

T-compound: An Agent-Specific Design Pattern

Eric Platon¹, Nicolas Sabouret², and Shinichi Honiden¹

¹ National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda, Tokyo 101-8430

² Laboratoire d'Informatique de Paris 6, 8, Rue du Capitaine Scott, 75015 Paris
{platon, honiden}@nii.ac.jp, nicolas.sabouret@lip6.fr

Abstract. This paper introduces the T-compound, an agent interaction pattern we propose as reusable architecture in the design of multi-agent systems. We explain why this model and its underlying concept of overhearing are relevant and specific to agents. Then, we describe two strategies of implementation, together with their qualities and weaknesses. We then identify the T-compound in various agent-based projects to confirm its properties of reusability and composability that turn it into design pattern. The paper ends with reports of other implementations that address overhearing and related concepts.

1 Introduction

The evolution from object-based to agent-based programming requires significant endeavours so that the software engineering community wishes to reuse as much as possible the grounds acquired in the object world. Foundations formed by object-oriented methodologies over the previous generations (procedural, functional, etc.) reached high levels of maturity and these achievements should be preserved to promote agent technologies, since neither workforce nor economic trends are ready to accept starting from a void. Examples of efficient and effective method traits are design patterns, these reusable and successful pieces of class architectures that can be applied to numerous cases at different stages of the software development process. In this paper, we aim at presenting a work compatible with this trend by proposing an original design pattern for agent systems named the T-compound. It represents aspects of the recent concept of overhearing [1, 2], and we will show along this paper it is actually a frequent pattern in Multi-Agent Systems (MAS).

To this end, we compile in Section 2 our motivation for this proposal and the issues it addresses. We detail the architecture of the T-compound in Sect. 3 with special care on the question of implementation. Then, we present in Sect. 4 related work and applications that inspired this research and support the idea of design pattern, and finally the paper ends with our concluding words in Sect. 5.

2 New concept, new needs

2.1 From Object- to Agent-Methodologies

Object-oriented programming introduced higher flexibility and modularity in software design, compared with previous engineering paradigms (procedural, functional, etc.). These advantages quickly enacted the compilation of successful and reusable pieces of software, namely the design patterns [3–5]. These ‘on-the-shelf’ architectures allow saving time by mostly binding the design stage to the selection of proper patterns that address the problem. Also, it ensures higher robustness of the resulting design since these patterns proved their stability over years of usage by professional engineers. Finally, the readability of the software is much clearer due to easily identified organisation of the code, and a mutual understanding among designers and cultivated readers.

Agent is a concept related but distinct from object [6, 7]. This new abstraction lies at a higher level and needs original foundations to exploit the potential it promises, especially in adaptive and intelligent distributed systems such as MAS. However, software engineers wish to reuse the object methodological capital with agents when it makes sense, and only extend it otherwise [8]. Design patterns being a successful achievement of the object paradigm with significant consequences on development, we shall explore similar methods with agents to cope with their specificities.

So far, object-oriented patterns have been reused to design agent systems. However, those patterns rely on the object definition of interaction, i.e. message passing or ‘stimuli’ [9, chapter 2.10.1]. Agents are endowed with some extent of autonomy implying that interactions generally rely on *intentions* in the theory of speech acts captured in the FIPA-AUML standard [10, 11]. Consequently, agent patterns should be considered as a new concern.

2.2 Concept of Overhearing

Overhearing has been recently introduced in the MAS community as an indirect interaction mechanism [1, 2]. Informally, overhearing occurs when two agents are discussing and a third one can listen to their conversation as depicted on Fig.1. This architecture originally appeared to address agent cooperation [1], monitoring MAS at runtime [12], and dynamic group formation [13].

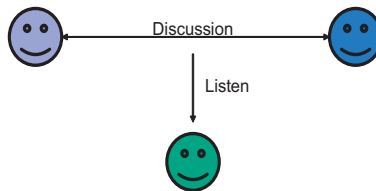


Fig. 1. Overhearing Model

With these projects, overhearing bred interests as an agent-specific interaction. In fact, objects cannot adhere to this mechanism as it relies on the *intention* of agents to hear or to be heard. Furthermore, overhearing is akin to eavesdropping and then becomes a hazardous metaphor. That is why in all projects cited above, a common hypothesis is to rely on cooperative teams of agents that trust each other. The general case of open MAS will be the subject of further research.

Following these early results, Gutnik and Kaminka proposed a formal model of overhearing specialised in conversation recognition [2]. Various general implementation techniques were developed explicitly for overhearing or, in our understanding, related ideas [14–16]. These first endeavours to exploit the notion confirms the potential of this mechanism. Yet, the question of systematic modelling to unify these proposals is not addressed. In this aim, we propose the T-compound as a bottom-up approach of overhearing that could serve as agent-specific design pattern.

3 Agent, Overhearing, and T-compound

We designed the T-compound as an interaction infrastructure in agent engineering. This section aims at presenting its architecture, the patterns we could infer from it, and the issue of actual implementation.

3.1 Architecture and Patterns

Our representation of the concept of overhearing on Fig.1 defines the T-compound, and the shape of this image is the origin of its name. The architecture relies on three agents including two actors engaged in a conversation, one listener, and the necessary links to model the discussion and the overhearing. The ‘discussion’ link between the actors is not necessarily bidirectional. The listener agent can actually be constrained to hear only part of a discussion, as we would usually hear only one side of a phone call performed by others [2]. The ‘listen’ link is the actual overhearing of the discussion and should appear for each listener, forming each time a different ‘T’. From the compound, we derived three reusable patterns from our study of agent system architectures that were implemented in various projects (see details in Sect. 4.1). Fig. 2, 3, and 4 show the cases we distinguished.

The first pattern on Fig.2 is the direct application of the T-compound. The listener agent has no acquaintance with the two other agents and only follows the interaction flows. This pattern could be applied to cases of monitoring, test and verification of MAS, or even the negative view of spies we avoid presently by considering only cooperative agents. In the former case, monitoring applications were already explored [12], and we expect extended results in the case of tests with further research, as concerns appear on interaction protocols to dynamically introduce these test agents and define their abilities.

The next pattern on Fig.3 exploits one more link between the listening agent and one of the discussion actors. This new acquaintance enacts another interaction channel, typically in the case of personal assistants. Such agent applications

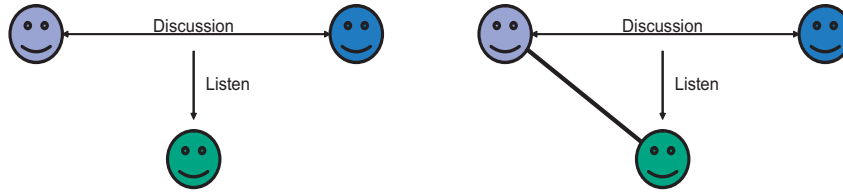


Fig. 2. Pattern 1 for monitors or spy agents **Fig. 3.** Pattern 2 for personal assistants agents

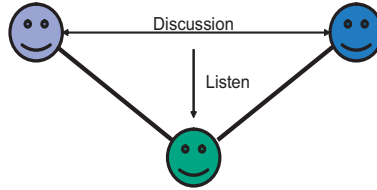


Fig. 4. Pattern 3 for mediators (referee) or global assistants

are increasingly frequent in research on human interfaces to support the user in its tasks, such as time management, online booking, and so forth [17,18]. This pattern also matches solutions of the active domain of automated negotiations whereby users delegate the actual negotiation process to their ‘proxy’ agent [19–21].

The last pattern on Fig.4 completes the previous one with the missing interaction channel. This ‘full mode’ endows the listener agent with mediator capabilities, such as referees in dispute resolutions, or global assistant that provides its service to users equally. This instance could also be appropriate for advertising agents that listen to subscribers’ discussions to contact them according to their preferences and habits. This pattern can be partially observed in projects like the Helper Agent [22] and M [23] that we present in Sect.4.1.

Beyond the architectures covered by our three patterns, there exists well-known models the T-compound cannot represent —and they could form other potential agent design patterns. For instance, matchmakers —which live in many agent systems to manage services similar to yellow pages— and brokers do not fit any of these three patterns. Indeed, brokers stand in-between the two discussing agents, route and sometimes filter the exchanged information; the matchmaker enables the discussion, but then disappears from the communication as its role is bound to initiate the interaction between the two demanding agents. Neither of these two agent types *listen* in the way supported by the T-compound.

3.2 Toward Implementation

In this section, we describe two implementation strategies of the T-compound, using a common example. The first is explicit as it relies on direct agent in-

tentions to participate in ‘T’; the second is implicitly setup by environmental constraints.

Illustrative Scenario. Our example specifies a simple multi-agent game simulation consisting of two players, one referee, and the audience. All entities are embedded in the system environment named ‘room’. Fig.5 depicts the actors and settings of our game. Traditional interactions are drawn with plain arrows, whereas overhearing appears with dashed ones.

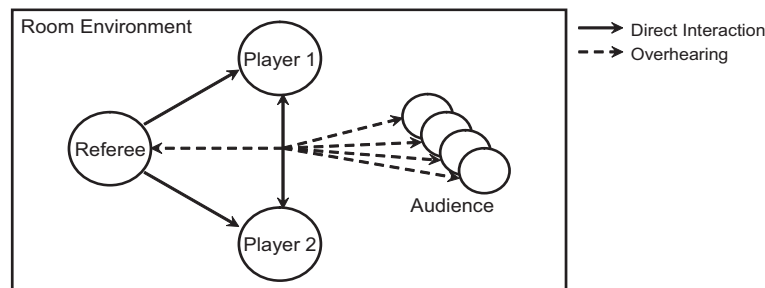


Fig. 5. Our example: a game with referee and audience

In this game, the referee chooses a topic at random; the players have to argue about the subject and the winner is elected when the opponent cannot answer back on time. Once the match starts, the players interact by sending arguments to each other directly. The referee overhears their discussions and enforces the game rules (e.g. argument validity, etc.). The audience also overhears the match to enjoy the performance.

In the following, we present two implementation strategies of the interactions required by this game. The methods end with ‘agent diagrams’, to be related to the usual class diagrams without agent internals.

Explicit listening and agent intentions. This first method implements the T-compound as a broadcasting of messages, and is directly inspired from Busetta et al. [14] (see Sect.4.2). Any message that can be overheard is sent at once to the intended recipient and the listeners that subscribed to the service. The sending is here performed and managed by the sender (it will be different with the next method). The corresponding interaction diagram is laid out on Fig.6 for a single listener. The implementation we can expect from this diagram is straightforward with current agent methodologies. Overhearing of α (dashed arrow) is coded as a traditional direct interaction h_α with the constraint to carry exactly the same messages as the listened channel, i.e. all messages through α must also be sent as copies through h_α . It appears the right execution of overhearing in this implementation relies on the agent intention to be listened. Such agent commits

to send copies of overheard messages to any registered listener, so that it becomes an explicit overhearing for the agent.

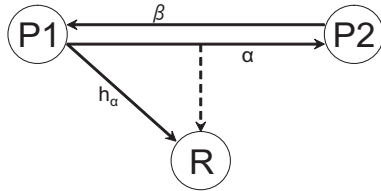


Fig. 6. Explicit Overhearing

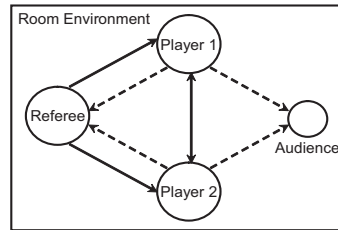


Fig. 7. Explicit Implementation Diagram

In our example, this implementation means that players exchange directly arguments for the game and send copies to the referee and the audience on the broadcast list. We detailed this implementation diagram on Fig.7. Note the indirect interactions disappeared to become direct interactions (dashed arrows) with constraints to specify the messages they carry are exact copies of messages between the two players.

Implicit listening and agent environment. In this framework, any agent that needs to participate or listen to conversations subscribes to the room environment through appropriate protocols to be defined, i.e. they enter the room and commit to its local rules. The implementation of this room-based overhearing requires a representation of this part of the environment. Consequently, this part is shown on Fig.8 as the entity E, a special facility that could be thought of as a dedicated mail server. E gathers all messages that roam into the corresponding room and routes them to their intended recipients and listeners. Conversely to the previous architecture, the message sender let E managing the delivery.

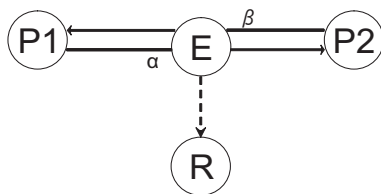


Fig. 8. Implicit Overhearing

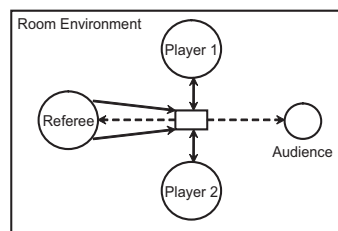


Fig. 9. Implicit Implementation Diagram

Therefore, we separate the agent from the message management and the environment ensures the distribution. The notion of room allows distinguishing

different groups that can listen to each other in a large MAS. This method is named implicit overhearing since indirect interactions become the standard of communication. A direct interaction becomes a private message that E delivers only to the intended recipient.

In the example, the room environment that limits the MAS receives all messages from the players and the referee, and routes them to their receivers. In the case of messages from players, the game requires the room to broadcast them to the referee and the audience.

3.3 Comparison

With explicit overhearing, the main advantage is to control from the agent point of view where messages end. The sender knows exactly which agents will receive the message since it decides the destination. However, all agents that accept overhearing must effectively send all copies of messages. This causes an overhead of managing the mailing lists. In our example, players have to focus on their game and spend most of the computing time on it. Handling messages is necessary for the system, but players cannot manage all situations, especially when the audience population evolves.

With implicit overhearing, properties are counterparts of the previous method. First, the agent loses control of message delivery. Instead, the environment manages it and ensures the behaviour expected by system rules is respected. However, agents generally trust the system institution they applied to, so that they should be willing to delegate such a management and focus on their proper functions. Consequently, agents also save the message handling overhead. Hence, players can concentrate on their game, while the environment manages where messages are ending, whatever population is listening. One drawback from the environment solution is to practically centralise messages. In larger systems, this can be an implementation issue, whereas broadcasting is generally well scalable.

4 T-compound and Design Pattern Properties

In this section, we review several types of agent systems, identifying the patterns that lead us considering the T-compound as a design tool candidate. This review serves at justifying the T-compound verifies the two main properties of design patterns, namely reusability and composability. Then, we report work related to our implementation issues.

4.1 Identification of T-compound in Past Agent Systems

Monitors. Gutnik and Kaminka proposed an overhearing agent able to monitor others' conversations and apply recognition algorithms in different situations (random lost of messages, etc.) [2]. In terms of interactions, the system is represented on Fig.10, and enters the category of Pattern 1 on Fig.2.

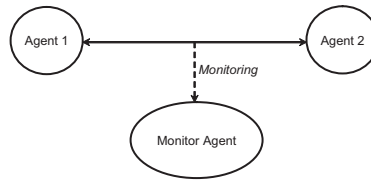


Fig. 10. Monitor System

In this case, the notion of trust between agents is a basic requirement to avoid considering overhearing as eavesdropping. Monitors must perform their tasks in cooperation with overheard agents in a non-intrusive fashion. For the moment, the common hypothesis is to consider trustworthy agents, but the first pattern will quickly require special care to prevent spy agents to multiply in real applications.

Personal assistants. The COLLABORATIVE AGENT proposed in [18] assists one user in the task of scheduling complex flight routes and matches our second pattern on Fig.3. In this project, the user plans a journey with an airplane company software client linked to the schedule database. As soon as no solution to the user request is available, COLLAGEN intervenes and suggests some alternatives. The agent actually observed the user in its planning process, reasoned about the actions and could identify problems together with solutions. Fig.11 shows the interaction structure of this software.

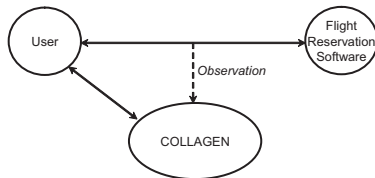


Fig. 11. COLLAGEN, a personal assistant

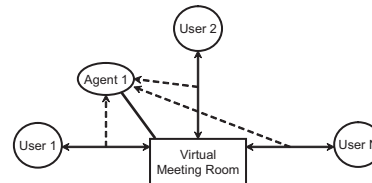


Fig. 12. Global assistant system

Rich uses the concept of observation instead of listening, but we think both can be unified in the software world. This example of personal assistant architecture exists in various other projects such as [17, 19–21].

Global assistants. To illustrate global assistants, we chose M [23] as representing the category, and related systems have similar architectures [22, 24] that match the third pattern on Fig.4. The system maintains in real-time a shared-working space in a virtual meeting room. Users create, modify and exchange documents, while agents check for integrity and organise the contents on each

interface depending on the global and local-user constraints. Fig.12 shows a simplified case of M, to maintain readability of the drawing.

The actual M system delegates one agent per user, and each of them can listen to all interactions to process the ones concerning its owner. Although these agents are personal, they can interact with peers and users to perform their service correctly, and so become global assistants. This image can be thought of as an example of weaving T-patterns together.

Another case with Pattern 3. The last example relies exclusively on software agents to complete the panel of applications of this section and show that the user has no compulsory role in the T-compound. In the Robocup, robots or software players are engaged in football matches [25]. The game is entirely automatic and motivates most areas of computer science, including multi-agent systems. In such games observation of opponents' behaviours is critical to develop efficient strategies dynamically and automatically as, for instance, in [26]. The T-compound can model the infrastructure of this process by enabling agents to 'see' or 'listen' to other's actions. One agent can adapt its trajectory when opponents pass the ball; it can also choose a strategic position when opponents 'talk' their intentions. However, agents are opponents in this context so that our implementations may lack security enforcement.

4.2 Other Related Work

Explicit Overhearing Busetta et al. proposed the 'LoudVoice' as practical implementation of the overhearing concept [14]. It relies on multicasting messages to recipients and listeners, and is the origin of our explicit case. Our approach only features the minimal requirements for overhearing in order to compare with the second method.

Busetta experimented this technique and showed it is efficient in terms of message delivery and scalability. We think it is a practical and complete way for overhearing, although it suffers from the issues we presented in Sect.3.2, that is agents have to manage overhearing by themselves. Depending on the system issues, this can however be the best solution. For instances, large-scale MAS can leverage the expected performance from this technique.

Implicit Overhearing This approach is akin to the metaphors and models of 'coordination artifact' by Omicini et al. [16], and 'field-based coordination' by Mamei et al. [27]. Here the metaphor is similar but in the case of 'communication artifact'. The main difference lies in the fact our model deals with direct and indirect interactions at once and in the exclusive case of communication. Also, coordination artifacts stand at a higher level than implementation and field-based coordination exploit tuple-spaces to deliver messages without use of the overhearing concept, although we think their technique could implement it. In cases that need more natural integration of overhearing, we think this type

of implementation is preferable. Mamei show tuple-spaces are an efficient solution, especially in the case we are interested in ('message tuple'). For instances, simulations and generally cooperative MAS can get benefit of this method.

5 Conclusion

Design patterns are one strength of object methodologies in constructing complex systems that remain clear, robust, and feasible in timely-fashion. Hence one wish to exploit these patterns in the agent world, and also propose specific ones. In this paper, we introduce the T-compound as candidate agent pattern for the soaring concept of overhearing. In support to this presentation, we showed it is already a frequent architecture that appears in many projects exploiting similar concepts. We also described two types of implementations inspired by existing or related work [14, 16, 27]. The former is based on broadcasting messages to recipients and listeners. It is efficient and straightforward application, but it may load agents with intricate message management. The latter technique relies on the agent environment to delegate message management by applying to 'overhearing rooms'. Performance may be lower than the previous technique owing to centralisation bottlenecks, but we need practical tests to confirm it. This technique looks more natural as overhearing is supported by the environment itself. The choice of the implementation depends on the design. We think the scalability of the first one is its main selection criteria, whereas the natural framework and advanced concept characterises the second.

References

1. Busetta, P., Serafini, L., Singh, D. and Zini, F.: 'Extending Multi-Agent Cooperation by Overhearing', International Conference on Cooperative Information Systems, 2001.
2. Gutnik, G. and Kaminka, G.: 'Towards a Formal Approach to Overhearing: Algorithms for Conversation Identification', AAMAS'04 Conference, 2004.
3. Gamma, E.; Helm, R.; Johnson, R; Vlissides, J.: 'Design Patterns', Addison-Wesley Professional Computing Series, 1995.
4. Schmidt, D.; Stal, M; Rohnert, H.; Buschmann, F.: 'Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects', Wiley Series in Software Design Patterns, 2000.
5. Magee, J.; Kramer, J.: 'Concurrency: State Models and Java Programs', Wiley, 1999.
6. Ferber, J.: 'Multi-Agent System: An Introduction to Distributed Artificial Intelligence', Addison Wesley Longman, 1999.
7. Huhns, M.; Singh, M. Eds: 'Readings in Agents', Morgan Kaufman, 1998.
8. Agent UML Specifications: <http://www.auml.org>, 2004.
9. UML Specifications Version 1.5: <http://www.uml.org>, 2003.
10. FIPA Specifications: FIPA Modelling: Interaction Diagrams, Working Draft, 2002.
11. FIPA Specifications: FIPA Communicative Act Library Specification, 2002.
12. Kaminka, G.A., Pynadath, D.V., Tambe, M.: 'Monitoring Deployed Agent Teams', Agents-2001, 2001.

13. Legras, F. and Tessier, C.: 'LOTTO: Group Formation by Overhearing in Large Teams', AAMAS'03 Conference, 2003.
14. Busetta, P.; Donà, A.; Nori, M.: 'Channeled Multicast for Group Communications', AAMAS'02 Conference, 2002.
15. Tummolini, L.; Castelfranchi, C.; Ricci, A.; Viroli, M.; Omicini, A.: "'Exhibitionists" and "Voyeurs" do it better: A Shared Environment for Flexible Coordination with Tacit Messages', E4MAS Workshop, AAMAS'04 Conference, 2004.
16. Omicini, A.; Ricci, A.; Viroli, M.; Castelfranchi, C.; Tummolini, L.: 'Coordination Artifacts: Environment-based Coordination for Intelligent Agents', AAMAS'04 Conference, 2004.
17. Mitchell, T.; Caruana, R.; Freitag, D.; McDermott, J.; Zabowski, D.: 'Experience With a Learning Personal Assistant', Communications of the ACM, 1994.
18. Rich, C.; Sidner, C.: 'COLLAGEN: When Agents Collaborate With People', First International Conference on Autonomous Agents, Marina del Rey, February 1997.
19. Chavez, A.; Maes, P.: 'Kasbah: An Agent Marketplace for Buying and Selling Goods', First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, 1996.
20. Tsvetovaty, M.; Gini, M.; Mobasher, B.; Wieckowski, Z.: 'MAGMA: An Agent-Based Virtual Market for Electronic Commerce', Journal of Applied Artificial Intelligence, Vol. 11, Number 6, pp. 501 - 523, September 1997.
21. Wurman, P.R.; Wellman, M.P.; Walsh, W.E.: 'The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents', Second International Conference on Autonomous Agents, Minneapolis, USA, 1998.
22. Isbister, K.; Nakanishi, H.; Ishida, T.; Nass, C.: 'Helper Agent: Designing an Assistant for Human-Human Interaction in a Virtual Meeting Space', CHI'00 Conference, 2000.
23. Riecken, D.: 'M: An Architecture of Integrated Agents', in Software Agents, J. Bradshaw Editor, pp 419, AAAI Press, 1997.
24. Chalupsky, H.; Gil, Y.; Knoblock, C.A.; Lerman, K.; Oh, J.; Pynadath, D.V.; Russ, T.A.; Tambe, M.: 'Electric Elves: Applying Agent Technologies to Support Human Organisations', in Proceedings of the 13th IAAI Conference, pp. 51-58, 2001.
25. The Robocup Official Website: <http://www.robocup.org/>
26. Kaminka, G.A.; Fidanboyly, M.; Chang, A.; Veloso, M.M.: 'Learning the Sequential Coordinated Behavior of Teams from Observation', Robocup 2002: Robot Soccer WorldCup VI, Springer-Verlag, 2002.
27. Mamei, M., Zambonelli, F.: 'Self-Maintained Distributed Tuples for Field-Based Coordination in Dynamic Networks', ACM SAC'04, March 2004.