Patterns for Negotiating Actors*

Michael Weiss
Carleton University, Ottawa, Canada
weiss@scs.carleton.ca

Babak Esfandiari
Carleton University, Ottawa, Canada
babak@sce.carleton.ca

1 Introduction

Negotiation is the process through which two or more autonomous actors (or agents) arrive at a mutually agreeable course of action. In the context of software systems, the need for negotiation stems from increased software modularity and increased autonomy of the resulting modules. Modules can consist of third-party (web) services or more sophisticated software agents. We shall use the term "actor" to designate such entities.

Negotiation is a search process. The actors jointly search a (multidimensional) space of actions in an attempt to find a point in the space at which they each meet their individual objectives. In this paper we present three alternative solutions for organizing the negotiation. Our focus is on how actors reveal their preferences, and whether they interact directly or not. We do not consider other design issues such as negotiation language, protocols or strategies, which are discussed eg in [12, 15].

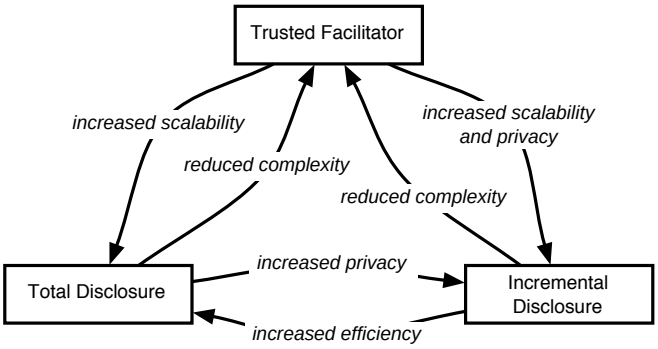


Figure 1: Relationships between the negotiation patterns

1.1 Overview of the Patterns

The solutions are documented in the form of patterns. The patterns and their relationships are shown in Figure 1. The labels on the arrows summarize the rationale for selecting one pattern over another. Rationale is expressed in terms of non-functional contributions. For example, the arrows linking TRUSTED FACILITATOR and TOTAL DISCLOSURE should be read as: leads to a more scalable, but also more complex solution.

As the patterns describe alternative solutions, there is not strict starting point from which to explore the set of patterns. Nonetheless, there is a logical progression to the order in which the patterns are presented. TRUSTED FACILITATOR is targeted towards the common situation where actors do not trust one another, and may also lack the knowledge to carry out a direct negotiation. TOTAL DISCLOSURE can be applied to situations where the negotiation is simple, and trusting actors interact directly with one another. INCREMENTAL DISCLOSURE aims at a situation where the actors interact directly, but it would put them at a disadvantage to reveal their preferences up front. Finally, the techniques of TOTAL DISCLOSURE and INCREMENTAL DISCLOSURE can be used internally in an implementation of TRUSTED FACILITATOR, when actors are represented by proxies.

The patterns are described using the Alexandrian form [1]. For each pattern we document its context, problem, forces, solution, consequences,

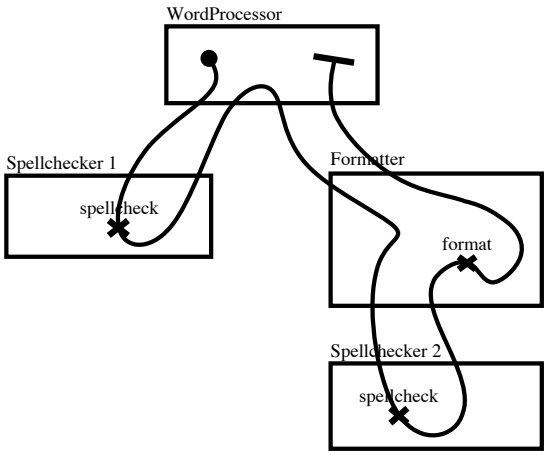


Figure 2: A composite word processing web service.

and related patterns. Problem and solutions are highlighted for emphasis. Diagram summarizes the solutions. In most of these we use the Use Case Maps [2] notation to indicate both structure and behavior of a solution.

1.2 Example

Throughout the paper, we will apply the patterns to a problem of aligning the behavior of the component services of a composite web service. Consider a word processing service (from [14]) that uses two third-party services, spell-checking and formatting, as shown in Figure 2.[1] This improves the maintainability of the web service, but also creates new problems, such as if, or how the user's language preference is passed between services.

Assume that the user has set her language preference for the word-processing service to Canadian English. But let us also assume that, hidden to the word-processing service, the formatting service itself incorporates a spell-checking service. This time, the formatting service does not specify

[1] This diagram shows service providers as rectangles (components). The X's (responsibilities) indicate actions, here corresponding to service invocations (eg, `spellcheck`) or local actions (eg, `format`). The path connecting responsibilities indicates their causal order (the start of a path is represented by a knob, and its end by a bar).

a language preference to the spell checking service. If the spell checking service uses a US English dictionary by default, the result of the service composition is that the incorrect language option will be applied.[2]

The Example Resolved sections of the patterns describe different ways of making the behavior of these web services consistent, that is, ensuring that the correct language preference is passed on to the formatter.

1.3 Decision Theory

This section provides a quick primer on decision theory concepts.

Decision theory is concerned with the behavior of self-interested actors [15]. To be self-interested means that each actor has its own preferences and desires about the environment in which it acts. Actors try to bring about *outcomes* in their environment (eg the outcome of a game).

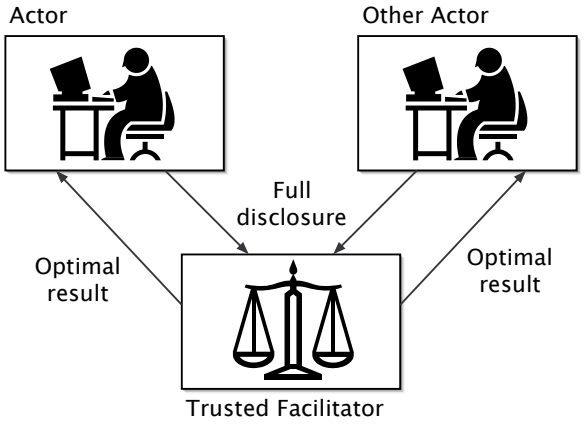
Actors prefer certain outcomes over others. An actor's preferences can be captured in the form of *utility functions*, which associates with each outcome a measure (utility) of how "good" the outcome is for the actor. The larger the utility the better. Utility functions lead to a *preference relation* over the outcomes. An outcome is preferred, if its utility is higher.

Each actor has a set of possible *actions*. When faced with the decision which action to take next, an actor, acting in isolation, will choose the action that leads to the outcome with the highest utility. However, outcomes are usually dependent on the actions of other actors as well.

Thus, in order for an actor to select the best action, it has to take account of the *choices* of all other actors. For each actor, this should be the action that meets the objectives of all actors, while forcing it to make only as little concessions (in terms of lost utility) as possible. *Negotiation* is the process through which the actors can arrive at a mutually agreeable outcome.

[2] A concrete example is the integration of the Safari web browser and the .Mac web mail service. Safari allows users to spellcheck web form input using the spellchecking service built into Mac OS X. The .Mac web mail service also integrates a spellchecker. As the user types their email, the Mac OS X spellchecker will check the spelling using the language preference of the OS (eg UK English). When the user asks the .Mac service to spellcheck the message, the language preference is not passed on to the .Mac service.

2 Trusted Facilitator



Each actor can choose between a set of actions, and in some order of preference. **Context**

In our example, the actors of interest are a word processing and a formatting service, and the actions to choose from correspond to the available language options. For simplicity, let us only consider two language options, Canadian (UK) and US English. The user of the word processing service prefers UK English, whereas the formatting service can support either language, but has has its default set to US English. **Example**

***

The actors need to arrive at a mutually agreeable course of action. That is, we need to determine the actions each of the actors should take, which meet the objectives of all actors and allow them to proceed, while forcing each actor to make only as little concessions as possible. **Problem**

The following forces need to be considered: **Forces**

- *Optimality*. Actors in your system pursue their own objectives, but need to be in mutual agreement on the next actions to take before they can proceed. These are the best individual choices, while satisfying the objectives of all actors (locally and globally optimal).

- *Privacy*. Actors do not want to exchange their preference relations. If they did, an untrustworthy actor could exploit the knowledge contained in the disclosed preferences, and offer an action that is optimal for them, but not optimal for the other actor. Actors could also exploit the knowledge for reasons unrelated to the negotiation.
- *Fairness*. The negotiation process should be fair. All actors should have access to the same information. In particular, they should be aware of the set of actions each actor can take. Otherwise an actor may want to reconsider their choice in light of the previously undeclared actions of others. Actors should also be truthful. There should be no incentive for them to misrepresent their intentions.
- *Uncertainty*. Actors do not have enough knowledge to reach a mutual agreement. They do not normally know each other's preferences, nor may they be aware of the set of actions available to the other actors. They may not know which actors to negotiate with, or they may not trust that the other actors will be truthful.
- *Scalability*. As a result of uncertainty, actors do not interact directly with one another. However, this means that all interactions need to be mediated by an intermediary, causing additional overhead. The intermediary can become a bottleneck if there is a heavy load.
- *Efficiency*. Efficiency is determined by the number of exchanges between actors. Less exchanges mean that the negotiation will be more efficient. However, the more information actors exchange per message, the less the actors' privacy will be preserved.

Therefore:

Solution

Use a trusted third party as a facilitator to select the best course of action. The facilitator has knowledge of all preference relations and selects the best action for each actor according to some fixed rules.

Figure 3 shows the structure of the pattern. The actors disclose their preference relations to the facilitator, ensuring that they are not made available to other actors. After the facilitator has selected the best course of action, it informs each actor about the action it should take. (In the UCM notation, a "stack" of actors indicates multiplicity. This diagram states that

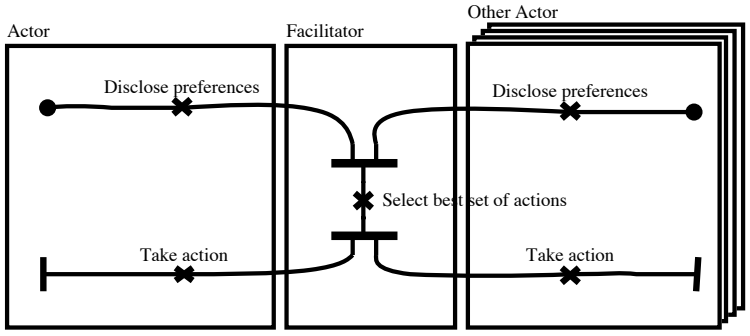


Figure 3: Structure of the TRUSTED FACILITATOR pattern

the negotiation can involve more than one other actor.)

The trustworthiness of the facilitator is of great importance. For the facilitator to function it needs to unbiased, and preserve the confidentiality of the actors. As in the world of security, trustworthiness can be established through various means. One approach is to maintain a hierarchy of trusted facilitators, in which one facilitator vouches for the proper operation of other facilitators. This hierarchy is "grounded" in a set of well-known facilitators. Another approach is to build a network of trusted facilitators through word-of-mouth among the actors to recommend facilitators.

Facilitators may mediate between actors who do not know one another. Thus they provide knowledge that the individual actors do not have. They may also act as gatekeepers, restricting the participation of actors in the negotiation. Facilitators usually also enforce ground rules that the negotiation has to follow, eg in an auction there are precise rules as to who can bid when, in what order bids must be made, or when an auction is won.

***

Consequences

Applying the pattern has the following consequences:

- *Optimality*. Since the facilitator knows the preference relations of all the actors, it can determine a best course of action. A facilitator also increases the "reach" of an individual actor, ie the number of actors

it is able to negotiate with. This improves the chances of the actors to obtain their most preferred choices (eg a higher price on a sale).

- *Privacy*. There is a risk of leaking the preference relations collected by the facilitator to an unauthorized actor. This somewhat limits privacy. However, if a hierarchy or network of trusted facilitators is in place malevolent misuse of user data will reduce the reputation of a facilitator. User privacy is also at stake if an intruder can gain unauthorized access to the data stored by a facilitator.
- *Fairness*. A facilitator eliminates the need for actors to misrepresent their intentions. For example, buyers participating in a facilitated auction have no incentive not to disclose to report a lower maximum bid than they are willing to make, because otherwise another buyer who discloses his maximum bid truthfully will win the auction.
- *Uncertainty*. The facilitator decreases the risk associated with not knowing which other actors are available for negotiation, and which actions other actors may choose. The facilitator may establish a standard of what information the actors have to disclose. A facilitator can also enforce ground rules of the negotiation, and ensure that preference information is only used for the purposes of the negotiation.
- *Scalability*. An individual facilitator can become a bottleneck under heavy load. This somewhat limits the scalability of the approach. However, this problem can be averted by adopting a federated approach, in which facilitators can be added to share the load.
- *Efficiency*. Since preference relations are disclosed to the facilitator in one step, the efficiency of the negotiation will be high.

Example Resolved

Applying the pattern to the example results in the trusted facilitator selecting the best course of actions that meets the preferences of both actors. One approach is to assign weights to preferences, for example, UK/1.0 and US/0.01 for the word processing service, and US/1.0 and UK/1.0 for the formatting service. Figure 4 illustrates this approach. The facilitator can then attach a rank to each combination, for example, the product of the weights each actor assigns to the *other's* choice. In this case, the combination (UK, UK) would result in a combined weight of $1.0 * 1.0 = 1.0$, but the combination (US, US) only in a weight of $1.0 * 0.01 = 0.01$.

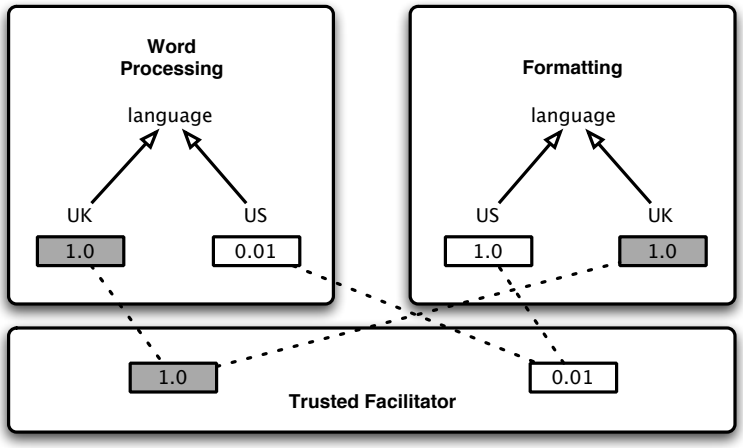


Figure 4: Details of the ranking procedure used in the example

Known Uses

Online auction service providers (such as eBay) perform the role of a trusted facilitator. They know each buyer's preferences (initial and maximum price) and seller's preferences (initial and minimum price), and automate the bidding process through which the auction winner is determined. (In eBay's case, proxy agents place bids on the actors' behalf.) The outcome of the auction best meets buyer and seller preferences.

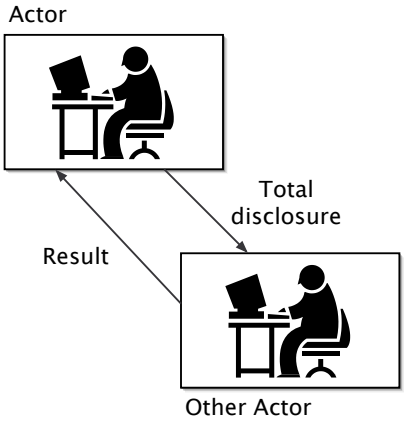
The concept of facilitators was introduced as part of the KQML protocol [6] for agent communication. Facilitators in KQML coordinate the interaction of agents, providing registry, forwarding, and matchmaking services. However, they do not implement higher-level interactions such as negotiation. These are implemented on top of the KQML protocol, using KQML performatives to define the negotiation protocol.

Related Patterns

The TOTAL DISCLOSURE pattern presents a decentralized approach to sharing preference relations that avoids the scalability problem. The INCREMENTAL DISCLOSURE pattern also addresses privacy concerns by describing how only a minimum amount of preferences can be released.

A number of specializations of this pattern for the electronic auction domain have been proposed in [8], however, only few details are given.

3 Total Disclosure



Each actor can choose between a set of actions, and in some order of preference. **Context**

In our example, the actors of interest are a word processing and a formatting service, and the actions to choose from correspond to the available language options. For simplicity, let us only consider two language options, Canadian (UK) and US English. The user of the word processing service prefers UK English, whereas the formatting service can support either language, but has has its default set to US English. **Example**

***

The actors need to arrive at a mutually agreeable course of action. That is, we need to determine the actions each of the actors should take, which meet the objectives of all actors and allow them to proceed, while forcing each actor to make only as little concessions as possible. **Problem**

The following forces need to be considered: **Forces**

- *Optimality*. Actors in your system pursue their own objectives, but need to be in mutual agreement on the next actions to take before they can proceed. These are the best individual choices, while satisfying the objectives of all actors (locally and globally optimal).

- *Privacy*. Actors trust one another, and are not concerned about exchanging their preference relations. However, once provided with the preference relations an untrustworthy actor can still exploit the knowledge they contain for reasons unrelated to the negotiation.
- *Fairness*. The negotiation process should be fair. The set of actions available to each actor are mutually known, and there is no concern of an unanticipated action leading an actor to reconsider their choice. Actors should also be truthful about their intentions.
- *Uncertainty*. The negotiation situation is relatively simple. The other actors, and the set of actions available to each actor are known. However, actors do not know whether other actors will be truthful. If the negotiation is repeated, the uncertainty will be reduced over time.
- *Scalability*. A TRUSTED FACILITATOR can become a bottleneck due to the overhead of requiring all interactions to be mediated by the facilitator. Direct interaction between the actors avoids this overhead. On the other hand, it assumes there is only a small number of actors, since otherwise the coordination overhead would be too high.
- *Efficiency*. Efficiency is determined by the number of exchanges between actors. Less exchanges mean that the negotiation will be more efficient. However, the more information actors exchange per message, the less the actors' privacy will be preserved.

Therefore:

Solution

Use a decentralized approach in which the actors interact in a peer-to-peer fashion, and reveal all of their preferences at once. The other actor selects a course of action that meets the preferences of both actors.

Figure 5 shows the structure of the pattern. The requesting actor discloses its full preferences to the other actor. The other actor selects the best course of action (from its perspective), then advises the first actor on which action it should take. This type of negotiation is also known as server-side negotiation. It involves only one round of negotiation.

An alternative to server-side negotiation is client-side negotiation. In client-side negotiation, the client requests a set of available actions from the server, then selects the best course of action. The responsibility of selecting

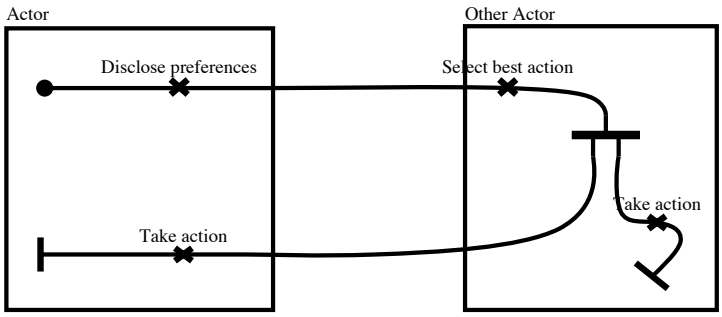


Figure 5: Structure of the TOTAL DISCLOSURE pattern

the action, therefore, remains with the client. Client-side negotiation has the advantage that the client typically knows better what action is most appropriate, but it also involves more overhead (an extra message).

An important variant of this pattern is parameter aggregation. In this approach an intermediary or gateway acts as an aggregator for preferences (the input parameters), but does not itself negotiate with the originating actor. Instead, it forwards the actor's preferences to the intermediary, possibly adding preferences of its own. Negotiation is then carried out between the intermediary and the other actor. An intermediary can also perform parameter filtering. Instead of adding preferences, the intermediary now removes preferences before passing them on to actors further downstream.

***

Consequences

Applying the pattern has the following consequences:

- *Optimality*. The choices made by the other actor may not be optimal, as the requesting actor may be in a better position to select the best action. The other actor may also intentionally select an action that is suboptimal for the requesting actor, but advantageous for itself.
- *Privacy*. When actors send their preferences in full, more of the actor's preference relations than necessary may be revealed. Requesting actors also cannot limit the amount of information the other actors collect about them beyond the purpose of the negotiation.

- *Fairness*. There is an imbalance between the actors. The other actor has more information than the requesting one: it knows its own as well as the requesting actor's preferences. The requesting actor cannot control whether the other actor will select its actions fairly. There is thus less incentive for the other actor to be truthful.
- *Uncertainty*. Requesting actors do not know whether the other actor will be truthful. However, trust can be established through repeated negotiation with the same other actor. An advantage of this style of interaction over using a TRUSTED FACILITATOR is that actors do not need to rely on second-hand experience with other actors. But the initial contact can be be established by a central facilitator.
- *Scalability*. Direct interaction between actors is more scalable than using a central facilitator, as long the overhead of coordinating the actors is not too high. The overhead grows with the number of actors that need to interact, which suggests that this pattern should only be used for small numbers of actors. Facilitators can be used in conjunction with a direct interaction style to coordinate certain aspects of system operation (eg locating actors, buffering messages, etc.).
- *Efficiency*. Since preference relations are exchanged in one step, the overhead in performing the negotiation is low.

Example Resolved

In the example, the word processing service submits its language preferences together with its request to the formatting service. The formatting service, in turn, can then select the appropriate language using a ranking procedure similar to how the example was resolved in TRUSTED FACILITATOR. The important difference between these approaches, though, is that no third party is involved. Figure 6 illustrates this approach.

Known Uses

This pattern is used for content negotiation by the HTTP protocol [5]. Content negotiation allows clients and servers to decide on the the most appropriate format to send a requested document. For example, the `Accept-Language` request header can restrict the languages to be used in the responses to the request. It lists the acceptable languages in order of preference. Each language option can be followed by a quality value (`q`), which is a number between 0 and 1, specifying its relative acceptability.

For example, the following header states that both UK English (`en-gb`)

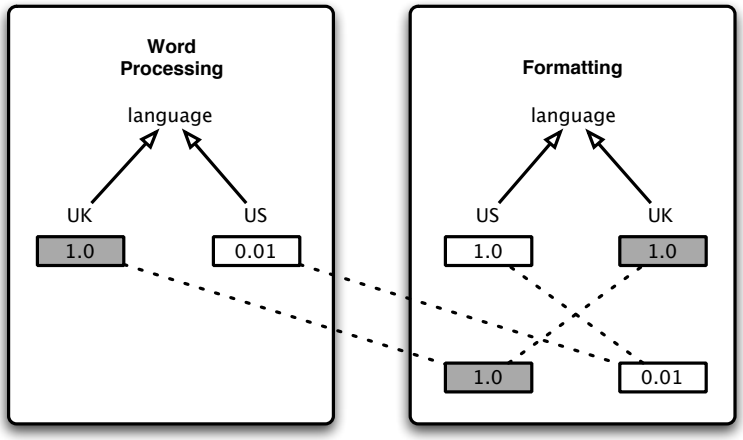


Figure 6: Details of how the conflict can be resolved in the example

and US English (`en`) are acceptable, but UK English is preferable:

```
Accept-Language: en-gb;q=1.0, en;q=0.01
```

HTTP supports both server-side (as discussed above), and client-side negotiation. In client-side negotiation, the client requests a resource without `Accept` headers. The server replies with a list of available contents, the client then makes an additional request for the desired format.

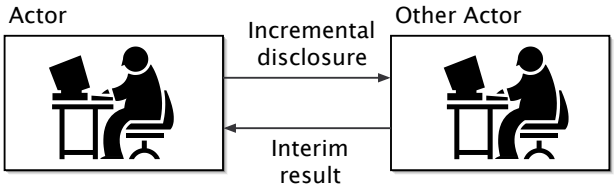
The pattern is also used by the Composite Capability/Preference Profiles (CC/PP) protocol for exchanging device capabilities [10].

The CC/PP protocol also uses parameter filtering. A downstream server may be responsible for selecting content according to the user-preferred language, while a gateway may be responsible for transcoding the content from HTML to WML. As stated in [10], the gateway might, therefore, choose not to forward all profile information to a downstream server.

Related Patterns

If you are more concerned about privacy than efficiency, you should apply the INCREMENTAL DISCLOSURE pattern, instead. Consider using a TRUSTED FACILITATOR, if managing the negotiation process locally within the actors becomes too complex (for example, too much information about the state of the negotiation must be shared between the actors, or it is difficult to select the set of actions that is a global optimum for all actors).

4 Incremental Disclosure



Context

Each actor can choose between a set of actions, and in some order of preference.

Example

In our example, the actors of interest are a word processing and a formatting service, and the actions to choose from correspond to the available language options. For simplicity, let us only consider two language options, Canadian (UK) and US English. The user of the word processing service prefers UK English, whereas the formatting service can support either language, but has has its default set to US English.

***

Problem

The actors need to arrive at a mutually agreeable course of action. That is, we need to determine the actions each of the actors should take, which meet the objectives of all actors and allow them to proceed, while forcing each actor to make only as little concessions as possible.

Forces

The following forces need to be considered:

- *Optimality*. Actors in your system pursue their own objectives, but need to be in mutual agreement on the next actions to take before they can proceed. These are the best individual choices, while satisfying the objectives of all actors (locally and globally optimal).
- *Privacy*. Actors do not want to exchange their preference relations. If they did, an untrustworthy actor could exploit the knowledge contained in the disclosed preferences, and offer an action that is optimal for them, but not optimal for the other actor. Actors could also exploit the knowledge for reasons unrelated to the negotiation.

- *Fairness*. The negotiation process should be fair. All actors should have access to the same information. In particular, they should be aware of the set of actions each actor can take. Otherwise an actor may want to reconsider their choice in light of the previously undeclared actions of others. Actors should also be truthful.
- *Uncertainty*. The negotiation situation is relatively simple. The other actors, and the set of actions available to each actor are known. However, actors may not trust that the other actors will be truthful. Often actors only interact once, ie they cannot learn from past interactions.
- *Scalability*. A TRUSTED FACILITATOR can become a bottleneck due to the overhead of requiring all interactions to be mediated by the facilitator. Direct interaction between the actors avoids this overhead. On the other hand, it assumes there is only a small number of actors, since otherwise the coordination overhead would be too high.
- *Efficiency*. Efficiency is determined by the number of exchanges between actors. Less exchanges mean that the negotiation will be more efficient. However, the more information actors exchange per message, the less the actors' privacy will be preserved.

Therefore:

Solution

Use a decentralized approach in which the actors interact in a peer-to-peer fashion, but reveal their preferences only gradually. As the actors exchange their preferences, they converge towards a mutually agreeable set of actions, while preserving as much privacy as possible.

Figure 7 shows the structure of the pattern. The negotiation starts with a proposed action from one of the actors. In response to the proposal the other actor(s) will either accept the proposal and take the proposed action, reject it and terminate the negotiation, or offer a counterproposal. There can be multiple rounds of counterproposals, until either an agreement on the course of action is reached (accepted), or not (rejected). Preferences are only disclosed with each step of the negotiation, that is, when accepting a proposed action, rejecting it, or making a counterproposal.

This solution forces actors to make their intentions explicit [4]. In response to the intentions disclosed by other actors, the actors may replan

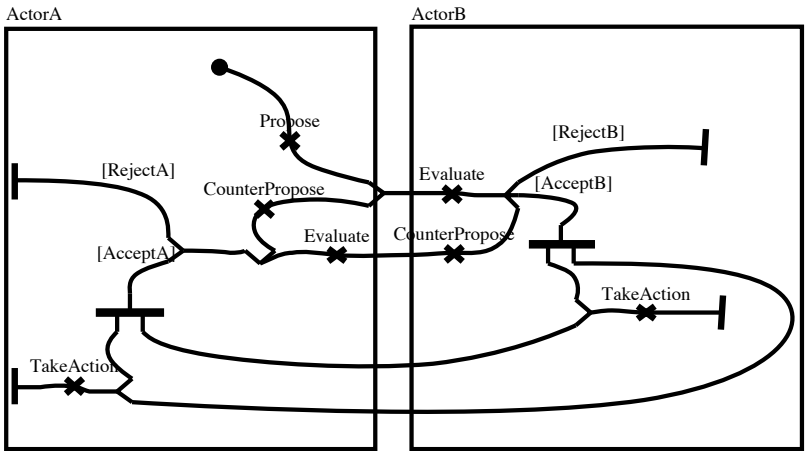


Figure 7: Structure of the INCREMENTAL DISCLOSURE pattern

their actions to avoid detected interactions with their own intentions. Replanning involves choosing among alternative courses of action.

In selecting alternative courses of action, actors are guided by maximizing their utility. The exchange of intentions, and subsequent replanning proceeds until the actors manage to align their actions, or if they reach an impasse which makes further negotiation impossible. A common approach to represent the alternative courses of actions is as an AND/OR tree.

***

Consequences

Applying the pattern has the following consequences:

- *Optimality*. Actors can choose actions that are optimal in light of the preferences already disclosed to them by others. However, the final negotiation outcome may still be suboptimal, as actors do not learn about all of the preferences of the others (eg if an actor bargains with multiple actors, the first acceptable offer may not be the best one).
- *Privacy*. The requesting actor retains control over which preferences it exchanges with other agents. Only the necessary information to reach an agreement is disclosed. Therefore, requesting actors can

also limit the amount of information the other actors collect about them beyond the purpose of the negotiation.

- *Fairness*. Incremental disclosure limits how much either actor can exploit the knowledge contained in the preferences to their benefit. It also avoids that the requesting actor releases more information than necessary for the negotiation to proceed, which could be misused.
- *Uncertainty*. Requesting actors do not know whether the other actor will be truthful. By keeping preference sharing to a minimum necessary, actors can reduce the risk of untruthful behavior, even if the actors have not previously interacted. To further reduce uncertainty, the contact can also be be established by a TRUSTED FACILITATOR.
- *Scalability*. Direct interaction between actors is more scalable than using a central facilitator, as long the overhead of coordinating the actors is not too high. The overhead grows with the number of actors that need to interact, which suggests that this pattern should only be used for small numbers of actors. Facilitators can be used in conjunction with a direct interaction style to coordinate certain aspects of system operation (eg locating actors, buffering messages, etc.).
- *Efficiency*. The privacy of the requesting actor is best protected by releasing preferences in small increments. However, this may result in a large number of messages between the actors. For efficiency, the increments should be larger. However, this will reduce privacy.

Example Resolved

Figure 8 shows an important variant of the pattern: in a permission-act cycle, a permission is requested before an action can be performed [14]. In Figure 8, the `spellcheck` action from Figure 2 has been replaced by a proposal of a `spellcheck(US)` action, followed by a counter-proposal of a `spellcheck(UK)` action. Only then the actual request is performed.

Known Uses

Negotiating actors are used in [11, 13] to address feature interactions in telecom systems. In this approach actors negotiate the details of setting up of a call. The actors do not want other actors to see details such as the names on a party's call screening list. Actors represent their preferences as a hierarchy of goals, which describe the space of alternatives the actors can choose from. During the negotiation actors exchange proposals in the forms of goals, and determine whether their goals are compatible.

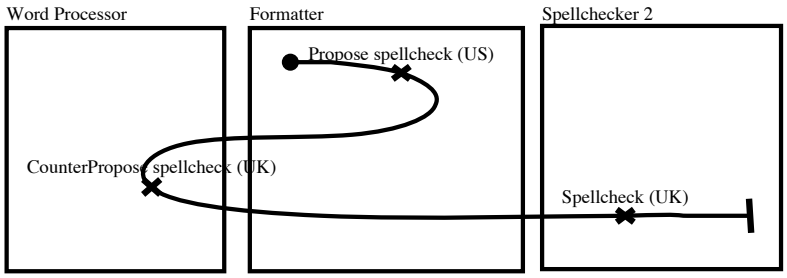


Figure 8: Details of how the conflict can be resolved in the example

More recently, negotiating actors have been applied to privacy protection protocols such as the protocol for the gradual release of preferences in [9]. In this protocol, actions are represented as attribute-value pairs, and a precedence relation is defined between attribute values, as well as between attributes, and constraints on those attributes. A canonical example is the release of preferences regarding desired features of a digital camera, to which the vendor responds with a set of matching products.

A classical example of using the pattern in an electronic marketplace is Kasbah [3]. In the Kasbah system users create agents to represent them in parallel bilateral negotiations. Users can be sellers of items (eg a book) or buyers. In each case, they instruct their agents with a starting price, reservation price, and bidding strategy. The agents then perform the user's bidding according to the user-selected strategy (reflecting their risk tolerance). As soon as one of the negotiations between a seller and a buyer agent finds a mutually agreeable price, its users are informed.

Related Patterns

The TOTAL DISCLOSURE pattern can be applied if efficiency is more important than privacy. TOTAL DISCLOSURE can be considered an extreme case of this pattern, in which all preferences are exchanged at once.

A related pattern is the NEGOTIATION pattern in [7]. Its main focus is on service discovery in distributed applications. It defines similar negotiation primitives (propose, offer, request, accept, and reject). Another is the NEGOTIATING AGENTS pattern [4], which deals with the detection and resolution of conflicting intentions between actors. However, neither pattern provides a discussion of privacy, fairness, or scalability issues.

Acknowledgements

Thanks to my shepherd Klaus Marquardt for his fruitful feedback.

References

[1] Alexander, C., A Pattern Language, Oxford University Press, 1977.

[2] Buhr, R., Use Case Maps as Architectural Entities for Complex Systems, IEEE Transactions on Software Engineering, 1131–1155, 1998.

[3] Chavez, A., and Maes, P., Kasbah: An Agent Marketplace for Buying and Selling Goods, Conference on Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM), 1996.

[4] Deugo., D., Weiss, M., and Kendall, E., Reusable Patterns for Agent Coordination, in: Omicini, A., Coordination of Internet Agents, Springer, 2001

[5] Fielding, R., Gettys, J., et al, Hypertext Transfer Protocol – HTTP/1.1, RFC 2616, 1999, http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[6] Finin, T., Fritzson, R., et al, KQML as an Agent Communication Language, Conference on Information and Knowledge Management, ACM, 1994.

[7] Gomaa, H., Designing Software Product Lines with UML, Addison-Wesley, 2005.

[8] Paschke, A., Kiss, C., and Al-Hunaty, S., A Pattern Language for Decentralized Coordination and Negotiation Protocols, E-Technology, E-Commerce, E-Service (EEE) Conference, IEEE, 2005.

[9] Smith, R., and Shao, J., Preserving Privacy when Preference Searching in e-Commerce, ACM Workshop on Privacy in the Electronic Society, 2003.

[10] Ohto, H., and Hjelm, J., CC/PP Exchange Protocol Based on HTTP Extension Framework, W3C Note 24, 1999, http://www.w3.org/TR/NOTE-CCPPexchange.

[11] Griffeth, N., and Velthuijsen, H., The Negotiating Agents Approach to Runtime Feature-Interaction Resolution, International Workshop on Feature Interactions in Telecommunications Systems, 217–235, IOS, 1993

[12] Singh, M., and Huhns, M., Service-Oriented Computing: Semantics, Processes, Agents, Section 17.5: Negotiation, 372-380, Wiley, 2005.

[13] Velthuijsen, H., Distributed Aritficial Intelligence for Runtime Feature-Interaction Resolution, Computer, August, 48–55, 1993.

[14] Weiss, M., and Esfandiari, B., On Feature Interactions among Web Services, International Conference on Web Services (ICWS), 88–95, IEEE, 2004.

[15] Wooldridge, M., An Introduction to Multiagent Systems, Wiley, 2002.