Easy Development of Communicative Behaviors in Social Robots

Chao Shi, Takayuki Kanda, Member, IEEE, Michihiro Shimada, Fumitaka Yamaoka, Hiroshi Ishiguro, and Norihiro Hagita, Senior Member, IEEE

Abstract—This paper presents a development framework for social robots with which developers can easily prepare communicative behaviors based on tag-based sentences. Previous literature in human-robot interaction has revealed various useful non-verbal behaviors. But for developers, integrating such a large amount of non-verbal behaviors each time they build a social robot is not realistic. The more repertory of non-verbal behaviors acquire, the larger is the burden faced by developers to implement the non-verbal behaviors. Our software, however, only requires developers to prepare sentences with simple markup language for controlling explicit non-verbal behaviors and utterances. The backend system analyzes the tag input and adds implicit non-verbal behaviors accordingly. With the help of previous literature, the system is equipped with various communicative behaviors, so that the robot autonomously adjusts its gazes, gestures, and standing points. The system’s effectiveness is demonstrated with examples on robot interaction where the system receives only simple sentences with scripting language to produce complex and communicative robot behaviors.

I. INTRODUCTION

Social robots are expected to serve as communicative partners in such daily environments as museums, shops, and homes. Their human-like body properties, such as a head, eyes, and arms, will be used for non-verbal interaction in addition to natural language utterances (Fig. 1) to enable human-like interaction that is as simple as “talking to a person.” Previous literature in human-robot interaction demonstrated the importance of pointing [1-3], gazing [4-7], nodding [8], and proximity [9-14]. In addition, the importance of timing between verbal and non-verbal behaviors has been demonstrated [15-17]. However, framework for such social robots hasn’t been proposed.

Since robot researchers will probably continue to accumulate a large amount of knowledge about non-verbal behavior for human-robot interaction, developers require assistance to use them. They are already busy with other robotics programming, including using various sensors, adjusting hardware parameters, integrating sensory output from many cognitive modules, combining sensory input and stored memory about interacting persons to decide the robot’s actions, and controlling actuators to express behavior. Such complexity can be seen in the recent cognitive architectures for social robots [18, 19].

This study addresses a framework to ease the development process of social robots by concentrating on the control of non-verbal behaviors. We explore the minimal input from developers and design architecture to autonomously control implicit behaviors while accepting control input for explicit gestures. Our approach’s effectiveness is demonstrated with examples where developer inputs are minimal in comparison with the complexity of the interactive behavior expressed by the robot.

II. RELATED WORKS

A. Development framework for ECA

Many researchers have worked on development framework for embodied conversational agents. In the study of embodied conversational agents (ECA), there are two main approaches for helping developers create interactive agents with input.

In one approach, the system only requires developers to provide text for speech and analyzes the text to add necessary gestures. Cassell proposed a system called BEAT [5], which can analyze the words of a speech and automatically generate motions corresponding to the conversation. Since developers are interested in creating believable behavior, they are greatly helped by adding motions.

In contrast, when we consider developing a social robot, developers require many gestures to provide information that cannot only be supplied with a text for utterance. When a robot points at an object and says, “look at this,” apparently the robot needs information about the target indicated by the spatial deixis “this” . Thus, we cannot build a fully automatic system to generate motion only from text for utterances. Instead, we need to ask developers to supply additional information to accompany the text for utterances.

In the second approach, a couple of studies proposed a system based on scripting or markup language in embodied conversational agents. For example, Kranstedt et al. proposed a system called Multimodal Utterance Representation Markup Language (MURML) that uses xml-like tags in combination with utterance text so that developers can...
specify arm movements and head motions that correspond to utterances [6]. MURML allows developers to specify such detailed motions as the orientation of the palm. Such scripting language was improved and extended into Behavior Markup Language (BML), which allows more in-depth description including synchrony [7]. Researchers have also studied agent architecture for generating communicative behaviors. For example, Function Markup Language (FML) models intention and other variables behind the generation of gestures [8].

A few studies have developed a scripting language for humanoid robots. Nishimura extended MPML and proposed a system called Multimodal Presentation Markup Language for Humanoid Robots (MPML-HR) that accepts the tag-based notation of motions accompanied with utterance text [9]. Moubayed et al. extended BML for AIBO, a dog robot [10]. However, since both studies just extended scripting language used in embodied conversational agents, the details of all motions need to be specified, and their systems are only concerned with robots staying at pre-defined locations.

B. Toward scripting language for social robots

Since the current ECA frameworks are not suitable for a social robot, we need to extend them into a new framework. To build such a system, at first we need to analyze the difference between ECA and social robots.

The major difference between embodied conversational agents and social robots is that the latter need to work in the real physical world. In such environments, however, major difficulties complicate in the development of scripting language. First, everything is mobile; people walk, and the target objects move. Unless the system can recognize the positions of mobile entities, the developer needs to manage the positions in higher-layer modules, including dialog management, which further complicates development. For example, MPML-HR [9] obtain moving and pointing functions, but it still requires information about target positions in the world system, i.e., x-y coordinates. We believe the system must manage such dynamic information itself.

Second, real-time response is critical in social interaction. Previous HRI literature revealed the need for such quick verbal and non-verbal responses as nodding, looking, moving arms, and adjusting the standing position [2, 11-13, 15-17, 20]. This also makes the control of behaviors more complex. Since such responding motions interfere with other intentional motions like gesturing, complex constraints among non-verbal behaviors need to be processed. No previous literature has addressed a development framework to support such complex arrangement among non-verbal behaviors.

One more important difference between robots and embodied conversational agents is that robots need real actuators, which force the degrees of freedoms to be small and the speeds of the motions to be slow. In embodied conversational agents, high degrees of freedom are often assumed [6-8]; but for robots such degrees of freedom are very expensive. While some forefront robots are being researched with high degrees of freedom [18], the most commonly used social robots for research have approximately ten degrees of freedoms; this limits the variety of gestures. Moreover, these robots have motors in a real world, so completing movements requires time, which is not a feature addressed in any scripting language for embodied conversational agents. Thus, developers for social robots are more interested in handling such time constraints and less interested in using such detailed gestures as controlling the palm.

III. Embodied behavior for communication robots

What kind of gestures and motions do social robots need to express? What kind of information does a developer need to provide to generate such gestures and motions? In this section, we overview previous literature in order to implement them into our software framework. Our basic policy of implementation is to minimize the input from developers while allowing them to control the non-verbal behavior in detail.

A. Explicit and implicit behaviors

We classified non-verbal behavior into explicit and implicit behaviors. Explicit behaviors require specification from developers, and implicit behaviors do not require such explicit specification, but they can be specified by developers. Thus, unless one explicitly requires information from developers, we categorized it as implicit to automate the generation of non-verbal behaviors as much as possible.

Even though our definition reflects a developer’s standpoint, it resembles the robot-oriented definition by Breazeal et al. They classified non-verbal behaviors as explicit and implicit, and their implicit behaviors reflect the robot’s internal state [21]. Different classification of behaviors means different results. For example, we categorize a greeting as explicit since it requires explicit specification from developers, but they categorized it as implicit since it only reflects the robot’s internal state without providing explicit information.

B. Reducing required information

1) For explicit behaviors

It is our definition that explicit behaviors require specification from developers. That is, developers need to provide information about how to use it; thus, we need to consider how to reduce the amount of required information that must be provided. For example, when a developer intends a robot to gesture at an object, the developer must
inform the system about the target object. This can be done in different ways: providing a pointing angle, providing x-y-z coordinate information in the absolute position, and providing the object’s label. Since a system that only accepts pointing angles will be very simple, developers always need to calculate pointing angles from much low-level information, such as the positions of the robot and the target object. If the system accepts the object’s label, developers are freed from these computations, while the system autonomously needs to process such information. Our basic policy allows as much abstracted information as possible so that less effort is required from developers.

2) For implicit behaviors

Based on the generation methods, we can categorize implicit behavior into three types. First, autonomous movements related to such speech processes as gazes, beat gestures, and idler motions while not speaking; second, movements required by explicit behaviors, including changing a standing position to point at an object when the system is explicitly required to do so; third, autonomous movements related to the partner’s movement, such as joint attention.

Our policy autonomously generates all of these implicit behaviors as defaults, while enabling a developer to inhibit the cause of the implicit behavior. In this way, we can minimize operator input and reduce the burden for remembering an excessively complex system to generate implicit behaviors.

C. Collection of non-verbal behaviors

We summarized the previous findings in human communication and human-robot interaction into a list of potential non-verbal behaviors to be included and considered whether to implement them as explicit or implicit behaviors. Table 1 summarizes the list of non-verbal behaviors. Each subsection 1)–7) below corresponds with the rows of table 1.

<table>
<thead>
<tr>
<th>Explicit / Implicit</th>
<th>Target</th>
<th>Corresponding information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Deictic gesture</td>
<td>Explicit/Implicit</td>
<td>Object</td>
</tr>
<tr>
<td>2) Iconic gesture</td>
<td>Explicit</td>
<td>-</td>
</tr>
<tr>
<td>3) Beat gesture</td>
<td>Explicit</td>
<td>-</td>
</tr>
<tr>
<td>4) Idler motion</td>
<td>Implicit</td>
<td>-</td>
</tr>
<tr>
<td>5) Eye contact</td>
<td>Implicit</td>
<td>Human</td>
</tr>
<tr>
<td>6) Joint attention</td>
<td>Implicit</td>
<td>Object</td>
</tr>
<tr>
<td>7) Positioning</td>
<td>Implicit</td>
<td>human/object</td>
</tr>
</tbody>
</table>

1) Deictic gestures (pointing)

In human communication, pointing gestures are used often with spatial deixis, such as this and that. Timing is crucial for such gestures. For example, when one says, “Look at this,” one always simultaneously points at the object while saying this. Pointing has been often used in human-robot interaction [22-26].

In addition, words in spatial deixis change based on proximity. Sugiyama modeled the Japanese terms, kore, sore, and are (in English, this and that) and developed a system to automatically switch the use of words in deictic interactions [20]. As shown on the left in Fig. 2, the model is based on a deictic map that decides which deictic word should be used at each place.

We need to ask developers to explicitly provide information about such pointing and utterances that include spatial deixis. Here, information about target objects should be included in the text where pointing should occur.

2) Iconic gestures

Iconic gestures draw the shape or a symbol of the target [27]. Robots might not draw a shape well with their fingers, but they can indicate an object’s size. Thus, we asked developers to explicitly specify the types of iconic gestures.

3) Beat gestures

When speaking, humans often make a gesture their hand that resembles beating out a tempo. This is called a beat gesture. McNeil suggested that humans use beat gestures when they are discussing important topics [26]. In embodied conversational agents, Cassell et al. implemented a system that automatically analyzes the part of utterances that should be accompanied by beat gestures [5]. While Cassell’s system assumes a knowledge base for adding gestures, we adopted a different approach in which developers tag places that need emphasis. This is because in our pilot study, we found that too many beat gestures cause a robot to move too much and the interaction becomes annoying.

4) Idler motion

We believe that idling robots need to express lifelikeness [18]. This can be implicitly done if the system can detect when the robot is neither speaking nor listening.

5) Eye contact

Gaze plays an important role in interactions, e.g. adjustment function of conversation flow, and monitor function to verify the reaction of the interlocutor [28]. For the adjustment function, gaze maintains eye contact with an interlocutor and its timing is driven by the state of the utterance. A speaker looks at his listener to garner attention to the beginning of his speech and looks at the listener again at the end of his speech to inform the listener that now he/she can speak [28, 29]. Mutlu et al. confirmed that robot gaze is useful for the adjustment function in the same way in communication as human gaze [27, 30].

6) Joint-attention

Gaze is also used to express attention, which is known as joint attention [31]. In situations where joint attention occurs, a speaker usually looks and points at the target, and a listener’s gaze follows it. Previous studies demonstrated that a robot can engage in joint attention interaction without receiving specific information from developers [19, 20, 22, 24, 25]. As argued in [32], the target of attention can be retrieved from a standing position as well as the gaze direction.

7) Positioning

Human communication literature has revealed that humans adjust their standing position based on the conversation’s
situation [33]. When people talk about an object, they form a
area known as the O-space where their attentions are focused
together [34]. Yamaoka et al. simulated this positioning in
human-robot interaction. For implementation in a robot, they
decomposed the O-space constraints into the following four:
• proximity to listener
• proximity to object
• listener’s field of view
• presenter’s field of view
Based on these four constraints, a suitable presenter position
is computed [32]. Once the system knows that it is going to
change its attention to a particular object, e.g., a robot refers
to the object, it can autonomously change the robot’s position
based on the above constraints. We designed our system to use
referencing commands to implicitly change the robot’s
standing position.

IV. SYSTEM

The system’s main component is its motion generation
module, which receives input from other modules that use this
behavior generating module. Here, developers provide the
input with simple scripting language, so that the behavior
creating module will autonomously satisfy other implicit
behaviors.

A. Simple Communicative-behavior Markup Language

We developed a scripting language called Simple
Communicative-behavior Markup Language that is used for
controlling both robot utterances and body motions. It is
interpreted in the motion generation module to control
utterances and motions. Our scripting language only requires
the following four basic tags to control explicit behavior, as
illustrated in Table 2.

- **Speak tags** make the robot say any sentence within the
  speak tag starting from `<speak>` and ending with `</speak>`.
- **Reference tags** make the robot refer to an object and are
  used as the interior tags of the speak tag. When this tag is
  used, the system autonomously controls the robot’s
  standing position to secure its sight line and pointing
direction toward the specified object. The system also
  controls the timing of pointing so that the robot extends its
  arm to point at the object just before starting the words in
  the `<reference> </reference>` structure. The reference
tags require a label of an object, e.g., `<reference
name="pencil">`.
- **Emphasis tags** make the robot express a gesture to
  emphasize a particular part of the utterance with a beat
gesture and are used as the interior tags of the speak tag.
The system controls the timing of beat gestures so that the
robot starts them just before starting the words in the
`<emphasis> </emphasis>` structure.
- **Iconic tags** make the robot express an iconic gesture and
  are used as the interior tags of the speak tag. The system
  controls their timing so that the robot starts its gesture just
  before starting the words in the `<iconic> </iconic>`
  structure. The iconic tag requires a parameter for the type
  of iconic gesture, e.g., `<iconic type="big">`. Note that
  abbreviation is allowed. One of the abbreviations for
  iconic tag we often used is `<ask>` tag, where the robot tilts
  its head to express a gesture of listening pause.

Note that interior tags can be used repeatedly inside one
sentence. In addition, the system has other tags for precisely
controlling various implicit behaviors.

B. Software architecture

Figure 3 shows the components related to the generation of

<table>
<thead>
<tr>
<th>Table 2 Simple Communicative-behavior Markup Language</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Speak</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Reference</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Emphasis</strong></td>
</tr>
<tr>
<td><strong>Iconic</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
communicative behaviors. We assume that at a higher layer a behavior generation module exists, which is prepared by developers, that could be a complex dialog management system or a very simple state transition machine. A behavior generation module sends the sentences written in scripting language to the motion generation module.

The main component in our software architecture is the motion generation module, which interprets the scripting language, adjusts the timing of utterances and gestures, and adds implicit gestures. We illustrated that all sensory inputs come from the environmental information module, which represents input from the speech and gesture recognitions. The environmental information module stores the position information of the object and the person around the robot. As summarized in Table 3, the following information is usually required for the motion generation module:

- **Object**: label and position
- **Human**: label, body central position, face position, and body orientation

The motion generation module receives information from the environmental information module to complete the positional information that dynamically changes, e.g., positions of people, and to generate implicit behaviors based on sensory information, including looking at an object if the person looks at or points at it.

In this paper, we used a motion capturing system that provided this information for the environmental information module. We can easily replace such sensory input and keep using the same mechanism. For example, for a field trial, we used a human tracking system based on a laser range finder [35] and a vision-based face tracking system for the input from the robot’s camera. Note that if part of the sensory information is not provided, the function is disabled that requires other information. For example, if we just use the human tracking system, since it only provides the human position, the robot adjusts its standing position toward the interacting person; this disables the eye contact function.

There is a low-level actuation module called the robot control module, which is prepared for each separate piece of robot hardware to conceal the low-level differences of hardware: arrangement of joints, length of arms, etc. Developers can use the same scripting language regardless of the robot hardware. For now, a robot control module has been prepared for three different robots: Robovie II (Fig. 1, left), Robovie R2-mini (Fig. 1, right), and Wakamaru.

The motion generation module communicates by a voice synthesis module to obtain accurate timing information and to control the timing of utterances.

### TABLE 3 REQUIRED SENSOR DATA

<table>
<thead>
<tr>
<th>OBJECT</th>
<th>POSITION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HUMAN</strong></td>
<td><strong>OBJECT</strong></td>
<td><strong>HUMAN</strong></td>
</tr>
<tr>
<td>Body_center_pos</td>
<td>Position of human body’s center</td>
<td>Human head orientation</td>
</tr>
<tr>
<td>Head_center_pos</td>
<td>Position of human head’s center</td>
<td>Human body orientation</td>
</tr>
</tbody>
</table>

**C. Generating implicit behaviors**

Three processes are related to the generation of implicit behaviors. And we created an operational mode for the system.

The operational mode includes the main internal variables possessed by the motion generation module. There are three states: Speaker mode, Listener mode, and Idler mode.

As shown in Fig. 4, transitions between idler mode and other modes are based on the presence of people around the robot. If someone is in the conversational area, the robot transits from idler to listener mode. When a speak tag is activated, the robot transits to speaker mode, and when an utterance specified in the speak tag is finished, the robot transits to the listener mode. When no one is inside the conversation area for one second, the robot transits to the idler mode.

1) Implicit behavior in speaker mode

In the speaker mode, the robot mainly complete the accompanying implicit behaviors based on the information in the reference tag. Four behaviors are mainly associated with the reference tag.

The reference tag controls pointing behaviors with an extended arm as well as gaze behavior. Literature on joint attention reports the importance of gazing at the object in addition to pointing ([36]); thus, when our robot points at a target, it simultaneously looks at it.

The reference tag also controls the use of reference terms employed in conversation, such as here and there. For referring to spatial entities, e.g., an object’s location, these terms change based on the position of the speaker, the addressees, and the target [37]. Some languages use two-way contrast (English has this and that or here and there), while others use three-way contrast (Japanese has kore, sore, and are). As shown in Fig. 3, Sugiyama et al. developed a model that chooses a word based on positional relationships [11]. We connected their model to our system to autonomously replace the reference term with an appropriate one.

The reference tag is also associated with the control of the standing position. We implemented the model developed by Yamaoka et al. [9] that dispatches the robot to the proper position to present the information about the target to the person. After the referencing behavior is specified, the robot keeps establishing the O-space by considering the positions of the person and the target object. The implicit control of position is valid even when no pointing behavior has been
specified yet. In this case, the robot analyzes the person’s attention from its body orientation and gazes to establish O-space.

In the speaker mode, the system controls the robot gaze. If a reference tag is specified, the robot looks at the target object while pointing; otherwise, it looks at the interacting person to maintain eye contact. People don’t keep eye contact for too long time. Mutlu et al. analyzed the distribution of gaze during interaction, and reported that people sometimes hold gaze on the listener, while sometimes widely distribute to a space around people [27]. We implemented this gaze distribution model into the robot as well.

2) Implicit behavior in listener mode

In the listener mode, the robot engages in joint attention while maintaining eye contact. When the interacting person is looking at/pointing at the target, the robot also looks at the same target to engage in joint attention and maintains eye contact when the person is looking at the robot.

Control of the standing position is active in the listener mode too. As control in the speaker mode, the robot establishes O-space by considering the positions of the person and the target object. The difference is in the listener mode, where the target object is not indicated by the scripting language, i.e., the reference tag, but is decided by the robot. We used the attention shift model developed by Yamaoka et al. to decide the target object [18].

3) Implicit behavior in idler mode

In the idler mode, the robot exhibits idling motion so that it is perceived as lifelike and communicative [18, 38]. We designed idling behaviors in which the robot often looks around to show that it is working and waiting for a person to come.

D. Synchronization of utterances and gestures

Three steps are involved to synchronize utterances and gestures. As shown in Fig. 5.

Step 1: Analyze the timing of each phoneme in the utterance.

The system generates an utterance first. It sends the text to the voice synthesis module and receives a sound file and detailed timing information for each phoneme.

Step 2: Analyze the timing of all non-verbal behaviors.

The robot control module has a set of commands to determine the time required to generate motions. The motion generation module sends a set of motions to the robot control module to generate non-verbal behavior and receives timing information for each motion.

Step 3: Compile and execute a command list of the timing chart.

There are rules for the timing requirements; pointing must be done to associate utterances with pointing. The system compiles a set of commands for the robot control module and adjusts the utterance’s start timing to satisfy all required timing. Then it starts to execute all commands.

V. EXAMPLE INTERACTION

Let us explain how our system works with two examples and to demonstrate the effectiveness of the architecture.

The first example is about the use of reference tag (Figure 6). We describe how explicit and implicit behaviors are generated based on the input sentence. In the example, the robot is going to give advice to a person who asked the robot to suggest a PC.

(1) The person came to the area, and the robot’s state
transited from idler mode into listener mode. The robot oriented its body to the person, and started to maintain eye contact with him.

(2) Here, the motion generation module received an input sentence from a module in higher layer, e.g. a behavior generation module, which manages dialog at higher layer. Here, the input is only the following sentence, prepared in advance by a developer: <speak> I think <reference name="PC">this laptop PC</reference> is a good choice </speak>

(3) Given this input, the robot began to move to the best place to execute the referencing behavior of target object, “PC”, to the person currently interacting with the robot, i.e. a place close to the person where they can look at the object together [10]. When the robot arrived to the position, it started to speak the sentence.

(4) While saying “I think,” it started to prepare for the referencing behavior. It adjusted its body orientation toward the object, and started to move its arm and head, since the robot needs to point at the object just before speaking about the referencing utterance.

(5) It started to speak utterance inside the reference tag. Since the target object is far from the robot, the system replace the spatial deixis this to that. The robot said, “that laptop PC.” Its arm is already arrived at the position of pointing. It also looked at the object while pointing.

(6) While saying “is a good choice”, the robot resumed implicit behaviors in speaker mode, i.e. eye contact. And after that the robot’s state transited back into listener mode.

Second example is to demonstrate that we can compose interaction with the robot easily with the proposed framework. The example is a simple application, realized by only six input sentences (Table 4). In this example, the environmental information module received sensory input from motion-capturing system.

Table 4 INPUT SENTENCES FOR EXAMPLE 2

| (1) | <speak>Welcome to the PC shop, I'm Robovie. Please have a look around.</speak> |
| (2) | <speak><reference label="laptop">This laptop PC</reference> is very popular now.</speak> |
| (3) | <speak>There is a desktop PC<reference label="desktop"> over there</reference></speak> |
| (4) | <speak><reference label="desktop">This desktop PC</reference> is currently on sale so it is <emphasis>very cheap</emphasis> now.</speak> |
| (5) | <speak>It's bigger than a laptop, but actually it only needs<iconic type="small">one square meter</iconic> of space.</speak> |
| (6) | <speak>If you find something you want to buy, please tell me. Thank you.</speak> |

Fig. 6 shows a scene of interaction realized by the input shown in Table 4. (Please see the multimedia attachment for the video of this scene). First, when the person came, the robot moved to him and greeted while maintaining eye contact (Fig. 6 (1)). The robot explained the laptop with deictic gesture (Fig. 6 (2)), and asked him to look at the desktop PC next. Since he moved to the desktop PC, the robot followed him and stood at the location closest to the PC and engaged in joint attention, i.e. look at the PC when he looked at it and looked at him when he looked at the robot (Fig. 6 (3)-(4)). These movements in the scene 3 and 4 are all implicit behaviors generated autonomously. The robot used emphasis gesture and iconic gesture to introduce the desktop PC (Fig. 6 (5)). At last, when he left from this area, the robot transited into idler mode (Fig. 6 (6)).

We believe that these examples provide an insight that this system enables developers to implement such complex interaction only with small input.

VI. CONCLUSION

This paper reports a development framework that enables developers to easily use the communicative behaviors of social robots. Based on the literature in human communication as well as human-robot interaction, the system autonomously controls various implicit behaviors, while accepting commands from developers for a few explicit behaviors. Since the system’s architecture is layered, we can easily replace modules for use in different humanoid robots. The usefulness of the development framework was demonstrated with examples. We will evaluate if our system
can ease the development process by measuring the time and effort saved with this development framework in the future.

REFERENCES


