the change request with the current state of the HiL prototype regarding the current state of the *NotifyTakeover* action and then calculates an adaptation plan to change the system to satisfy the request. This adaptation plan contains a set of (structural) actions to change the current system configuration: unbind the high attention level interaction mechanisms and deploy and bind the slight attention level interaction mechanisms (6). The planner sends this adaptation plan to the *executor*, who uses the *effector* layer of the managed system to perform the change actions described in the

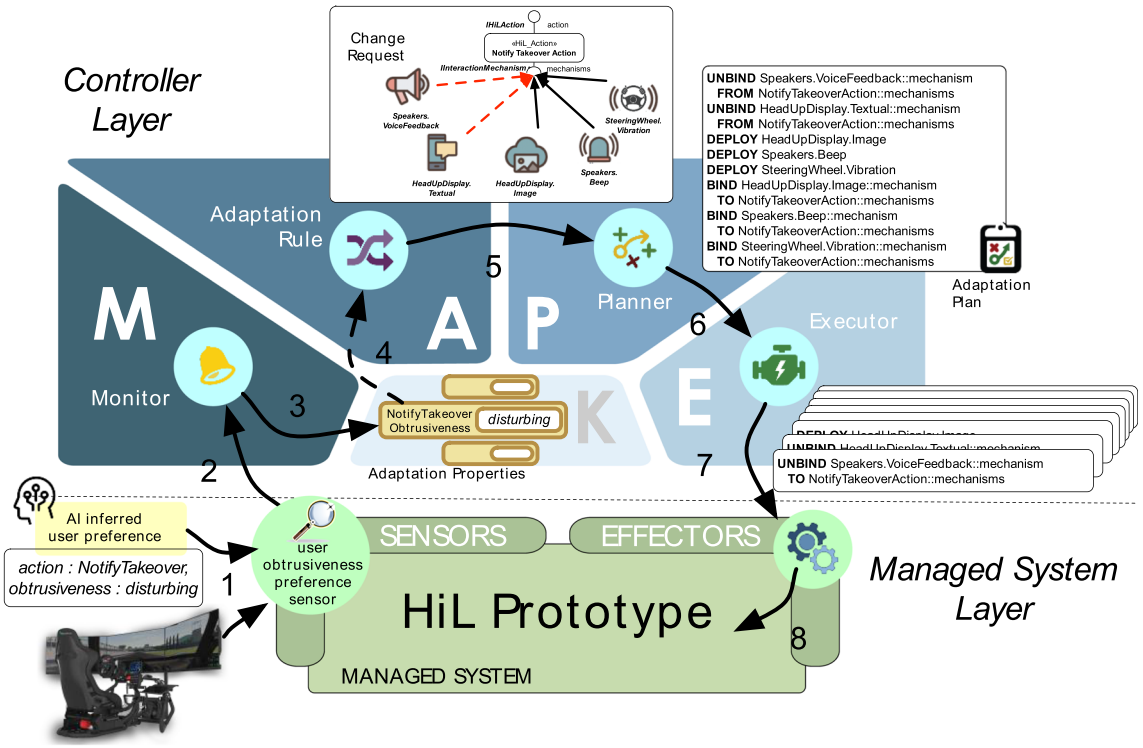


Fig. 13 HiL prototype with a MAPE-K control loop

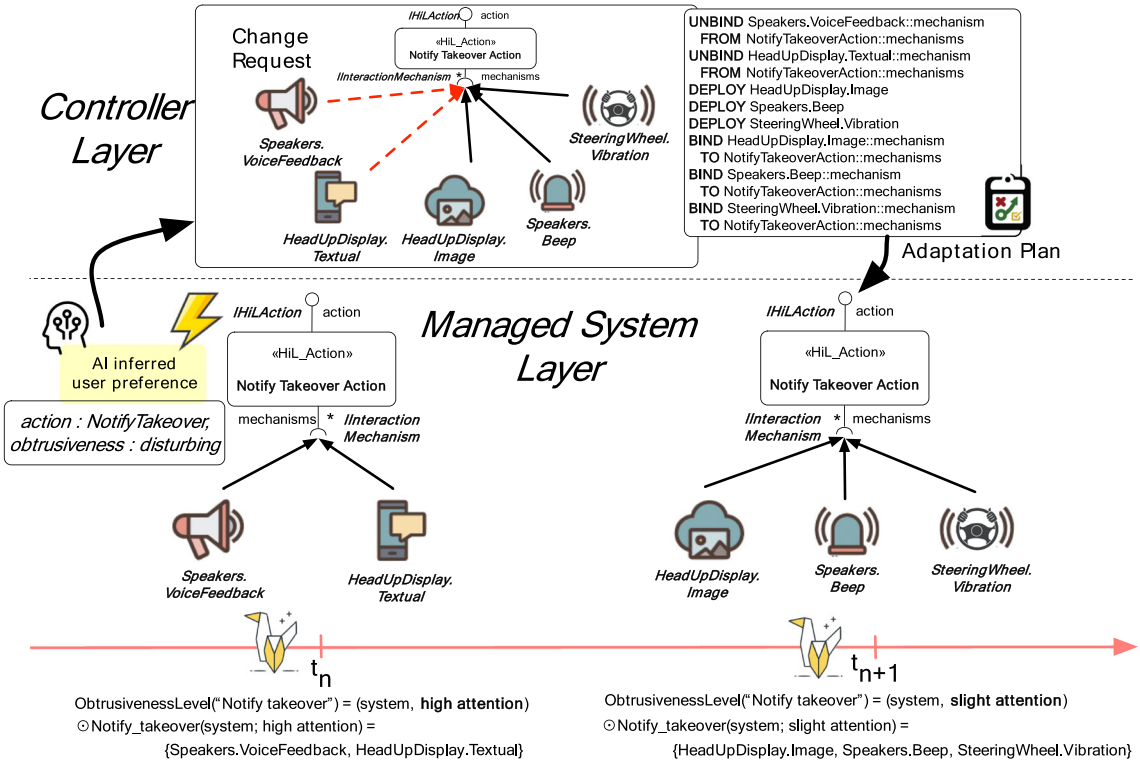


Fig. 14 An example of self-adaptation of the HiL prototype

adaptation plan (7). Finally, the effectors perform the changes within the running system (8) to obtain the next system configuration at t_{n+1}, which is a configuration that is less obtrusive than the previous one.

6 Validating the proposed method

The method proposed in this paper is conceived for helping interaction designers to design, prototype, and validate HiL systems. In this section, we present an experiment to analyze the perception of interaction designers regarding to which extent they would accept or not the design method proposed. This acceptance is related to the perceived usefulness, the perceived ease of use, and the intention to use in the future. In this experiment, several interaction designers applied the proposed method to design, prototype, and validate the HiL interactions in one of the most typical tasks of an autonomous car, the *Takeover task*, which has been used as a running example throughout the paper. This experiment followed the guidelines presented by Kitchenham et al. [51] and Wohlin et al. [52].

6.1 Goal

According to the Goal/Question/Metric template [38], the objective of the experiment was to:

Analyze our method to design, prototype, and validate HiL interactions
For the purpose of evaluating the acceptance for using the method
With respect to the perceived ease of use, the perceived usefulness, and the intention to use
From the viewpoint of the researchers
In the context of interaction designers

6.2 Research question and hypothesis formulation

The research questions and the null hypothesis proposed for the experiment are:

- RQ1: *Is our proposed method perceived as easy to use*? The null hypothesis tested to address this research question is: H_{01}: The proposed method is perceived as difficult to use.
- RQ2: *Is our proposed method perceived as useful*? The null hypothesis tested to address this research question is: H_{02}: The proposed method is perceived as not useful.
- RQ3: *Is there an intention to use our proposed method in the future*? The null hypothesis tested to address this research question is: H_{03}: There is no intention to use the proposed method in the future.

Table 3 Questions to evaluate the dependent variables PEOU, PU, and ITU

Item	Item statement
PEOU1	The process proposed by the design method was simple and easy to follow
PEOU2	Overall, the design method was easy to use
PEOU3	The design method was easy to learn
PEOU4	It was easy for me to apply the design method to the autonomous car example
PEOU5	It was easy for me to understand the rules of the design method
PEOU6	I believe that I can apply the method in practice
PU1	I believe that this design method would reduce the effort required to design HiL interactions
PU2	I believe that the HiL interactions designed with this method are easy to understand for users
PU3	This design method would help to obtain HiL interactions adapted to the user's preferences
PU4	Overall, I believe that the design method is useful
PU5	I believe that the use of this design method would help me to design HiL interactions
PU6	Overall, I think that the use of this design method provides an effective means for designing HiL interactions
PU7	Overall, I believe that this design method would improve the design of HiL interactions
PU8	I believe that the use of this design method would help interaction designers to design HiL interactions adapted to the user's preferences
ITU1	Definitively, I would use this design method to design HiL interactions
ITU2	I intend to use this design method in the case I would have to design HiL interactions in the future

6.3 Identification of variables and metrics

We identified two types of variables:

- **Independent variables:** Variables that affect the dependent variables. The process that is applied to design the HiL interactions was identified as a factor (aka independent variable) that affects the dependent variables.
- **Dependent variables:** Variables that correspond to the outcomes of the experiment. In this experiment, we evaluated our method with regard to the following dependent variables:
- *Perceived ease of use (PEOU)*, evaluated by means of *RQ1*, refers to the degree to which an interaction designer believes that using our method would be effort-free.
- *Perceived usefulness (PU),* evaluated by means of *RQ2*, refers to the degree to which an interaction designer believes that our method will achieve her/his intended objectives.
- *Intention to use (ITU)*, evaluated by means of *RQ3*, refers to the extent to which an interaction designer intends to use the method.

The instrument used to measure these variables was an adapted Technology Acceptance Model (TAM) [53], as well-known and thoroughly validated model for evaluating information technologies. This instrument defines a 5-point Likert with 16 items. Table 3 shows all the adapted items. PEOU1 to PEOU6 are associated with the dependent variable PEOU, PU1 to PU8 are associated with the dependent variable PU, and ITU1 and ITU2 are associated with the dependent variable ITU.[6]

6.4 Experimental context

Experimental subjects. The experiment was conducted in the context of the Universitat Politècnica de València (Spain). Six subjects participated in the experiment (3 males and 3 females) between 35 and 47 years old who were researchers of the PROS Research Center.[7] The background and experience of the subjects with regard to the tools and techniques used in the experiment were found through a demographic questionnaire handed out at the beginning of the experiment. According to the questions included in the demographic questionnaire, we concluded the following:

- All of the subjects had an extensive background in Java programming and modeling tools.
- All of the subjects had knowledge about HCI, but only some of them had experience in developing HiL interactions.

Experimental objects. The experiment was conducted using the running example used throughout the paper, i.e., the autonomous car. The object used in the experimental investigation was a requirement specification created for this purpose.

[6] We are aware that Likert scales are qualitative data, but some studies propose converting them to quantitative data to work with statistical tests [54]

[7] http://www.pros.webs.upv.es/

It contained the description of the *Takeover* task.[8] In order to shorten the evaluation process, we provided the subjects with an example of the *Handover* task (in this task, the driver gives the order to the system for it takes over control of the car) designed with our method to guide the design of the *Takeover* task.

Instrumentation. The instruments that were used to carry out the experiment were the following:

- A demographic questionnaire: A set of questions to know the level of the user's experience in Java/OSGi programming, modeling tools, and task modeling. This document included questions containing Likert-scale values ranging from 1 (strongly disagree) to 5 (strongly agree).
- Task description[9]: A document that describes the work/activities to be performed in the experiment using our method. This document contains guidelines to guide the subject throughout the experiment.
- Satisfaction in use questionnaire based on TAM: A questionnaire with the 16 items of Table 3 containing Likert-scale values ranging from 1 (strongly disagree) to 5 (strongly agree) to evaluate satisfaction with the entire process.

6.5 Experiment design and procedure

In this experiment, we followed a simple design (one factor with one treatment), where all subjects were exposed to the treatment. The subjects must design, prototype, and validate the HiL interactions of the *takeover* task of the autonomous car applying the proposed design method. To start the process, the subjects were provided with a requirements specification where the HiL interactions of the *takeover* task were described in natural language.

The study was initiated with a short presentation in which general information and instructions were given. Then, the experiment started with the demographic questionnaire to capture the users' backgrounds. Afterwards, the HiL task description document was given to the subjects, and they started to design the HiL interactions following our method. After the final design of the HiL task, subjects filled in the satisfaction in use questionnaire. Specifically, the activities carried out to apply our design method were the following (see Fig. 15):

1) *Designing HiL interactions*. The participants designed the HiL task using our conceptual framework according to the HiL task requirements specification document provided. We provided the subjects with a tutorial where the design language and the provided tools were explained. The output of this activity was the HiL task design. The experimenters supervised this phase with the aim of helping participants to obtain a suitable HiL design.
2) *Generating a HiL prototype*. Using the software infrastructure of our method, the participants generated the prototypes for the autonomous car from their HiL task design automatically.
3) *Validating the HiL prototype and automatically updating the HiL designs and prototypes*. The prototypes were tested by two end-users during 30 min. The end-users where two experimenters, which took the role of end-users to test the prototypes obtained by each participant. The two experimenters behaved the same for all participants. By means of the AI tool and the autonomous capabilities of the generated prototype, the HiL task design and the prototype were automatically enhanced according to the end-users' preferences and needs. In this activity, the participants only observed the end-users testing the prototype.
4) *Obtaining the final HiL design and the prototype*. When the experimenters finished testing the prototype, the participants analyzed the adapted HiL task design of both experimenters and modified the initial HiL task design proposed to fit the preferences of both experimenters. Then, from the final HiL task design, the final prototype was automatically generated.

6.6 Threats to validity

The various threats that could affect the results of this experiment and the measures that we took were the following:

- *Conclusion validity:* This validity is concerned with the relationship between the treatment and the outcome. Our experiment was threatened by the *random heterogeneity of subjects*. This threat appears when some users within a user group have more experience than others. This threat was minimized with a demographic questionnaire that allowed us to evaluate the knowledge and experience of each participant beforehand. This questionnaire revealed that participants had a very similar experience in the tools and techniques to be used in the experiment.
- *Internal validity:* This validity concern is related to the influences that can affect the factors with respect to causality, without the researchers' knowledge. Our evaluation had the *maturation* threat that refers to

[8] The requirement specification document can be downloaded from: http://hil.tatami.webs.upv.es/docs/methodEvaluationExperiment/RequirementsSpecification.pdf
[9] The task description document can be downloaded from: http://hil.tatami.webs.upv.es/docs/methodEvaluationExperiment/TaskDescriptionDocument.pdf

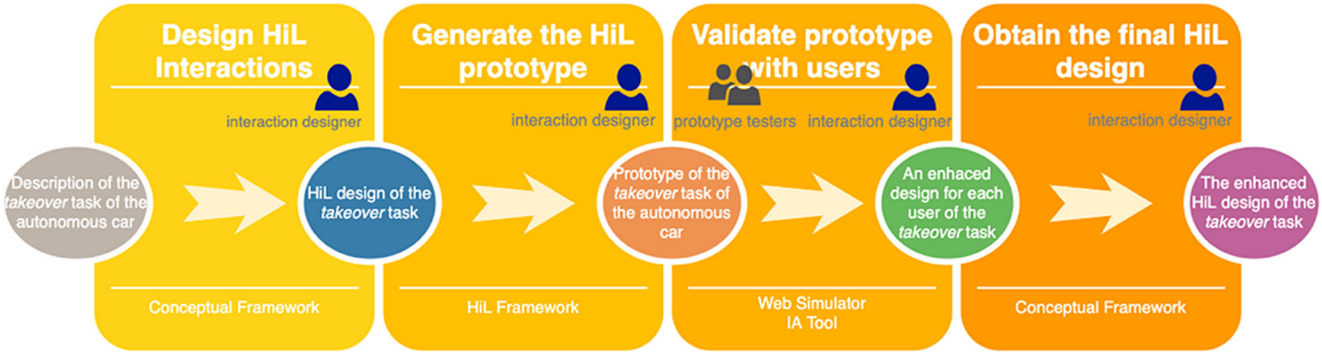


Fig. 15 Steps of the performed experiment

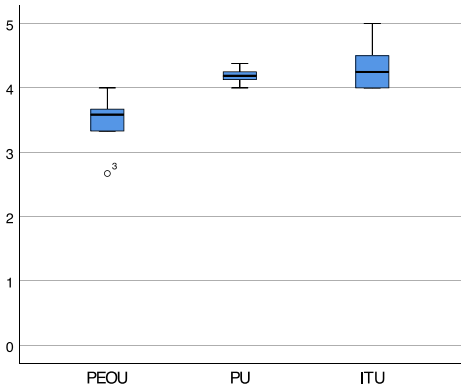
the effect that users react differently as time passes (because of boredom or tiredness). We solved this threat limiting the evaluation to 2 h. Also, the *instrumentation* threat could affect the evaluation due to an incorrect interpretation of the tasks to be done and the questionnaires. This threat was minimized by the researcher, who helped the subjects to understand the tasks.

- *Construct validity:* Threats to construct validity refer to the extent to which the experiment setting actually reflects the construct under study. Our experiment was threatened by the *hypothesis guessing* since users might try to figure out what the purpose and intended result of the experiment are, and they are likely to base their behavior on their guesses. We minimized this threat by hiding the goal of the experiment.
- *External validity:* This validity concern is related to conditions that limit our ability to generalize the results of the experiment to industrial practice. Our experiment might be affected by the *representativeness of the results* threat. Even though the experiment was performed in an academic context, the results could be representative with regard to interaction designers with no experience in our method. With respect to the use of researchers as experimental subjects, several authors suggest that the results can be generalized to industrial practitioners [55].

6.7 Data analysis and interpretation of results

The analysis was performed using the SPSS v.26 statistical tool. First, a descriptive study was performed, where we studied the response variables in terms of their descriptive statistics. Figure 16 shows the results obtained. The descriptive statistics indicate that the average values of PEOU (mean = 3.47), PU (mean = 4.19), and ITU (mean = 4.33) exceed the neutral score of 3, that is the midpoint of the 5-point Likert scale. These results indicate that designers perceived our method as a useful and easy-to-use method, and they will intend to use the method in the future if they had to design HiL interactions. Even though, the PEOU is the worst-rated variable, which indicates that a weak point of our method is the ease of use.

To deeply analyze the descriptive statistics, Fig. 17 presents the results obtained for PEOU, PU, and ITU variables according to the 16 items of the users' satisfaction questionnaire shown in Table 3. In the figure, we can show that most of the items was rated with the 4–5 score that means a great acceptance of our method by the users. The PEOU items were the worst rated. PEOU1 stated that *the process proposed by the design method was simple and easy to follow* and PEOU3 stated that the *design method was easy to learn*. As some participants did not have experience in developing HiL interactions, they had more difficulties to learn the design method and to follow the process. Also, some users (33.3%) perceived



Descriptive Statistics

	PEOU	PU	ITU
Min.	2.67	4.0	4.0
Max.	4.0	4.38	5.0
Mean	3.47	4.19	4.33
SD	0.45	0.13	0.40

Fig. 16 Box-and-whisker plot and descriptive statistics of PEOU, PU, and ITU

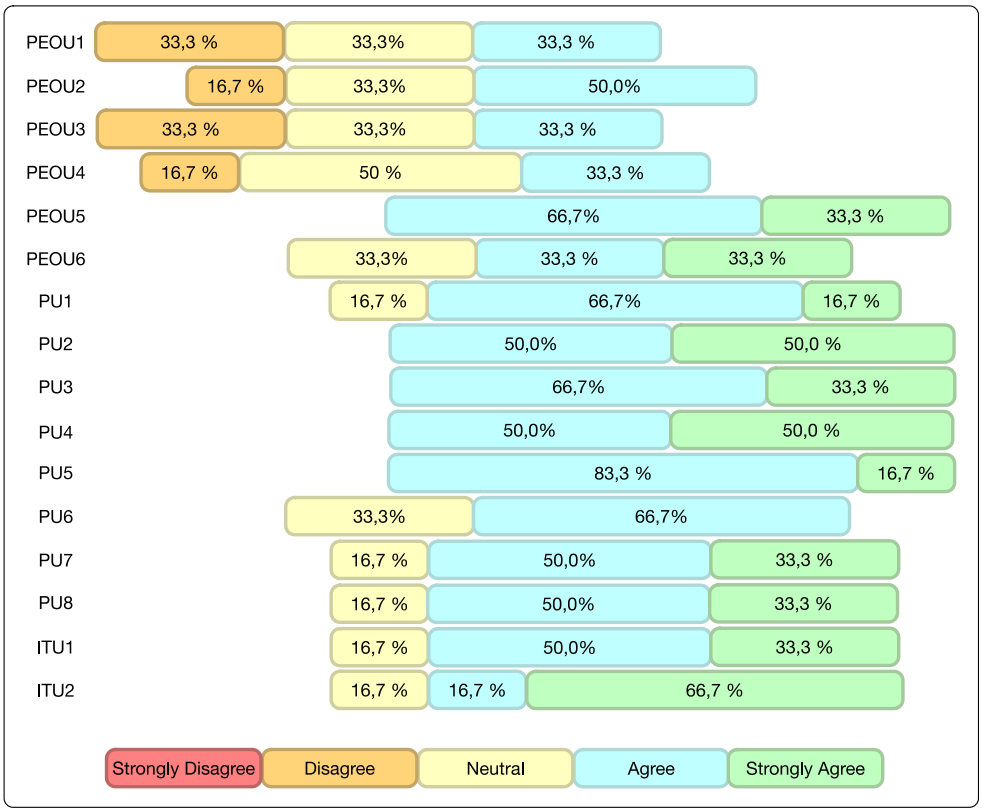


Fig. 17 Results of the user's satisfaction in use questionnaire

the method not easy to use (PEOU2) and found difficulties to apply it to the autonomous car example (16.7%, PEOU4). However, once they learnt it, they perceived the design method as useful (see PU4). In PU6 (*Overall, I think that the use of this design method provides an effective means for designing HiL interactions*), some users also commented us that did not rated the item with the high score because then cannot compare the method with another existing method to design HiL interactions. Despite this, in PU7 and PU8, 83.3% of users considered that the design method would improve the design of HiL interactions and they believed that the use of this design method would help interaction designers to design HiL interactions adapted to the user's preferences.

To confirm the observations of the descriptive study, we carried out the analytical study to each variable applying a statistical method. The statistical method allowed us to verify if the responses of the participants were significantly different from the neutral value (3) of the Likert scale.

A normality test using the Shapiro Wilk test was required to verify whether the data was normally distributed. We used this test as our numerical means of assessing normality because it is more appropriate for small sample sizes (< 50 samples). Then, using the Shapiro Wilk test, we obtained the following p values: *PEOU p value* = 0.456, *PU p value* = 0.834, and *ITU p value* = 0.091; therefore, as all the p value > 0.05, we retained the null hypothesis of population normality. Therefore, we can apply the t test. The t test was configured with the hypothesis value equal to 3 (the neutral value). The result obtained with this test was as follows:

- *Perceived ease of use:* The value obtained for the PEOU variable was p value = $0.05 >= 0.05$. Therefore, we cannot reject the null hypothesis H_{01} and can conclude that our method is perceived as not easy to use. However, as the results of the questionnaire show, some users perceived the method not easy to use at the first time, but they rated the method as useful.
- *Perceived usefulness.* The value obtained with this test for the PU variable was p value = $0.000 < 0.05$. Therefore, we can reject the null hypothesis H_{02} and can conclude that our method is perceived as useful.
- *Intention to use*. The value obtained with this test for the ITU variable was p value = $0.000 < 0.05$. Therefore, we can reject the null hypothesis H_{03} and can conclude that our method is perceived as useful.

The experiment has revealed that the method is useful, and participants have expressed their intention to use it, but they have not perceived it easy to use at a high degree. Feedback from questions PEOU1 to PEOU4 shows a need to provide a more guided process for applying the method. Therefore, we want to improve the usability of the method by providing tool support to help and guide interaction designers to specify the HiL task designs. In this way, we are planning to develop an

editor to build and validate HIL designs and an assistant (e.g., a dashboard) to guide interaction designers through the proposed method.

7 Evaluation of the smart prototyping technique

This section presents an evaluation of the usefulness of the "smart" prototyping technique. On the contrary to the previous section, which focuses on obtaining an evaluation of the own method by interaction designers, this section focuses on the product obtained by our technique. We used the case of the autonomous car that has been used throughout the paper. To perform the evaluation, we compared the initial HiL design that the designers defined with the enhanced version that the smart prototyping obtained in terms of users' satisfaction. This evaluation allowed us to explore whether the "smart" prototyping technique was helpful in achieving a HiL design that adapts to users' preferences and needs.

7.1 Variables

We identified two kinds of variables in the experiment: *independent variables*, which are the variables that are manipulated and controlled in the experiment, and *dependent variables* (or response variables), which are those variables that we want to study to see the effect of the changes in the independent variables. The variables in our experiment are described below.

Independent variables: The HiL design implemented in the HiL prototype was identified as a factor that affects the dependent variables. This factor had two treatments:
• The "initial HiL design" specified by designers
• The "adapted HiL design" where the run-time refinement had been applied after the "smart" prototyping process

Dependent variables: The dependent variable was the subjects' satisfaction regarding the proposed HiL prototype. The users' satisfaction was measured for the two aspects related to the challenge of managing user attention. These aspects are as follows:
• *Human attention achievement*. To measure this aspect, we used:
– The number of times that the fallback plans have been launched. One of the reasons for launching a fallback plan is the inability to achieve the human's attention in order to involve the human in the task. Therefore, the less the fallback plans were launched, the more human attention was achieved.
–The time period that the preparatory actions have been executing. Preparatory actions aim at preparing the human to maximize the success of the task. Therefore, the shorter the time period that the preparatory actions were executing, the more human attention was achieved.
• *Perceived obtrusiveness*. To measure this aspect, we used a 5-point Likert scale to assess the users' satisfaction.

7.2 Participants

The experiment was conducted at the Universitat Politècnica de València (Spain). Fifteen students that were recruited randomly by the authors' colleagues participated in the experiment. Their education level was balanced. They were between 21 and 35 years old. The background and experience of the subjects were determined using a demographic questionnaire that was handed out in the first session of the experiment. This instrument consisted of 10 questions on a 5-point Likert scale. The participants had to be subjects with different profiles (different ages, gender, and expertise) since we considered it necessary to conduct the experiment with a heterogeneous group in order to avoid any bias in the sample. According to the questions included in the demographic questionnaire, we concluded:

- The subjects were between 21 and 35 years old (10 males and 5 females).
- All of the subjects had taken a human-computer interaction (HCI) course as part of the curriculum of the Computer Science degree.
- None of the subjects had used an autonomous car or a simulation of it.

7.3 Tasks

The participants made use of the HiL simulator to interact with the HiL prototype. The HiL prototype was composed of six tasks for an autonomous car at L3 of autonomy [56]. Within an L3 autonomy, a car is capable of traveling thousands of kilometers without human intervention under restricted conditions (specially marked roads and good weather). Nevertheless, the driver is required to be ready to intervene at any moment in conflictive situations [57]. Thus, the tasks we designed that require human integration were the following[10]:

- T1 *Supervised Autonomous Driving*: This represents the collaborative work that the system and the human perform

[10] The task specification document with the initial HiL design of the autonomous car can be downloaded from: http://hil.tatami.webs.upv.es/docs/AutonomousCarSpecification.pdf

in AutoPilot mode. The AutoPilot (an autonomous system process) is responsible for performing the automated driving task (lane keeping with adaptive cruise control) in an operational situation. However, the L3 AutoPilot mode requires a driver to always be in attentive mode and ready to take over (if required).

- T2 *Supervised Manual Driving*: This represents the collaborative work that the system and the human do in Manual mode. The human is responsible for performing the driving task. However, the L3 Manual mode requires the system to supervise the state of the human (and take the appropriate actions when necessary).
- T3 *Handover*: The driver gives the order to the system for it to take over control of the car.
- *T4 Emergency Handover*: The system realizes that the human cannot continue driving (e.g., it realizes the driver has fallen asleep) and takes control of the car in a rush.
- T5 *Takeover*: The system transfers the control of the car to the human in a situation that does not require hurry (e.g., the car is approaching a city and the car cannot continue driving autonomously).
- T6 *Emergency Takeover*: The system must leave the car in a safe situation while it transfers control of the car to the human. This task takes place in emergency situations (e.g., a sensor fails and the system cannot continue driving autonomously).

7.4 Instrumentation

The instruments that were used to carry out the experiment were as follows:

- *A demographic questionnaire*: This questionnaire contains a set of 10 questions on a 5-point Likert scale that provide knowledge about the background and experience of the subjects.
- *A script to recreate the user context situation*: This is a description of the initial user context situation.
- *A users' satisfaction questionnaire*: We used a questionnaire of 11 questions containing a 5-point Likert-scale to assess the user's perception of obtrusiveness of the interactions when using the HiL simulator. This questionnaire was created based on similar instruments used in [58, 59]. Table 4 shows the users' satisfaction questionnaire that was used to measure the obtrusiveness aspect.
- *A HiL prototype for the autonomous car simulator*: We generated a HiL prototype for the autonomous car according to what is described in Subsection 5.1. This prototype was an implementation of the six collaborative tasks introduced above.
- *A HiL simulator for the autonomous car*: We used the HiL simulator introduced in Subsection 5.2. This simulator recreated the execution of the tasks that were implemented in the HiL prototype.
- *A Mac Pro 2017 Server with MacOs 10.6 (Catalina)*: This PC was used to run the HiL simulator.

7.5 Procedure

Each participant took part in the experiment throughout several validation sessions on different days. The experiment was conducted in our laboratory. Before starting the experiment, the participants were briefly presented the experiment and the main features of the autonomous car, and each participant signed a consent form. Then, they filled in the demographic questionnaire. During the validation sessions, the participants were alone in the laboratory with the experimenter. The experimenter observed the process and guided the participant when necessary.

Table 4 Users' satisfaction questionnaire

General acceptability and presentation	Q1. General acceptability	(Not acceptable—very acceptable)
	Q2. Presentation	(Not acceptable—very acceptable)
	Q3. Preferred presentation for actions	(Less intrusive—more intrusive)
General obtrusiveness of actions	Q4. Appropriate attention level	(Strongly disagree—strongly agree)
	Q5. Some actions have disturbed me	(Strongly disagree—strongly agree)
	Q6. I was not aware of some actions	(Strongly disagree—strongly agree)
	Q7. I would prefer more intrusive actions	(Strongly disagree—strongly agree)
	Q8. General obtrusiveness	(Not acceptable—very acceptable)
Interaction mechanisms	Q9. The interaction mechanisms were appropriate	(Strongly disagree—strongly agree)
	Q10. I like using the interaction mechanisms	(Strongly disagree—strongly agree)
	Q11. The context-dependent interactions were useful	(Strongly disagree—strongly agree)

Each *Sb* subject that was recruited for the experiment participated in the validation sessions according to the following procedure:

- *Sb* participated in a first validation session where the car simulator executed the prototype generated from the initial HiL design. During the validation session, the feedback gathered by the simulator was provided as input to the AI tool to infer whether the HiL actions that were executed for each task were appropriate for *Sb*. The HiL prototype self-adapted itself at runtime according to the inferences of the AI tool and refined the HiL design accordingly.
- *Sb* continued participating in validation sessions until $\delta(p_{ini}, p_{end}) = 0$, where p_{ini} was the prototype at the beginning of the session, p_{end} was the prototype at the end of the session, and δ is an operation that calculates the difference between p_{ini} and p_{end} in terms of the number of changes performed. Thus, *Sb* finished when p_{end} equaled p_{ini}, which meant that the prototype had not changed during the session since it was already fitted to the user's preferences. The last version of the prototype was achieved in a minimum of two sessions and a maximum of four sessions.

After the first validation session and the last validation session, the participants filled in the users' satisfaction questionnaire. Each session took around 30 min. At the beginning of each session, the participants were given a script to recreate the initial user context situation (e.g., user in the driver seat, attentive to the road, etc.). During the session, the user was able to change the context variables and check how the system reacted to these changes. The user was also able to keep an eye on his/her smartphone or perform other activities to not be excessively and unnaturally attentive to the system (this was particularly meaningful during the validation of the supervised autonomous driving task). All of the sessions started in the autonomous driving mode. From this mode, the set of tasks being validated were performed according to the following plan:

- T1 *Supervised Autonomous Driving*. The car is autonomously driving for 5 min.
- T5 *Takeover*. We simulate a situation where the car requires human participation. Therefore, the car transfers control to the user. When this task ends, the car is in the manual driving mode.
- T2 *Supervised Manual Driving*. The participant manually drives the car for 5 min.
- T3 *Handover*. The car prototype is in the manual driving mode. The experimenter asks the driver to transfer control to the car. When this task ends, the car is in the autonomous driving mode.
- T1 *Supervised Autonomous Driving*. The car is autonomously driving for 5 min.
- T6 *Emergency Takeover*. The car prototype is in the autonomous driving mode and transfers control to the driver because of an emergency situation. When this task ends, the car is in the manual driving mode.
- T2 *Supervised Manual Driving*. The participant manually drives the car for 5 min.
- T4 *Emergency Handover*. The car prototype is in the manual driving mode. The experimenter asks the driver to focus his/her attention on their smartphone so that the system will take control. When this task ends, the car is in the autonomous driving mode.
- T1 *Supervised Autonomous Driving*. The car is autonomously driving for 5 min.

Figure 18 shows a scenario of the execution of the *Takeover* task. The figure shows the sequence of the screenshots of the interactions that occur during a task execution that was performed during the experiment. The first interaction is shown in the upper-left quadrant of the figure, which matches with the *Take the wheel* preparatory action (the task starts with the user not having his/her hands on the wheel). According to the obtrusiveness level of this action (high, system), the system shows a text on the head-up display indicating to the user that he/she should take the wheel and the speakers read out this warning. Then, the user takes the wheel by clicking on the wheel element (see upper-right quadrant). When the user takes the wheel, the system initiates the task with the first core action, *Notify takeover* (see the center-left quadrant of the figure), which is also performed by the system at the high attention level. Therefore, the head-up display shows the notification textually on the head-up display, and the speakers read out the notification. Then, the *Confirm takeover* action is activated, and the human confirms the takeover (center-right quadrant), which is performed by the human at the slight attention level. For this action, the user simulates making pressure on the steering wheel. Once the human confirms the takeover, the head-up display shows an icon indicating that the control transfer has been successfully performed, which is shown in the lower-left quadrant of the figure. Finally, the system informs the user about the driving mode by activating the feedback action *Inform about driving mode* (see lower-right quadrant).

Note that the adaptations performed by the prototype for each user only impact the interaction mechanisms selected for carrying out an action, since the only aspect we focus on in this work is the attention level of the actions. Thus, the adaptations performed by the prototype for the fifteen users were wide ranging. For one user, the prototype changed only one interaction mechanism in one task; for another user, the prototype changed the interaction mechanisms of several actions in all of the tasks.

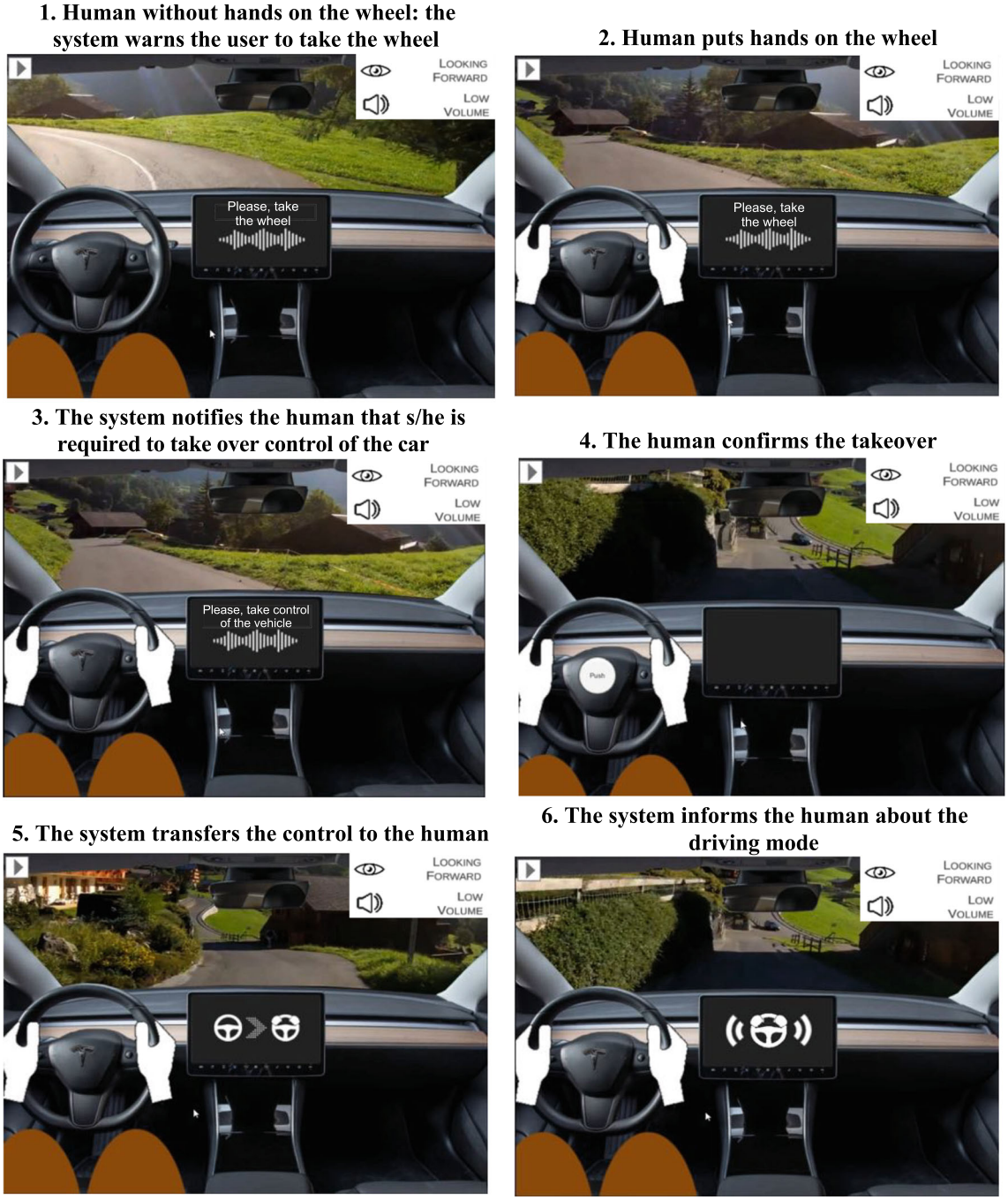


Fig. 18 Simulation scenario for the takeover task

7.6 Analysis of results and discussion

In this section, we analyze the results of the measurements performed in the experiment: human attention achievement and user perception of obtrusiveness. Although both results can be influenced by the expertise that the participants had acquired during the sessions (internal threat to validity) and the use of a subjective measure (conclusion threat to validity), we minimized these threats by spacing the sessions over time and using objective measures.

Human attention achievement. To analyze the amount of human attention that the tasks achieved, we used two measures: (1) the number of times that the fallback plans were launched and (2) the time period that the preparatory actions were executing.

Figure 19 shows the results obtained for the number of times that the fallback plans were launched during the first session of the validation (initial HiL design) and during the last session (adapted HiL design). The graph dots show the number of launched fallback plans for each participant during

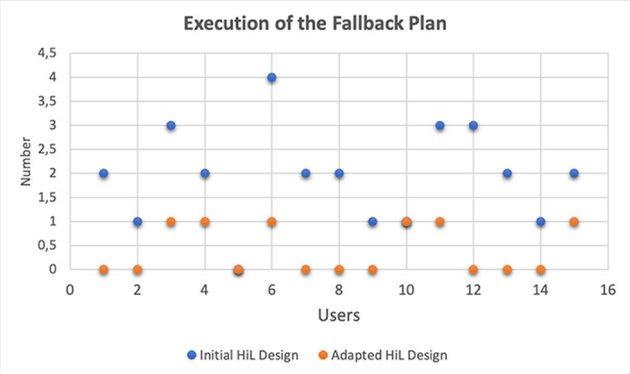


Fig. 19 Results of the execution of the fallback plan

the first session (blue dots) and for the last session (orange dots). Note that nine tasks were executed in one session. As the figure shows, the number decreased for most of the participants during the last session. Figure 20 shows a similar graphic for the time period that the preparatory actions were being executed during the first session and during the last session. In this case, the period decreased less significantly since the preparatory actions have a time constraint. The obtained results suggest that the simulator captured the attention of humans better during the last session than during the first session.

User perception of obtrusiveness. Figure 21 present the results obtained for the user perception of the obtrusiveness according to the eleven questions of the users' satisfaction questionnaire shown in Table 4. It can be observed that there is notable user satisfaction in the last session with the adapted HiL design.

Specifically, we observed the following:

- The majority of users (80%) rated the adapted HiL design as generally more acceptable (Q1), and 53% of users considered its presentation to be more acceptable (Q2). In both versions, they preferred a balanced presentation for actions that was a little more intrusive in order to be aware of them (33.4% for the initial version and 40% for the adapted version, Q3). They stated that they were more aware of actions in the last session.
- Most of the users (60%) considered the attention level of actions to be more appropriate in the adapted HiL design since it was adapted according to their obtrusiveness preferences (Q4). However, 53.3% considered that some of the actions in this design disturbed them a little bit more than the actions of the initial HiL design (20%, Q5). This is because, in the first session, most of the users (66.7%) were not aware of some of the actions (Q6) and this caused the HiL design to be adapted to be more intrusive (as users stated in Q7). With this adaptation, the general obtrusiveness in the last session was more acceptable for users (93.3%, Q8).
- The users considered the interaction mechanisms to be appropriate for both HiL designs (86.6% for the initial version and 93.3% for the adapted version, Q9), although a little bit more appropriate for the adapted one. For both HiL designs, they liked using the interaction mechanisms (60% for the initial version and 66.7% for the adapted version, Q10), and they considered the context-dependent interactions in both versions to be useful (86.6% in both versions, Q11).

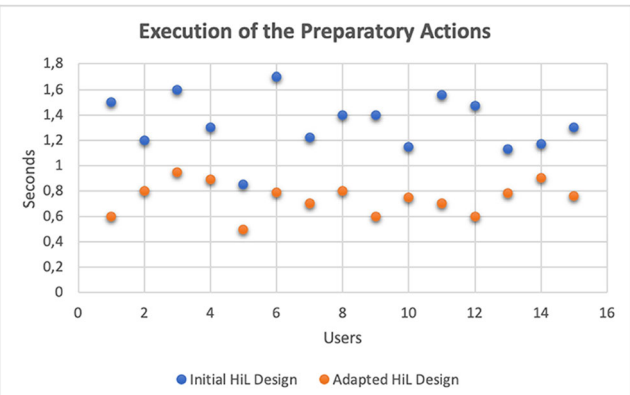


Fig. 20 Results of the execution of the preparatory actions

The results obtained in the experiment revealed that the autonomous car HiL design was significantly enhanced with the "smart" prototyping technique proposed. Both human attention achievement and obtrusiveness perception were improved. Therefore, we conclude that the "smart" prototyping technique helps achieve a HiL design that better adapts to users' preferences and needs. We interpret these results as an initial validation of the path that we have opened in this paper, where we use "smart" prototyping as a technique for achieving designs that are adapted to users' preferences and needs.

8 Conclusions

AmI environments require human involvement in a manner that conforms to the concept of *ambience*. Integrating humans into these systems is a complex task that encompasses numerous concerns. Therefore, the human participation in AmI systems must be addressed from the early development stages of the software lifecycle. In this paper, we propose a method to face the challenge of designing human involvement in AmI systems. Our framework advocates the use of high-level specifications for the design of the human-system collaboration and for user validation with prototypes enhanced with AI techniques and reconfiguration capabilities in order to "smartly" adapt these specifications to the users' preferences and needs at runtime. This way, designers have an engineering

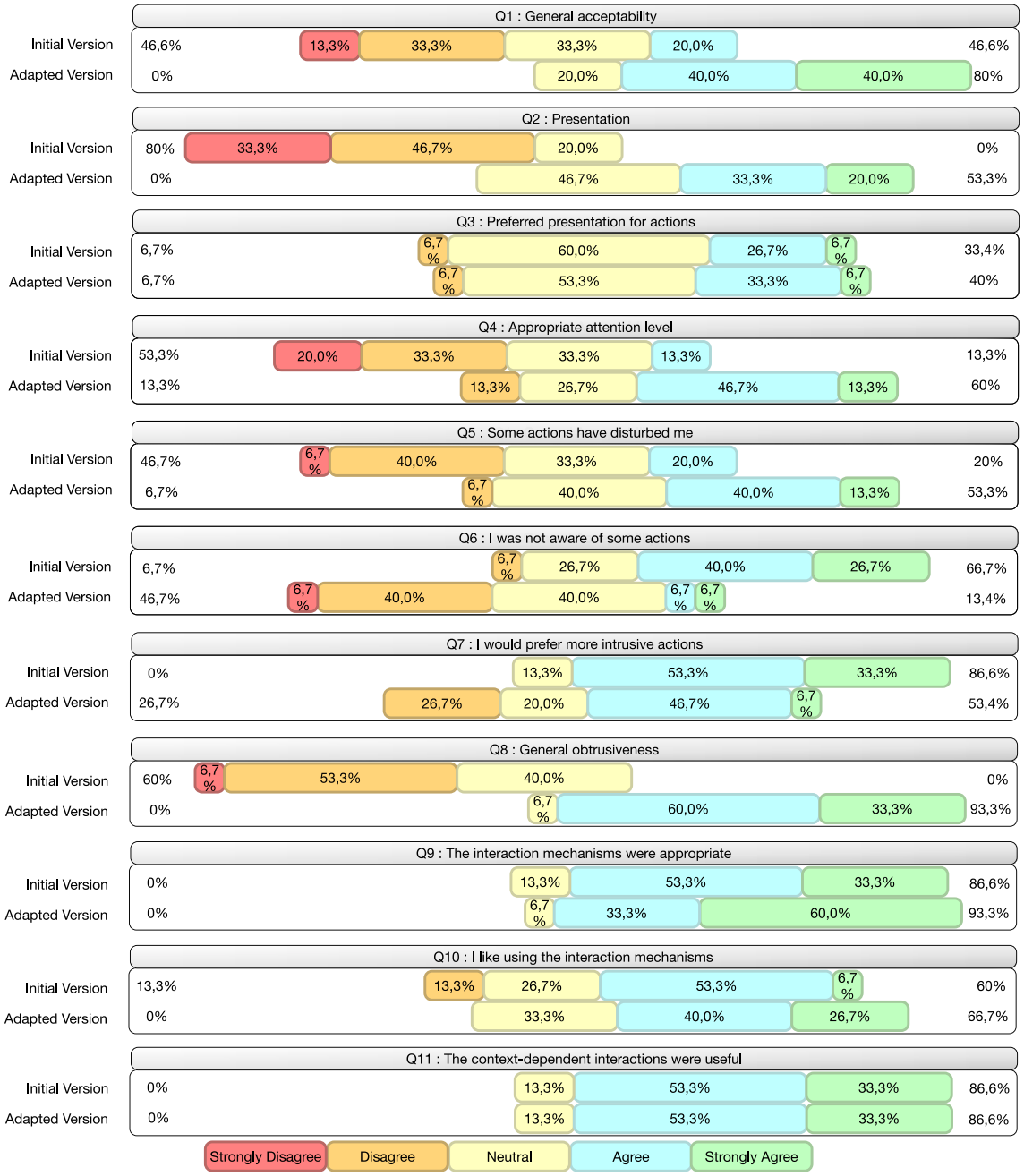


Fig. 21 Results of the users' satisfaction questionnaire

approach that assists them in systematically building human-in-the-loop AmI solutions. A key aspect of our approach is the management of human attention as a first-class concept in the specification of human-system collaboration. This leads to a design that maximizes the likelihood of success of the collaboration between the human and the system.

The results of the evaluation of the "smart" prototyping technique have allowed us to make an initial validation of this work. The results show that our "smart" prototyping method is more useful than traditional prototyping where designers observe the users interacting with a prototype and extract knowledge manually to enhance the prototypes. With the AI tool that we propose, the usage data can be used to infer new knowledge and to adjust the specified human-system collaboration automatically. At first, we tried to use this knowledge to automatically obtain a final system specification of human-system collaboration. Nevertheless, this was not possible since the technique extracts the needs and preferences of each user (not the sum of all the users that test the prototype), and the final specification should be a suitable combination of all these preferences. Therefore, we used the knowledge extracted by the AI tool to help designers build the final system

specifications. More advanced works on this issue could lead to applying AI techniques of this kind to adapt the final system specification. More complete datasets with additional information could also lead to extracting fine-grained knowledge, which could lead to other kinds of adaptations of human-system collaboration specifications. This work has allowed us to ratify a path for reducing the time to design and validate HiL specifications. Similar approaches could have a major impact on reducing the time to market and improving the quality of software systems in general.

Further work will be dedicated to developing several case studies in different domains and performing more thorough validation with designers. This validation can help us to enhance the conceptual framework by including new concepts that complete the specification of the collaborative work between humans and systems and also to improve the components of the software infrastructure that we provide for prototyping, such as the HiL simulator or the AI tool.

Acknowledgments This work has been developed with the financial support of the Spanish State Research Agency and the Generalitat Valenciana under the projects TIN2017-84094-R and AICO/2019/009 and co-financed with ERDF.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Nilsson T, Crabtree A, Fischer J, Koleva B (2019) Breaching the future: understanding human challenges of autonomous systems for the home. Pers Ubiquit Comput 23:287–307. https://doi.org/10.1007/s00779-019-01210-7
2. Streitz N, Charitos D, Kaptein M, Böhlen M (2019) Grand challenges for ambient intelligence and implications for design contexts and smart societies. Journal of Ambient Intelligence and Smart Environments 11:87–107
3. Farooq U, Grudin J (2016) Human computer integration. ACM Interactions 23(6):26–32
4. Waytz, A.: How humans and machines can live and work together (May 2019),
5. Gams, M., Yu-Hua Gu, I., Härmä, A., Muñoz, A., Tam, V.: Artificial intelligence and ambient intelligence, Tenth Anniversary Issue, Journal of Ambient Intelligence and Smart Environments 11 (2019), 71–86. IOS Press
6. Streitz, N., Privat, G.: Ambient intelligence. Final section "Looking to the future", in: The Universal Access Handbook, C. Stephanidis, ed., CRC Press, 2009, pp. 60.1–60.17
7. Gil M, Albert M, Fons J, Pelechano V (2019) Designing human-in-the-loop autonomous cyber physical systems. Int J Hum Comput Stud 130:21–39
8. Gil M, Albert M, Fons J, Pelechano V (2020) Engineering human-in-the-loop interactions in cyber physical systems. Information Software Technology 126:106349
9. Fitts, P. M: Human engineering for an effective air-navigation and traffic-control system, National Research Council (1951)
10. Sheridan, T. B: On how often the supervisor should sample, IEEE Transactions on Systems Science and Cybernetics 6 (2) (1970) 140–145
11. Lee JD, See KA (2004) Trust in automation: designing for appropriate reliance. Hum Factors 46(50–80):50–80
12. Stanton, N. A., Young, M. S: Vehicle automation and driving performance, Ergonomics 41 (7) (2010) 1014–1028
13. Cámara, J., Moreno, G., Garlan, D.: Reasoning about human participation in self- adaptive systems. In: 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 146–156 (May 2015)
14. Dorn, C., Taylor, R. N. "Coupling software architecture and human architecture for collaboration-aware system adaptation," in Proceedings of the 2013 International Conference on Software Engineering, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 53–62
15. Evers, C., Kniewel, R., Geihs, K., Schmidt, L.: "The user in the loop: enabling user participation for self-adaptive applications," Futur Gener Comput Syst, vol. 34, no. 0, pp. 110–123, 2014
16. Cranor, L., A framework for reasoning about the human in the loop, in UPSEC'08 conference on usability, psychology, and security (2008)
17. Nothwang WD, McCourt MJ, Robinson RM, Burden SA, Curtis JW (2016) The human should be part of the control loop? In: Resilience week (RWS)
18. Nunes DS, Zhang P, Silva JS (2015) A survey on human-in-the-loop applications towards an internet of all. IEEE Communications Surveys and Tutorials 17(2):944–965
19. Weiser M (1991) The computer for the 21st century. Sci Am:66–75
20. Gibson JJ (1986) The ecological approach to visual perception. Lawrence Erlbaum Associates, London **(first published in 1979)**
21. Norman, D.A.: Affordance, conventions and design, Interactions 6(3) (1999), 38–43. ACM Press
22. Christakis N (2010) The face and others: issues of communication and social psychology. Papazisis Publications, Athens
23. Chin, J., Callaghan, V., Allouch, S.B.: The Internet of things: reflections on the past, present and future from a user centered and smart environments perspective, Tenth Anniversary Issue, Journal of Ambient Intelligence and Smart Environments 11 (2019), 45–69. IOS Press
24. Langley, P: Machine learning for adaptive user interfaces. pp. 53–62. Annual Conference on Artificial Intelligence. 1997
25. Garcia-Ceja E, Riegler M (2019) Kvernberg. A. K., Torresen, J.: User-adaptive models for activity and emotion recognition using deep transfer learning and data augmentation, User Modeling and User-Adapted Interaction
26. Miller, C.A., Parasuraman, R.: Designing for flexible interaction between humans and automation: delegation interfaces for supervisory control, The Journal of the Human Factors and Ergonomics Society 49(1), pp. 57–75, March 2007

27. Cheng, B.H., et al.: Software engineering for self-adaptive systems. pp. 1–26. Springer-Verlag (2009)
28. Mirnig AG, Gärtner MA, Laminger A, Meschtscherjakov S, Trösterer M, Tscheligi R, McCall F (2016) McGee: control transition interfaces in semiautonomous vehicles: a categorization framework and literature analysis. In: International conference on automotive user interfaces and interactive vehicular applications (AutomotiveUI '17)
29. Dey AK (2001) Understanding and using context. Personal Ubiquitous Comput 5(1):4–7
30. Eskins D, Sanders WH (2011) The multiple-asymmetric-utility system model: a framework for modeling cyber-human systems, in: proceedings of the 2011 Eighth International Conference on Quantitative Evaluation of SysTems. IEEE Computer Society, Washington, DC, USA, pp 233–242
31. Buxton, B.: Integrating the periphery and context: A new model of telematics, in: Proceedings of Graphics Interface, 1995, pp. 239–246
32. Horvitz E, Kadie C, Paek T, Hovel D (2003) Models of attention in computing and communication: from principles to applications. Commun ACM 46(3):52–59
33. Ju W, Leifer L (2008) The design of implicit interactions: making interactive systems less obnoxious. Des Issues 24(3):72–84
34. Beruscha, F., Augsburg, K., Manstetten, D., Schwieberdingen, R. B. G.: Haptic warning signals at the steering wheel: a literature survey regarding lane departure warning systems (short paper) (2011)
35. Chun J, Lee I, Park G, Seo J, Choi S, Han S (2013) H. Efficacy of haptic blind spot warnings applied through a steering wheel or a seatbelt. Transport Res F: Traffic Psychol Behav 21:231–241
36. Trivedi MM, Cheng SY (2007) Holistic sensing and active displays for intelligent driver support systems. Computer 40(5):60–68
37. Winkler, S., Kazazi, J., Vollrath, M.: Distractive or supportive – how warnings in the head-up display affect drivers' gaze and driving behavior, in: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, 2015, pp. 1035–1040
38. Basili, V., Caldiera, G., Rombach, H. Goal question metrics paradigm, in: J. Marciniak (Ed.), Encyclopedia of Software Engineering, Wilely, 1994, pp. 528–532
39. Cao Y, Theune M, Nijholt A (2009) Modality effects on cognitive load and performance in high-load information presentation, in: proceedings of the 14th international conference on intelligent user interfaces, IUI '09. ACM, New York, NY, USA, pp 335–344
40. Lemmelä S, Vetek A, Mäkelä K, Trendafilov D (2008) Designing and evaluating multimodal interaction for mobile contexts, in: proceedings of the 10th international conference on multimodal interfaces, ICMI '08. ACM, New York, NY, USA, pp 265–272
41. Obrenovic Z, Abascal J, Starcevic D (2007) Universal accessibility as a multimodal design issue. Commun ACM 50(5):83–88
42. Bernsen NO (1994) Foundations of multimodal representations: a taxonomy of representational modalities. Interact Comput 6(4): 347–371
43. Reeves LM, Lai J, Larson JA, Oviatt S, Balaji TS, Buisine S, Collings P, Cohen P, Kraal B, Martin J-C, McTear M, Raman T, Stanney KM, Su H, Wang QY (2004) Guidelines for multimodal user interface design. Commun ACM 47:57–59
44. Savio, N., Braiterman, J.: Design sketch: the context of mobile interaction, in: Proceedings of MobileHCI 2007, 2007, pp. 248–286
45. Mohri M, Rostamizadeh A, Talwalkar A (2012) Foundations of machine learning. The MIT Press
46. Nair V (2010) Hinton. G. Rectified Linear Units Improve Restricted Boltzmann Machines, ICML
47. Goodfellow, I., Bengio, Y., Courville, A. 6.2.2.3 Softmax Units for Multinoulli Output Distributions. Deep Learning. MiT Press. pp. 180–184 (2016)
48. Diederik P (2015) Kingma and Jimmy Ba. Adam, A Method for Stochastic Optimization, International Conference on Learning Representations
49. de Lemos, R., Giese, H., Hausi A. Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M. Villegas, Thomas Vogel, Danny Weyns, Luciano Baresi, Basil Becker, Nelly Bencomo and Yuriy Brun, Software Engineering for Self-Adaptive Systems: A second Research Roadmap, in Software Engineering for Self-Adaptive Systems II, LNCS, Springer, January 2013, vol. 7475, pp. 1–32
50. IBM, An architectural blueprint for autonomic computing, Autonomic Computing,White Paper 2005
51. Kitchenham B, Pickard L, Pfleeger SL (1995) Case studies for method and tool evaluation. IEEE Softw 12(4):52–62
52. Wohlin C, Runeson P, Host M, Ohlsson MC, Regnell B (2000) Wesslen. A. Experimentation in Software Engineering, An Introduction, Springer US
53. Davis FD (1989) Perceived usefulness. Perceived ease of use and user acceptance of information technology, MIS Quarterly 13(3): 319–340
54. Jamieson S (2005) Likert scales: how to (ab) use them. Med Educ 38:1217–1218
55. Runeson, P. Using students as experiment subjects – an analysis on graduate and freshmen student data, in: Proceedings 7th International Conference on Empirical Assessment & Evaluation in Software Engineering, 2003, pp. 95–102
56. Litman, T.: Autonomous vehicle implementation predictions: implications for transport planning (2013)
57. Muller J No hands, no feet: my unnerving ride in Google's driverless car. Forbes
58. Gil M, Giner P, Pelechano V (2012) Personalization for unobtrusive service interaction. Personal Ubiquitous Computing 16:543–561
59. Vastenburg, M., Keyson, D.V., de Ridder, H. Considerate home notification systems: a user study of acceptability of notifications in a living room laboratory. International Journal of Human-ComputerStudies 67(9):814–826