Information Sciences 264 (2014) 230–259

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Using agents to manage Socio-Technical Congruence in a Global Software Engineering project

Javier Portillo-Rodríguez [a,*], Aurora Vizcaíno [a], Mario Piattini [a], Sarah Beecham [b]

[a] Alarcos Research Group (University of Castilla-La Mancha), Paseo de la Universidad, 4 13071 Ciudad Real, Spain
[b] Lero-The Irish Software Engineering Research Centre, University of Limerick, Ireland

ARTICLE INFO

Article history:
Received 17 June 2012
Received in revised form 13 October 2013
Accepted 3 January 2014
Available online 11 January 2014

Keywords:
Global Software Engineering
Socio-Technical Congruence
Software agents

ABSTRACT

In Global Software Engineering (GSE) there are a number of communication and coordination challenges which are brought about by the factor of distance. Measuring Socio-Technical Congruence (STC) is, however, presented as a suitable technique for helping to resolve those issues. This leads us to believe that applying STC measurements to GSE might be beneficial, improving communication and coordination. However, after studying existing tools that use STC measurements, we detected some gaps, both in the way they measure STC and in the features offered by the tools for the GSE environment. That is why we have designed an Agent Architecture for coordination and communication that aims to fill the gaps found in the current tools and includes features adapted to GSE characteristics. This is achieved by taking advantage of the special features that agents offer. Moreover, this proposal has been validated in a case study performed at Indra Software Labs, a global software development company. Results show that, by using our proposal, it is possible to improve coordination and communication in a distributed environment.

1. Introduction

Modern software engineering, and more specifically Global Software Engineering (GSE), which takes place in globally dispersed teams, presents specific advantages and challenges to the software industry [21]. Working in a globally-distributed team is fast becoming the norm for many software engineering organizations, as they seek to attract new customers, strive to be nearer to their existing customers, broaden their skills base and reduce their labor costs. Realizing these benefits, however, goes hand in hand with additional challenges such as increased complexity and communication overheads [12,28].

In Global Software Engineering, coordination becomes more difficult; this is due to the problems derived from distance (i.e. different working times, different understandings or difficulties in communicating efficiently). A key challenge for GSE companies is to coordinate and allocate tasks, and ensure that important information is successfully communicated to all partners [6,10,5,24]. Despite this need, there is little help available for GSE managers to assess their organization structure and to decide which coordination and communication processes need to be strengthened. If organizations cannot assess how well their communication and coordination strategies are working, it will be extremely difficult to pinpoint where changes are needed. An independent measure of how people coordinate their efforts could help to solve this problem, allowing GSE organizations to leverage the benefits of GSE and increase productivity and quality [33].

* Corresponding author. Tel.: +34 645921751.
E-mail addresses: javier.portillo@uclm.es (J. Portillo-Rodríguez), aurora.vizcaino@uclm.es (A. Vizcaíno), mario.piattini@uclm.es (M. Piattini), sarah.beecham@lero.ie (S. Beecham).

0020-0255/$ - see front matter © 2014 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2014.01.009

One technique designed to measure coordination is that of Socio-Technical Congruence [8,23]. The main idea of Socio-Technical Congruence is based on Conway's Law [9], which says that the structure of a software product reflects the physical layout of the development organization. Based on this law, Socio-Technical Congruence (STC) can be defined as a measure or a technique that assesses the ''fit'' between the structure of a software system and the structure of the organization that develops it [30] or ''an intuitive way to compare required coordination effort within a software development project with the actual ongoing coordination'' [23].

There is now a growing effort to demonstrate empirically the benefits of gaining a good level of congruence between coordination requirements and the coordination activities actually performed, where, in theory, a high level of congruence should relate to improvements in productivity [7] and quality [22]. However, a high level of congruence can also be associated with increased risks and costs, where too many interactions can overload and overwhelm team members [4]. This means that there is clearly a need to balance the number of interactions and time spent on actual development tasks.

On the other hand, there are tools available to help to measure STC and STC-related aspects and achieve a suitable socio-technical balance. Tools such as Ariadne [29] or Tesseract [26] have been implemented, but the majority of STC tools have been designed to visualize coordination relationships and coordination gaps (lack of coordination interactions where coordination is needed) but they do not help the organization to maintain a good level of STC. Moreover, they use STC measurements that are based on collocated development, and because of this, they are not suited to a Global Software Project.

In this paper we present an Agent Architecture that aims to manage (measure, control and maintain) the Socio-Technical Congruence, in order to improve coordination and communication in a Global Software Project. We introduce a new approach with which to measure STC, by combining current STC measurements and agent properties, such as autonomy, reactivity, pro-activity and social ability. These properties can be used to perform tasks like monitoring, reporting and measuring Socio-Technical Congruence levels. In summary, agents could provide a much-needed systematic approach that can be incorporated and used to improve current STC tools. Focusing on agent architecture and STC, therefore, we aim to answer the following research question:

RQ1: Can software agents be used to measure Socio-Technical Congruence in Global Projects?
RQ2: How can agents improve current Socio-Technical Congruence tools to make them more useful for a Global Development Project?

To answer these questions we present how, SCT can be measured, by using the agents' properties. We also demonstrate how, by using the proposed Agent Architecture, agents are able to perform tasks to calculate, control and maintain STC, improving the current tools and extending their usage to global settings. This proposal has been validated in a case study performed at the Indra Software Labs, showing how it is possible to improve coordination and communication by using our architecture, which combines agent technology with STC measurement.

This paper is structured as follows: In the next section, Section 2, central terms such as Congruence, Coordination, Communication (important aspects in GSE), as well as the relationships between them, are introduced. In Section 3, the main benefits and risks of measuring and maintaining a high level of STC are presented. Section 4 presents a review of current approaches to measuring Socio-Technical Congruence, with a look at some of the existing tools that measure STC or STC-related aspects. In Section 5 we present our proposal (an Agent Architecture) for managing STC and improving the existing tools in a global setting. We also explain, in Section 5, why agents might be an appropriate technology with which to measure STC in GSE. Section 6 describes the results obtained in the case study performed. In Section 7, the main limitations found and assumptions made in our proposal are discussed. Finally, in Section 8, we outline the conclusions obtained and lines for future research.

2. Congruence, coordination and communication

To clarify why measuring Socio-Technical Congruence is related to the improvement of coordination and communication, in this section we define the terms *congruence*, *coordination* and *communication* and how they inter-relate.

Congruence has several definitions in literature, but they all refer to ''the match between a particular organization design and the organization's ability to carry out a task'' [8]. We have taken the definition given in [23], where it is defined as ''an intuitive way to compare required coordination effort within a software development project with the actual ongoing coordination'' or ''the fit between an organization's coordination requirements and an organization's social interactions''.

From this definition, we can already draw out the main relationship between coordination and Socio-Technical Congruence (STC). Basically, what STC measures is the alignment between **coordination needs** (extracted from technical dependencies) and the **real coordination** performed (drawn from socio-technical aspects). This means that to measure STC it is necessary, first of all, to obtain the coordination needs and then check the coordination that the team members are actually performing.

Coordination needs (represented in Fig. 1) can be obtained in different ways, but they are usually found by combining technical dependencies between Technical Entities and dependencies between Technical Entities and stakeholders of the project.

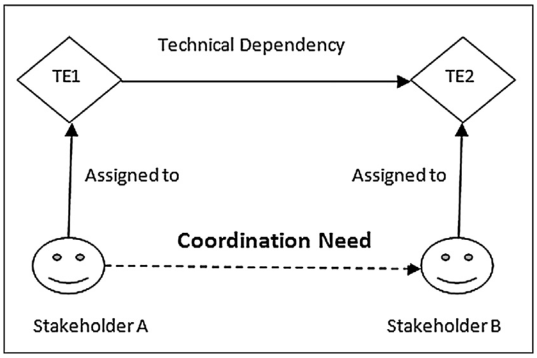


Fig. 1. Coordination need.

A Technical Entity (TE) is "an entity in a project that can be worked on by a person" [22]. Depending on the context where STC is calculated, we can consider different kinds of Technical Entities. Examples of TE are Tasks, Requirements, Bugs, Source Code Files, Modification Requests, Tests, etc.

Thus, by extracting the dependencies between these TEs (for instance, requirements that depend on each other) and associating these dependencies with those between stakeholders and TEs (for instance, person P is assigned to requirement R), it is possible to obtain the Coordination Needs that indicate which particular individuals should be coordinated with which other specific people.

Although in Fig. 1 only one coordination need is represented, it is necessary to obtain all the coordination needs between all the stakeholders involved. The set of stakeholders to be taken into account will be defined by the focus of the study; that is, if the STC is measured for the whole project (in this case, a GSE Project) the set of stakeholders will be all the stakeholders participating in the project.

If, on the other hand, the STC is measured for just the coding phase of the GSE Project, the set of stakeholders will contain those stakeholders involved in that phase. In order to measure STC, these Coordination Needs should be compared to the real coordination that stakeholders are actually performing.

To obtain real coordination, usually called **Actual Coordination** [8,7], social interactions are used. The assumption made is that those individuals having social interaction might be coordinating themselves. As explained in [22], the actual coordination "does not need to represent coordination at all, but merely a relationship of interest between two people in an organization". Here is where communication is related to STC, because one way of obtaining social interaction is by obtaining communication interaction.

Moreover, as stated in [16,17], informal communication is important for coordination. Specifically, a dense communication network is associated in [17] with fewer coordination problems in distributed teams.

We can thus see that one way, but not the only one, of obtaining Actual Coordination is by checking communication interactions. Here is other important assumption, consisting in supposing that all communication interactions are related to work aspects. In practice, not all communication interactions are actually work-related interactions, but the percentage of work-related interactions is usually enough to accept this assumption.

In a nutshell, the relationship among Socio-Technical Congruence (STC), Coordination and Communication can be represented as shown in Fig. 2, where STC is measured by comparing Coordination Needs and Actual Coordination. Coordination Needs are obtained by using Technical dependencies between technical entities. Moreover, Actual Coordination is obtained by using social interactions, where communication interactions are a subset of these.

Given the relationship between STC, Coordination and Communication, measuring STC might be an indicator of the level of coordination, as well as of communication. This level is usually translated to the number of "gaps" (the lack of interactions between two people that should be coordinating); that is, wherever there are no gaps the level of congruence is supposedly the highest.

On the other hand, when there are gaps, it could mean that there are mismatches between the needed coordination and the actual coordination; it might therefore be necessary to close them if coordination breakdowns are to be avoided.

Now that the main concepts of this work, along with the relationships between them have been explained, in the next section these relationships are used to explain why measuring STC might be important or beneficial and what particular practical uses of measuring STC exist at the present time. We also include some of the disadvantages of measuring STC.

3. Measuring Socio-Technical Congruence: Pros and cons

Coordination between team members is needed in every software project. In some cases, however, such as in distributed or global software development, coordination becomes more difficult, due to the problems derived from distance (i.e. different working times, different understandings or difficulties in communicating efficiently). All these factors make it necessary

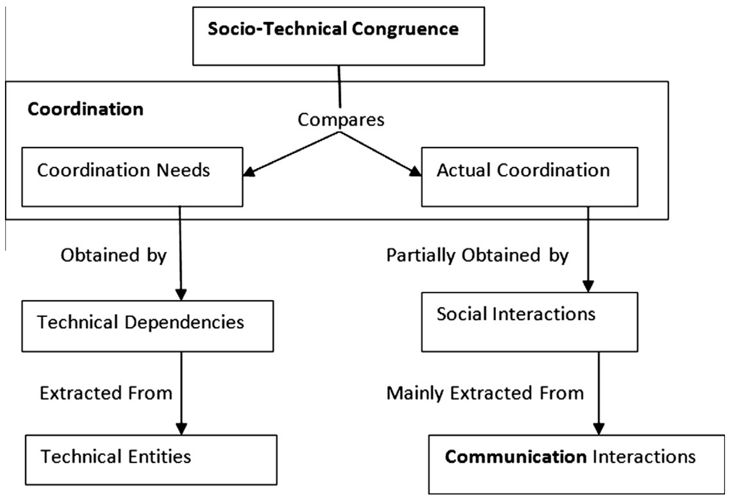


Fig. 2. Relationships between STC, coordination and communication.

to apply techniques that help to assure a good level of coordination in the effort to avoid coordination breakdowns that might have a negative influence on the success of the project.

Socio-Technical Congruence (STC) is directly related to coordination, as stated in Section 2. To be specific, it is directly related to coordination needs and how these coordination needs are actually performed. Thus, one of the main benefits that can be obtained by measuring STC is to assure that coordination needs are being fulfilled. In other words, the higher the level of congruence, the higher the percentage of fulfilled needs, which in the end can be translated into the following assertion: measuring STC can result in a better level of coordination.

As mentioned in Section 2, STC is also related to communication. In this sense, measuring STC is also a way of monitoring and improving communication between distributed teams. To measure STC it is usually necessary to look at how people are communicating; this implies that it might be a way of detecting problems in this area, such as the lack of communication, or communication overload.

There are empirical studies that certify specific benefits (in specific contexts) of measuring STC. In [8], the analysis of congruence shows that it helps to reduce the amount of time required to perform tasks such as modification requests, and that the most productive workers reach higher levels of congruence than the less productive.

Moreover, measuring STC is an effective way of detecting "gaps", in order to close these and avoid their negative effects (for instance, the number of code changes increases when communication gaps exist [13]).

There are several practical applications of measuring STC; for instance, a manager can use it to monitor the alignment of his teams' coordination with technical dependencies [23]. It can also be used to provide a ranking of missing coordination, suggesting which gaps are the most important to fill [23]. In [8], there is a description of a set of implications for collaborative tool design.

For instance, tools might assess the congruence between coordination needs and communication in order to monitor, control and improve the flows of communication. Moreover, tools could provide real-time information of the set of workers that a person may need to communicate with.

We can summarize all this by saying that, due to the need for efficient coordination (especially in distributed teams), Socio-Technical Congruence (STC) is a useful technique with which to study coordination and thereby assess to what extent coordination needs are fulfilled.

Moreover, a high level of STC has been reported in many studies [7,13, 22] or [15], with an improvement in the outcome of development activities that require a large amount of coordination. Measuring STC can, nevertheless, introduce additional costs and risks that should be considered. For instance, if we try to eliminate gaps by augmenting coordination and communication this might increase an overload of information in team members, and individual productivity could decrease [4].

Bearing in mind all the benefits explained above, in terms of what a software development organization might obtain through high Socio-Technical Congruence, it is vital to deploy mechanisms with which to measure the level of congruence in an organization. Taking that into account, the approaches found for calculating or measuring STC are presented in the following section, together with some of the tools available.

4. Related work: Measurements and tools

In this section, we introduce some of the existing Socio-Technical measurements and tools. In relation to our proposal, we will glean possible tool improvements from this section, to study whether any of the measurements presented can be used by our Agent Architecture in a GSE Project.

4.1. Socio-Technical Congruence measurements

In this section we present four approaches for measuring Socio-Technical Congruence, as sourced from literature. According to Brooks [4] measuring congruence implies taking into account ‘properties’ of congruence [25]:

Congruence represents a state at a particular moment in time; that is, an organization may be congruent at a particular moment in time and this does not assure that the organization will be congruent at another point in time. **Congruence is dynamic,** because dependencies, coordination needs or social structures, among other things, may change over time. **Congruence is multi-dimensional;** measurements should specify which dimension or dimensions they are dealing with.

Dimensions are related to the ways of coordinating the work, such as comments left in tools or direct communication with dependent stakeholders. Finally**, congruence is also a multi-level measurement**; this means that it can be measured, for instance, from individuals or teams, right through to organizations.

To measure congruence it is also necessary to gather information such as team structures, communication interactions, process and work practice information, current coordination actions taken through tools, tacit knowledge, task assignments and location of people, among other things [25]. Depending on the dimensions and/or levels, chosen, the information needed may vary.

To obtain the desired information, existing software tools such as forums, instant messaging tools, email tools or software repositories can be used. In such cases information may be obtained automatically, but sometimes it may be necessary to use surveys or interviews as well.

There are ways to calculate or measure the Socio-Technical Congruence (see Table 1). In the case of Cataldo et al.’s measurement, presented in [8,7], Coordination Needs are found by using Task Dependencies, and Actual Coordination is obtained in four ways: (1) *Structural Congruence,* when two workers belong to the same formal team, (2) *Geographical Congruence,* when two people are working in the same physical place, (3) *Modification Request Communication Congruence,* when two workers have made comments on the same modification request report, and finally and (4) *IRC communication congruence,* constructed by examining the IRC logs manually; this exists when two workers make reference to the same modification request.

Kwan et al. [23,22] present a congruence measurement that takes limitations found in Cataldo et al’s first proposal into account; that is, every dependency is considered equally important, and any coordination action is sufficient to fulfill a coordination need.

Kwan et al. build on Cataldo et al’s solution by adding a weighted congruence measurement [23]. Weights are given to dependencies between tasks, as well as to between tasks and people. Dependencies between tasks are established by using “the ratio of dependencies a task has with another over the number of total dependencies that task has on others” [23].

Dependencies between tasks and people are weighted by using the number of hours that a worker is supposed to spend on a task. In this case, to obtain Actual Coordination a communication network can be used, weighted by the amount of ongoing communication.

Moreover, several distance measurements may also be employed, such as physical distance, time difference or the number of organizational units that separate two individuals. This weighted congruence measurement, presented by Kwan et al., is supposed to have some benefits with respect to gaps. The approach is able to detect where the gaps are and assess the importance of the gaps, in order to identify the most important ones to close.

In [13], Ehrlich et al. present a different approach to measuring congruence, which focused on gaps in communication in their effort to study coordination in distributed teams. Moreover, Valetto et al. in [32] explain how the information needed to use this congruence measurement can be obtained by using software repositories.

To calculate congruence, they use a social network, where they combine information about communication interactions, mapped as undirected graphs, and relationships between software artifacts, mapped as directed graphs, as well as work relationships; these too are represented as directed graphs, by relating people and software artifacts. They also consider the possibility of including weights in the arcs, by using the number of changes to the same artifact made by a worker, or the number of dependencies between source files.

In addition, they propose the use of other attributes apart from the weight, such as confidence among the workers, or the size of contributions. Finally, they calculate congruence by measuring the proportion of software artifact relationships that are mirrored by the people interaction network and to do so they developed an algorithm, which is explained in [32].

Another approach that also uses social networks is presented by Gokpinar et al. in [15]. This network, called ‘Product Architecture Network’, is created by considering the sub-systems as nodes, and links as dependencies between sub-systems. Gokpinar et al. also use the dependencies between engineers and subsystems to find out where the engineers are performing communication, so as to obtain a network called Organizational Coordination Network that represents the Actual Coordination matrices presented previously. Finally, to obtain congruence (called “Coordination Deficit” in [15]), both networks are compared.

In Table 1 we have included information about the authors of each measurement and the references where the measurement is explained (first column). We have also included the project phase where the measurement has been applied and where the associated data has been extracted, along with the approach used to formalize the measurement, in this case matrices or social networks, and whether the congruence measurement take into account weights, also noting which weights have been used in each case. Moreover, dimensions considered in each proposal are included; the last column indicates whether the congruence measurement has been used in any empirical study to check advantages, benefits or

Table 1
Key features of congruence measurement approaches presented in this section.

Congruence measurement	Project phase	Approach	Weighted	Weights	Dimensions considered	Empirically tested
Cataldo et al. [8,7]	Coding	Matrices	No [8] Yes [7]	Number of source-code files dependencies	Team Structures, Geographical Location, Modification Request Comments, IRC communication	Yes [8,7]
Kwan et al. [23,22]	Coding	Matrices	Yes	– Dependencies a task has with another – Hours that a worker is supposed to spend in a task – Expertise a person has about a task	Communication interactions, Physical Distance, Temporal Distance	Yes [22]
Valetto et al. [32] and Ehrlich et al. [13]	Coding	Social networks	Yes	– Number of changes to the same artifact made by a worker – Number of dependencies between source files	Communication interactions, Software Repository information	Yes [13]
Gokpinar et al. [15]	Architecture design	Social networks	Yes	Number of distribution lists where the engineer is included	Communication through distribution lists	Yes [15]

improvements that can be obtained by using it in comparison to, for instance, not using any congruence measurement, or using another different existing congruence measurement.

For our architecture, we believe that the most appropriate measurement is that of Kwan et al., for several reasons:

- Its features can be adapted to the global context where the agents of our architecture will operate. For instance, this approach takes into account the geographical and temporal distance, two important factors in a global environment. In fact, as we will see later, we have also included the socio-cultural factor as a third weight.
- Kwan's approach has been tested by using the tasks as the main technical entity; this fits well with our context, since we need to use a measure that is valid for all GSE processes, and tasks can be considered in all software life processes. For instance, Valetto et al. [32] and Ehrlich et al. [13] have tested their proposal, looking at aspects very close to the coding phase, such as the number of dependencies between source files, or Software Repository information.
- It should also be said that to achieve our goals we do not need to represent the dependency networks, which means that an approach that uses social networks is not indispensable for us. We believe that considering an approach based on a social network would offer more analysis possibilities, but at the same time it would make data management more difficult, without improving our approach.

In short, since we are not creating a new STC measurement we have chosen the one whose weights and dimensions best fit our requirements.

These approaches present a solution for measuring STC; however, there are also some practical implementations (tools), based on the idea of Socio-Cultural Congruence, that should be taken into account before proposing a new approach.

4.2. Tools supporting Socio-Technical Congruence

In order to provide knowledge about the existing implementations related to Socio-Technical Congruence measurement, in this section we present a set of tools or prototypes, also indicating which points might be improved if new tools are to be developed. These improvement points will be taken into account when presenting our proposal. To obtain the set of tools, we have performed a non-systematic review in STC literature, based mainly on the information taken from papers presented in the different editions of ICSE workshops on Socio-Technical Congruence. The set of tools discovered is the following:

- **Ariadne**: [29,2]: This is an Eclipse plugin that analyzes Java files to identify dependencies and authorship information: it does so by using a configuration management repository. With this information, the tool is able to identify software dependencies between software developers and visualize these dependencies finally, to create a bridge between technical and social dependencies. The goal of generating these visualizations is to minimize coordination problems.
- **TraVis**: [18]: This tool has the same goal as Ariadne, but it makes use of dependencies between different artifacts and their users, instead of using source code authorship only. With this information TraVis creates a semantic network, called "traceability network", with task-related entities, artifacts, and users. TraVis thus uses this network to display the dependency network of specific artifacts, developers or activities, in the quest to increase awareness within a GSD project.

- **WorldView**: [2]: This is a software development tool designed to provide managers, team leaders and developers with a central repository that can visualize the structure of distributed teams, the availability of its members and their location. The tool draws out, from a number of sources, the social dependencies between teams and represents each dependency that is extracted as a color-coded line between two dependent teams, or between two team members.
- **Tesseract**: [26]: Tesseract is a socio-technical dependency browser that makes it possible to show social and technical relationships between different project entities such as developers, code or bugs. It also visualizes the match or mismatch of socio-technical relationships and allows the interactive exploration of these relationships over time. In other words, this tool analyzes different project entities to determine the socio-technical relationships. These relationships are displayed via four panels, enabling easy exploration of the data set. This tool makes use of the Cataldo approach to calculate Socio-Technical Congruence.
- **"Fonseca Tool":** [27]: This tool has been designed to identify dependencies between pieces of code and their authors automatically, using Ariadne [29]. With this information, the tool creates a social network of developers that establishes the dependencies between developers. By using these two types of dependencies, the tool can identify situations where there is a mismatch between the dependency and communication networks.
- **CodeSaw:** [14]: This is a social visualization tool for distributed software development. CodeSaw visualizes a software community by using code repositories and project communication. This tool introduces the feature of Spatial Messages, which consist in allowing users to leave comments on the visualization itself.

Our study of the tools presented has led us to obtain some interesting conclusions that could help to improve the current tools:

- The first conclusion is that there are some tools, such as Ariadne, that take into account only one single data source; they do not compare the technical dependencies with social interactions (for instance, communication interactions); these tools therefore make a less thorough analysis than those that use both.
- The second conclusion is that almost all the tools have the goal of visualizing dependencies and/or of indicating lack of coordination. This means that no new visualizations tools are needed. The only problem is that they just visualize the situation; they do not apply actions automatically in order to improve a negative situation.
- Thirdly, those tools calculating STC, such as Tesseract, use an approach (Cataldo et al's) that was not designed to be used in a global environment; this being so, they incorporate all the disadvantages of this approach (as presented in Section 4.1) such as the fact that the measurement does not include weights and it is not possible to establish, among other things, a gap ranking.
- The fourth conclusion is that some of these tools highlight gaps, but they do not analyze these gaps in time, nor do they relate these gaps to the level of coordination by using STC values, for instance.
- Fifthly, we conclude that there are tools, such as the "Fonseca Tool" that take into account communication interactions only (in this case by email). This can be considered an incomplete way to measure STC, because email interactions are not the only way of communication and/or coordination. For instance, in a Global Environment, indirect communication through development tools (for instance, comments attached to issues in Issue Management Tools) is also performed, and these interactions should also be included when measuring STC. This is stated in [11], which asserts that assuming that coordination is performed by direct communication only is a mistake.

Considering these points, together with the features of the STC measurements presented, we can give a set of reasons to outline why our architecture, based on agent technology, may improve the way STC is calculated and used by the tools studied:

- The first reason is related to the environment where STC is being calculated and used, i.e., Global Software Engineering. In this sense, agent technology has been specifically designed to be used in distributed environments where information flow and coordination should be controlled.
  In our context, the distributed environment is a GSE project; agents collaborate to calculate STC, with the global goal of measuring coordination and of maintaining a good level of this coordination. First of all, then, we can say that agent technology fits perfectly with the real environment where STC will be calculated and used.
- The second reason is related to the general limitation of the existing tools to work proactively. That is to say, the existing tools do offer support to highlight automatically the dependencies that exist between team members and to indicate where there are possible coordination and communication gaps. However, they do not perform actions automatically to close the gaps found, for instance. In our case, agent technology has been designed specifically to work as proactive software that does not wait for user intervention to perform some kind of actions. For instance, by using agents, the gaps detected can be analyzed in a collaborative way, to decide which action would be the best one to take in order to close the gaps (without human intervention). In our case, as explained later, agents will collaborate to offer each team member possible solutions for closing the gaps detected.

- The third reason is closely related to the second one, since agents have also been designed to work as personal assistants (for instance recommenders). The possible solutions given by the agents will thus also be adapted to user preferences. That means that information given by the agents will not be the same when the gap has no importance as when the gap is really important. Similarly, the solutions offered will vary depending on user preferences. For instance, to close a coordination gap the agent may recommend to two team members how and when communication between them should take place, based on each user's preferences.

Finally, we can say that the use of agents allows us to include all the features that are needed to calculate STC in a way that fits perfectly with the GSE context, permitting us to include some improvements with respect to the existing tools.

Our proposal puts forward its goal of improving coordination and communication by measuring coordination through the use of STC and by offering distributed team members a set of features that are not available in current implementation; this is all possible thanks to the use of agent technology.

In the next section, we present our proposal, giving more details about the way agent features are used to apply the improvements that have been detected as needed.

5. Proposal: An agent architecture to manage Socio-Technical Congruence

In the first part of this section, that is, in Section 5.1, we are going to present Agent Technology as a technology that can be used to manage Socio-Technical Congruence. In addition, we will describe (in Section 5.2) an Agent Architecture, explaining how this is able to apply some improvements over existing tools. By doing all this we will respond to the research questions RQ1 and RQ2.

5.1. Using agents to measure STC

In general, agents are software programs with special properties, and although different definitions of intelligent agents can be found in literature, one way of distinguishing agents from other types of software applications and of characterizing them is to describe their main properties [34]:

- Autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal states.
- Social ability: agents interact with other agents (and possibly humans) via some kind of agent communication language.
- Reactivity: agents perceive their environment and respond in a timely fashion.
- Pro-activeness: in the sense that the agents can take the initiative and achieve their own goals.

By using these properties, agents are able to perform tasks that might be useful when managing SCT:

- Software agent technology can monitor and coordinate events and meetings and disseminate information [3] or build and maintain organizational memories [1]. Another important issue is that agents can learn from their own experience. Consequently, agent systems are expected to become more efficient with time, since the agents learn from their previous mistakes and successes [35]. In doing all this, agents can discover the most appropriate actions to eliminate gaps, for instance.
- Most agents today employ some type of artificial intelligence technique to assist the users with their computer-related tasks, such as reading e-mails, maintaining a calendar, and filtering information[20]. This might be very useful for registering users' interactions and for obtaining actual coordination.
- Agents can exhibit flexible behavior, providing knowledge, both "reactively", on user request, and "pro-actively", anticipating the user's knowledge needs. They can also serve as personal assistants, maintaining the user's profile and preferences. Agents can thereby perform actions to improve congruence, coordination and communication and adapt these actions to the users' preferences and group necessities.

To summarize, the properties and uses of agent technology in the area of information management have encouraged us to consider agents as a suitable technology to manage Socio-Technical Congruence. This is because agents might control congruence at different levels by taking into account user, group and context characteristics, in order to apply actions that help to maintain congruence and hence project coordination and communication. Thus, at user level, agents can play the role of user assistants by taking the user's coordination needs into account, suggesting possible solutions to users when potential coordination or communication problems appear. These solutions might be adapted to the user's needs by considering, for instance, user profiles or user preferences. At group level, agents can cooperate to maintain congruence of the group or extract valuable information such as who the most active or the most-consulted users are.

Moreover, it is our belief that because agents are able to do all the tasks on their own, without needing too much user intervention, they thus avoid overloading users with more tasks. Another aspect that makes agent technology suitable for this work is the scalability of agent systems. The number of agents in a multi-agent system can vary, depending on the needs. For instance, if an agent is needed to register interactions, this agent might become a bottleneck, because all interactions are registered by the same agent and the number of interactions could increase considerably. It is possible, however, to increase in real time the number of agents in charge of registering interactions, thus avoiding bottlenecks. The agent system can hence be adapted to a setting such as a Global Environment where we cannot predict initially the number of interactions.

Taking all these reasons into account, we can say that agents could be used to measure STC because they are able to manage it, and apply actions that avoid risks when measuring STC; existing implementations are thus improved. In the next section we present the Agent Architecture that we propose.

5.2. An agent architecture to manage Socio-Technical Congruence

Based on what we have explained in Section 5.1, we argue that agentś properties can be used in the context of managing STC. To that end we have designed an agent architecture, with the main goal of **managing STC in order to monitor and improve coordination and communication in a Global Software Project**.

5.2.1. Architecture goals

To reach the main goal, we have defined a set of sub-goals that agents will reach by following a non-intrusive approach. That is, agents will perform their tasks, trying not to disturb or overload users with too much information, while adapting the information to each user's preferences. The sub-goals defined in order to improve coordination and communication between team members are the following:

- **Calculation, Control and Maintenance of Socio-Technical Congruence:** As we have explained before, a high level of STC is associated with better coordination and communication between team members. Agents will thus be in charge of calculating, controlling and maintaining STC in order to improve coordination and communication. To calculate congruence, the agent will use the approach given by Kwan et al. [22] because this is the one that is best adapted to the global context; these authors use temporal distance, geographical distance and communication interactions to calculate congruence. The next goal is directly related to this first one, because it derives from the measuring and control of STC.
- **Detection, analysis and elimination of gaps:** Taking into account that agents are calculating and monitoring (controlling) the level of STC, they are also able to detect which team members are not interacting with those they should be, based on their coordination needs; in other words, agents are able to detect gaps. Given the negative effects for coordination wherever gaps exist, agents will detect important gaps, seeking to close them and maintain a good level of STC and coordination. Moreover, they will check which actions have been the best ones in closing these gaps.

To achieve these goals, we have designed an agent architecture composed of a set of modules that will manage the agents' behavior. Before explaining each architecture module, it is important to know how agents obtain and share information with other agents and/or other software entities, if we are to understand the data/information flow in the architecture.

Agents are usually represented as "Input-Process-Output" entities, since they receive information from other agents or systems; they then process that information, and finally they generate an output that is usually translated into an action. The way that agents receive the input is by using messages. In our case, these messages will come from other agents or from the user (by using a tool), and will contain data/information that agents should process in order to generate an output (an action). If the user wants to update personal data, for instance, he/she will use a tool, which will send this information to the agent using a message.

5.2.2. Scenario

This agent architecture has been designed for use in a global software environment, i.e., in a Global Software Project, where different teams may exist, and where team members can be distributed geographically in different countries. A specific example of a scenario where agents can be used is the one represented in Fig. 3.

In Fig. 3 there are three teams in charge of implementing a software system. This system has been divided into three parts; each team should implement each part. These parts are dependent; coordination is needed between the teams and their members.

Each team is composed of team members; members from a given team should coordinate and communicate with those members from the other teams that are working on dependent parts of the system.

In this kind of scenario, each user would have a personal agent, who would be in charge of registering user preferences, user coordination needs and user interactions (to obtain actual coordination), calculate congruence, etc. Using this information, each agent would control the coordination gaps of each user and would select appropriate actions to eliminate any gaps. The scenario presented in Fig. 3 could thus be represented as follows (see Fig. 4).

In this case, Fig. 4, each "smiley" represents an agent (a team member); the linking lines represent the information exchange between agents. This exchange of information between agents is necessary to maintain congruence, as well as to improve coordination and communication.

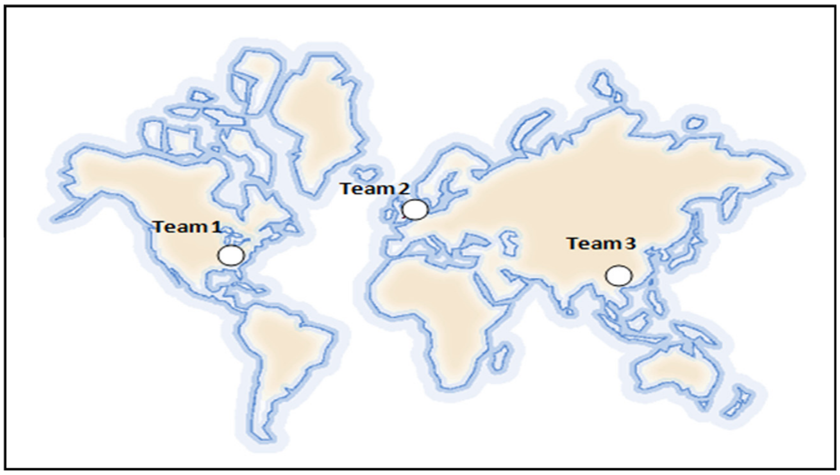


Fig. 3. Scenario.

To understand how agents are able to achieve the specified goals, in the following sections we set out the architecture structure in detail and describe how the architecture modules interact. We also indicate the data flow between modules for some of the main features.

5.2.3. Architecture structure

Bearing in mind the model used by the agents to obtain and share information, the architecture modules (shown in Fig. 5) can be presented:

- **Interpreter:** This is a commonly-used module in agent architectures, in charge of checking the type of input (message) that agents receive. Depending on the type of message, the Interpreter will send the message (or the information included on it) to one of the connected architecture modules in order to process the input.
- **Profile Manager:** As we have said before, one of the key aspects of the architecture is to follow a non-intrusive approach and adapt the information to each user's preferences. This being the case, we have defined a module called Profile Manager that will be responsible for checking the user's preferences so as to adapt the way agents interact with users. This profile manager will thus manage information about, for example, when the user prefers to be contacted, what his/her working hours are, or what the user's role is in the project. This information will be managed to carry out actions such as to adapt agent's notifications to the user's preferences, for instance.
- **STC Manager:** This module is divided into three sub-modules and its main goal is to obtain enough information to calculate STC. The sub-modules are the following:

– **Interaction Manager:** To measure Socio-Technical Congruence, the actual coordination that team members are really performing needs to be obtained; this is extracted from team members' interactions. This module is thus in charge of monitoring, and registers the interactions of each user. This module should be adapted to the specific use of the architecture, since it might draw interactions from different sources such as the email database, discussion forums, and chat messages.

– **Coordination Manager:** Another of the elements needed to measure Socio-Technical Congruence is the coordination needs. This module will capture and monitor those over time. These needs may be extracted automatically by connecting this architecture module to a Project Management tool. For example, it could be connected to Jira[1] by using the Jira connectivity options (plugins or web services) to extract coordination needs from the relationships between tasks and workers.

– **STC Calculator:** This module processes the information obtained by the agent, in order to calculate the STC and considers this to be a general indicator of the user's coordination and communication level. In this case, the agents calculate the STC by using a weighted approach based on that presented by Kwan et al. [22]. Rather than proposing a new STC measurement, we use a modification of Kwan's approach, so as to adapt the existing STC measurement to the GSE context. Kwan's approach has been adapted to our highly-distributed environment through the inclusion of a set of factors that influence the way in which globally-distributed team members perform coordination and communication activities. These factors have been included as weights that determine the STC level of each user.

[1] www.atlassian.com/JIRA.

In the original approach, the dependencies which one task has with another, the time that a worker is supposed to spend on a task and the expertise that a person has as regards a task were considered as weights. However, in our case, we considered the following factors as weights:

- **Socio-Cultural Distance (SCD):** This factor is calculated by using the countries from which the team members originate and by consulting a table, presented in [19], in which 50 countries are rated according to a set of socio-cultural factors that influence the way in which team members work in each country.

  This factor influences the STC level as follows: Interactions performed by team members with a high SCD are considered less important than interactions performed by team members with a low SCD. By using this factor in this manner, STC levels will be higher if the SCD is low than if the SCD is high.
  In summary, our agents consider that team members with a high SCD need more interactions to maintain a good level of coordination (a good level of STC) than team members with a low SCD. By way of example, if two team members belonging to the same city perform 4 interactions in a week, this will be considered as a better coordination level (higher STC level) than in a case in which the same numbers of interactions are performed by two team members belonging to different countries, because in the second case more interactions are already considered necessary.

- **Temporal Distance (TD):** This factor is used in a more analogous manner than SCD, and is measured as the difference in time between locations. In this case, it is considered that team members who are separated by a greater time difference need more interactions to maintain a good level of coordination (a good level of STC) since they do not work at the same time-zones. Our agents will therefore consider any coordination interactions performed by team members separated by a greater number of hours to be of less importance and this way, to obtain a better STC level, these team members will need more interactions and thus we will be fostering interactions among separated team members.
- **Geographical Distance (GD):** This last factor is measured by the amount of kilometers between locations, and is used in a similar way to the previous factors. Interactions performed by team members separated by a great geographical distance will therefore be considered less important than interactions performed by co-located team members, since our agents will consider that co-located team members need fewer interactions to maintain coordination, and also that these interactions are easy to perform.

Although these three factors are used in a similar way, it is necessary to use all of them, since a high SCD does not imply a high TD or a high GD. For instance, Spain and Argentina are separated by a high TD and GD but the SCD is not very high because, among other things, the language is the same. The best case, supposedly, in terms of STC, will thus be when team members belong to neighboring countries, and the worst will be when team members belong to highly separated cities (countries) with very different cultures.

The last factor that we take into account is **Task Priority**, in order to consider technical aspects of the global project. We believe that it is better to perform a higher number of interactions when task priority is high. This is translated to a higher level of STC, and our agents will consider that coordination interactions performed to finish high priority tasks are of less importance, thus ensuring a higher number of interactions to obtain a good coordination level.

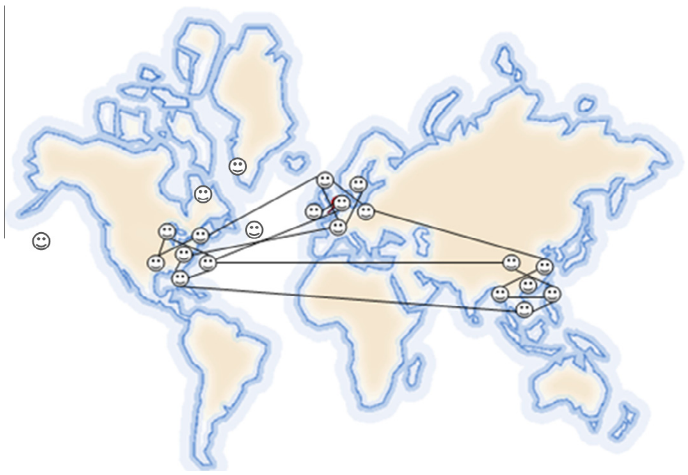


Fig. 4. User agent distribution.

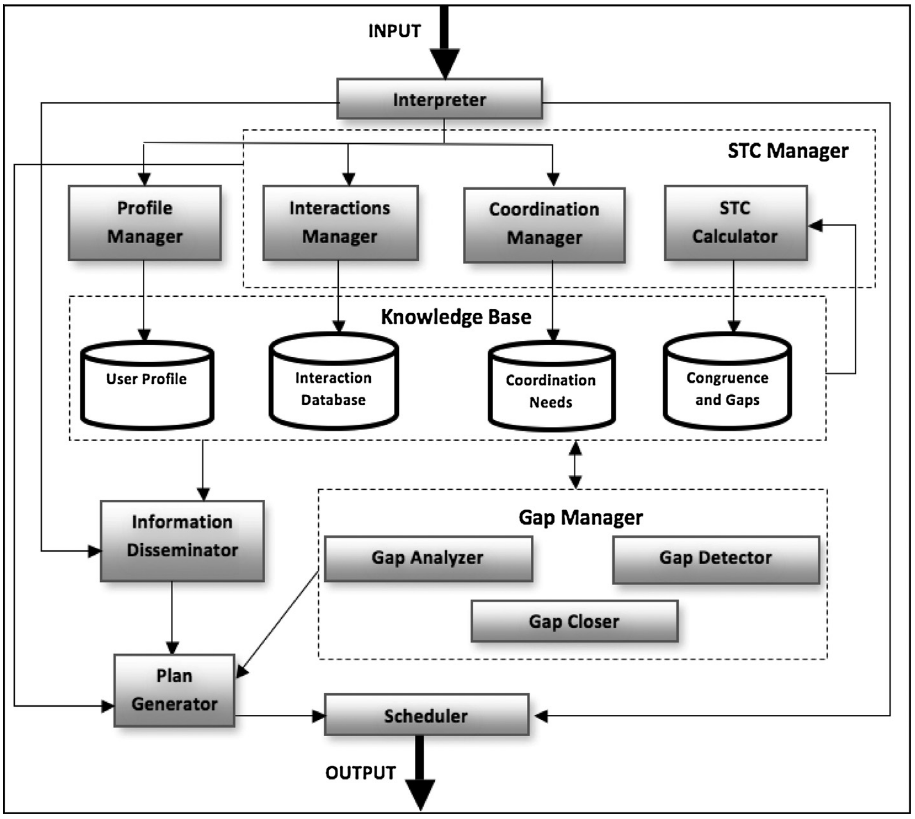


Fig. 5. Agent architecture.

Another modification that has been made to Kwan's approach is related to the fact that approach was defined for project developers. In our case, the scope is any project stakeholder working on project tasks or issues. The scope of the original approach has therefore been extended.

Apart from the aforementioned changes made to Kwan's original approach, our architecture also considers entities when calculating STC. These are the following:

- **Technical Entity:** The technical entity that will be used by the agents will be that of issues. In this work, issues represent the technical entity in which tasks are defined and may therefore include tasks such as the performance of a new development, or the resolution of any simple support task.
- **Technical Dependency:** Of all dependencies between issues, this architecture considers two main types of dependencies by default: a parent–child dependency and a general "depends-on" dependency, signifying that two issues are dependent, but that the reason for the dependency is not specified. The architecture database can be used, nevertheless, to increase the number of dependency types in the effort to support specific issue relationships.
- **Coordination Need (CN):** As stated previously, when explaining the Coordination Needs Module, agents will consider that there is a coordination need between two project stakeholders when there is a dependency between two issues, when one issue is assigned to one of the project stakeholders and the other is assigned to the other stakeholder. The agent retrieves all of the user's coordination needs to generate a coordination needs matrix. The coordination needs matrix is weighted by using all of the aforementioned weights. A coordination need is thus stronger if team members are highly distributed. Bearing that in mind and in order to calculate Coordination Needs, we need to calculate: (1) Assignment Matrix- representing which tasks are assigned to whom- and (2) Task Relationships Matrix- representing dependencies between tasks, we apply the weights as follows:
- Assignment Matrix (A) values are weighted by using task priority. This way we give more importance to assignment of tasks that should be finished with greater priority.
- Task Relationships Matrix (D) values consider only two values (0,1), representing that there is no dependency between two tasks.
- We construct the Coordination Needs matrix by using the following formula:

$$\mathbf{CN} = \mathbf{A} \times \mathbf{D} \times (\mathbf{A})^{\mathbf{t}}$$

Finally, we use as final weights the normalized sum of the distance factors to adjust the coordination needs values that have been obtained.

The final effect that we obtain by calculating the coordination needs in this way is that there will be a greater need for coordination when task priorities and distance between users are higher.

- **Actual Coordination (AC):** The Actual Coordination matrix is calculated by using the same approach as Kwan's. The agent generates a matrix of user interactions that represents the Actual Coordination matrix. This matrix is weighted by considering the amount of ongoing communication, which is normalized by using the largest value of communication between two team members (used in Kwan's approach), in addition to the weights mentioned previously (our adaptation).
- **Congruence calculation:** Congruence is calculated by comparing the Actual coordination matrix and the Coordination Needs matrix. If an edge exists in the Actual Coordination matrix and it also exists in the Coordination Needs matrix, the agent considers that there is congruence. Finally, in order to extract a general indicator of the level of coordination and communication, the agent calculates the percentage of coordination needs that the user is fulfilling, by using the following formula:

$$\mathbf{Diff}(\mathbf{CN},\mathbf{AC}) = \mathbf{card}\{\mathbf{diffij}|\mathbf{Cnij} > \mathbf{0}\ \&\ \mathbf{Acij}\ >\ bf0\}$$

$$\mathbf{LC} = \mathbf{Diff}(\mathbf{CN},\mathbf{AC})/|\mathbf{CN}|$$

With this approach we obtain as a result that people related by high priority tasks, who are at a great distance from each other, will need more interactions to achieve their coordination needs, which means a good level of coordination. This is done because we consider that people working in geographically-distant places, talking different languages, etc. will need more interactions to keep coordinated.

- **Gap Manager:** One of the main goals of measuring Socio-Technical Congruence is to detect and close important coordination gaps. Given that actual coordination is related mainly to communication interactions, this module will be in charge of detecting both coordination and communication gaps. Apart from that, this module will eliminate the gaps detected by also taking into account the non-intrusive approach. To achieve this goal, this module has been divided into the following sub-modules:
- **Gap Detector:** This module will check the fit between coordination needs and actual coordination, to obtain possible gaps. This information is taken from the Knowledge Base once the STC Manager has stored this information.
- **Gap Analyzer:** This module will analyze the gaps detected in order to find out which are most important and need to be closed quickly; that is, it will make an analysis of the relative importance of gaps.

The importance of gaps is calculated by using a variation of Kwan's approach [22], weighting the gaps with a normalized value of the sum of SCD, TD, and GD. To be specific, this module will adjust the importance of the gaps calculated by multiplying the values obtained by one, plus a normalized sum of the four explained weight values between each pair of users. Our agents thus place more importance on those gaps in which team members have most difficulties as regards communication and coordination.

In the example shown in Fig. 6, there are two gaps which are rated with 0.2 (Gap A) and 0.12 (Gap B). If we imagine that the weights values calculated (normalized sum of SCD, TD and GD) are 0.05 (representing people with a low distance value) and 0.8 (representing a high distance value) respectively, then the agent finally obtains 0.21 for Gap A (A' in Fig. 6) and 0.216 for Gap B (B' in Fig. 6. This means that, although Gap A was initially more important, Gap B is eventually more important, owing to a higher socio-cultural distance and/or temporal distance and/or geographical distance.

- **Gap Closer:** This last module will select the right actions that the agent will perform to close the gaps. To select the right actions (the right plan) this module will take into **account preferences** of its user and preferences of the other users implicated. For instance, if the agent wants to close a gap by connecting two users via email communication, it will consider when the users prefer to be contacted, as well as the working hours of each user. To do this it will use the information stored in the User Profile Data Base by the Profile Manager.
- **Information Disseminator:** This module will access the information stored by the agent and will share it when is needed by another agent. The information that agents will share will be stored in five databases that make up the Knowledge Base of each Agent.
  - **User Profile:** This Knowledge Base (KB) will contain the information related to user preferences or user profile. It will thus contain, for example, the role, native language, work-place or contact preferences, among other information.
  - **Actual Coordination:** This KB will contain the interactions that each user is actually performing, including when, how and with whom each interaction has been performed.
  - **Coordination Needs:** This KB will contain the coordination needs of each user. These coordination needs will be associated with a period of time, in order to log the changes over time.

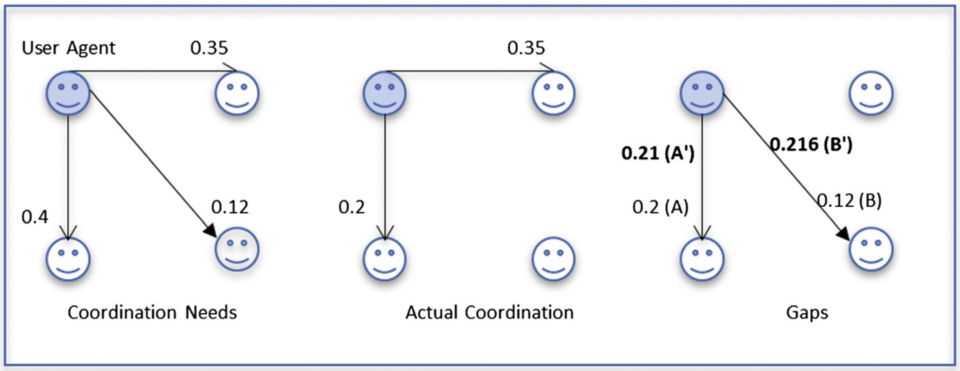


Fig. 6. STC and gap calculation.

- **Congruence and Gaps:** In this case, this KB is used to store more complex information. Its goal is to store information related to calculated values of STC and gaps. It will therefore contain the gaps detected, as well as the importance of the gaps, along with the effectiveness of the plans chosen to close the gaps. Agents can thus learn the best plans for each type of gap, ensuring that they can at the very least apply the most effective plans first of all. Moreover, agents will be also able to check whether the plans applied are helping to obtain better levels of congruence.
- **Plan Generator:** Taking into account that in the agent context a plan is a set of one or more actions, this module contains all the possible actions that the agent can perform; depending on the needs, this module will select a set of actions to generate a plan. The other modules will then request the appropriate plan that helps to achieve the agent's goals. For instance, if the Interaction Manager Module wants to check interactions in the email database, this request is sent to the Plan Generator Module and this latter will select the set of actions needed to perform that task. Basically, it works as a predefined plans store, and it is a module that is typically used in agent architectures.
- **Scheduler:** This module is also one that is typically used in agent architectures. It is in charge of managing the priority of the actions that agents have to perform. So, for example, if the agent has not got enough information to calculate STC, the scheduler will set this task as a priority, because without enough input information the architecture outputs (actions) might not be accurate. On the other hand, if another agent has requested information about the current agent, the scheduler will give more priority to this task, so as to offer the requested information quickly and allow the other user's agents to get on with their tasks.

For a better understanding of the architecture, it is important to explain how the agents are able to achieve their goals and improve current tools by using the architecture. However, before explaining this, it is necessary to present the scenario where agents will be used.

5.2.4. Using the architecture

We have seen the goals, structure and scenario of the use of the architecture. In this section, we are going to explain how the architecture achieves the main goals and how agents include improvements, thus completing the answers to the research questions that we indicated in the introduction section.

How Agents Calculate and Control Socio-Technical Congruence: Each agent will calculate individual STC values for each team member, by making use of our approach. This, as explained previously, is an extension of the Kwan et al. approach [23,22]), which consists of matrix representations of coordination needs and actual coordination; it also makes it possible to create an importance ranking of the gaps.

The agent will obtain the coordination needs (A process in Fig. 7) by using the ***Coordination Manager Module*** to request the information needed. The Coordination Needs Module does this by requesting the execution of the appropriate plan (set of actions) from the **Plan Generator Module**, which sends the plan to the **Scheduler** to be executed when possible.

Once the external module has sent back the information requested, the ***Interpreter Module*** receives this information and sends it to the ***Coordination Needs Module,*** where it is processed and stored in the ***Knowledge Base (KB)***. In this case, the information required is the dependencies that the tasks on which the user is working have with other tasks, along with the people assigned to the tasks involved and the priority of each task, in order to obtain the task importance. This information could be extracted from a Project Manager tool, or it may be introduced manually. In the example shown in Fig. 7, the information extraction is carried out by using web service technology from a Project Management Tool.

At the same time as the agent starts the process of retrieving coordination needs, it also begins to search for user interactions (B process in Fig. 7) by using the ***Interaction Manager Module.*** This is possible since each plan is managed as a thread, and several threads can be executed at the same time. In the example, this is simplified, in that the agent extracts interactions only from an email server.

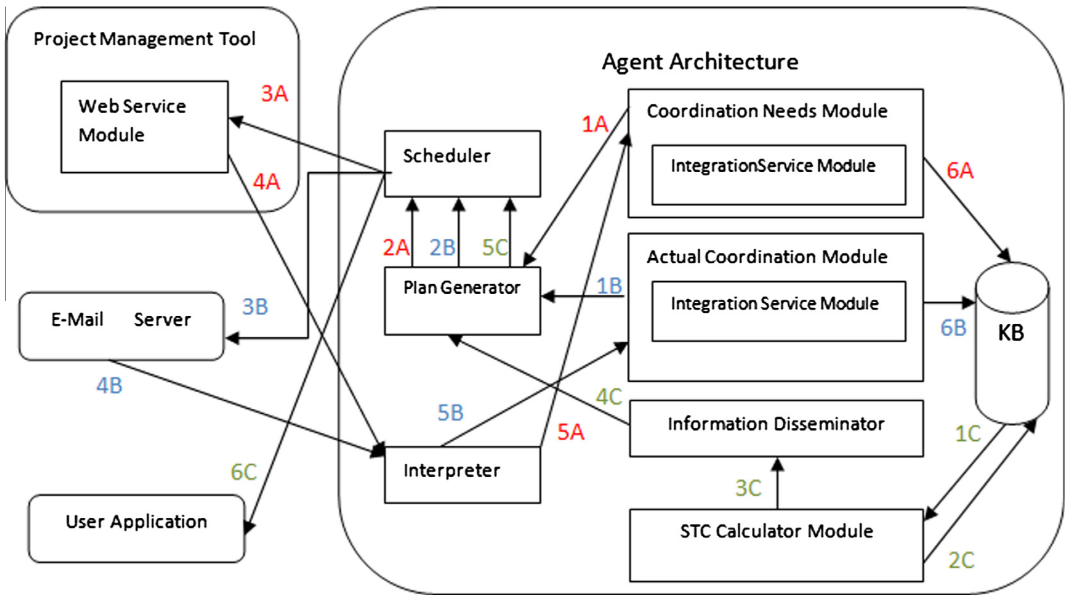


Fig. 7. Module interactions to calculate STC.

In order to ensure that the agent is specialized in each searching method, a plan (a set of actions that an agent should perform to achieve a concrete goal) has to be implemented for each method in the ***Plan Generator Module***.

By way of example, an agent might use one plan to extract interactions from an email database, and a different one to extract interactions from a forum. In this case, the agent executes a plan to obtain interactions from emails. Thus, when the ***Interaction Manager Module*** starts the process, it will send the order to obtain interactions to the Plan Generator, which will select the plans available that are related to the extraction of interactions (from emails, forums, comments, etc).

Once the search for plans has been executed, the agent will receive the interactions in the aforementioned XML format through the **Interpreter Module**, which will in turn send the information to the **Interaction Manager Module,** which eventually stores the data in the ***Actual Coordination Data Base***. Once the agent has gathered sufficient information, it will use the **STC Calculator Module** to calculate congruence and store the value obtained in the database (Steps 1 and 2 of the C process in Fig. 16). The database can later be used by this module to extract the history of the STC values, in order to check whether the congruence level is high at that time, and that the coordination needs are therefore being fulfilled.

This module will calculate the STC by considering data belonging to the time interval in which the current dependent tasks have to be performed. The agent will therefore obtain an STC value each time a task, or a set of tasks, has been performed (see Fig. 8).

The idea is to control congruence when there are dependencies between tasks and where there are consequent coordination needs between team members. In the example shown in Fig. 17, coordination needs are registered at t1 and t3, the actual coordination is gathered between t1–t2 and t3–t4, and the STC values are calculated at t2 and t3.

In this example the user is fulfilling all the coordination needs; that is, STC = 1 (STC is a value between 0 and 1). It is assumed that task dependencies are available via a Project Management tool or are manually introduced by, for instance, the project manager. This can be useful as an indicator of the team members' coordination level. For example, the agent could send this information to the Project Manager by using the ***Information Disseminator Module*** (Steps 3,4,5 and 6 of the C process in Fig. 7) and the manager may check whether there is any problem with this team member.

The information used to calculate STC can otherwise be sent by all the agents to a central repository to obtain the STC level of the team, the project or the organization if it is needed. This is also an improvement on previous tools, since they do not take into account the multi-level nature of congruence. A project manager could thus access information related to one or more team members, or information related to a whole team or a project.

How Agents maintain Socio Technical Congruence: Another goal of the architecture is to maintain the STC. The agents will attempt to maintain a high level of congruence by following the example in Fig. 8, from t1 to t2. To achieve this goal, the agents will help users to interact with those people who have coordination dependencies. It is important, at least at the beginning of a project, to know who needs to be coordinated with whom [31]. This task would appear to be easy in a co-located team; in a Global Environment, however, in which team members are separated and it is difficult to establish personal relationships, agents will help team members to get to know the other people. As explained above, the agents will calculate which people have coordination needs with the user, and will also offer useful information such as how to contact, when it is better to contact (taking into account user preferences) or personal information; all of this will create a more confident relationship. This information will be extracted from the ***User Profile Data Base*** by using the **Profile Manager**

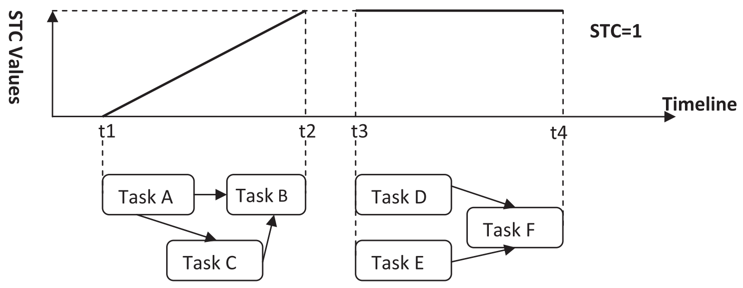


Fig. 8. STC measure over time.

Module. The agent will obtain the other users' preferences by communicating with the other agents via a request message. These other agents will respond with a message including the profile of each user, and this information will eventually be given to the user. To perform this task, the **STC Calculator** module should first detect a decrease in the user's STC level, or the user should request contact information through an implemented tool connected to the agent. For instance, after a user request has occurred, this module requests from the **Plan Generator** the specific plan needed to request contact information from other agents. This plan, as always, is sent to the Scheduler to be executed when possible. Next, when the other agents receive the request, the **Interpreter Module** detects that it is an information-request message and sends the content of the message to the **Information Disseminator Module**. This module checks which information has been requested, takes it from the **User Profile Data Base,** since it is contact information that has been requested, and connects with the **Plan Generator Module and the Scheduler** in order to return the information requested. Finally, the requester agent receives the information and sends it to its Information Disseminator, which is in charge of showing it to the User (see Fig. 9). This technique is performed to keep team members connected, make interactions possible and therefore help to maintain a high level of STC. By using this technique, agents will also assist in improving the communication between team members. These actions that the agents perform make up one of the additional improvements on other implementations in which the tool simply visualizes STC-related data, but does not perform actions to maintain the level of STC. Moreover, the agents will attempt to maintain the level of STC by managing the possible gaps.

How Agents detect, analyze and eliminate gaps: The agent architecture has a Knowledge Base that includes data concerning coordination needs, actual coordination, gaps and user profiles. Through its three modules, the **Gap Manager Module** is able to use this information to detect, analyze and perform actions to eliminate gaps.

Firstly, the ***Gap Detector Module*** is in charge of checking the existing gaps in real time by using the information stored in the **STC and Gaps Data Base**. Once a gap has been detected by this module in the database, it is sent to the ***Gap Analyzer Module,*** in order to analyze whether the gap is sufficiently important for it to be worth disturbing users with extra information or tasks. This module analyzes the gaps detected by using calculated weights which, as explained previously, are based on socio-cultural distance, ongoing communication and task priority. This module finally creates a gap importance table, in which each gap is associated with its importance value.

The ***Gap Closer Module*** will use this value to notify, over time (from t1 to t2 in Fig. 10), the users involved, informing them of important coordination needs. The number of notifications will be proportional to the importance of the gap detected. The gap importance has been divided into three levels, in which 0–0.33 is considered to be the Low Level, 0.33–0.66 the Medium Level and 0.66–1 the High Level.

More specifically, when a gap is detected, the agent will notify the user of this situation, as follows (see Fig. 10):

- If the gap importance is Low (L), the agent will notify the user of this gap once in the middle of the time that the agent is monitoring (A).
- If the gap importance is Medium (M), the agent will notify the user of this gap at a quarter, and three quarters, of the time period.
- If the gap importance is High (H), the agent will notify the user of this gap at a quarter, a half, and three quarters, of the time period.

In the example shown in Fig. 10, the gap is detected at the beginning, but if it were detected at the middle, the remaining time would be divided in the same way and the notifications would be closer. This gap detection, analysis and notification are further improvements on other implementations. Other tools in existence at the moment do not consider the importance of the gaps; neither do they analyze gaps and perform tasks to close them. This is made possible by the autonomy feature of software agents, which is not included in the other tools; in them it is the user who should analyze and decide which actions can be performed to close possible gaps.

To summarize, in this Section 5.2.4, we have presented how agents are able to achieve the goals presented in Section 5.2.1. We have also presented some of the improvements that agents introduce with respect to other implementations. In doing all this, the research questions are completely answered.

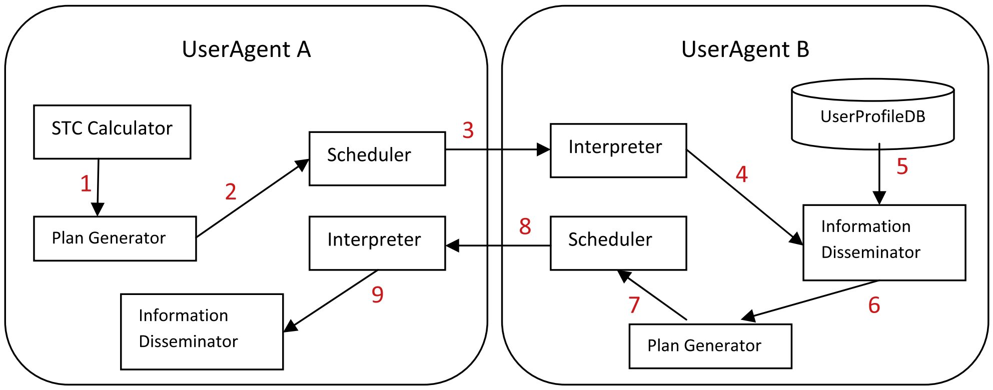


Fig. 9. Retrieving contact information.

6. Validation: Case study at ORIGIN project

The aim of this case study is to show the feasibility of the architecture goals in a real project called ORIGIN, in which different distributed teams need to communicate and coordinate to perform their tasks. This study was performed by developing a system based on the architecture; this was then integrated into the project. The specific research questions to which the case study attempts to respond are shown in the following sub-section.

6.1. Research question

In order to validate our approach the following research questions are applied to our case study:

RQ1: Is it feasible to construct a system based on our architecture that will help distributed team members to improve communication and coordination?
RQ2: Can the tool constructed make use of the architecture modules in order to manage data automatically without user intervention?

6.2. Organizational context

The specific details of the context in which the case study was performed are explained in this section. We specifically outline the details of the ORIGIN project, that is, the project into which the system developed was integrated.

6.3. General setting

The ORIGIN project is an initiative of Indra Software Labs (ISL) who, in collaboration with several research groups such as the Alarcos Research Group, works to offer solutions for globally distributed teams, signifying that new methodologies and tools are developed to solve or alleviate GSE problems or challenges.

Of all the activities performed in the ORIGIN project, this case study is focused on the development activities performed by Indra Software Labs, specifically on the communication and coordination activities that ISL perform in the design and development processes.

The design and development processes are supported mainly by an issue and project-tracking tool called Jira. In the ORIGIN project, Jira is the “connection point” between the design and the development team, who are geographically distributed. Jira will therefore also be our core system with which our system should be integrated and in which coordination and communication activities will be analyzed by our agent-based system. Another important communication tool will also be included in the project: the email tool.

The way in which the different teams work with Jira is the following: Information concerning new modules that should be developed is received by the development team in the form of Jira issues which specify the specific functional design or functionality that should be developed; in other words, the functional requirements that the module should fulfill. This process usually includes the participation of several teams on several sites when it is necessary to develop an integration module, i.e., a module whose main functionality is to integrate two or more tools. The design and development of a module that integrates GRv2 tool (requirements management tool) and the Jira tool itself were chosen for this study.

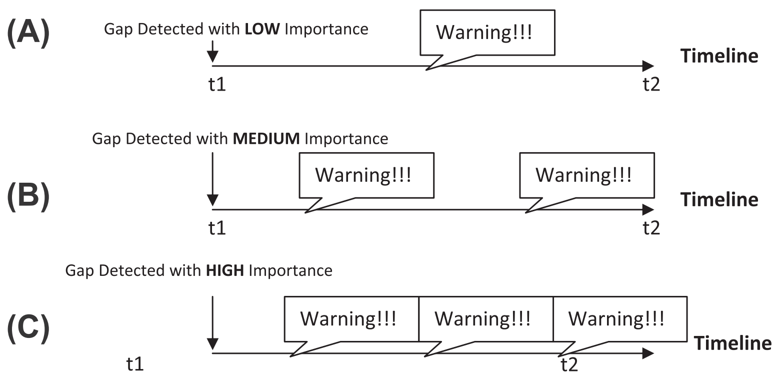


Fig. 10. Gaps notifications depending on their importance.

6.4. ORIGIN framework

The development process in ORIGIN is driven by an agile development framework based on SCRUM. Like other agile methodologies, this methodology is based on several principles described in the agile manifesto [2].

The two main challenges that the architecture, and therefore the system developed based on the architecture, should support are: On the one hand, the system should confront changes to coordination and communication needs quickly, since software requirements may change continuously as the result of new customer needs or design changes. On the other hand, our proposal should support distributed communication, since face-to-face communication, which is a key principle, cannot be performed between distributed teams when the distance is sufficient to hinder it.

As will be shown later, the system developed by using the architecture offers support to both challenges by monitoring coordination and communication needs and user interactions, sending notifications for important coordination or communication gaps and offering information about possible ways of closing these gaps.

6.5. Distributed teams setting

During the design and development of the integration module, coordination and communication should be performed by at least two teams. In our study the process includes three teams; one design team and two development teams.

The first is the development team located in Ciudad Real; it is in charge of developing the sub-module that allows the tools from GRv2 to be connected. The second team is the development team located in Malaga, which is in charge of developing the sub-module responsible for enabling integration from EA, while the third is the design team that is located in Madrid. Apart from design, this team is also in charge of controlling the progress of the development, and therefore the progress of the project.

Table 2 shows how these three teams are geographically distributed. This scenario does not present a typical globally-distributed project, and aspects such as time or language differences cannot be taken into account. However, Table 2 shows that the three teams are sufficiently geographically distant to be able to test the architecture, since there are hundreds of kilometers between each team. These distances were considered to be sufficient to perform the case study, since there are enough coordination and communication needs between the three teams to make our proposal useful.

Although there are three teams in this case study, not everyone belonging to these teams was available to perform the case study. That meant that there were 16 participants, distributed as follows: 6 participants from the development team in Ciudad Real, 4 participants from the development team in Malaga and 6 participants from the design team in Madrid. These participants were the people interviewed, following the process explained previously. Table 3 shows a list of the participants (interviewees) and the roles played by each of them in each team.

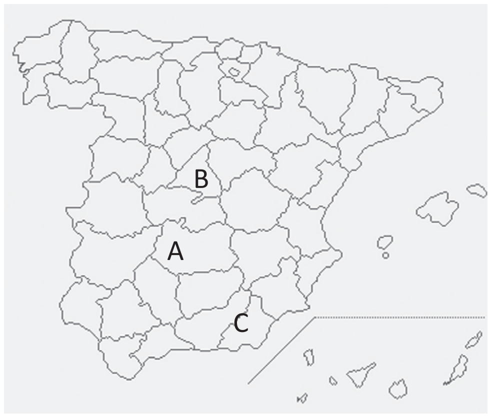
When considering the available participants and roles, it is important to know that there is a Project Leader (H) who coordinates the work by communicating with team managers of each site via weekly meetings. It must also be pointed out that each team manager should coordinate the work performed in each team with the help of the Development Manager, who needs to coordinate with the other Development Managers only for development issues. Bearing these coordination aspects in mind, in the next section we shall describe the data extraction process.

6.6. Data extraction

This section shows how the data extraction process needed to conduct the case study was performed. In this case, the data extraction process was performed by using semi-structured interviews: (1) to detect relevant findings that will show how our tool is able to improve coordination, and (2) to study the technological environment and adapt our tool to it. Our tool was also used to extract automatically the data needed to perform the STC analysis.

Table 2
Distribution scenario for the case study.

Distance (km s)	A	B	C
Development Team 1 (A)	–	217	348
Design Team (B)		–	538
Development Team 2 (C)			–



The system will specifically extract data related to coordination, control and communication between the case study stakeholders, thus retrieving communication interactions, task relationships or control elements, among other things. This has been made possible thanks to the availability of a set of tools that permit the extraction of these elements. These are: ***Jira,*** which is used to extract communication, coordination and control elements, together with communication tools such as ***MS Outlook***.

However, before deploying the system based on our architecture, it was necessary to extract some other kinds of information regarding the technological environment. That information would give us a general idea of how our system could extract the aforementioned data automatically, and it was obtained by using semi-structured interviews.

A study of the technological environment enabled us to find out which tools were used for communication and coordination and which artifacts were used to organize the work. The study was necessary for the implementation of the system, since this information provided us with the main features of the environment with which the agents should be connected. As will be shown, all the details extracted here are key aspects that should be considered during the implementation of what was defined in the architecture as the **Integration Service,** which connects the architecture to external systems that provide useful data.

It is essential to study details concerning communication tools and artifacts, coordination tools and artifacts, and control tools and artifacts. Artifacts are considered to be those elements that store information about tasks and communication, coordination and control interactions. In this case, we shall consider artifact files, API objects or database data from which information about tasks and interactions can be extracted by the agent.

The technological environment is made up of the following configuration:

Communication tools and artifacts: The three teams use five principal communication tools. The first is that of email communication through Microsoft Outlook, which is the most frequently-used means of communication among the team members in our case. The second is instant communication through the use of the Microsoft Office Communicator; this is also in common use, although it is not employed as frequently as the former tool. The third is voice communication, also using the Microsoft Office Communicator (MOC). This third means is not commonly used on a daily basis, since not everyone has headsets; or it may be that the tool with which to make calls has not been configured. The fourth is the commenting tool offered by Jira, which allows users to write comments on issues in order to, for example, request details of the issue specification. The last is that of telephone calls. The project leaders commonly use this communication tool, since ISL provides them with a company mobile phone. The rest of the team has a voIP telephone that is usually used when important development problems arise.

In order to make use of these communication interactions in our analysis, we need to know which artifacts the tools use to store interactions, thus, as is shown in Table 4, email interactions can be extracted from the outlook email database, and MOC Instant Messaging interactions can be obtained from conversation files that store conversations in XML format. MOC Voice calls are also registered by the tool, but are not available for external access. Telephone calls, however, are registered on invoices, but, although it is possible to extract a set of interactions by checking who called whom from these invoices, they were not available for this study. Finally, issue comments are stored by Jira in its own database and can be accessed by using the Jira API.

The Information Relevance and Percentage of use indicators were analyzed, in the effort to discover the overall reliability that the agent-based system would have in relation to communication analysis.

The analysis of both indicators (information relevance and percentage of use) had to be extracted manually by studying the organization's communication interaction options. In this case, the information relevance level was found by considering

Table 3
Interviewees' roles in the case study.

Interviewee	Project role	Team
A	Team Manager	Ciudad Real (A)
B	Development Manager	Ciudad Real (A)
C	Software Engineer	Ciudad Real (A)
D	Senior Software Engineer	Ciudad Real (A)
E	Senior Software Engineer	Ciudad Real (A)
F	Software Engineer	Ciudad Real (A)
G	Team Manager	Madrid (B)
H	Project Lead	Madrid (B)
I	Development Manager	Madrid (B)
J	Software Engineer	Madrid (B)
K	Senior Software Engineer	Madrid (B)
L	Senior Software Engineer	Madrid (B)
M	Team Manager	Malaga (C)
N	Development Manager	Malaga (C)
O	Software Engineer	Malaga (C)
P	Senior Software Engineer	Malaga (C)

which types of interactions would provide us with most information about the Jira issue on which the teams were working.

The percentage of use was obtained by interviewing the team members. In these interviews, people were asked about the number of times they used each communication type with other distributed team members. This information was then used to discover the average of the percentages of use of each type of communication interaction, resulting in the percentage shown in Table 4.

The analysis of these indicators may provide an idea of how reliable the agent-based system will be. An ideal environment would be one in which all interactions could be registered automatically by the agent and, for each interaction, the task (issue in our case) to which the interaction is related could also be extracted automatically.

However, although this will be possible one day, in our case study, the agent system developed was able to extract interactions automatically from two communication tools of the set: email and issue comments. These tools are supposed to represent 58% of interactions, which is a high percentage. What is more, both tools have been classified in medium and high levels of information relevance, signifying that the agent's actions will be based on reliable data.

Coordination tools and artifacts: Since our agents also analyze coordination, it is important to know which coordination tools and artifacts are used by the team members to extract sufficient information to perform a reliable analysis.

In the context of this case study, the main tool that the team members use to maintain coordination is that of Jira. Jira allows the creation of issues and the definition of issue relationships in its effort to coordinate the work. In this case, team members associate Jira Issues with Tasks or Bugs. When it is necessary to perform a new task, a new issue is usually created, which also occurs when a bug has to be fixed. From here on, the term *issue* will therefore be used to refer to tasks or bugs.

Issue coordination takes place by dividing Jira issues into sub-issues, thereby creating a parent–child relationship. Jira also permits the definition of other relationships between tasks (issues) through the use of the issue-linking feature.

Issue Linking permits the creation of an association between issues on either the same or different JIRA servers. For instance, one issue may *duplicate* another, or its resolution may *depend* on that of another. Jira includes the following types of links:

- **relates to**/is **related to:** This relationship means that two issues are related. The reason for the link is not defined.
- **duplicates/is duplicated by:** This relationship means that an issue duplicates, or is duplicated by, another issue. It allows the user to know with whom s/he should be coordinated so as not to duplicate work.
- **blocks/is blocked by:** This relationship defines the order of execution of issues, since it is possible to know which issues should be resolved or completed, in order to start with another issue that is blocked by the first.
- **clones/is cloned by**: This link is usually used when two tasks should be carried out in different projects. Rather than creating a new issue, the existing issue is cloned in the new project.

Jira issues can also be used in task coordination to link team members by using the assignee field. In our project, the project leader (Reporter) fills in this field and Jira automatically sends an email to the team member to whom the issue has been assigned (Assignee). By combining this information (issues assignees and issues links), our agent system is able to extract the coordination needs that should be fulfilled by team members. This means that an Issue A can, for example, be connected to an Issue B by a blocking relationship (Issue B is blocked by Issue A), signifying that it is necessary to complete Issue A in order to complete Issue B. Moreover, taking into account that Issue A is assigned to User A and Issue B is assigned to User B, the agent will consider that there is a coordination need between User A and User B.

Table 4
Classification of interactions sources at ORIGIN.

	Outlook e-mail	MOC instant messaging	MOC voice calls	Telephone calls	Issue comments
Artifacts	E-mail database	Conversation files	Calls register file	Invoices	Jira database
Easily accessible	Yes	No	No	No	Yes
Information relevance level	Medium	Medium	Low	Low	High
Percentage of use (%)	35	10	7	25	23

Control tools and artifacts: In this case study, the principal tool that is used to obtain control elements is Jira, once more. Jira includes sufficient information to be able to control the progress of the project. In this case, the agent will have at its disposal the following artifacts with which to extract control data:

- **Issue Time control:** In order to control the evolution of issues over time, it is possible to extract the time spent on each issue, the original estimation of time needed to finish each issue, the due date of the issue, the theoretical and real start date, and the theoretical and real end date.
- **Project progress control:** Since the design and development process in ORIGIN follow an agile methodology, the progress of the project can be extracted by checking which issues were planned to be finished in a Sprint. It is also possible to check at any moment of the Sprint, such as at the end, whether the issues planned for that Sprint have been finished and closed.

By considering both time and project control, our agent will be able to get together the data it needs to establish the interval of time in which the STC will be controlled, as well as the set of issues that will be controlled for any user. The agent will achieve this by dividing the time into Sprints, using the start and end date of each sprint to establish the intervals in which the STC is controlled.

The agent will also extract which issues, and therefore which relationships between issues or tasks, will be controlled by using the set of issues that are planned to finish in the sprint. Figs. 8 and 10, shown in Section 5.2.4, are therefore translated by the agent to Figs. 11 and 12.

Fig. 11 shows how the agent will extract the Sprint 1 start and end dates in order to create the first interval of time in which the STC is controlled and how the chosen issues (A, B and C) will be selected, since these are included in Sprint 1. This information is also gathered in a similar way from Sprint 2, and so on.

In Fig. 12, the Sprint information, that is, Sprint Start Date (SSD) and Sprint End Date (SED) is also used to obtain the number of notifications that users will receive, depending on the importance of the gaps. In the example shown in Fig. 12, the SSD is on 9/1/2012 and the SED is on 9/15/2012. Given that a gap is detected on 9/5/2012, that the gap importance is low and that there are 10 days until the end of the sprint, the agent will send just one notification at the middle of the interval, that is, after 5 days.

After a first data extraction step in which data was automatically gathered by the tool we developed, this data was analyzed. As explained previously, a second data extraction step was performed by using the second interview. The information obtained from this interview is summarized in the results section.

Since part of the data analyzed was extracted automatically by a tool that has been developed on the basis of our architecture, this tool and its main features will be shown in the following sub-section before we present the results obtained.

6.7. STCM tool

In this section we show some of the interface and usability aspects of the tool developed, called STCM (Socio-Technical Congruence Manager), to provide a better understanding of the results. The tool, which was installed in each of the participants' PCs, was designed to include a simple interface hidden in the tray bar of the operating system (see Fig. 13).

The first time the tool is run, the user has to log in by using the same login data used at ISL. Employing the company's id allows the user identification to be unified; in other words, the user is the same in Jira, our tool and the other company tools. After login, the tool shows the user a form (see Fig. 14) in which the following data has to be filled in:

- **User Id:** Each user (and agent) is identified by a unique id. The agent then uses this id to identify agents and users and for login aspects. In this case, the id is the same as the one that users have for the company.
- **Password:** The password is also used for login aspects and is encrypted by the agent. It is also the same password used in the company.
- **Personal Information:** This information includes name, surname, city, country and language, along with a profile photo.
- **Working hours:** The user should include information about his/her working hours.
- **Preferred Contact Hours:** From the set of working hours, the user should also include information about when it is better to contact him/her.

- **Holidays:** This field includes those days on which the user will not be available.
- **Alternate Contacts:** For those days on which the user will not be available, this field should contain a list of one or more users that can be contacted.
- **Contact Methods:** The list of methods by which the user can be contacted.
- **Preferred Contact Methods:** The user's favorite contact methods from the list.

Once the profile information has been filled in, the user can access the main window of the tool (Fig. 15). This window contains four tabs:

- **STC and Gaps:** This screen, shown in Fig. 15, offers two main functionalities. The first is a graph in which the evolution of the STC is presented. This graph can be generated for one of the chosen Sprints (in Fig. 15, for the current Sprint). It is also possible to select several Sprints in order to see a global evolution of STC over time and to control coordination and communication levels.

Furthermore, at the bottom of the screen the user has a list of unresolved gaps, with the option of resolving them by choosing a gap and clicking on the Resolve button. This button opens a new window (see Fig. 16) in which the tool shows tips to help the user to close the gap. These tips include information about how to contact the other users, such as contact hours, preferred means of contact or alternative contacts when other users may be in a holiday period. The user can also use our tool to open the Jira Issue by clicking on the Open Issue button.

This screen (the Resolve Gap screen) also includes an option with which to close the gap directly. This option should be used when interactions for the associated issue were made using other types of media that are not included in the tool scope. For instance, in this version of the tool, telephone calls are not analyzed, so if the user sees a gap related to ISSUE-X and he/she knows that interactions for this issue were made by using the telephone, this gap can be directly closed by the user.

- **Profile:** This screen shows the same information as that displayed in Fig. 14. This screen can be used to update the user's profile and preferences.
- **Notifications:** This tab is used only to display a list of the notifications that have been shown to the user. In this version of the tool, the only types of notifications are Gap Notifications, i.e., notifications to alert the user of the existence of communication or coordination gaps. Notifications in this version of the tool are shown using the Tray System Bar, as shown in Fig. 17.
- **Configuration:** This last tab is included to configure the tool. The screen includes the server information and the number of times that the agent checks the STC values and frequency of notifications.

Having presented the tool used in this case study to obtain the results, these findings will be presented in the following section.

6.8. Data analysis and results

This section shows the results obtained from this case study by using our tool in the ORIGIN project. More specifically, a description will be provided of the results obtained in certain specific situations, which show how the tool, and therefore the architecture, is able to improve communication and coordination in distributed teams.

The results presented in this section and the data used originated from two Sprints, one from 09/01/2012 to 09/16/2012 and the second from 09/17/2012 to 10/04/2012. The first Sprint includes 55 issues and 16 participants: 6 participants from the development team in Ciudad Real, 4 participants from the development team in Malaga and 6 participants from the design team in Madrid.

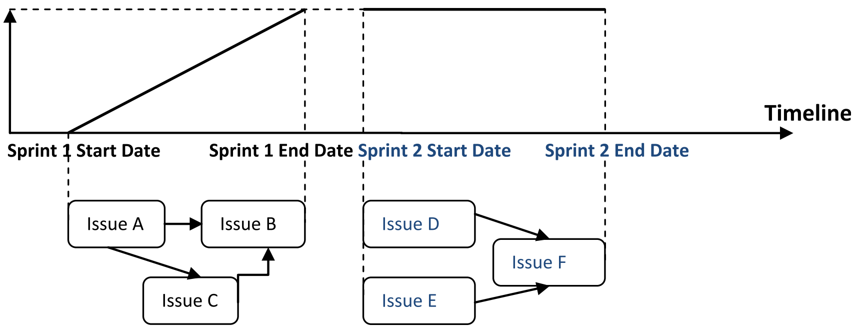


Fig. 11. STC control intervals.

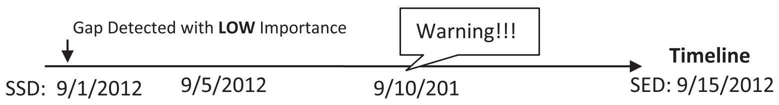


Fig. 12. Gap notifications depending on sprint dates.

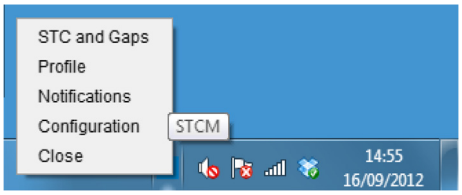


Fig. 13. TrayBar Icon.

Before starting the case study, all the participants were informed via email of the realization of the case study; the message told them that during these two sprints they had a new tool at their disposal. They were also provided with an explanation of the main features of the tool and how to use it. Those taking part were invited to employ the tool and take notes on how useful it was as regards helping them in aspects of communication and coordination.

Although all the participants used the tool during this period of time, it was not possible to obtain sufficient notes from which reliable results and conclusions could be drawn. This was simply because the people working on the project were busy during these sprints. A semi-structured interview (available in Appendix III) was therefore also conducted at the end of the second sprint to elicit the interviewee's ideas and opinions about how useful the tool was in improving communication and coordination- The aim was also to contrast the results obtained by the tool with the interviewees' point of view.

This interview was conducted face-to-face with all the participants in Ciudad Real and the 4 participants in Madrid, and by phone with the rest, since a face-to-face meeting was not possible.

It is important to note that the STC evolution graphs shown in this section were constructed by considering only those interactions between users with coordination needs. The graphs therefore show days on which no STC values are registered, because there were no coordination needs on those days. It is also important to note that this does not mean that the users have not interacted with other team members. We decided to use this means of considering STC because it is easier to show the relationship between STC and the gaps detected.

At this point, we will discuss the most relevant answers/comments given by the study participants as regards how our tool (architecture) may help to improve coordination. We have also included the special situations detected in which the tool might be useful for coordination. A presentation of the explanations given by some of the interviewees describing why and how coordination might be improved by using the tool is therefore provided below.

During the first Sprint, one of the participants from the development team in Ciudad Real found that the use of the tool might help to improve coordination with other team members as follows:

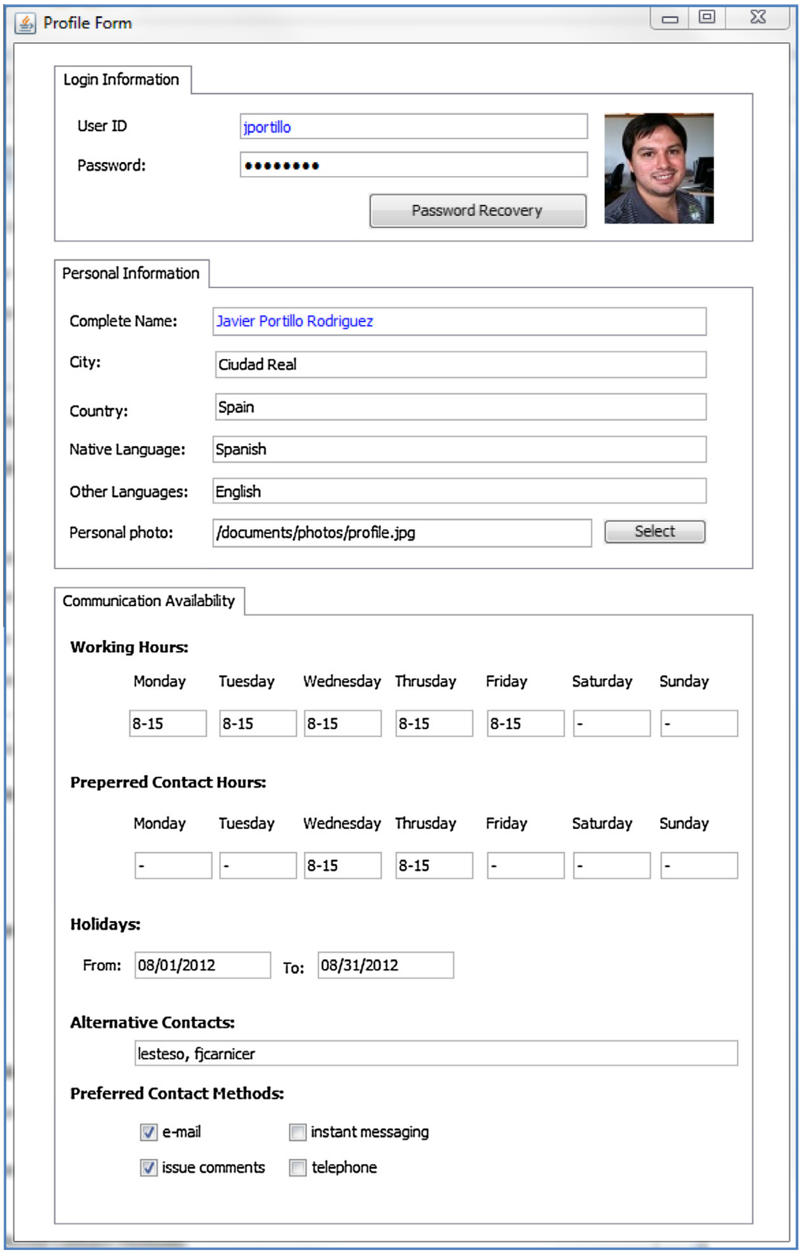
Sometimes, an issue is created (in Jira) that has no relationship with any other issues, and the user starts to work on that issue. After a time, another issue is created which depends on the first issue created. The user working on the last issue knows that there is a coordination dependency between the last issue and the first, since this relationship was set when the last issue was created. However, the first user is not notified about this issue relationship, or about the corresponding coordination dependency.

The above situation was described by one of the participants when he was asked about coordination problems that this tool might alleviate. He also said that this kind of situation "could result in a coordination breakdown and sprint delays".

This person went onto explain that our tool might help to avoid this coordination breakdown because "by using the gap notifications it is easy to check the relationship between your issues and others and therefore to be aware of the people with whom you should be coordinated". He also said that "coordination awareness is clearly improved".

Based on these comments and the data obtained in the STC evolution graphs, we detected one of the situations in which this tool/architecture might be useful for improving coordination. This scenario occurs when team members start working on issues that are not related to other ones (with no coordination needs) and where, for instance, new issues are created and coordination needs arise in the middle of the Sprint. This situation is quickly managed by our tool/architecture, since new coordination needs are quickly registered and gaps related to these needs are also notified speedily, especially when the priority of these related issues is very high (blocker in this case study).

This situation is shown in Fig. 18. Between the 1st and the 6th of September, this user did not have any issues related to other ones, and no STC values were therefore registered. However, on the 7th of September a new issue was created (ORIGIN-1334) and this issue had a blocking dependency with one of the user's issues (ORIGIN-1311). As the user told us in the interview, "thanks to the quick response of the tool, I was aware of this dependency on the same day as the other issue was created…".


Profile Form

Login Information

User ID	jportillo
Password:	••••••••

Password Recovery

Personal Information

Complete Name:	Javier Portillo Rodriguez
City:	Ciudad Real
Country:	Spain
Native Language:	Spanish
Other Languages:	English
Personal photo:	/documents/photos/profile.jpg

Select

Communication Availability

Working Hours:

Monday	Tuesday	Wednesday	Thrusday	Friday	Saturday	Sunday
8-15	8-15	8-15	8-15	8-15	-	-

Preperred Contact Hours:

Monday	Tuesday	Wednesday	Thrusday	Friday	Saturday	Sunday
-	-	8-15	8-15	-	-	-

Holidays:

From: 08/01/2012 To: 08/31/2012

Alternative Contacts:

lesteso, fjcarnicer

Preferred Contact Methods:

[✓] e-mail [ ] instant messaging
[✓] issue comments [ ] telephone

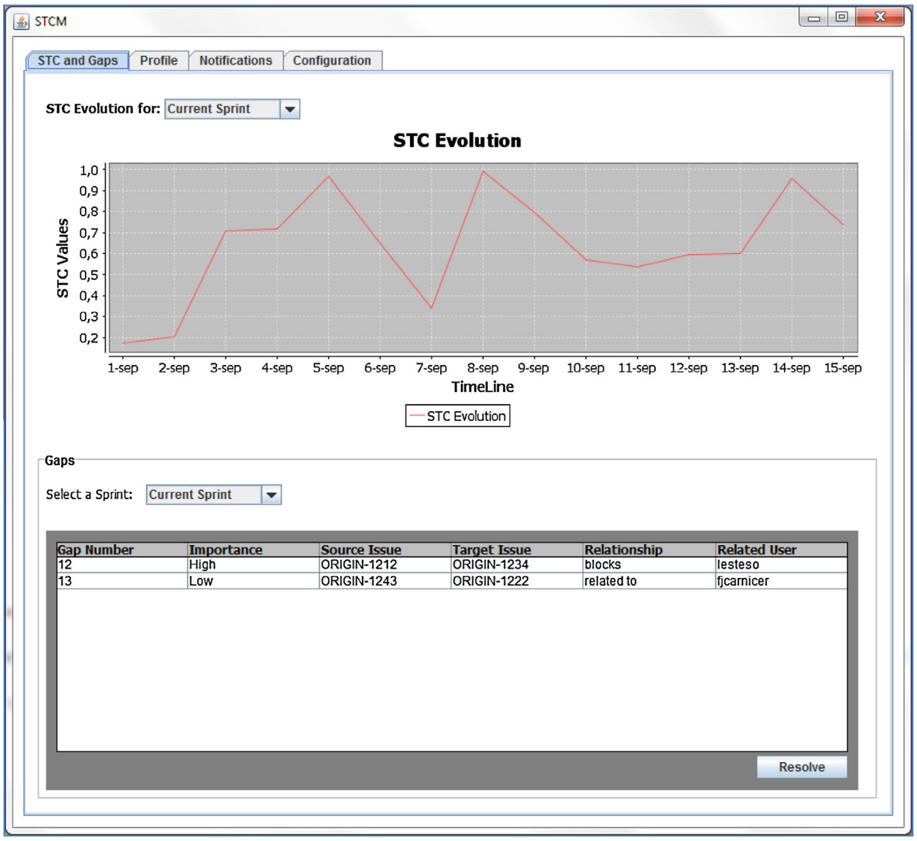
Fig. 14. User profile interface.

Moreover, as is shown in the STC Evolution graph in Fig. 18, the user's STC level started to grow and, given that only one gap had been detected, there was coordination to resolve these two issues.

Another interviewee pointed out that the tool may help to avoid work overlap since "by knowing what relationships exist between your issue and others, you can establish contact with other team members to organize the work between issues that are closely related". This comment can be checked against the data registered on the detected gaps, as is shown in Fig. 19.

Fig. 19 shows the *Gaps screen*, where there is a gap between the user and user *mjhernandez*. This means that these users should be coordinated and, even more importantly, that they should be closely coordinated, because their issues are linked by a "duplicates" relationship. This means that there are two issues that specify that the same or a very similar functionality should be implemented.

This last real example is a clear depiction of a situation in which this tool might be useful for improving coordination. More specifically, this improvement consists of users being notified quickly about this coordination need. Without this, this



Gap Number	Importance	Source Issue	Target Issue	Relationship	Related User
12	High	ORIGIN-1212	ORIGIN-1234	blocks	lesteso
13	Low	ORIGIN-1243	ORIGIN-1222	related to	fjcarnicer

Fig. 15. STCM main window.

relationship might be set in Jira, but if users are not aware of it then the coordination need might not be fulfilled or it might be detected late on. In this latter case, the sprint may be delayed, since people would waste time developing modules that had already been implemented.

One of the design team members told us that, from the point of view of the project manager, this tool might help reduce the amount of work needed to coordinate team members. In this case study, the design team is in charge of communicating issue relationships, particularly when these issues are blocker issues (very important) or when the due date to finish the issues is close. In this respect, this person found that when she was going to communicate information about issue relationships, some people already knew this information, since they had seen it in the tool. This confirms that the tool was being used correctly and, moreover, that it was useful for the participants.

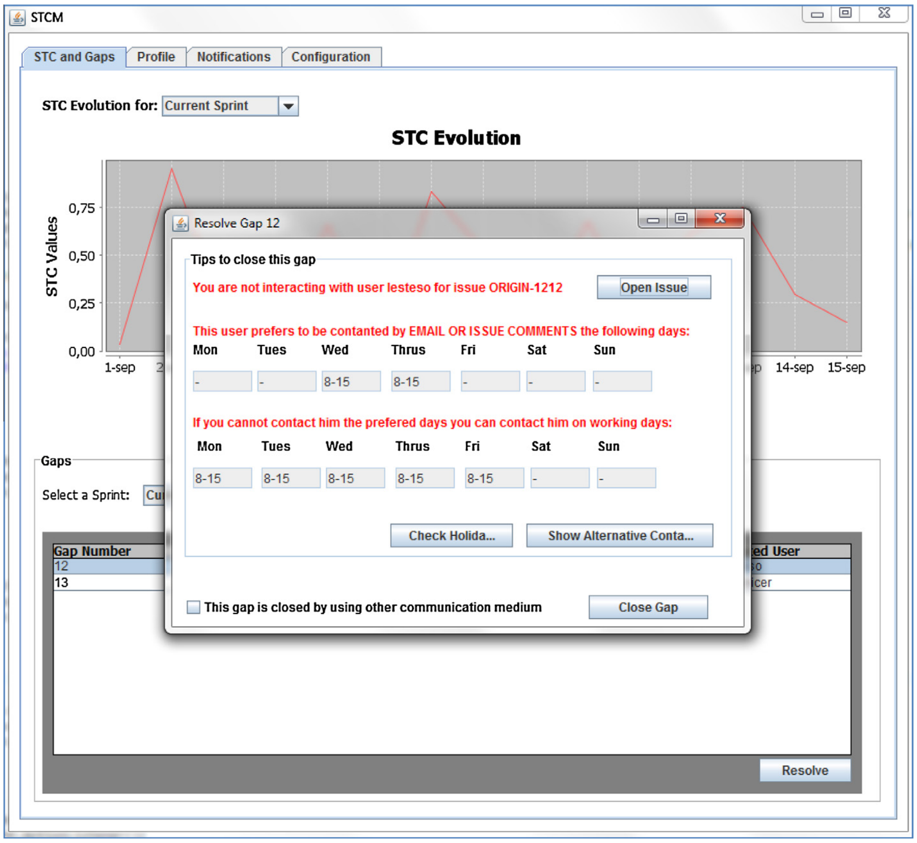
Some interviewees identified that the STC evolution graph may help to facilitate the detection of important coordination breakdowns, or low levels of coordination. They said that, although by looking at the graph it is not possible to gather why the STC is high or low, it is easy to check coordination breakdowns or low levels of coordination quickly.

Another interesting situation in which coordination was improved by our tool was detected during the interviews. The data shown in Fig. 20, which were extracted from one of our interviewees, show a low level of coordination during the first sprint. However, during the second sprint, we can see how this data changed drastically and that the STC level was high. We asked the user why this change had taken place, and he replied that obtaining a low STC during the first sprint encouraged him to improve coordination so as not to obtain a low coordination level and be considered a "bad" worker.

The user also explained that he had improved his STC level by looking at the tool's recommendations and seeing that the other users considered issue comments to be one of the preferred ways of maintaining coordination (information offered by our tool). He then started to coordinate with the other users by using issue comments. We can therefore state that the tool might be useful in encouraging improvement in coordination.

One of the comments repeated by several interviewees was that periodic gap notifications were considered to be a useful means of preventing important coordination breakdowns.

In terms of coordination, these are the most relevant user comments/situations that demonstrate how our tool (architecture) is able to improve some of the ORIGIN team members' coordination activities.



Tips to close this gap

You are not interacting with user lesteso for issue ORIGIN-1212

This user prefers to be contanted by EMAIL OR ISSUE COMMENTS the following days:

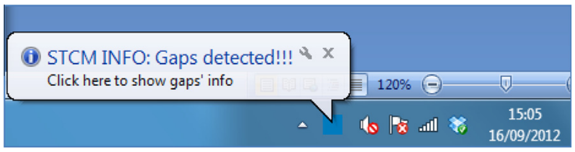
Mon	Tues	Wed	Thrus	Fri	Sat	Sun
-	-	8-15	8-15	-	-	-

If you cannot contact him the prefered days you can contact him on working days:

Mon	Tues	Wed	Thrus	Fri	Sat	Sun
8-15	8-15	8-15	8-15	8-15	-	-

This gap is closed by using other communication medium

Fig. 16. STCM resolve gap screen.



STCM INFO: Gaps detected!!!
Click here to show gaps' info

Fig. 17. STCM notifications in the Tray System Bar.

6.9. Threats to validity

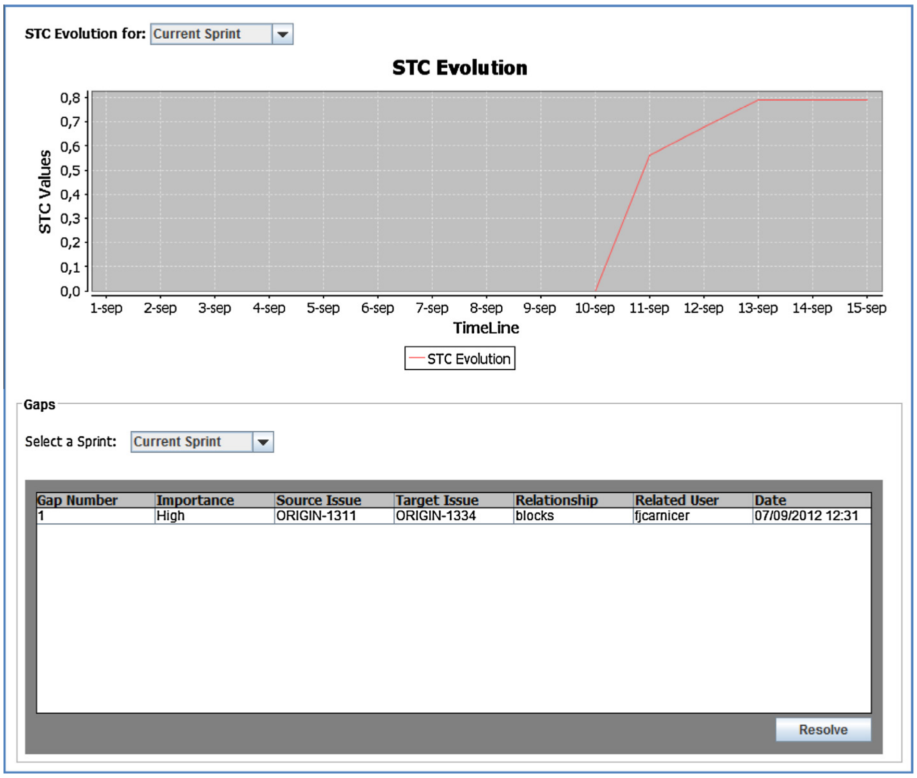
This section provides a discussion of the validity of the results presented. The case study has been performed in a distributed environment but not in a global environment, signifying that possible communication and coordination problems derived from cultural or language differences are limited, or simply do not exist.

We can state that coordination may be low, but, we can also affirm that if the participants have detected some communication and coordination improvements derived by the use of the tool in a scenario in which problems are few in number, we can expect that in a global environment with a higher level of communication and coordination problems, the tool, and therefore the architecture, would be more useful.

Our adapted STC measurement considers, among other things, Socio-Cultural Difference, Temporal Distance and Geographical distance to adapt them to GSE. In this case study, however, given that all the team members work in the same country, only Geographical Distance has had an influence on the results obtained.

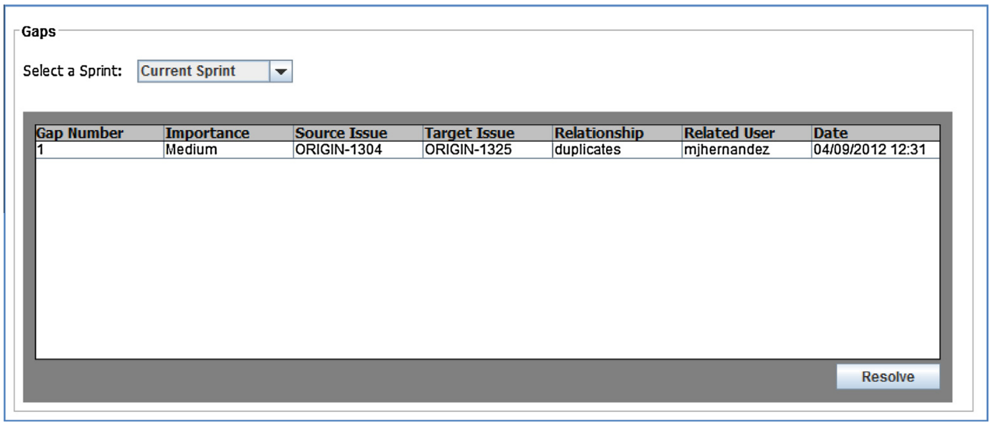
A further limitation was that the tool was tested for only two Sprints and with only 16 participants. This also limits possible coordination problems, since the number of issues is small in comparison to larger distributed teams, and two sprints are not enough to test the architecture in all possible situations. However, these two sprints were sufficient to obtain enough situations to show the improvements made by our proposal.

In terms of the data extracted, although the results are based on two of the most important communication channels (emails and issue comments), important communication interactions which were performed using other communication



Gap Number	Importance	Source Issue	Target Issue	Relationship	Related User	Date
1	High	ORIGIN-1311	ORIGIN-1334	blocks	fjcarnicer	07/09/2012 12:31

Fig. 18. Detecting important coordination gaps.



Gap Number	Importance	Source Issue	Target Issue	Relationship	Related User	Date
1	Medium	ORIGIN-1304	ORIGIN-1325	duplicates	mjhernandez	04/09/2012 12:31

Fig. 19. Detecting gaps that avoid work overlap.

channels, such as the telephone, were lost during the sprints. This also reduces the reliability of the gap analysis, since agents can consider gaps to be situations in which interactions are performed by a channel that is not controlled.

Our proposal is also limited with regard to how the data obtained has been processed, since agents do not analyze the content or semantics of interactions. Thus, if an interaction is performed by email, agents cannot associate this interaction with a specific task, since the email content is not analyzed.

6.10. Conclusions

This section presents a summary of the conclusions reached from the case study, considering the results obtained and the threats to their validity.

The case study addresses two main research questions:

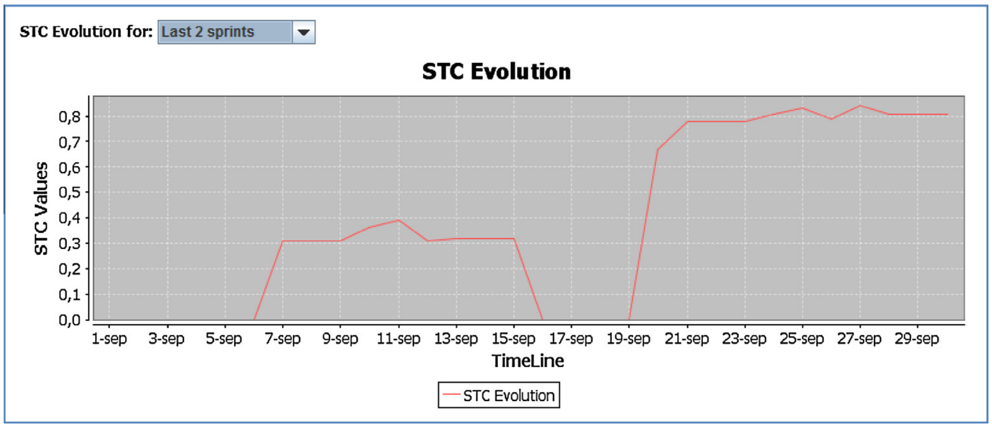


Fig. 20. STC level improvement owing to tool features.

- **RQ1:** Is it feasible to construct a system based on our architecture that will help distributed team members to maintain communication and coordination?
- **RQ2:** Can the tool constructed make use of the architecture modules to manage data automatically without user intervention?

RQ1 and RQ2 have been answered by constructing a tool called STCM, which includes all the features needed to extract automatically the interactions, tasks, relationships between tasks, etc., so as to be able to calculate STC and find communication and coordination gaps, with a consequent improvement in coordination and communication.

The ways in which this tool can improve coordination and communication have also been outlined in the results section. These results allow us to state that the architecture/tool can be useful in distributed environments, such as those of global software projects in which tasks are very dependent on communication between team members; this communication is almost always performed using software tools.

The analysis performed by the agents included in the tool depends entirely on how many data sources can be connected to our tool (architecture). We can therefore state that this tool (architecture) depends on the capability of the tools that already exist in the project in terms of extensibility. This is because it is necessary to connect our tool to tools that already exist to extract data concerning tasks, interactions, etc. In our case, we have seen that Jira is a tool whose extensibility options permit it to be integrated with almost any kind of tool through the use of its web service and plugin modules.

Prior to applying our tool, it is highly advisable to carry out a study on the means of communication and coordination used by the team/s, since this will assess whether it is possible to gather sufficient data and determine whether or not our tool/architecture might be useful; for example, if it is an environment in which only interactions performed by telephone can be extracted, it may not be worth deploying our architecture.

The analysis of user comments about the use of the tool has allowed us to draw various conclusions. For instance, the provision of the communication availability of other team members was considered to be one of the most useful features. The use of STC graphs to summarize the coordination and communication level was seen by project leaders as a simple way to monitor complex data and information.

With this case study we consider that we are validating the most important aspects of our proposal.

7. Limitations and discussion

In this paper we have presented an Agent Architecture with which to manage Socio Technical Congruence in a Global Software Project. However, it is important to discuss some of the limitations and the assumptions taken.

One of the main limitations is that we are assuming that a high percentage of the interactions that take place when people are working are work-related interactions, helping them to coordinate their work. In addition to that limitation, we have to recognize that we do not extract the content of interactions, for instance the text of an email, to analyze whether the interaction is work-related or not.

An important assumption is that agents will have all the information needed to create the coordination dependencies, the actual coordination and the weights to make a gap ranking. In the text we explain that part of this information could be taken from Project Management tools, or directly introduced manually by, for instance, the Project Manager. In particular environments it might be difficult to obtain all the interactions needed to calculate STC (for instance telephone interactions). We do believe, however, that although it is sometimes impossible to extract all interactions, we may be able to obtain enough interactions (from repositories, forums, emails, instant messaging tools, etc) to apply this technique.

Another limitation of this study is that when comparing our proposal with previously implemented tools related to STC the set of tools might not be complete because, although great effort to extract the available tools has been made, the search method was not systematic. We make a similar observation about the methods for measuring STC that have been used.

8. Conclusions and future work

In this paper, measuring Socio-Technical Congruence (STC) has been presented as a technique with which to measure coordination, by checking the alignment between the coordination needs in a project and the actual coordination performed by team members. We have also seen how communication interactions can be analyzed through the use of STC.

Measuring STC has consequently been presented as a suitable technique with which it is possible to improve coordination and communication. These are precisely two of the aspects that present most difficulties in a GSE environment, due to the problems derived from distance (i.e. different working times, different understandings or difficulties in communicating efficiently).

This being so, we believe that using this technique in a GSE environment, where people are distributed and communication is usually performed by the use of technology, would be advisable and beneficial.

It was with this goal in mind that we studied a set of measurements which are currently used to measure STC, along with a set of tools that make use of STC. We set out to identify their features and check which ones might be the best ones for GSE and its special characteristics.

From the set of STC measurement studied we can say that congruence analyses have almost always been applied to the Coding phase of the project; consequently, the technical entity is the source code file which is used most. Moreover, the most commonly-used dimension for calculating Actual Coordination is communication interaction (for instance, through distribution lists, comments, IRC communication). We can also say that there are two approaches to calculating STC; one that uses matrices to represent coordination needs and technical dependencies, and another that employs Social Networks.

On the other hand, if these measurements are going to be used in real-time, there are data, mainly related to the weights that may be difficult to obtain. For instance, in the Kwan et al. proposal, they use the expertise of team members to weight the matrices. In a real-time analysis of congruence this information might not be available or, where it is available, it may not be objective.

We have also concluded that the most appropriate measurement for use in a Global Environment is that of Kwan et al., because it is related to major aspects of GSE, such as geographic and temporal distance.

By studying the current STC tools, we concluded that no single tool is designed in its entirety for GSE environments and we believe all tools need to be adapted. We have thus come up with a set of improvements that might be useful in a GSE environment when managing STC, since the current versions of the tools studied are not adapted well enough for GSE.

From the result of this study we proposed an Agent Architecture that combines agents' properties, such as autonomy, proactivity or reactivity and Kwan's measurement for managing STC in a Global Software Project. The proposal sets out to improve current STC tools in some aspects such as the monitoring of STC levels, by performing actions that would maintain a good level of STC, with ensuing improvement in communication and coordination in GSE. Our belief is that this approach could help to alleviate some of the problems derived from distance in a GSE environment.

Moreover, use of the architecture in our case study leads us to conclude that agent properties could be very useful in managing STC, as well as in improving the current tools, by endowing the tools with new features that would also help to improve productivity and quality.

Concretely, the performed case study showed us that our proposal may improve coordination and communication by: (1) improving coordination awareness, (2) avoiding overlapping and repeated work, (3) detecting lack of coordination, (4) offering useful contact information or (5) making easier communication for newcomers.

Acknowledgements

This work has been funded by the PEGASO/MAGO Project (Ministerio de Ciencia e Innovación MICINN, TIN2009-13718-C02-01) and co-funded by FEDER. It is also supported by ENGLOBAS (PII2I09-0147-8235) and GLOBALIA (PEII11-0291-5274), funded by Consejería de Educación y Ciencia (Junta de Comunidades de Castilla-La Mancha) and co-funded by FEDER (Spain) and ORIGIN (IDI-2010043 (1-5)) funded by CDTI and FEDER. This research is also supported, in part, by the Science Foundation Ireland Grant 10/CE/I1855. Finally, we would also to thank the anonymous reviewers for their useful and constructive comments.

References

[1] A. Abecker, V. Dignum, L.V. Elst, Agent-mediated knowledge management, Lecture Notes in Computer Science, vol. 2926, Springer, 2003.
[2] B. Al-Ani, E. Trainer, R. Ripley, A. Sarma, A.v.d. Hoek, D. Redmiles, Continuous coordination within the context of cooperative and human aspects of software engineering, in: Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering, ACM, Leipzig, Germany, 2008, pp. 1–4.
[3] S. Balasubramanian, R.W. Brennan, D.H. Norrie, An architecture for metamorphic control of holonic manufacturing systems, Comput. Ind. 46 (1) (2001) 13–31.
[4] F.P. Brooks, The Mythical Man-Month: Essays on Software Engineering, 20th Aniversary ed., Addison-Wesley, Reading, MA, 1995.

[5] B. Bruegge, A.H. Dutoit, Communication metrics for software development, in: Proceedings of the 19th International Conference on Software Engineering, ACM, Boston, Massachusetts, United States, 1997, pp. 271–281.
[6] E. Carmel, Global Software Teams: Collaborating Across Borders and Time Zones, Prentice Hall PTR, 1999. p. 269.
[7] M. Cataldo, J.D. Herbsleb, K.M. Carley, Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity, in: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, Kaiserslautern, Germany, 2008, pp. 2–11.
[8] M. Cataldo, P.A. Wagstrom, J.D. Herbsleb, K.M. Carley, Identification of coordination requirements: implications for the Design of collaboration and awareness tools, in: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, ACM, Banff, Alberta, Canada, 2006, pp. 353–362.
[9] M. Conway, How do committees invent?, Datamation (1968)
[10] C. Cramton, The mutual knowledge problem and its consequences for dispersed collaboration, Organ. Sci. 12 (3) (2001) 346–371.
[11] C.R.B. de Souza, D.F. Redmiles, The awareness network, to whom should i display my actions? And, whose actions should i monitor?, IEEE Trans Software Eng. 37 (3) (2011) 325–340.
[12] C. Ebert, Global software engineering: distributed development, outsourcing, and supplier management, in: IEEE Computer Society Books, Wiley, Los Alamitos, USA, 2010.
[13] K. Ehrlich, M. Helander, M. Valetto, G. Davies, C. Williams, An analysis of congruence gaps and their effect on distributed software development, in: Socio-Technical Congruence Workshop at ICSE conference, Leipzig, Germany, 2008.
[14] E. Gilbert, K. Karahalios, CodeSaw: a social visualization of distributed software development, in: Proceedings of the 11th IFIP TC 13 International Conference on Human-Computer Interaction – Volume Part II, Springer-Verlag, Rio de Janeiro, Brazil, 2007, pp. 303–316.
[15] B. Gokpinar, W.J. Hopp, S.M.R. Iravani, The impact of misalignment of organizational structure and product architecture on quality in complex product development, Manage. Sci. 56 (3) (2010) 468–484.
[16] J.D Herbsleb, A. Mockus, T.A. Finholt, R.E. Grinter, Distance, dependencies and delay in a global collaboration, in: ACM conference on Computer supported Cooperative Work, ACM, New York, NY, USA, 2000.
[17] P. Hinds, C. McGrath, Structures that work: social structure, work structure and coordination ease in geographically distributed teams, in: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, ACM, Banff, Alberta, Canada, 2006, pp. 343–352.
[18] A.v.d. Hoek, D. Redmiles, P. Dourish, A. Sarma, R.S. Filho, R.d. Souza, Continuous coordination: a new paradigm for collaborative software engineering tools, IEE Semin. Digests 2004 (902) (2004) 29–36.
[19] G. Hofstede, G. Hofstede, G. Hofstede, Cultures and Organizations: Software of the Mind, McGraw-Hill, 2004.
[20] J. Jensen, Using an Intelligent Agent to Enhance Search Engine Performance, 2002 June. <http://www.firstmonday.dk/issues/issue2_3/jansen/index.html>.
[21] A.A. Keshlaf, S. Riddle, Risk management for web and distributed software development projects, in: Fifth International Conference on Internet Monitoring and Protection, Barcelona, Spain, 2010, pp. 22–28.
[22] I. Kwan, A. Schr, D. Damian, Does socio-technical congruence have an effect on software build success? A study of coordination in a software project, IEEE Comput. Soc. (2011) 1–20.
[23] I. Kwan, A. Schr, D. Damian, A Weighted congruence measure, in: Workshop on SocioTechnical Congruence, 2009, pp. 1–4.
[24] A. Powell, G. Piccoli, B. Ives, Virtual teams: a review of current literature and directions for future research, SIGMIS Database 35 (1) (2004) 6–36.
[25] A. Sarma, J. Herbsleb, A.v.d. Hoek, Challenges in Measuring, Understanding, and Achieving Social-Technical Congruence, 2008.
[26] A. Sarma, L. Maccherone, P. Wagstrom, J. Herbsleb, Tesseract: Interactive visual exploration of socio-technical relationships in software development, in: Proceedings of the 31st International Conference on Software Engineering, IEEE Computer, Society, 2009, pp. 23–33.
[27] B.F. Sebastiao, R.B.d.S. Cleidson, F.R. David, Exploring the Relationship between dependencies and coordination to support global software development projects, in: ICGSE '06. International Conference on Global Software Engineering, 2006, 2006.
[28] H. Spanjers, M.t. Huurneç, B. Graaf, M. Lormans, D. Bendas, R. van, Tool support for distributed software engineering, in: International Conference on Global Software Engineering (ICGSE'06), Florianopolis, Brazil, 2006, pp. 187–198.
[29] E. Trainer, S. Quirk, C.D. Souza, D. Redmiles, Bridging the gap between technical and social dependencies with Ariadne, in: Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology eXchange Eclipse 05, 2055, pp. 26–30.
[30] G. Valetto, S. Chulani, C. Williams, Balancing the value and risk of socio-technical congruence, in: Proc. Socio-Technical Congruence Workshop at ICSE, 2008.
[31] G. Valetto, S. Chulani, C. Williams, Balancing the value and risk of socio-technical congruence, in: Proceedings of the ICSE Workshop on Socio-Technical Congruence (STC 2008), Leipzig, Germany, 2008.
[32] G. Valetto, M. Helander, K. Ehrlich, S. Chulani, M. Wegman, C. Williams, Using Software repositories to investigate socio-technical congruence in development projects, in: Proceedings of the 29th International Conference on Software Engineering Workshops, IEEE Computer, Society, 2007, pp. 25.
[33] G. Valetto, M. Helander, K. Ehrlich, S. Chulani, M. Wegman, C. Williams, Using software repositories to investigate socio-technical congruence in development projects, in: Proceedings of the Fourth International Workshop on Mining Software Repositories, IEEE Computer, Society, 2007, pp. 25.
[34] M.J. Wooldridge, N.R. Jennings, Intelligent agents: theory and practice, in: Knowledge Engineering Review, vol. 10(2), Cambridge Univ Press, 1995.
[35] G. Zacharia, A. Moukas, P. Maes, Collaborative reputation mechanisms for electronic marketplaces, Decis. Support Syst. 29 (4) (2000) 371–388.