An extension to iStar framework as alternative to support design decisions in the task analysis performed in the Human Computer Interaction Area (HCI)

DOI: http://doi.org/10.17981/ingecuc.18.2.2022.05

Artículo de Investigación Científica. Fecha de Recepción: 13/09/2022, Fecha de Aceptación: 20/09/2022

Elizabeth Suescún-Monsalve
Universidad EAFIT - GIDITIC Research group. Medellín, (Colombia)
esuescu1@eafit.edu.co

Cesar Jesús Pardo-Calvache
Universidad del Cauca - GTI Research Group. Popayán, (Colombia)
cpardo@unicauca.edu.co

To cite this paper:
E. Suescún-Monsalve, C Pardo-Calvache "An extension to iStar framework as alternative to support design decisions in the task analysis performed in the Human Computer Interaction Area (HCI)". DOI:
http://doi.org/10.17981/ingecuc.18.2.2022.05

Abstract

The goal of this work is to present i* framework as alternative to support design decisions in the Human Computer Interaction Area (HCI), mainly, in the process of task analysis. We show i* framework a modeling language suitable for an early phase of software system modeling to understand the problem domain, also, that provides methods to represent tasks, as well as support design decisions the development that satisfies the process requirements from the beginning until the end. To achieve our goal, approaches addressed to analyze notations aimed at task analysis and their representation. Finally, we present suggestions for improvement in the i* framework through a proposed extension for refining task model, instantiating examples and justifying our proposal through peer review.

Keywords

Task Analysis, GOMS, Intentional Modeling, Framework i*, HTA

I. INTRODUCTION

Task analysis is an approach that involves different techniques addressed to describe interaction between users (people) and environment in a systematic way. For that reason, task analysis is useful to elicit descriptions of what users do and represent the information on different notations. As well as define that a user must do in terms of cognitive actions or processes to achieve some goal. Indeed, task analysis could be named as a methodology which supports a set of techniques to help in the analytical process of collecting information, to organize and use this information to perform assessments or design decisions, all of that to understand user behavior [1].

Task analysis in the context of HCI aims to know who the user is and what activities they do in terms of the system. How this information is picked up through task analysis and represented on the notation [2]. That means, that notations allow the information to be formalized and analyzed and it is important to provide information about the system.

Nevertheless, techniques in requirements engineering (RE) or Software Engineering (ES) generally do not collect this kind of information support design decisions related to HCI concepts. and so do meaningful and useful early phase of software system modeling [3], to obtain information necessary to understand the activities of the user (analysis phase or high-level task) and how to represent this information on an appropriate model when designers aim to provide precise design indications (modeling phase or design phase).

Using task analysis, it is possible to get a formal description of the set of actions that a user must perform to achieve a particular goal. In a similar fashion, task analysis is based on cognitive theories that help the HCI for representation of the user and the interaction through the interface. It shapes the understanding, knowledge, intention, and processing mechanism of the user [4]. Based on these concepts, this work aims to analyze and contrast two approaches. One which is well known in HCI named GOMS model [4, 5, 29, 30] and i* (also known as iStar) modeling language or conceptual modeling languages, focusing on the intentional, used to model organizational process and interaction between actor or software agents, to represent social and strategic dimensions. This is widely known to elicit requirements in RE [6, 7, 8, 9].

This work aims to presenting framework i* and motive its use to the concepts of task analysis of HCI and show how tasks could be modeled by this framework. i* is a goal-oriented language [6], RE notation and define tasks, they "represents actions that an actor wants to be executed, usually with the purpose of achieving some goal" [1]. This work is organized in six Sections. Section 2 gives an overview of concept basis and describes the approaches used. Section 3 emphasizes what were the limitations found in the literature. Section 4 describes a questionnaire used to evaluate i* notational system and results. Section 5 presents an empirical evaluation to identify strengths and weaknesses of GOMS. Section 6 points out our proposal, finally Section 7 presents our conclusion and future works.

II. THEORETICAL FRAMEWORK

According to [10], the authors described that is necessary that the software solutions be well designed thus they can be accepted and widely used. To find this acceptance software solutions must be designed according to necessities and capabilities of the target users. For that reason, decisions related to design have a strong effect on the users and the way how software solutions are been using. Thus, software engineering researchers are interested to improve the physical, it means, aspect of software interface and somehow ensure the success of the software product.

Physical aspects (user-friendly), easy to execute the specific tasks, psychology process of people when they interact with the computers are topics widely explored HCI. In other words, HCI is addressed to study, project and design the interaction between users and computer. HCI is supported from different focuses, all of these, to develop or improve the safety, utility, effectiveness, and usability of systems that include computers [10].

In HCI area, the design has been user-centered, incorporating issues related to cognitive engineering and usability [11, 12, 28]. This last one uses cognitive models to represent two sides of the interface, the first, the system itself and the last one, the user. In a similar facet is the system from the user fashion, how a user who does not be experts executes tasks (complexity or nor), it is considered as an activity of resolution problems. And this complexity is related to the different variables involved in the environment. As an illustration, a user has goals and intentions, which are associated with his/her necessities. So, the tasks are executed based on the interaction between this user and the physical system, as a result, there are changes not only in physical variables but also in system state. Consequently, this user interprets the result based on his/her psychology goals, therefore, this user transduces the psychology intentions to physical actions on the available system mechanics [12].

According to [10] if the interface has an appropriate design (system side) and this design is based on learning and experience. In an ideal situation the psychological effort should be reduced. These issues are concerned with cognitive engineering which associate them to three kinds of knowledge design, development and technologic. Consequently, the knowledge of user tasks is an important element in these issues because this (task) will be exhibited to the user in communication with the interface.

For that reason, our analysis focuses on two proposals, first, in cognitive engineering to tasks representation with model GOMS, it has been one of the few widely known theoretical concepts in HCI to model user behavior [5], which is also used to describe, prescribe, predictive and estimate time and the way a user performs tasks on a system [29, 30]. The other is based on organizational and requirements modeling named i* [6] in the next section.

A. i Concepts and Notations*

Intentional Modeling is already used to model and implement software agent-based and organizational processes. Furthermore, it is starting to be used in conventional software development. There are different kind of intentional modeling, in this work we show the intentional modeling based on i* Framework [7, 9]. According to [6], the i* framework is based on the dependency relationships among actors.

Framework i* has two models: the Strategic Dependency (SD) model and the Strategic Rationale (SR) model. The Figure 1 shows the summary of the notation used to model with Framework i* based on the wiki-based collaborative site [7].

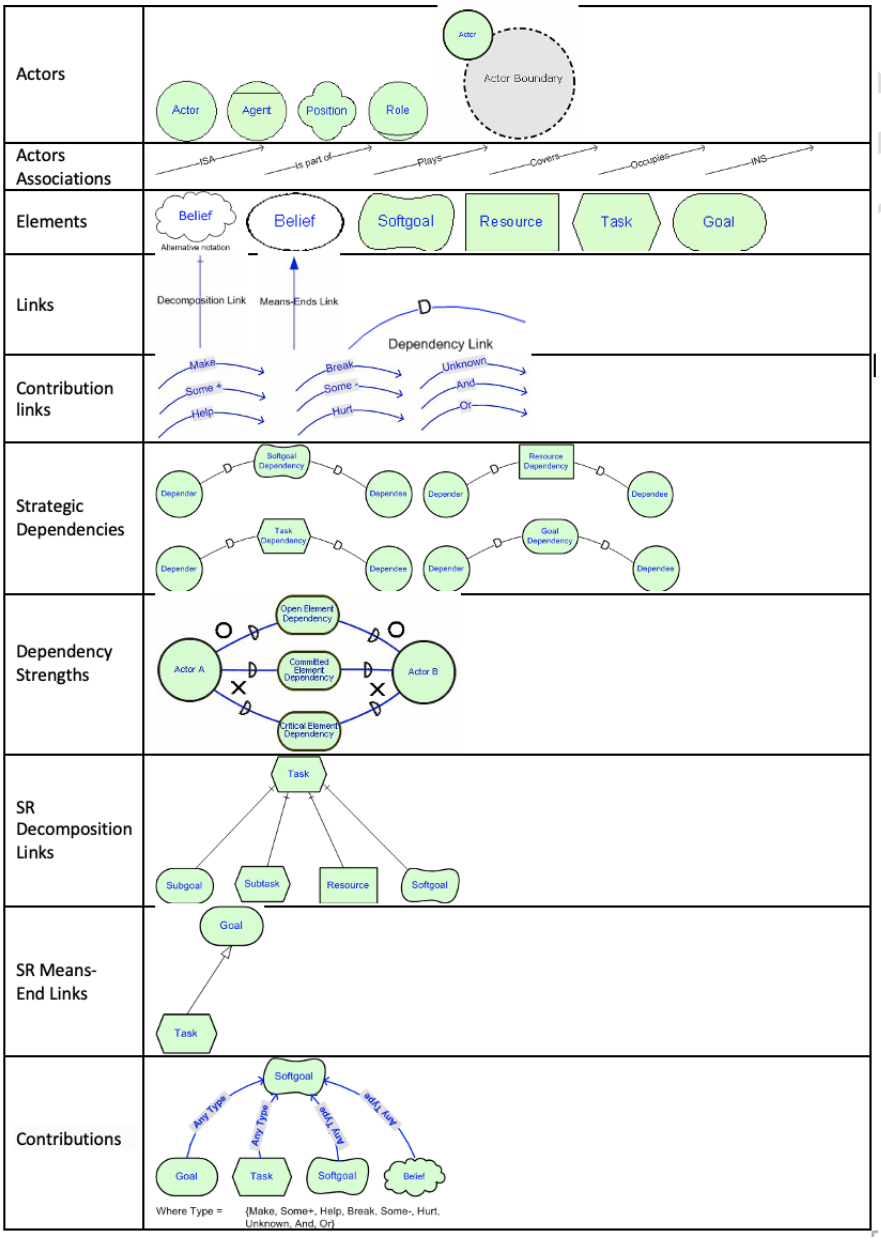

Fig. 1 Summary of i* Notation [4]

In fact, the main idea of i* involves, actors who depend on each other for goals to be achieved, for resources to be provided, for tasks to be performed, and for softgoals to be satisfied related to version 1.0 of notation [6] and qualities to 2.0 version [2]. The SD model shows the organizational context of the system as a network of dependency relationships among actors. This network contains a set of nodes and links where each node represents an actor and each link map out one dependency between two actors. Therefore, a dependency is a relationship where one actor (the depender) depends on another actor (the dependee) to achieve a goal, to perform a task, to provide a resource or to achieve a softgoal;

reflecting distinct types of freedom can be achieved by relationship. Another model is the SR which shows the representation of the "internal rationale". Specifically, the relationships behind the actors. Thus, it offers more details with the representation of the intentional relationships that are internal to the actors.

In addition, the i* modeling framework according to [6] helps to elicit goals at early stages because it represents the dependency relationship among actors and interaction between them. In the same way, i* helps to represent actors' reasons to achieve goals. These goals are decomposed into different intentional elements such as tasks. Analysis and representation of tasks are important from the perspective of this work, for that reason, this work shows how framework i* is used to represent actor tasks and to know how the i* notation works, mainly how to represent those tasks. To illustrate, the Figure 1 shows in the last item named SR Decomposition Links how the task is represented and the different elements (subtask, subgoal, resource, softgoal) to decompose this.

Intentional Modeling is already used to model and implement software agent-based and organizational processes. Furthermore, it is starting to be used in conventional software development. There are different kind of intentional modeling, in this work we show the intentional modeling based on i* Framework [7, 9]. According to [6], the i* framework is based on the dependency relationships among actors.

Framework i* has two models: the Strategic Dependency (SD) model and the Strategic Rationale (SR) model. The Figure 1 shows the summary of the notation used to model with Framework i* based on the wiki-based collaborative site [7].

In fact, the main idea of i* involves, actors who depend on each other for goals to be achieved, for resources to be provided, for tasks to be performed, and for softgoals to be satisfied related to version 1.0 of notation [6] and qualities to 2.0 version [2]. The SD model shows the organizational context of the system as a network of dependency relationships among actors. This network contains a set of nodes and links where each node represents an actor and each link map out one dependency between two actors. Therefore, a dependency is a relationship where one actor (the depender) depends on another actor (the dependee) to achieve a goal, to perform a task, to provide a resource or to achieve a softgoal; reflecting distinct types of freedom can be achieved by relationship. Another model is the SR which shows the representation of the "internal rationale". Specifically, the relationships behind the actors. Thus, it offers more details with the representation of the intentional relationships that are internal to the actors.

In addition, the i* modeling framework according to [6] helps to elicit goals at early stages because it represents the dependency relationship among actors and interaction between them. In the same way, i* helps to represent actors' reasons to achieve goals. These goals are decomposed into different intentional elements such as tasks. Analysis and representation of tasks are important from the perspective of this work, for that reason, this work shows how framework i* is used to represent actor tasks and to know how the i* notation works, mainly how to represent those tasks. To illustrate, the Figure 1 shows in the last item named SR Decomposition Links how the task is represented and the different elements (subtask, subgoal, resource, softgoal) to decompose this.

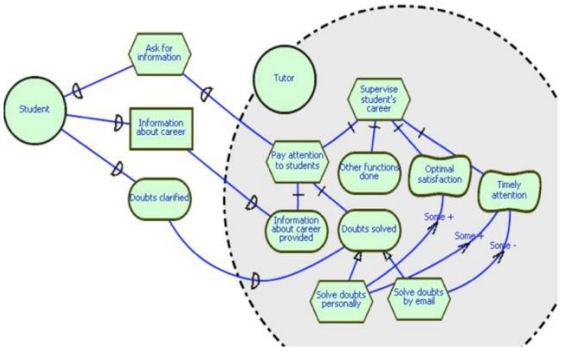

Fig 2. An i* model to illustrate Task-decomposition links [8]

To illustrate, the Figure 2 shows an i* model for an academic tutoring system. This model represents the decomposition of a task into different intentional elements. According to the authors in [8] Firstly, on the left-hand side are the SD

dependencies between a student and a tutor. That means, what the actors share to achieve the tasks. In this case, Ask for information. Secondly, on the right-hand side are the internal intentional elements of the SR model of a tutor. That means, what internal rationale the actor (tutor) must achieve the main task Ask for information. For that reason, the model includes the decomposition based on Supervise student's career task. The reader can notice that i* is highly visual and has an adequate quantity of elements to possibly facilitate human communication and problem solving.

B. GOMS Concepts and Notations

The GOMS model [13] proposed by Card/Moran is based on the mechanism of human reasoning to solve problems. For that motive, it is possible in GOMS represents activities (physical and mental) that involve work (solve problems). Each task is described as the Goal to satisfy the set of Operations that are provided by the system to the user interaction. Also, Methods are sequences of operators that accomplish a Goal. There could be more than one method available to accomplish a single goal. Finally, there is rules set of selection that determines the better alternative to each case (rules are described based on the control structure if-then). Furthermore, each task could be decomposed in other primitive subtasks building a hierarchical tree. Finally, GOMS is a group of languages that includes NGOMS and KLM [31], they are based on the point of view of the user which shows the system like an information processor. In other words, model human processor according to [13, 30], the Table 1 shows a typical example the GOMS notations and used.

The goals are "user desires" that are proposed to get something. The goals could be useful if it is necessary as a point of reference in some "error" case. Goals have information related to user intention. For that reason, it is necessary to create a series of basic operations. These operations are basic units of perception, motor, or cognitive acts whose execution is required to change some aspect of the user's mental model, or to modify the environment. As a result, this kind of action could influence the system, (click a key) or only the user's mental state (read some dialog). There is a degree of flexibility in the granularity of the operations (extent of each operation). To carry out these operations, there are several possibilities of decomposing a task into subtasks. For example, a window manager, you could close the window using the mouse on a menu or keyboard (shortcut). Each of these possibilities would be a method.

GOAL: CLOSE-WINDOW . [select GOAL: USE-MENU-METHOD . MOVE-MOUSE-TO-FILE-MENU . PULL-DOWN-FILE-MENU . CLICK-OVER-CLOSE-OPTION GOAL: USE-CTRL-W-METHOD . PRESS-CONTROL-W-KEYS]
For a specific user: Rule 1: Select USE-MENU-METHOD unless another rule applies Rule 2: If the application is GAME, select CTRL-W-METHOD
If there is more than one alternative, we could suggest series of conditions and rules to take the best options (Method): METHODS: IF (EXPERT-USER)USE-KEYBOARD-METHOD ELSE USE-MOUSE-METHOD
We could decompose the goals in subgoals: GOAL: EDIT-DOCUMENT GOAL: OPEN-DOCUMENT

Table I A typical example of GOMS notations, it was taken from [13]

As a matter of fact, the task decomposition provides us with an understanding of strategies for solving problems of the application domain. The main goal of task analysis is to produce a decomposition of tasks so that users (designers, end-users) can follow a step-by-step method of resolution.

If this is the case, GOMS can also be used to measure performance [13]. The depth of subtasks can be used to estimate the requirements of short-term memory (STM) and even to estimate response time (assuming constant time for each operation).

Briefly, the GOMS model was one of the first methods used for user interface design, having great impact on subsequent research. To describe how an expert performs tasks and breaks them down into subtasks. However, one of its weaknesses is that it only considers error-free behavior and sequential tasks. These limitations will be described in the next sections.

III. LIMITATIONS OF THE NOTATIONS

A. Limitations on the Concepts

According to [14], "to be complete", such a methodology should provide methods to support the development as well as a design that satisfies the requirements. In a similar fashion, methods and techniques that accompany a methodology should help developers or designers to validate the design based on requirements. This is due to difficult to find a methodology to define requirements and at the same time find design elements of the system based on understanding of the user tasks. Particularly, i* and GOMS include concepts and processes to support analysis and design of tasks, but they could be complemented to improve the communicability of the models. This session discusses some limitations that were found in some researchers and based on an empirical evaluation.

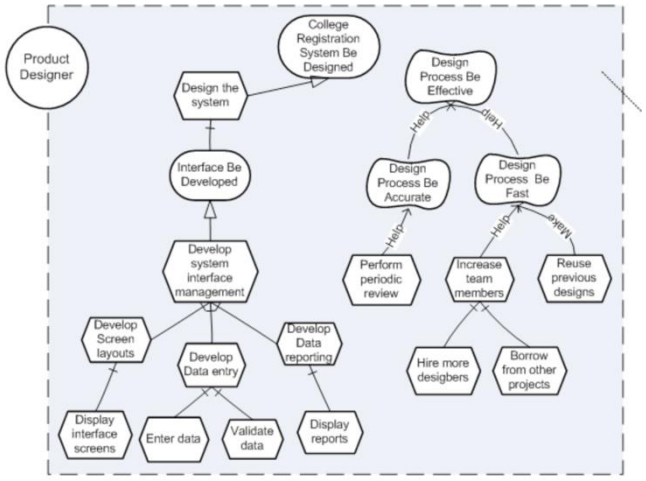

Fig. 3 An i* model to illustrate Goals-decomposition links [7]

i* modeling partially tackles task analysis by allowing designers or simply engineers to capture tasks as noted earlier in section A. i* Concepts and Notations. To complement, in [14] is defined task as an i* concept that represents, at an abstract level, a way of doing something and it also could be a means for satisfying a goal. In a same way, GOMS as is described in [15] is a description of the knowledge that a user must have to carry out tasks, in other words, it is a representation of the "how to do it" describing goals, operators, methods, and selection rules in order to learn and understand to perform these tasks.

Briefly, to assist designers in task analysis both i* and GOMS cover moderately this approach. But, to further illustrate the lack of adequate task analysis, consider the i* diagram in Figure 3, although the tasks are clearly shown, it is not possible to identify the option more adequately that user should choose, or if all tasks should be execute, or if some tasks should be omitted or simply what sequence they (tasks) should follow in order to satisfactorily be executed and how to finish this execution.

Similarly, someone might argue that GOMS could capture or represent sequentially as well as details of how to use the system. However, constraints are not represented. Namely, constraints represent, conditions or preconditions, rules, limitations, or restrictions toward the achievement of a goal. Consider the basic method used in most of Macintosh Finder to manipulation of File, Table 2 taken of [15], it seems that its weaknesses, because it only considers error-free behavior and sequential tasks, as well as, the kind of user and his/her unpredictability are disregard, it seems that each user is expert or the model is addressed to suppose that a user will know what to do at any given point.

Method for goal: drag item to destination.
Step 1. Locate icon for item on screen.
Step 2. Move the cursor to the item icon location.
Step 3. Hold the mouse button down.
Step 4. Locate the destination icon on screen.
Step 5. Move the cursor to the destination icon.
Step 6. Verify that the destination icon is reverse-video.
Step 7. Release mouse button.
Step 8. Return with a goal accomplished.

Table II To illustrate, the basic method used in most of the Macintosh Finder to manipulation of file [15].

In a similar fashion, there are other limitations, which are related to the notations. This work used the same approach presented in [15] by Moody, D. The author describes a set of principles for designing effective visual notation: these principles are optimized for understanding and problem solving. Both compose a design theory called "Physics of Notations" and this aims to explicit how visual notations communicate. All this addressed to visualizing, specifying, constructing, and documenting software systems. Visual notations are an important concern in communicating with users, customers, and novices. This work also argues that using visual notations is useful in software engineering because diagrams can convey information more concisely and precisely than ordinary language (like text), mainly to non-technical people, novices, or end-users. Due to the picture superiority effect, the information represented visually is more likely to be remembered.

Visual notations are only human-oriented representation according to [15]. Due to, there are two important issues to consider: "good visual notation". The first is cognitive effectiveness, that determines the ability of visual notation to both communicate with business stakeholders and support design and problem solving by software engineers or designers. Being that the visual notations share the same perceptual and cognitive hardware and software. And finally, design rationale, which is concerned with the process of documenting design decisions mode and the reasons that were taken. It is important because it provides traceability in the design process.

In short, the communicability of visual notations is based on theories from communication, semiotics, graphic design, visual perception, and cognition. And one the most important concern is the interpretation, because effectiveness is defined from the receiver's rather than the sender's viewpoint. That means, the intended message.

Moody, D in [15] presents enough evidence and provides a basis for explaining and predicting why some visual representations will be more effective than others. In this manner, Moody, D et al in [5] conducts a systematic analysis of the i* visual notation. Problems point out in [5] offer important contribution to our work, to name only a few: a) semiotic clarity because i* guide does not define an explicit metamodel; b) symbols overload (homographs) is happens because each graphical links has to represent more than 5 different types of relationships and they can connect different type of elements; c) perceptual discriminability, the shapes used to represent goal and belief, agents and roles, actor relationships are highly similar. As well as, the letter "D" attached to each side of the dependency is to symmetrical; d) complexity management is one the most serious limitations in practice identified, because i* does not scale well and this impacts directly the communication with end users and novices, it is shown in Figure 4 and it is a typical problem in graphical notations; e) perceptual directness, the four dependency types goals, softgoals, tasks and resources are neither discriminable nor mnemonic with the exception of the symbol for belief because this last one uses a widely recognized convention.

On the other hand, GOMS does not has visual notations but also was pointed out in [16] the textual languages encode information using sequences of characters and they are processed in a serial way. In a subsequent section will be evaluated both approaches GOMS and i* to show other focus used to evaluate notations.

B. Limitations on The Process

In addition to previous limitations, both i* and GOMS have limitations regarding the modeling of design issues. The information related to the system and user is quite ad hoc. Remembering that routes and sequences of tasks are based on the assumption, or the best or more typical options designed by the designer. Also, these methodologies do not allow conflicts to be identified between design decisions and requirements. Therefore, it is unclear how designers can capture

the knowledge about the user tasks and how they can develop an appropriate implementation that successfully meets both requirements and design decisions. In other words, the methodologies do not define how tasks modeled during the analysis process can be transformed into the design decision of the system during the design phase; consequently, this task analysis cannot be traced back to the early stage and vice versa. To conclude, early requirements should meet design decisions not only to identify conflicts but also to improve or define traceability as well as support the development of a design that satisfies these requirements in an effective way.

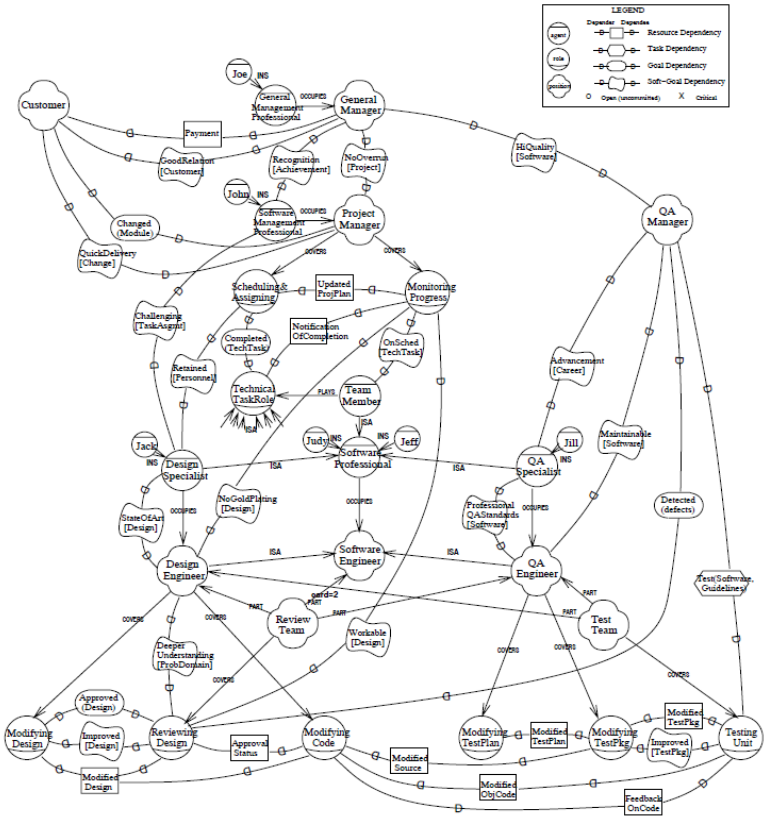

Fig 4 Complexity Management i* taken of [16]

IV. USING A COGNITIVE DIMENSIONS QUESTIONNAIRE TO EVALUATE I* NOTATIONAL SYSTEM

A. *Background*

The main goal addressed to the Cognitive Dimension Framework or CDs [17] is how to evaluate the usability of information based-on artifacts and notations, as well as it is proposed as a user-centered discussion tool to make quick but useful evaluations.

CDs proposes 12 dimensions, as described in [18] each dimension describes an aspect of an information structure, and each dimension can be manipulated independently of each other. Therefore, it is necessary to consider the following, in each evaluation must always consider the activities to be supported (what the system notational supports). All and all, CDs are intended for analyzing and discussing issues like doing a kind of design activity, internal dependencies contained in the language and how much the order's user actions are restricted and represented.

Specifically in this work was used the dimension related to evaluate system notational. System notational is focused to evaluate the world of information artifacts or the properties of user activities information artifacts. For that reason, the aim addressed in the CDs framework is to answer this question "are the users' intended activities adequately supported by the structure of the information artifact?" to provide an approach to usability analysis that could be used for anyone, even non-specialist who will find it easy to understand and easy to use.

Moreover, usability is not enough because the information artifacts exist in a social or organizational situation as is explained by the theory of i* [6]. In a similar fashion as it was pointed out in [19] software design educators suggest that

design tools are not sufficient for the design of novel user interfaces. For that reason, notations should be intended to represent at various abstraction levels, or alternatively, be well defined.

B. *Cognitive Dimensions Questionnaire Features*

As noted earlier, this work is intended to propose and suggest framework i* as an alternative to support design decisions related to concepts of HIC in early stage that satisfies the process requirements, more precisely to task analysis. Therefore, this work presents recommendations to extend the notations based on the evidence found in the literature cited. before that, we will do an analysis the notation in i* based on the guidelines suggested in [17]. The motivations cited in [19] are important to highlight in our work. To name only a few, the authors describe that it is necessary for usability, features like, the structural properties of a notation, the properties and resources of an environment and the type of activity. They are essential because the preconditions to usability are derived from them.

The CDs framework is an approach that according to [19] is useful to apply because it offers a compressible evaluation in a general way, uses terms that were readily comprehended by non-specialists. As well, it can be applied not only to interactive devices, but also to paper-based notation and other non-interactive information systems, which are focused on HCI. And finally, to distinguish between the needs of different types of "user needs", for instance, the difference between dictation tasks and design tasks.

C. *Evaluation of i* by pairs*

The evaluation by pairs [20] of i* was based on a set of features that have been considered important in the context of task analysis and understanding of the notation. It is important to note that the evaluation was addressed for people who are used to dealing with framework i*. After all, we expected that the discussion generated was as productive as possible. In all, eight research who are expert users of i* framework, they were invited in PUC-Rio (Pontificia Universidade Catolica do Rio de Janeiro, Rio, Brazil), the RE-Group (requirements engineering group) and affiliated to the group Software Engineering, they accepted to participate in the evaluation.

To name only a few of skills and characteristics related to the participants:

About Section 2 - Definitions:

How long have you been using i* notation? Between 8 years and 4 years

Do you consider yourself proficient in its use? most of them consider themselves proficient.

Have you used other similar systems? (If so, please name them) most of them have not used similar systems. Systems refers to i* notational system. The evaluation had begun with a brief introduction and instructions. In order to assure the correct selection of features to be evaluated, we based our evaluation on relevant features that have been proposed in [17] to evaluate notation systems. The questionnaire adapted according to our necessities is available at: https://bit.ly/3lOLfjK, besides, the consent is available at: https://bit.ly/3yZEbJg.

Once the participants had filled the questionnaire, the aim was to generate the discussion. Participants made a judgment about concepts, definitions, criteria and i* notation. The dynamic of the evaluation was represented in Figure 5, it was described according to iTrace notation describes in [21].

To name only a few of the important issues raised and the general agreement in the discussion we presented Table 3. When using the system, what proportion of your time (as a rough percentage) do you spend:

In general, on these points, the answer was according to participants' proficient. Section 4 – Questions about the main notation is available at: https://bit.ly/3GjpEJL.

What task or activity do you use the system for?	Most of them have used it to model Multi-agents System, Requirements and intentions.
What is the product of using the system?	SR and SD Models
What is the main notation of the system?	A graph

Table III Parts of your system

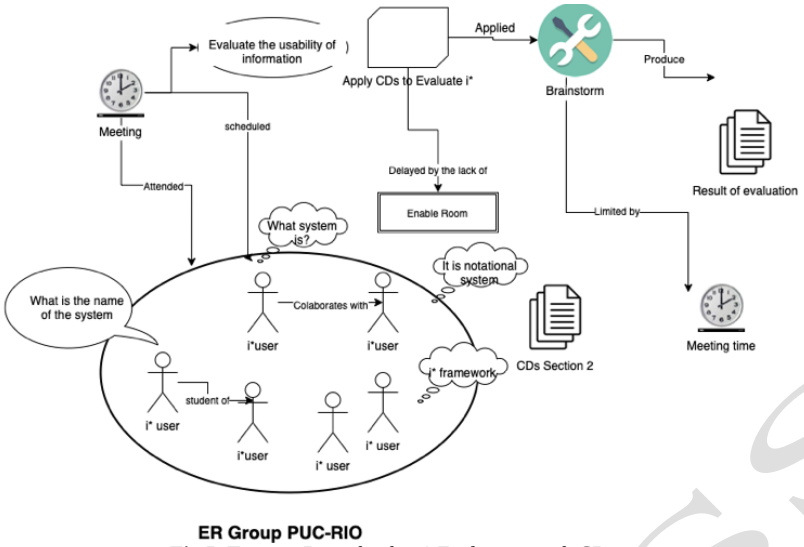

Fig 5 iTrace to Describe the i* Evaluation with CDs

*D. Considerations about Evaluation of i**

In short, this discussion tool offered elements to generate good concepts and capture enough important aspects of i* framework and its notational system. Furthermore, CDs was useful because allowed experts and general users to make judgments and reach agreement among themselves about i* framework. This last one focused on task analysis. With this activity, we had noticed that users when judged the i* notational system they were a little conditioned to the tool where they usually model with i*. Experts are more confident to use i*, consequently they proposed less changes. When the questionnaire used the word "system" this had generated misunderstanding among participants because they thought about software systems nor notational systems. Finally, some questions into the questionnaire are addressed to method nor to the language or notation. It could be a suggestion for improvement for CDs but that is outside the scope of our work.

V. USING AN EMPIRICAL EVALUATION TO IDENTIFY STRENGTHS AND WEAKNESSES OF GOMS

We have used the same approach present in [3] which shows an empirical evaluation of the i* framework. The definitions related to the evaluated features are also available there. Indeed, we also aimed to analysis the same features but in a shallow way in order to contrast GOMS and i* framework. As follow:

Feature 1, Refinement Evaluation: Not well supported. The same as reported to i*, there are several limitations when the GOMS model grows in size and complexity. It also was pointed out in [16], there is described the well- known problem of some notations. The reason is that both notations do not scale well. It is an important problem to consider because non-experts (novices) and end-users are less equipped to deal with complexity.

Feature 2, Modularity Evaluation: Well Supported. There are mechanisms in GOMS which are used to build blocks that can carry out with specific functionality and represent fragments of process.

Feature 3, Repeatability Evaluation: Not well Supported. GOMS is a textual language, it uses sequences of characters in natural language without formal pattern for descriptions. For that reason, similar settings can be described using different primitives. As a result, models derived from GOMS can be misunderstood, or incorrectly treated and used or according to the user' proficient.

Feature 4, Complexity Management Evaluation: Not well Supported. There are not mechanisms to deal with large and complex problems, as well it does not have high-level and lower-level view, dependencies between elements are not possible to model. These weaknesses are related to modularity features too.

Feature 5, Expressiveness Evaluation: Not well Supported. This feature is widely justified in [15]. This work argues that visual representations are more effective than textual languages. At the same time, GOMS is not equipped with enough elements to represent the whole behavior system.

Feature 6, Traceability Evaluation: Not Supported. It is possible to add new sentences to enrich the model but there are no mechanisms to allow evolution modeling to be monitored.

Feature 7, Reusability Evaluate: Not Supported. As mentioned above, GOMS is a modeling based on textual language, hence, it uses natural language in its notation. For that reason, it is too difficult to find and apply mechanisms for properly managing reusability of parts of a system model. Despite this, GOMS allows models to be built in blocks. It is complicated to reuse certain fragments of a model.

Feature 8, Scalability Evaluation: Not Supported. GOMS and other notations have the same problem, which is hard to solve because small systems are well represented but when problems or systems grow in size and complexity, it is hard to use and analyze. The scalability problem is also a direct consequence of the lack of mechanisms for modularization and complexity management, as well, it was reported to i* in [3].

Feature 9, Domain applicability Evaluation: Well Supported. GOMS literature shows several examples in which it is well applied to help processes in HCI mainly in task analysis [4, 5].

In brief, Table 4 present a summary of the results obtained from both evaluations. The first reported in [3] and the second one the empirical evaluation made in our work. Both approaches have similar strengths and weaknesses. Nevertheless, being i* a visual notation carries on with advantages based on theory and empirical evidence about cognitive effectiveness of visual representations, in a similar fashion, i* has a lot of elements to represent widely the system behavior.

Evaluation criteria	Evaluated Issue	i*	GOMS
Modeling Language	1. refinement	Not well supported	Not well supported
	2. modularity	Not supported	Well supported
	3. repeatability	Not well supported	Not well supported
	4. complexity management	Not well supported	Not well supported
	5. expressiveness	well supported	Not well supported
	6. traceability	Not well supported	Not supported
	7. reusability	Not supported	Not supported
Pragmatics	8. scalability	Not supported	Not supported
	9. domain applicability	well supported	well supported

Table IV Results of the Empirical Evaluation adapted [3]

VI. EXTENDING I* TO SUPPORT HCI CONCEPTS

As noted earlier, although task analysis can be modeled with i* [22] and other notations [10, 13, 23], few task analysis techniques have been proposed related to requirements processes and how this could be support to HCI processes to identify design decisions and requirements through the understanding of the users' tasks. Furthermore, the complexity that involves i* modeling has not allowed novices, end users or non-experts to understand and follow easily what is represented through of the model. To be fair, inherent complexity of i* modeling is still a new proposed in requirements engineering literature [3, 15, 16].

Task analysis should be focused to represent both user interface design and performance to reach user goals. As was noticed in [24, 28] usability is aimed to interactive systems to "focus on the users and their tasks", to be more specific, to represent the logical activities that should support users in reaching their goals.

In order to be more meaningful and useful, task analysis should be developed through and interdisciplinary collaborative effort, as noted earlier, involving the various viewpoints of ER and HCI. Furthermore, it should be intended to represent various abstraction levels as functional aspects and of interaction. Alike, trying to be more effective and reach a greater number of users like experts, non-experts, novices, and final users.

It is declared in [24], when designers want to specify only requirements, then task analysis could consider only the main high-level task, in contrast, when designers aim to provide precise design indications then task analysis should represent a greater granularity, this statement also drives our own investigation.

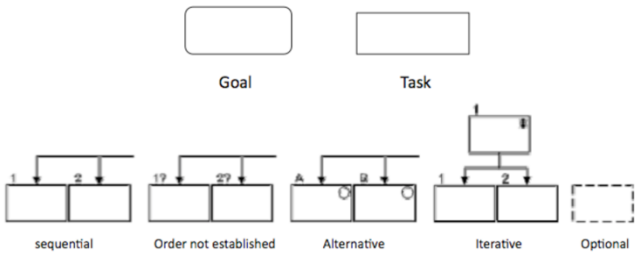

Fig 6 Notation in HTA [26]

For that motive, extensions to i* are proposed to enhance its task analysis capabilities. The extensions are motivated by HCI theory and cognitive theory showed in the II. Theoretical Framework section Concepts and Notations. We have also analyzed GOMS in different ways, presented concepts and notations, limitations on the process and used and

empirical evaluation to identify strengths and weaknesses of GOMS in order to contrast with i* framework. Furthermore, we showed that i* framework has the elements to support HCI concepts when task analysis is required. Our aim is to present the motivation to improve i*, some coming of literature cited and our own improvements based on analysis in this work. Within these improvements, it is necessary to introduce Hierarchical Task Analysis (HTA) as described in [25, 26].

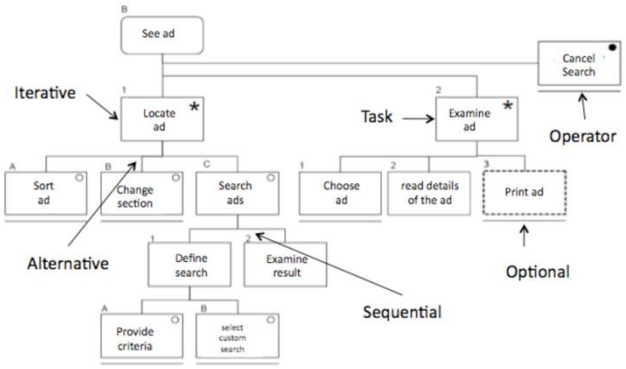

Fig 7 Example with HTA Notation [26]

Being HTA an approach top-down which considers goals of high-level and actions to reach goals, we found relevant to use in our work. Just like, the decomposition is similar as shown in i*, because HTA allows tasks to be characterized, as In short, the extensions to i* modeling with task analysis techniques that enable the cohesion of HCI concepts and Cognitive Engineering to be assessed, are detailed further in this section:

According to [16] dependencies could be represented more clearly using conventional arrows, it is shown in Figure 8. Using arrows that are universally understood could solve the directness problem.

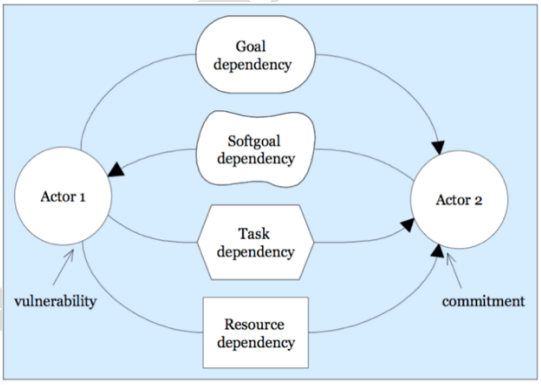

Fig 8 Strategic Dependencies, Suggested Improvement in [16]

In a similar fashion, in [16] is proposed to take maximum advantage of the power of human visual processing. It is shown in Figure 9 that represents with a thick red line a critical dependency between Actor 1 and Actor 2.

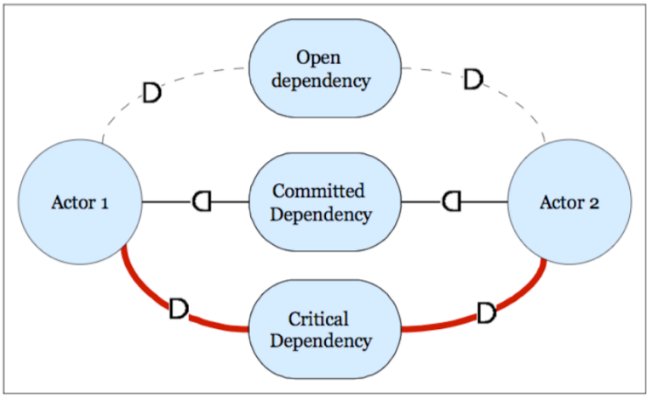

Fig 9 Graphical Encoding of Dependency Strengths, Suggested Improvement in [16]

With regard to our suggested improvement, this is addressed by HTA. The notation in HTA is a point to referent to solve some problems pointed out earlier in this work to respond the next question:

Are the tasks clearly shown? it is not possible to identify the option more adequately that user should choose, or if all tasks should be executed, or if some tasks should be omitted or simply what sequence they (tasks) should follow in order to satisfactorily be executed and how to finish this execution. The result of our proposal is shown in Figure 10 as an instance of the example shown in [26], this proposal could be used to improve i* usability and effectiveness, mainly for communicating with non-experts, novices and end-users who possibly are familiar with conventional notations. Likewise, GOMS could be useful when textual notation will be required to clarify the process represented. Finally, the approach presented in [27] could be used to solve problems like Complexity Management reported in [16] because this approach proposes to decompose each diagram in a single monolithic diagram. To get better visual representation.

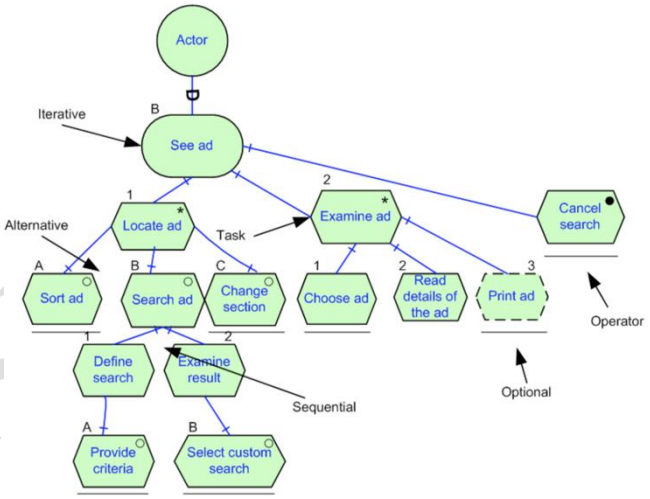

Fig 10 Using HTA Notation into the i* Diagram

VII. CONCLUSION AND FUTURE WORKS

The main contribution of this work has been to introduce framework i* as alternative to support design decisions in the Human Computer Interaction Area (HCI), mainly, to task analysis. We are aware that it is hard to find some notation which meets all necessities into the software development process. As regards our propose, it is possible to combine the more relevant and accepted features, predominantly in HCI notations to increase the i* notation. All of that is intended to improve the communication between different users like experts, non-experts, novices and end-users.

As mentioned in [16] framework i* is one the most important goal-oriented and organizational modeling in the RE field. Unfortunately, it does not consider the design decisions related to HIC process.

In order to be more meaningful and useful, task analysis should be developed through and interdisciplinary collaborative effort, as noted earlier, involving the various viewpoints of ER and HCI. Furthermore, it should be intended to represent at various abstraction level of both functional aspects and of interaction. When designers want to specify only requirements, then task analysis could consider only the main high-level task, in contrast, when designers aim to provide precise design indications then task analysis should represent a greater granularity. We agree with [24], task models should represent the intersection between user interface design and more engineering approaches by providing designers with a means of representing and manipulating a formal abstraction of activities that should be performed to reach user goals.

In addition to we analyzed advantages and disadvantages in both approaches, GOMS and i* framework and we found high-level of acceptance when users use them, as well the characterization of each notation. Being i* framework more advantageous due to its graphical and enrich notation.

To be fair, we take some improvements reported in literature [3, 15, 16] and put them together with our own suggested improvements which are based in HTA notation and GOMS modeling.

As future works, we propose to use Cognitive Dimensions Questionnaire to evaluate GOMS, thus generate a similar discussion as that reported with framework i*. In a similar fashion, it is necessary to extend the empirical evaluation to identify strengths and weaknesses of GOMS as reported in [3], this could find out more evidence than these produced in this work. As well, it is required to make experiments using the modifications proposes in i* notations to identify as suitable is the change. Finally, it necessary to make new analysis of i* notations but it should be focused on visual representation aspects, which the CDs framework had excluded.

ACKNOWLEDGMENT

Professors PhD. Elizabeth Suescún and PhD. César Pardo is grateful for the contribution of the EAFIT University (Medellín) and the Universidad del Cauca (Popayán), where they currently work as Assistant Professor and Associate Professor, respectively.

VIII. REFERENCES

[1] X. Yang, J. Hyup Kim, and R. Nazareth, “Hierarchical task analysis for driving under divided attention,” Proc. Hum. Factors Ergon. Soc. Annu. Meet., vol. 63, no. 1, pp. 1744–1748, 2019.

[2] D. Kieras, “GOMS models for task analysis. The handbook of task analysis for human-computer interaction,” pp. 83–116, 2004.

[3] O. Pastor, H. Estrada, and A. Martinez, “Strengths and Weaknesses of the i* Framework: An Empirical Evaluation”,” in Social Modeling for Requirements Engineering, The MIT Press, 2011.

[4] F. Dalpiaz, X. Franch, and J. Horkoff, “iStar 2.0 Language Guide,” arXiv [cs.SE], 2016.

[5] D. Kieras and B. John, The GOMS Family of Analysis Techniques: Tools for Design and Evaluation”, CMU-HCII-94-106. 1994.

[6] E. Yu, “Modeling strategic relationships for process reengineering,” in Social Modeling for Requirements Engineering, The MIT Press, 2010.

[7] “I* wiki,” i* Wiki. [Online]. Available: http://istar.rwth-aachen.de/tiki-index.php?page=iStarQuickGuide#Basic_i_Notation. [Accessed: 24-May-2022].

[8] C. Cares, X. Franch, E. Mayol, and C. Quer, “A Reference Model for i*”,” Social Modeling for Requirements Engineering, vol. 17, 2011.

[9] “Human-computer interaction, 3rd edition,” Pearson.com. [Online]. Available: https://www.pearson.com/uk/educators/higher-education-educators/program/Dix-Human-Computer-Interaction-3rd-Edition/PGM577781.html. [Accessed: 24-May-2022].

[10] S. D. J. Barbosa and B. S. Silva, “Interação Humano-Computador”,” Design e Avaliação de Inter- faces Humano-Computador. SBC Serie, 2010.

[11] D. A. Norman, Cognitive engineering. User centered system design. 1986.

[12] Ayalon, O., & Toch, E. (2021). User-centered privacy-by-design: Evaluating the appropriateness of design prototypes. International Journal of Human-Computer Studies, 154, 102641.

[13] S. K. Card, T. P. Thomas, and A. Newell, The Psychology of Human-Computer Interaction, Lon- don: Lawrence Erbaum Associates. Lon- don: Lawrence Erbaum Associates, 1983.
[14] H. Mouratidis and P. Giorgini, "Secure Tropos: Extending i* and Tropos to Model Security Throughout the Development Process"," in The MIT Press (ed) Social Modeling for Requirements Engineering, 2011.
[15] D. Moody, "The 'Physics' of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering"," IEEE Trans. Software Eng, vol. 35, no. 6, pp. 756–779, 2009.
[16] D. Moody, P. Heymans, and R. Matulevicius, Improving the Effectiveness of Visual Representations in Requirements Engineering: An Evaluation of i* Visual Syntax", 17th IEEE Inter- national Requirements Engineering Conference. Atlanta, Georgia, USA, 2009.
[17] "Cognitive Dimensions Framework," Cam.ac.uk. [Online]. Available: http://www.cl.cam.ac.uk/%7Eafb21/CognitiveDimensions/. [Accessed: 24-May-2022].
[18] T. R. G. Green, "Instructions and descriptions: some cognitive aspects of programming and similar activities"," in ACM Press Proceedings of the Working Conference on Advanced Visual Inter- faces, 2000, pp. 21–28.
[19] A. Blackwell and T. Green, "Notational systems -- the cognitive dimensions of notations framework"," in HCI Models Theories and Frameworks Toward a multidisciplinary science, vol. 288, Morgan Kaufmann Publisher, 2003, pp. 103–134.
[20] C. Blanco and P. Sánchez, "Aplicando evaluación por pares: análisis y comparativa de distintas técnicas," Jornadas de Enseñanza de la Informática, 2012.
[21] M. Serrano and J. C. S. Leite, "A rich traceability model for social interactions", in InTEFSE '11 Proceeding of the 6th international workshop on Traceability in emerging forms of software engineering, 2011.
[22]A. Sutcliffe, "Analyzing the Effectiveness of Human Activity System with i*"," in The MIT Press (ed) Social Modeling for Requirements Engineering, 2011.
[23] R. E. Eberts, User Interface Design. London, England: Prentice-Hall, 1994.
[24] S. Munir, "Cognitive modelling for user interface design in HCI: A comparative analysis on cognitive models," Research Journal Of Computer Science And Information Technology, vol. 4, no. 2, pp. 117–129, 2020.
[25] R. O. Prates and S. D. J. Barbosa, "Introdução à Teoria e Prática da Interação Humano Computador fundamentada na Engenharia Semiótica"," in Tomasz Kowaltowski and Karin Breitman (orgs.) Atualizações em informática. XXVII Congresso da Sociedade Brasileira de Computação. Jornadas de Atualização em Informática (JAI), 2007.
[26] F. Paternò, "ConcurTaskTrees: An Engineered Approach to Model-Based Design of Interative Systems". The Handbook of Analysis for Human Computer Interaction, Publisher: Lawrence. Erlbaum Associates Publishers, 2002.
[27] A. P. A. Oliveira, "Engenharia de Requisitos Intencional: Um Método de Elicitação, Modelagem e Análise de Requisitos. Tese de Doutorado," Puc-rio.br. [Online]. Available: https://www.maxwell.vrac.puc-rio.br/colecao.php?strSecao=resultado&nrSeq=13061@1. [Accessed: 24-May-2022].
[28] J. Nielsen, Usability Engineering. Morgan Kaufmann, 1994.
[29] X. Zhou, F. Teng, X. Du, J. Li, M. Jin, & C. Xue, H-GOMS: a model for evaluating a virtual-hand interaction system in virtual environments. Virtual Reality, 1-26, 2022.
[30] P. Setthawong & R. Setthawong, R. Updated Goals Operators Methods and Selection Rules (GOMS) with Touch Screen Operations for Quantitative Analysis of User Interfaces. Int. J. Adv. Sci. Eng. Inf. Technol., 9(1), 258. 2019.
[31] Cunha, D., Duarte, R. P., & Cunha, C. A. KLM-GOMS Detection of Interaction Patterns Through the Execution of Unplanned Tasks. In International Conference on Computational Science and Its Applications (pp. 203-219). Springer, Cham, 2021.

Elizabeth Suescún-Monsalve PhD in Computer Science at the Department of Informatics, Master degree in Computer Science and System and Informatics Engineer. Professor Universidad EAFIT. Member of the GIDITIC research group. https://orcid.org/0000-0001-7872-7638

Cesar Jesus Pardo-Calvache Systems Engineer, Master in Engineering and Computing, PhD of Engineering. Professor Universidad del Cauca. Popayán, Colombia. Member of the GTI research group. https://orcid.org/0000-0002-6907-2905