Component of Communication

Duarte Gouveia[1], David Aveiro[1]

[1] University of Madeira & Madeira Interactive Technologies Institute,
Caminho da Penteada 9020-105 Funchal, Portugal
duarte.gouveia@m-iti.org, daveiro@uma.pt

Abstract. This work aims at contributing to the design of an Information System (IS) that adequately support human interactions in organizations in an message-based asynchronous network. We move forward based on previous work [1], presented at the CIAO! 2015 Doctoral Consortium, where two protocols were given for DEMO Engines: PSI and Tell&Agree. In this work, we propose a unification of these two protocols based on the development of a Core Component of Communication. This component can be used to create state machines for complex communication ensembles, like PSI and Tell&Agree (which are used in its validation), but is generic enough to a broader range of application. The Core Component of Communication establishes that each interaction between parts starts with an Initial State, and can successfully end, either in a Quit State or one of the available Answer State(s). Although the Core Component is unique, it can have several possible configurations obtained by joining several Quit/Answer States in unique terminal states. This solution is a message-based protocol to be used in asynchronous networks, and its main advantages are increased flexibility and modeling power, and, we believe, it moves DEMO closer to Habermas "ideal speech situation".

Keywords: organizational engineering; agreements; state machine.

1 Introduction

This work aims at contributing to the design of an Information System (IS) that adequately support human interactions in organizations, supporting a business logic, in an message-based asynchronous network, like the Internet.

Searl established [2] that the minimal unit of human communication is not a sentence or other expression, but a certain kind of language acts, called speech acts, such as state, assert, describe, warn, remark, comment, command, order, request, criticize, apologize, censure, approve, welcome, promise, express approval, and express regret.

Flores [3] defined communication as an exchange of speech acts, assuming shared interest, and aiming at fulfillment, regardless of the conditions used to reach fulfillment.

Habermas criticized Searl's taxonomy of speech acts for not distinguishing perlocutionary acts form illocutionary acts. Perlocutionary acts are strategic actions that may be misleading to counterparts, for example, based in the claim of power. Illocutionary acts are based on the claims to truth, justice and sincerity.

Habermas established the Theory of Communicative Action, as he believes it is possible to reach mutual understanding and agreements using the desirable scenario of "ideal speech situation" [4], following the illuministic tradition. In such scenario:

1) Every part has an equal opportunity to speak at anytime, proposing solutions and expressing opinions, desires and needs.

2) Every part is free to support or question any assertion performed by others.

3) No one can be coerced into accepting the counterpart statements (1) and 2)).

4) Every part acts with the desire to reach a agreement.

These requirements can be regarded as ontological and foundational to design IS that support human interactions in organizations. They lead to more strict design constraints and what conditions are required to successfully support rational coordination of action [5].

On previous work [1], discussed at the CIAO! 2015 Doctoral Consortium, we presented two protocols for DEMO Engines: PSI and Tell&Agree with the purpose to provide state machines that helped the implementation of Information Systems (IS) that fully comply with DEMO theory [6]. This was an important contribution to the research field because existing implementations did not fully comply with theory [1].

The state machine for implementing DEMO/PSI in IS [1] requires shared memory to keep the state between participants. This constraint is determined by the existence of a single state-machine to keep the state of the DEMO/PSI transaction for both participants, as discussed in [1]. Having copies of the state-machine in the IS of each participant is not feasible, as incoherent states would occur by the almost coordination acts performed at short intervals of time by different participants.

Using a different approach, the state machine for Tell&Agree presented in [1], was built with the goals of complying with "ideal speech situation" and of operating with a message-based approach in asynchronous networks, which required the introduction of acknowledge acts. A message-based approach is used, where each participant keeps it's own state-machine in it's IS environment and only when coordination acts – messages - are acknowledged by the counterpart, they are considered valid. Tell&Agree introduced flexibility in the roles performed by participants and also the number of participants in agreements. But we realize that Tell&Agree introduced "a radical change regarding the current DEMO transactions" [1]. In this work we aim at closing the gap between these two protocols and reaching a more consensual solution, keeping the benefits of Tell&Agree, but better mapping it to existing DEMO/PSI transactions.

This paper is organized as follows: Section 2 states the problems to be addressed in this work; Section 3 frames the work done in research methodology applicable to this work; Section 4 presents the authors contribution to solve addressed problems; Section 5 analyses the findings and evaluates them according to the method described in section 3; Section 6 concludes summarizing the major contributions.

2 Research Problem

Based on the future work section of [1], we list a set of problems that are not yet properly solved by current theory.

- According to Demo Theory [6], requests are coordination acts that include all the possible details for the transaction. This is an unrealistic constrain, as the initiator is certainly not aware of the full context for the transaction, namely the technological, economical of societal context from the executor. The initiator might not know what is the current best solution for its need as he is not specialized in that area. We therefore discuss the requisite of fully aware **requests** in DEMO theory [6].
- The importance of allowing the **power of initiative** of all participants to start and move forward in a dialog towards an agreement at anytime, as expressed in "ideal speech situation" by Habermas [4]. Current DEMO theory establishes that only the initiator can perform a request. Having the executor as proponent of a solution that the initiator accepts is not a valid solution with current DEMO/PSI protocol.
- Tell&Agree [1] introduced the possibility of agreements with more than two participants. We address this issue and their implications regarding new entrants and leavers during the life cycle of the transaction.
- We revisit the **acknowledge** acts issue [1] and their ontological importance for the message based approach.
- We address several acts beside **production** and **coordination** that we believe are relevant [1][7] to transactions like **informative** acts, **meaningful** acts, **question** acts, **material** acts, **delegation** acts, **advice** acts, **identification** acts. Some of these acts, like delegation, have extensively been discussed in existing literature [1].

Our research question is: Can we build upon existing knowledge to improve the way we model social world transactions addressing the stated problems?

3 Research Design

In a broad sense, an empirical study [8] is an "act or operation for the purpose of discovering something unknown or of testing a hypothesis". It is still challenging to acquire new knowledge and test hypothesis through empirical methods in the research fields of software engineering and information systems [9].

In order to make research design options explicit we adhere to the extension of this pattern presented in [9] and depicted in Figure 1. This decision-making structure encompasses three phases: 1) Strategic Phase – deciding on research outcome, logic, purpose and approach; 2) Tactical Phase – deciding on research purpose and chosen method(s); 3) Operational Phase – deciding on data collection and analysis. The gray checked options, in Figure 1, correspond to the specific chosen path for this work. The unchecked options were considered and dismissed for the current work.

We start with the research questions from the problem statement, already presented in section 2, and end with the research findings that shall be presented in section 5.

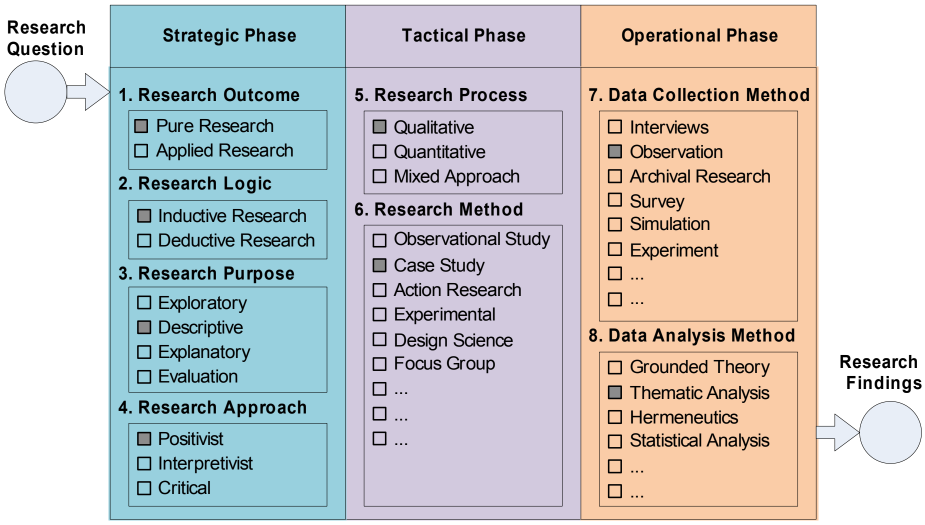

Figure 1: Research path using decision-making structure (adapted from [9])

On the ***Strategic Phase*** we aimed for a **1) Pure Research Outcome**. The emphasis of Pure research (basic research) is the understanding of a general problem and generating new knowledge, rather than providing a solution for a specific problem like in applied research [9]. We choose **2) Inductive Logic**, that moves from the more specific to the more general (bottom-up). It starts from observed data or phenomena and infers theoretical concepts, patterns or theories [9]. The alternative logic uses theory to generate hypotheses and test them with the collected data. We opted for a **3) Descriptive Purpose** which aims at describe a phenomenon or characteristics of a problem. [9] We prefered a **4) Positivist Approach** which assumes there is a objective truth and reliable paths that different researchers can follow to reach similar conclusions. [9]

On the ***Tactical Phase***, we choose a **5) Qualitative Research Process**. Quantitative research processes can be difficult to use in a reliable and useful way for this kind of research. Qualitative measures are more appropriate for accessing the features and properties of the devised theory. We selected two **Case Studies** as **6) Research Method**. In the validation section we apply the devised constructors and rules to modeling DEMO/PSI protocol and Tell&Agree protocol, both presented in [1].

On the ***Operational Phase***, we used **Observations** as **7) Data-Collection Method**. We observed the application of the proposed solution to the two case studies in order to get the information to perform analysis. We preferred **Thematic Analysis** as **8) Data Analysis Method**. It is quite common to use thematic analysis in the development of IS as it deepens the knowledge and understanding obtained from data. [9]

4 The Core Component of Communication

4.1 The Black Box model

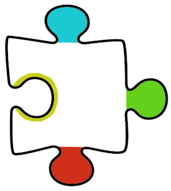

Figure 2: Puzzle piece as metaphor for black box state machine for Core Component of Communication

In order to fully present the Core Component of Communication it in a way easier to understand, we choose to first abstract it as a black box, graphically representing it using the metaphor of puzzle piece. The choice of this graphic notation is based on the idea that a puzzle piece is a building block, that can be joined to similar others, to build a much bigger logic set, like a business logic in an organization.

Internally each puzzle piece is a state machine, but from the outside only the initial and terminal states are shown, using the intrusion and extrusions of the puzzle piece with different colors. Each puzzle piece, i.e. each state machine, implements the "ideal speech situation" as proposed by Habermas [4] as will be shown in section 4.2.

In this metaphor the connections (intrusions and extrusions) are all compatible, which means that you can always connect an additional piece by joining the extrusion of current piece with the intrusion of another piece. The colors shown in the puzzle pieces are not constraints to the connections, but indications on the result reached in the internal state machine that exists in each puzzle piece.

Each piece has a single **initial state (I)**, represented in Figure 2 in the yellow intrusion. Regardless of the internal complexity, it is possible in due time to get to a definite agreement or a definite disagreement between parts through the exchange of messages and moving into terminal states. These possible terminal states are represented as extrusions, in the case presented in Figure 2 there are three possible terminal states, each with a distinctive color. Red represents the **quit state (Q)** – the disagreement resulting state when it was not possible to reach an agreement.

The other remaining states, collectively called **answer states** (for agreement), can be more than one, all distinct. In this sense we move away from DEMO/PSI protocol, where the only possible agreement is agreeing with what was requested. We follow Habermas "ideal speech situation" and allow that from discussion between parts, new agreements may emerge. However there should exist pre-existing default agreements solutions, established in configuration phase, that allow pre-building business logic, through combination of building blocks, based on those standard answers states. The exact mechanisms on how new possible answer states are created and agreed upon will be explained in the white box section of this metaphor, in section 4.2.

In Figure 2 there are two answer states: **Yes (Y)** represented in green; **No (N)** represented in blue. These answer options are kept along this document but they are just examples of possible answers.

This black box model has configuration options, namely:

- Is the **quit** state (red) required to exist?
- What are the initial answer states? (can be one or more)
- Which answer states get you back to the initial state?
- Do we allow new answer states to arise in the discussion?
- How many people may participate in this decision?

- What is the voting process to establish a decision? Options: unanimity, majority (50%+1), qualified majority, and many more complex voting options.
- Are new participants allowed to join this decision process after its creation?
- Are participants allowed to leave the discussion before a decision is reached?
- Is it allowed to change the person performing the coordinator role?
- Does the coordination also vote (even if tie braking vote) or just coordinates?

Many of these configuration options can only be fully discussed in the White Box model, in section 4.2. For now we will just focus on the first three questions.

The first important configuration option is if a **quit state** exists. In many negotiations one of available options is not to have an agreement – a quit state. Other times an answer state has to be reached in order to move forward in the business process. Both situations exist and the appropriate choice depends on the context.

The second and third configuration options are: what answer states exist for this "puzzle piece" (business transaction section), and which move you forward to a new terminal state (an extrusion in the puzzle piece) or move you back to the original initial state. Sometimes answers lead you back to the origination state for that "puzzle piece". For example, someone may request a cancellation to a established agreement. If the cancellation is allowed that leads to a new state. If the cancellation is refused that gets you back to the original state.

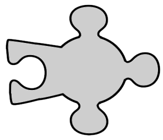

Figure 3: Graphical representation of an extension for allowing multiple attachments to current state

New answer states are reference positions where one or more pieces can be attached. Unlike normal puzzles pieces, where only one piece can be attached to the existing piece, in order to keep the existing metaphor we need to add an extension puzzle piece.

Extension puzzle pieces, represented in gray in Figure 3, are used exclusively to allow multiple attachment to an existing terminal state without breaking the chosen metaphor. They do not add any inner complexity, they just graphically allow multiple pieces to be attached to a existing state.

In section 5, we use an additional graphic syntax, by adding arrows between extrusions of different puzzle pieces to represent that they are actually the same state.

Based on the first three configuration options presented above, we can derive 5 typical patterns, presented in Figure 4. The 3 options presented on the first row have a quit state (red), while the ones on the second row do not. The patterns in the second row correspond to the ones in the same position on row one, but without the quit state.

When an answer state or a quit state get the process back to its originating position, then the corresponding color is represented next to the yellow ring in the initial state, as can be seen in Figure 4. You can see three example cases in the second and third column in Figure 4.

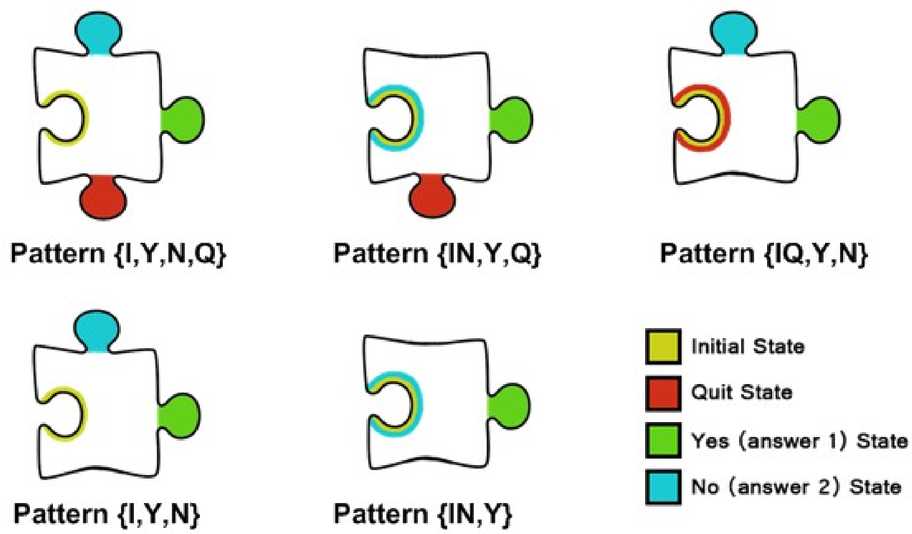

Figure 4: The 5 typical patterns obtain from alternative configurations

The way to identify each of the five patterns in a textual form is through the letters I, Q, Y, N. They correspond to the four states: Initial state, Quit state, Yes (answer 1), No (answer 2). If the quit state is absent then that letter is also absent. If a state gets back to the initial state position, then that letter is joined with the one that represents the initial state (I). The possible combinations are: {I,Y,N,Q}; {IN,Y,Q}; {IQ,Y,N}; {I,Y,N} and {IN,Y}.

Answering the configuration questions lead to distinct state-machines shown in Figure 4, but with a common interface to the exterior – the initial state and the terminal state. In this work we present these 5 typical patterns we were able to detect, but others may exist, either for different answers or for additional questions that were not yet identified. Therefore we do not claim for completude of these presented patterns. On the contrary, we embrace it as an incomplete solution that enables internal evolution while keeping a consistent interface between these building blocks.

A full business process can be constructed by combining these patterns in sequences depending on the desired options for the business process. As each of these puzzle pieces correspond to a specific state machine, combining the pieces results in a much more complex business process, that is succinctly described by the pieces as a black model. The connections between the pieces correspond to the shared states.

In order to promote reuse, the images of the five patterns above will be made available in http://www.duarte-gouveia.info/core_component/ .

4.2 The White Box model

4.2.1 Pattern {I,Y,N,Q}

It is now time to get into the inner complexity of Core Component and its common patterns. We will extensively describe the {I,Y,N,Q} pattern, including the challenges posed by the configuration questions presented in section 4.2, and then just present the differences in the remaining patterns comparing to this base pattern.

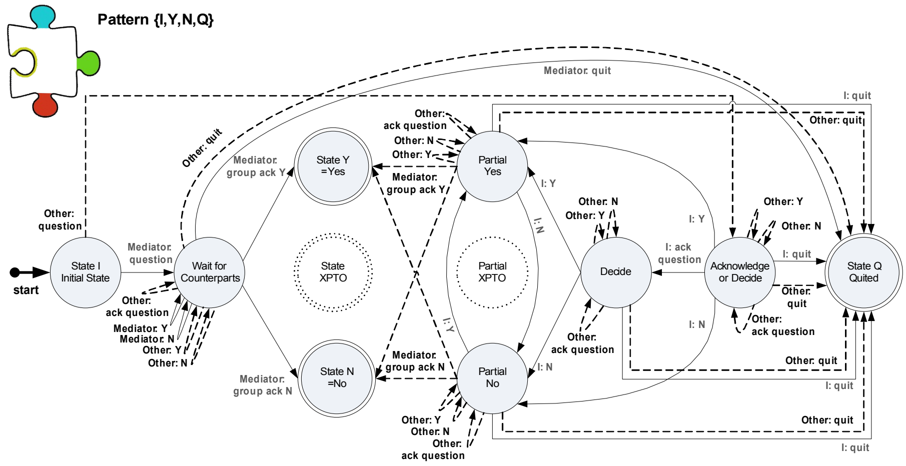

Figure 5: Pattern {I,Y,N,Q} of Core Component of Communication

To overcome space limitations bigger versions of Figure 5, 6 and 7 will be made available in http://www.duarte-gouveia.info/core_component/state_machines/ .

Two or more participants can be part in Core Component. For each one of them there will exist one state machine instance in the IS like the one presented in Figure 5. Solid lines are possible transitions a participant might take on it own state machine, but the state machine also keeps track of actions performed by other participants. Transitions performed by other participants are represented by dashed lines. States are represented as circles with solid lines. The initial state is market with a thick arrow and the terminal states are marked with double circles. States with dashed lines are latent new states that might emerge in the process.

The first participant to move from the initial state will take the coordination role. The coordination role has the responsibility of making the final transition to reach a decision state, provided that necessary and sufficient conditions are met. All the others participants will play a supporting role, stating their position and proposing new additional answers to be decided upon collectively.

Following Figure 5, from the initial state, the first mover will initiate the process by posing the question or motion to move forward. The first mover will take the coordinator role moving to “Wait for counterparts” state. All other participants will follow the dashed line to “Acknowledge or decide”.

Participants can acknowledge the question/motion at hand without taking a decision for now. By acknowledging they move from the “Acknowledge or Decide” state to the “Decide” state.

All participants are aware of others voted (and acknowledges) and keep an internal counting of which answer votes are being chosen by whom. That is represented by the endo-arrows that link each state to itself (both in participant and coordinator states). Participants can change their minds during the process without constraints. This allows to have a dynamic process where participant express their options even before a voting call is requested. This allows discussions to move forward faster as a “sense of the room” is known during discussion.

If the configuration determines that the coordinator is not able to vote, then those corresponding acts in the “Wait for counterparts” will be disabled.

Only when the necessary and sufficient voting conditions are met, the coordinator can perform the specific Group Acknowledge act to the admissible vote, reaching a terminal state. When that happens all participants move to that terminal state through the corresponding dashed line. The voting act by the coordinator (if possible) is a different transition that exist in “Wait for counterparts” state.

The necessary and sufficient conditions can be unanimous vote on one answer, majority (50%+1), qualified majority (for example 75%), or many other configurations. If the decision is made as part of a stable body in an organization, there might be constraints about the need of a certain percentage of acknowledgments – which correspond to “being in the room”, even without a vote – or other constraints regarding minimal time for discussion, voting periods or qualified votes, as participants might not have the same voting number or quality. However, notice that any voting option other than unanimous vote is moving away from the “ideal speech situation” as defined by Habermas. Acording to these situation, no participant can not be coerced to accept a solution he does not agree with.

New answer states (partial and terminal dyadic pair) can be added by participants (or coordinator) to allow alternative solutions that are able to generate consensus. Generating new rules might also generate new timing constraints to allow everyone to get to know the new existing options. The generation of new rules are represented in Figure 5 with the new states “Partial XPTO” and “State XPTO”. If a new solution is added this pair of states is added to the state machine and the transitions equivalent to the existing answer states, mutatis mutandis.

Also, existing solutions previously presented might be withdrawn, moving the participants that had chosen that option to move back to “Decide” state. This also allows multiple rounds in decision making by removing the least voted options – as exists in Australia's voting system.

An option this model does not directly support is the multiple vote, where each participant can vote simultaneously in more that one option. But it can be modeled with additional “de-vote” acts to allow multiple choice.

As an implementation detail it should be possible for participants to comment on each answer option, stating its merits or problems, in order to persuade participants on their choices. There should also be a general thread of comments, not attached to each particular answer but to the general transaction. All comments are just tell acts, as expressed in section 4.3, that do not produce state transitions in the transaction.

A complex situation might exist by allowing participants to enter or leave the decision during the process. For the new participant the situation is relatively simple: the new participant will be placed in the “Acknowledge or Decide” state and he can proceed from then on without any problem. Leaving a transaction is also relatively simple. Both entering and leaving should generate events notifications for other participants.

Typically the quit vote from a participant ends the transaction for all. A quit vote, if exists, has a semantic meaning that is different from the “leave the transaction” act.

There are also no relevant issues regarding voting procedures, as the rules can easily be applied in any circumstance, namely not allowing a vote if not enough participants are currently in the transition.

The difficult situation exists if a participant has already voted on some option and then the participants in the transaction change. The participant vote might be dependent on the context – the set of participants in the transaction. Agreeing with a particular solution with a particular set of participants does not guarantee that the member has the same position with a different set. Even so, as a general case, it doesn't seem to be justifiable, as a general solution, to move that participant back to the “Decide” state every time there is a change in the participants. This problems should be addressed as an implementation detail in the configuration options for each transaction/participant.

Finally, there is the question of changing the participant that is performing the coordination role. This is an important issue in the literature as absence of action from the coordinator, either voluntary or involuntary (due to a system failure) might lead to a blocked transaction. There is still no undisputed solution in the literature for this problem, therefore we shall leave it as an open question for the implementation. In general the problem is the consensus decision to change the coordinator and the process/criteria to choose the new coordinator.

The actual change of coordinator withing this pattern is easy to handle. The new chosen coordinator leaves its current state and is placed in the “Wait for counterparts” state. The former coordinator moves to “Acknowledge or Decide” state.

4.2.2 Pattern {I,Y,N}

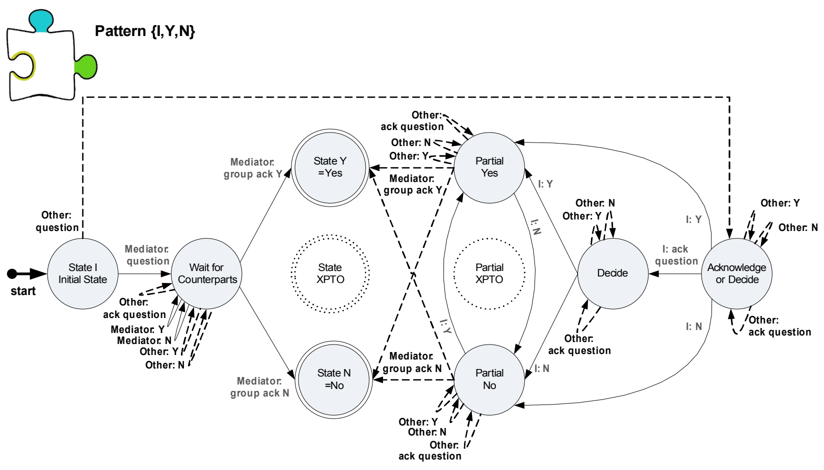

Figure 6: Pattern {I,Y,N} of Core Component of Communication

4.2.3 Pattern {IN,Y,Q}

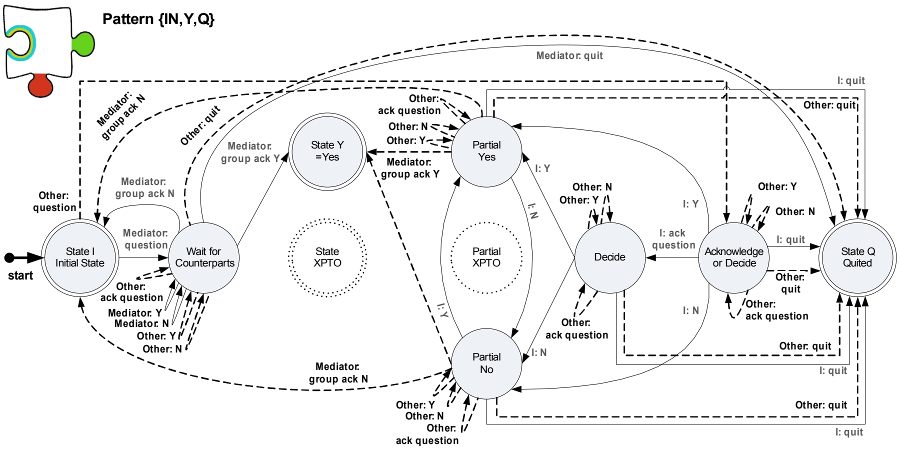

Figure 7: Pattern {IN,Y,Q} of Core Component of Communication

The pattern {IN,Y,Q}, presented if Figure 7, has several differences to the base pattern {I,Y,N,Q} presented in 4.2.1. By combining the terminal state that corresponds to No - answer 2 with the initial state, all transitions that lead to this state have been redirected to the initial state. Notice however that although each answer has two states: a partial and a terminal, only the terminal state was redirected. The partial keeps its functionality as described in the base pattern {I,Y,N,Q}, namely the ability each participant has to change its opinion before a final move to a terminal state happens.

4.2.4 Other Patterns

The remaining two patterns {IN,Y} and {IQ,Y,N} follow the same reasoning, and therefore, considering the space limitations, will not be shown.

4.3 The importance of Acknowledge in Tell, Ask/Answer and Propose/Agree

When two or more people communicate, there are typical patterns in order to transmit information (tell), ask for information and establish agreements, depicted in Figure 8.

In Figure 8 he counter-
part are in

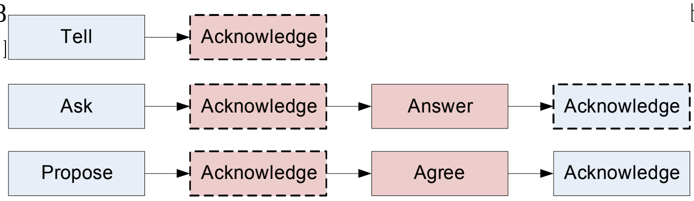

Figure 8: Typical pattern in comminication

In face to face communications people search for hints that acknowledge that the counterpart received the message. On IS, the acknowledge facility is usually not implement, which leads to uncertainty, as the initiator is not able to know if the counterpart has read and interpreted the message until a new message is received. Sometimes the new message is just to acknowledge without any new information, but it performs an ontological role in stating that the content was, not only received, but understood.

In a message to give information (tell), acknowledge is not usually mandatory, unless explicitly requested in the message. If acknowledge was requested and not received the initiator assumes that the message has not been read.

In a question (ask), a answer is expected, therefore the acknowledge is implicit when the answer arrives. Acknowledge in this case can be used as a social justification to a delayed response by the counterpart, but even so it is an ontological act. The answer typically doesn't require an acknowledge, unless explicitly requested in the message.

Contrary to previous cases, in an agreement, as depicted in Figure 1, although the first acknowledge is optional – just line in ask, the second one is mandatory, as both parts need to know that the agreement is established in order to be obliged to fulfill it.

In asynchronous digital communications over networks, we should facilitate the acknowledge acts to make the communication through messages more similar with face to face communications.

When you are working in a shared memory model, like in DEMO/PSI protocol [1], you can ignore that acknowledge acts, as both participants can check at all time what is the current state of the transaction.

We assume that communicative actions in social interactions aim to reach a consensus [6], by sharing information and committing to agreements. In order to reach an agreement we need negotiation where parts express thoughts, needs, concerns and constraints in order to check whether there is a common ground that is acceptable by all. Assuming that the initiator of the transaction is able to perform a full detailed request without establishing a dialog is a poor model. The initiator needs to know the adequate solution for him within the current existing constraints, that only the specialized executor can provide. Leaving that negotiation part out of the transaction model is losing a fundamental component of the business transaction.

In order to model a negotiation, we need to model messages being exchanged between parts. Many messages may be needed until an agreement is reached. Many of the preliminary messages/proposals may not be included in the final agreement, as they were not acceptable, or even if accepted, better solution may have risen during negotiations. But all those messages belong to the context that allows the full understanding of the details of the agreement, and therefore can't be thrown away.

In previous work a solution to this problem was presented in Tell&Agree protocol, by using a state chart [7] where each message had it's own state machine. Although this solution is feasible, we are now able to propose a simpler solution.

4.4 Aditional Acts

In this section we introduce the new kinds of acts, presented in section 2, and check how they can be implemented with the Core Component of Communication.

We are aware of the several speech act taxonomies, namely the ones proposed by Searl [2] and Habermas [4], with the Latin naming given by Dietz [4]. We believe that naming has a teleological purpose, and therefore should be used according to the intended audience. First of all there is no established standard for naming the different acts. Secondly, the names: assertives; directives; comissives; expressives; declaratives from Searl, or imperativa; constativa; regulativa and expressiva by Dietz, are too complex to normal users of IS in organizations. We choose simpler names, with more common language, with higher affordance with the intended use. Moreover we identify new kinds of acts that were not specifically addressed in that literature, and therefore did not have a specific name.

DEMO theory defines two kinds of acts: production and coordination. We believe there are other kinds of acts that are essential to organizational engineering. We shall succinctly present each of them answering the questions: Why this act is useful? How can it be implemented?

Informative Acts

Informative acts correspond to the tell pattern presented in section 4.3. Sometimes we want to transmit some information to counterparts. In other circumstances we want to transmit information's to ourselves in the future – to take notes for future reference. In that sense, not all informative acts are to transmitted to other participants.

In Informative acts, acknowledge is possible, but not mandatory. In current design options, informative acts can take advantage of the comment feature that was described in section 4.2 for possible answers and for the transaction.

Meaningful Acts

Meaningful acts is a special case of informative acts. They are used when a responsible and authorized participant in a transaction takes a decision that overrides a pre-established design options in the system or rules on undecided cases. This can be the case with generating or accepting new answer states in a pre-established business transaction for exceptional cases, or with no established processed defined. Meaningful acts use organization values as arguments for the choice taken.

They are similar to informative acts, but are usually visible only to internal members of the organization, that in due time might evaluate the choice taken which might lead to organizational change in business processes or clarify ruling for the future.

Meaningful acts can be implemented in the IS as a special kind of comment that is only visible to members of specific organization groups in the organization the participant belongs, and using organization values tags as additional information for that comment.

Question Acts

Also within the umbrella of informative acts, question acts follow the general pattern described in section 4.3 and are useful for gathering information from other participants in order to facilitate reaching a solution. We could consider a question act as two informative acts in sequence executed by counterparts. If the question requires a open answer then it can be implemented using the comments feature. But if closed answers is the choice, especially if a more significant question form is required with fixed questions and typified answers, then an additional voting option should be available for comments, either in a categorical single vote, categorical multiple vote or establishing a range when using answers that can be ordered.

Material Acts

A material acts exists when a participant performs an act in the real world which is relevant in a business transaction, but that material act was not registered in the IS by the person that performed the act. Another participant that has witnessed that act, or has evidence of that act being performed, may register that act impersonating other participant as a material act. This is not the same as delegation acts, because the person that received the delegation performed the act itself. Material acts are registers for acts that for some reason the person that performed the act was not able to perform.

The actual participant can deny material acts performed by others in his name, which is a serious social misconduct, unless it was just a mistake.

To implement material acts with Core Components, it should be possible to register material acts in the name of other participants. This should be an exception and not a normal procedure. Meaningful acts may be attached to material acts to justify its use.

It should be possible to allow several material acts registered by several participants for the same act. It would be the same has having several witnesses for a act, although this can be a challenge in terms of implementation of IS.

Delegation Acts

A delegation acts occurs when some participant is acting on behalf of other. There are two forms of delegation: transaction design delegation and transaction ad-hoc delegation.

Delegation by design happens when the assignment of roles to participants is done at design phase as a standard option for the implementation of business logic. In this case, it is assumed that all counterparts are aware of the terms the transactions will be performed, including the persons/organizations performing each role.

Ad-hoc delegation happens when a certain participant was expected to perform an act, but then it chooses to delegate the actual performance of that act to a third person. In that case, counterparts must be given the choice of accepting that delegation, as it may interfere with the expected terms for the service.

Ad-hoc delegation can be requested at any time and should be implemented with a {IN,Y} pattern, where the Y state actually also leads to the initial state.

Advice Acts

It is quite common in organizations to have participants in training mode or with assigned mentors. The idea is that actions being performed by the mentee can be either continually monitored by the mentor, or monitored by request of mentee. When that situation occurs, the mentor and mentee should be able to communicate with special informative acts in the comments section, but those special comments are only visible for mentor and mentee pairs, although they are placed within the proper context and timely ordered as all other comments.

The mentor could advice mentee on the way to proceed, but the actual act would be performed by the mentee, therefore we are still within the "ideal speech situation".

Activating and deactivating monitoring can be performed either by mentor or mentee, except if configured in a non standard way.

Identification Acts

Identification acts occur when a participant gives enough information to be identified as a known user in the IS. This shall allow the access to previous interactions with the organization, which allows to know previous configuration options, services required and other context information that might help the organization to provide a better service.

Identification acts can be requested at any time and should be implemented with a {IN,Y} pattern, where the Y state actually also leads to the initial state.

5 Validation with two case studies

In order to validate these Core Component of Communications we choose to model the Tell&Agree and the DEMO/PSI protocols with these building blocks. Both state machines were presented in [1] and are not included here due to space limitations.

5.1 Validating Tell&Agree

In Figure 9, we used two pieces to model Tell&Agree. The first one is the {IN,Y,Q} pattern and allows parts to reach an agreement or quit from the initial state. The second piece, to allow the cancellation of an agreement uses the {IN,Y} pattern, where there is no quit option, and not reaching an agreement on cancellation corresponds to moving back to the pre-established agreement position.

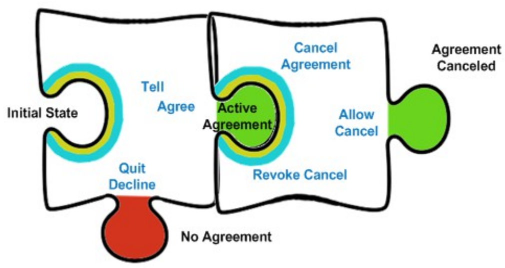

Figure 9: Tell&Agree [1] built with Core Components of Communication

The use of the Core Components is a perfect match for Tell&Agree protocol, complying with all its requirements and flexible and powerful features and with only 2 pieces.

The Core Component shows an additional benefit when compared with Tell&Agree [1] solution: it allows the messages exchanged to be naturaly enclosed within a context as new answers and comments within each piece, while in the Tell&Agree solution, that enclosure had to be provided as a add-on to group the several messages using a state chart [10].

5.2 Validating DEMO/PSI

In Figure 10 we can see the application of Core Components to DEMO/PSI, as specified in [1]. Other versions (older or newer) of DEMO/PSI might require adaptations,

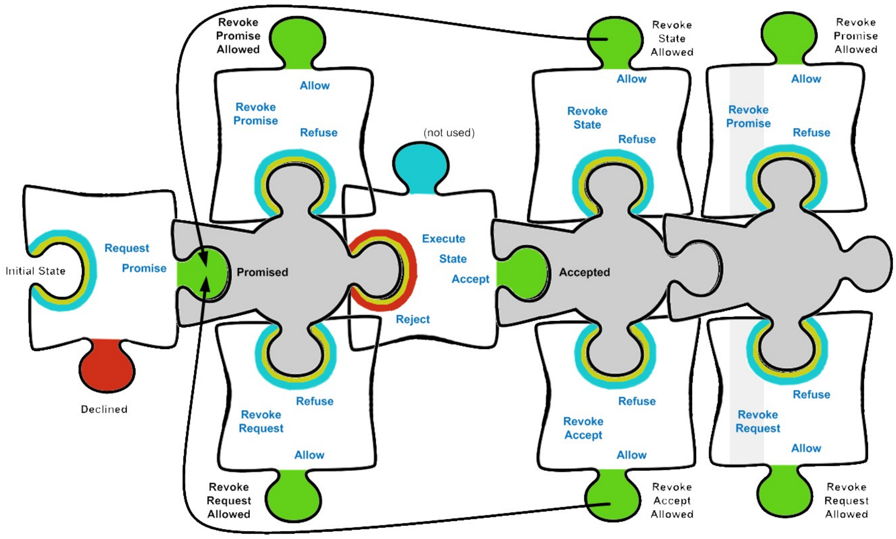

Figure 10: DEMO/PSI [1] built with Core Components of Communication

namely the patterns used to setup the desired solution.

In this case study we can see that many more pieces were required, namely the {IN,Y,Q} as the initial piece to perform the Request-Promise agreement, with the quit state corresponding to the terminal state reached through Decline. To model the State-Accept agreement we used the {IQ,Y,(N)} patterns, as the Reject act leads back to the original Promised state. The N option is not actually used and should be deactivated.

To implement all revoke requests we used the {IN,Y} pattern, as refusal of revoke requests keeps the state in the original position. Revoke Request and Revoke Promise at the Promised and Accepted states lead to terminal states. However, Revoke State and Revoke Accept lead to Promised state, as depicted by the arrows that mean that both states are the same.

This case study shows that it is possible to design DEMO/PSI with Core Components. This is specially interesting as DEMO/PSI is a memory shared protocol, and with this configuration it was transformed into a message-based solution with all the flexibility of Core Components – which are the same as Tell&Agree, namely: acknowledge acts; flexible power of initiative; ability to discuss agreements terms and not rely on rigid and imutable requests; allow agreements with more than two participants and the new kind of acts described in section 4.4.

Is there something lost with this new modeling? Yes, Core Components assume that the pairs Request-Promise and State-Accept are of dyadic nature. Only when a promise/accept exists a real change of state occurs, everything else before that terminal state is not reliable for enabling action rules or activating requests in sub-transactions. That is the “price to pay” for having flexible requests with a flexible discussion embedded. Another limitation is: it's no longer possible to call revoke promise and revoke accept from the “Stated” state, that is after the state has been performed. The State-Accept is dyadic and is an internal state that is not exposed as a terminal state.

These constraints previously stated assumes that the State-Accept is an agreement that is subject to discussion. If the business in question is such where a State could be performed without discussion, due to the nature of the provided produce, then we could replace the {IQ,Y,(N)} pattern by two {IN,Y} patterns in sequence, therefore exposing the Stated state.

6 Conclusion

This work shows that there is a building block we called Core Component of Communications that follows Habermas “ideal speech situation” and that can be customized into typical five patterns, although more specializations can be performed.

These five patterns allow to describe complex business processes (like Tell&Agree and DEMO/PSI) through the combination of those building blocks. We used the metaphor of puzzle pieces as compact graphical representation for each individual pattern.

Combining the Core Components with specific shared states, allows to directly derive the complete state machines for complex business processes, allow the implementation of IS with message-based approach to be used on asynchronous networks. This building block can be used to implement many complex systems. We will use it in future work with a construction of IS for more extensive validation.

This Core Component is focused on the problem of establishing agreements, and not particularly on individual speech acts. Nevertheless we presented several common communicative needs that organizations feel to coordinate work and describe business processes.

The benefits already identified for the Tell&Agree protocol [1], are now able to be used in DEMO/PSI, showing a positive answer to our research question: we were able to improve the way we model social world transactions solving the problems expressed in problem statement section.

7 References

[1] Gouveia, D., & Aveiro, D. Two Protocols for DEMO Engines: PSI or Tell&Agree.
[2] Searle, J. R. (1969). *Speech acts: An essay in the philosophy of language* (Vol. 626). Cambridge university press.
[3] Flores, F., Graves, M., Hartfield, B., & Winograd, T. (1988). Computer systems and the design of organizational interaction. *ACM Transactions on Information Systems (TOIS)*, *6*(2), 153-172.

[4] Habermas, Jurgen. "Discourse Ethics: Notes on a Program of Philosophical Justification." Moral Consciousness and Communicative Action. Trans. Christian Lenhart and Shierry Weber Nicholson. Cambridge: MIT Press, 1990, pp. 43-115.
[5] Dietz, J. L., & Widdershoven, G. A. M. (1991). Speech acts or communicative action?. In *Proceedings of the Second European Conference on Computer-Supported Cooperative Work ECSCW'91* (pp. 235-248). Springer Netherlands.
[6] Dietz, J.L.G. (2006) Enterprise Ontology – Theory and Methodology
[7] Gouveia, D. P. B. (2014). *Organizations redesign and building of information systems*.
[8] Basili, V. R., Shull, F., & Lanubile, F. (1999). Building knowledge through families of experiments. *Software Engineering, IEEE Transactions on*, *25*(4), 456-473.
[9] Wohlin, C., & Aurum, A. (2015). Towards a decision-making structure for selecting a research design in empirical software engineering. *Empirical Software Engineering*, *20*(6), 1427-1455.
[10] Harel, D. (1987). Statecharts: A visual formalism for complex systems.*Science of computer programming, 8(3), 231-274.*

arditi agência regional para o desenvolvimento da investigação tecnologia e inovação

This work was developed with financial support from ARDITI (Agência Regional para o Desenvolvimento da Investigação, Tecnologia e Inovação), in the context of program Madeira 14-20 -FSE.