

Managing Interaction for Multimedia Collaboration: Through the Keyhole of Noughts and Crosses Games

Siti Z. Z. Abidin, Min Chen and Phil W. Grant
Department of Computer Science
University of Wales Swansea
Swansea, SA2 899, UK
{cssiti, m.chen, p.w.grant}@swansea.ac.uk

Abstract

Interaction management is concerned with the protocols that govern structured interactive activities among multiple users or agents in distributed and collaborative environments. Research in this area has largely been carried out in specific application domains, such as 3D virtual environments. We present an abstraction of various collaborative applications in the form of variations of noughts and crosses. We examine the needs in these games for programming interaction protocols, and propose a comprehensive collection of program constructs for supporting interaction management. We demonstrate, through these variations of the game, the usefulness of these language constructs.

1. Introduction

Interaction management is concerned with the protocols that govern structured interactive activities among multiple users in distributed and collaborative environments. The implementation of the protocols is often not a trivial task in the development of an application involving structured communication. The provision of protocol control mechanisms is the weakness of most existing programming languages and development tools. Software engineers need developer support which can alleviate the burden in implementing correct and reliable codes for managing an online meeting, a teamwork exercise or a multi-user web camera in a structured manner. In this paper, we identify a collection of useful interaction protocols that are common in many multimedia collaborative applications. We consider an abstraction of various multimedia collaborative applications in the form of the noughts and crosses game and its variations. We examine the needs in these games for programming interaction protocols and propose a comprehensive collection of program constructs for supporting interaction manage-

ment. These constructs are incorporated into JACIE (*Java-based Authoring language for Collaborative Interactive Environments*)[7, 8], a scripting language designed to support rapid prototyping and implementation of distributed collaborative applications. We demonstrate, through variations of the noughts and crosses game, the usefulness of these language constructs.

2. Related Work

Human-human interaction in a distributed multimedia collaborative environment often requires coordination in a structured manner. Research on interaction management has been conducted largely in the context of specific management, including floor management in video conferencing [12], group coordination [4], web-based camera control [2], 3D collaborative virtual environment [6] and agent-based collaboration [10, 11, 3]. Protocols that have been deployed in such applications include: contention, round robin, token-based, reservation and prediction [5]. Recently, a more formal approach has been applied to interaction management, and this includes GTRBAC [9], which introduces the notion of *role enabling* and *role activation*, and TILCO[1], which separates the *external* view of the process of interaction from its *internal* view. JACIE represented one of the first attempts to provide interaction management in collaborative environments through high-level language constructs [7, 8]. However, the major question which remains unanswered is to what extent a collection of interaction protocols is considered adequate to address the needs of most collaborative applications.

3. Generalizing Noughts and Crosses

We first define a set of abstract notations for modelling the variations of the noughts and crosses game and the corresponding interaction protocols in Section 4. A summary of the games is given in Table 1 highlighting their main

Name	Turn Control	Cell States	Applications
Generalised game	round robin	$\square, \blacksquare, \circ, \otimes$	QA UI, menu, board games, tele-info.
Five-in-a-line	round robin	$\square, \blacksquare, \circ, \otimes$	as above.
Connect-4	round robin	$\square, \blacksquare, \circ, \otimes$	online form filling.
Three Stones	round robin	$\square, \blacksquare, \circ, \otimes$	as above.
Hasty Battle	contention	\square, \circ, \otimes	shared database access, printing queuing.
Vicious Battle	contention	\square, \circ, \otimes	shared whiteboard, co-authoring.
Gentlemen's Battle	tapping	$\square, \blacksquare, \circ, \otimes$	discussion forum.
Dictator's Entertainment	master	$\square, \blacksquare, \circ, \otimes$	e-learning class and testing, online meeting.
First-come, first served	reservation	$\square, \blacksquare, \circ, \otimes$	web camera control.
Opportunity Knocks	round robin	$\square, \blacksquare, \circ, \otimes, \boxtimes, \boxtimes$	emotional IQ test.
Secret Switch	round robin	$\square, \blacksquare, \circ, \otimes$	distributed database access.
Group games	round robin & group protocol	$\square, \blacksquare, \circ, \otimes$	group work and team games.

Table 1. Variations of the noughts and crosses game, and their main features

protocol features in turn control and domain control, and linking them with real life multimedia collaborative applications. For example, the shared whiteboard, on-line discussions, as well as group work and team games were already implemented in JACIE [8] using the small set of interaction protocols as built-in language constructs. With the new design of the language constructs, we have implemented all the imaginary versions of the noughts and crosses games. Therefore, JACIE may be applied to design almost all of the multimedia applications listed in the table.

3.1. Definitions

A *generalised game board* for noughts and crosses is defined over a grid of $N_x \times N_y$ cells. Each cell may exhibit one of the three basic visual states; *empty*, \circ , or \times . Cells may be in two different modes indicating whether or not the visual state of the cell is modifiable. There are six valid states for cells $\{\square, \blacksquare, \circ, \otimes, \boxtimes, \boxtimes\}$. Formally, the symbols are elements of the set $OX \times Mod$ where $OX = \{empty, \circ, \times\}$ and $Mod = \{mod, unmod\}$, e.g., $\langle \circ, mod \rangle$ represents \circ .

At time $t \in T = [t_{start}, \infty)$ in a distributed, collaborative environment of $K + 1$ processors, $p_0, p_1, p_2, \dots, p_K$, each processor has access to a board $G_k(t)$, a local version of a primary board $G_{prim}(t)$. G is a function of type $G : T \times \mathbb{K}_0 \times \mathbb{N}_x \times \mathbb{N}_y \rightarrow OX \times Mod$, where $\mathbb{K}_0 = \{prim, 0, 1, \dots, K\}$, $\mathbb{N}_x = \{1, \dots, N_x\}$ and $\mathbb{N}_y = \{1, \dots, N_y\}$. $G_k(t)(i, j)$ denotes $G(t, k, i, j)$ so each $G_k(t) : \mathbb{N}_x \times \mathbb{N}_y \rightarrow OX \times Mod$. In such an environment, it is necessary to have correctly designed and implemented interaction protocols to ensure a consistent state transition of $G_{prim}(t)$ and $G_k(t)$ for each k .

At time t , p_k can attempt to perform an action by trying to assign a \circ or \times in a cell. The actions are described by the function $act : T \times \mathbb{K}_0 \times \mathbb{N}_x \times \mathbb{N}_y \rightarrow OX \cup \{no_op\}$. no_op is used to indicate that no attempt is being made to change the state of a cell. A curried form of G encapsulat-

ing all the boards at time t , is of type

$$\mathcal{G} : T \rightarrow \mathbb{K}_0 \times \mathbb{N}_x \times \mathbb{N}_y \rightarrow OX \times Mod$$

where $\mathcal{G}(t)(k, i, j) = G(t, k, i, j)$. An *interaction protocol* is a temporal function, ψ , that computes all states $\mathcal{G}(t)$ as: $\mathcal{G}(t) = \psi(t, A(t_{start}, t), \mathcal{G}(t_{start}))$ where $\mathcal{G}(t_{start})$ represents the initial states of the game board on different processors and $A(t_a, t_b)$ is the graph of *act* with t restricted to $[t_a, t_b)$.

It is not feasible to maintain a continuous change of $\mathcal{G}(t)$, so $G_k(t)$ is updated in a discrete manner as $G_k(t_1), G_k(t_2), \dots, G_k(t_s), G_k(t_{s+1}), \dots$, where $t_{start} < t_1 < t_2 < \dots < t_s < t_{s+1} < \dots$. Ideally, one would like to simplify ψ to a function that operates on a discrete time series, $\dots < t_s < t_{s+1} < \dots$, as $\mathcal{G}(t_{s+1}) = \psi(t_{s+1}, A(t_s, t_{s+1}), \mathcal{G}(t_s))$. However, the time series operating at each processor is based on its local events (including clock, interaction and communication events), hence is not synchronised with that of other processors. This poses the major challenge to the design and implementation of any interaction protocols for net-centric collaborations.

In a distributed collaborative environment, the implementation of ψ has to be realised using a set of concurrent sub-functions, ψ_a, ψ_b, \dots , which operate in different processors in a distributed and co-ordinated manner. In the following, we assume a client-server model, with one server p_0 , and K clients p_1, p_2, \dots, p_K , only one user (player) at each client, and only one sub-function ψ_k at each processor.

3.2. Variations of the Noughts and Crosses

As in Table 1, Five-in-a-line, Connect-4 and Three Stones are very similar to noughts and crosses but with different number of cells, rules and size of board game or even board shapes. From the perspective of interaction management, Connect-4 introduces the notion of “unmodi-

fiable” cells, \blacksquare , as illustrated in Figure 1. The mechanism for changing the state of an empty cell from \blacksquare to \square is also interesting. The rest of the games in the table are imaginary variations of the noughts and crosses with different rules. Therefore, the games require different turn control and some differences in the cell states.

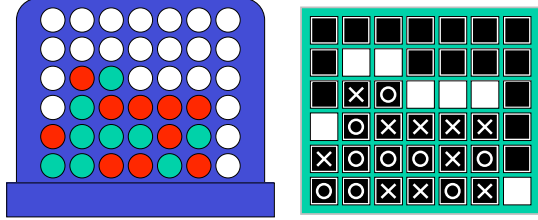


Figure 1. Connect-4 Board and its Representation

4. Interaction Protocols

Due to space limitations, we give only one example of the syntactic specification of the language construct protocol `master` described in Section 4.3. Other protocols use the same syntactic structure as `master`.

protocol master [server] [random user]
protocol master [server] [random user] [turn δ_{turn}][overall $\delta_{overall}$] [silence $\delta_{silence}$]
protocol master user [η_{client}]
turn set client η_{client}

In JACIE most arguments (or extensions) of a protocol are optional, facilitating “fast scripting” for simple and commonly-used protocols, and the extensibility when introducing new variations and extensions. The timer option `turn`, which is governed by a variable δ_{turn} , restricts the length of time during the turn control. `overall`, which is activated in the session start, can be used to restrict each client to have a fixed amount of total time for holding turns. `silence`, represents the maximum amount of time to pass for starting an action in order not to lose a turn. All the timer options are managed by each client sub-function ψ_k , and have the default value ∞ . All the protocols have these three timer options.

4.1. Round Robin

In a basic round robin protocol, only one user, at most, is authorised to alter the states of the game board at any time during a game. The protocol management function ψ resides mainly at the server p_0 as ψ_0 . The order of clients being activated is organised according to the order of their registration with the server, and is on the first-come, first

served basis in a circular manner. For efficiency reasons, JACIE facilitates an additional sub-function ψ_k at each client p_k , which validates each action $act(t, k, i, j)$ against the current $G_k(t)$.

4.2. Contention

Interactions under a *contention* protocol is perhaps considered as almost unmanaged since all clients are authorised, at almost any time during a game, to alter the game board, though it does not guarantee the success of every action. The protocol management function ψ almost totally resides at the server. Each user action is first validated by the client sub-function ψ_k against the current $G_k(t)$. It is then forwarded to the server, entering at the end of an action queue. As it is possible that actions from different clients may not be compatible with each other because of concurrent activities, all actions need to be reexamined at the server. As long as the queue is not empty, the server sub-function ψ_0 continues to fetch actions one by one from the head of the queue. Each action is revalidated against the current game board state, $G_0(t_s)$, stored at the server. If the action is invalid, it is simply discarded. Another timer option called the `rest timer` is introduced to ease the pressure on protocol management by restricting each client to “rest” for a small period between two consecutive actions.

4.3. Master

For this protocol, the `master`, whose job is to determine the turn control, has all the power to set the turn. The default master is the server. The random choice indicates that the server determines the order of the user turn at no specific order. The other choice, the `protocol master user` allows one of the users to become the master and then the `usermaster` can determine the turn protocol. Below is an example of a code fragment in JACIE.

```
JACIE {
  applet name ncDictator;
  configuration { ...
    number of users 2;
    protocol master random;
  } ...
  client implementation {
    declaration { ... } ...
    on session { ...
      on MOUSECLICK { ...
        // process an action
        pass turn; } ...
    } ...
  }
  server implementation {
    declaration { ... } ...
    on session { ...
      on TURN { ...
        // process an action } ...
    } ...
  }
}
```

To change the protocol, it is only necessary to change one line of code in the configuration section.

4.4. Reservation

This protocol allows clients to make their requests to the server for their turns in a round robin fashion and all the requests are put into a queue. The server checks the contents of the request queue and if not empty, the turn control will follow the order of the requests. After all the requests are fulfilled, another set of requests is asked from all the clients.

4.5. Tapping

In `tapping`, the order of the turn is determined by the client who currently gets the turn control. It is different from the protocol `master user` since in this protocol, every user is the master instead of just one master for all.

4.6. Group

The group protocol is used for selecting a member to represent the group that has the turn control. All groups can be assigned the same protocol or it is also possible to have different protocols for different groups. The following are the group protocol options: (a) **group userdefined** – In JACIE, there is a set of built-in channels such as canvas channel, message channel, chat channel and video channel. Since all the group members will have the turn control, they can communicate through one of these channels to do the selection. (b) **group round robin** – Each member of a group has an equal opportunity to represent the group in circular manner starting from the first member. (c) **group random** – For the group in control, the server selects one of the members at random. (d) **group master** – The same user represents the group as the group master for every turn. The master can be set by the JACIE provided statement, or by default is the first member.

5. Conclusions

We have built our technical discussions around variations of noughts and crosses, which serve as an “abstract” collection of multimedia collaborative applications. This enables us to focus on the interaction management, rather than the context-specific details in the applications. We have presented a set of formal notations for modelling the spatio-temporal activities in noughts and crosses. Based on the model, we have developed a comprehensive collection of interaction protocols that are capable of addressing the protocol needs in all variations of the noughts and crosses game described, and thereby the related applications. Our main

contribution in this respect is the adventurous attempt to provide language constructs for specifying a variety of interaction protocols. These have been incorporated into JACIE, a scripting language designed for prototyping collaborative applications.

References

- [1] P. Bellini and P. Nesi. Communicating TILCO: a model for real-time system specification. In *Seventh IEEE International Conference on Engineering of Complex Computer Systems*, pages 4–14, Skovde, Sweden, June 2001. IEEE.
- [2] G. W. Daniel and M. Chen. Interaction control protocols for distributed multiuser multicamera environments. In N. Callaos, A. D. Sciallo, T. Ohta, and T. Liu, editors, *Proc. of 7th World Multiconference on Systemic, Cybernetics and Informatics, SCI2003*, volume 1, pages 448–453. International Institute of Informatics and Systemics, July 2003.
- [3] Y. Demazeau, O. Boissier, and J. L. Koning. Using interaction protocols to control vision systems. In *International Conference on Systems, Man, and Cybernetics*, volume 2, pages 1616–1621. IEEE, Oct 1994.
- [4] H. Dommel and J. Garcia-Luna-Aceves. A novel group coordination protocol for collaborative multimedia systems. In *International Conference on Systems, Man, and Cybernetics*, volume 2, pages 1225–1230. IEEE, October 1998.
- [5] H.-P. Dommel and J. Garcia-Luna-Aceves. Group coordination support for synchronous internet collaboration. *IEEE Internet Computing*, 3(2):74–80, April 1999.
- [6] P. Garcia, O. Montala, C. Pairo, R. Rallo, and A. G. Skarmeta. Move: Component groupware foundations for collaborative virtual environments. In *COLLABORATIVE VIRTUAL ENVIRONMENTS*, pages 55–62. ACM Society, September 2002.
- [7] A. S. Haji-Ismail. *JACIE - A Scripting language for internet-based multimedia collaborative applications*. PhD Thesis, Department of Computer Science, University of Wales Swansea, UK, 2001.
- [8] A. S. Haji-Ismail, M. Chen, P. W. Grant, and M. Kiddell. JACIE - an authoring language for rapid prototyping network-centric, multimedia and collaborative applications. *Annals of Software Engineering*, 12:47–75, December 2001.
- [9] J. B. Joshi, E. Bertino, and A. Ghafoor. Temporal hierarchies and inheritance semantics for GTRBAC. In *Proceedings of the Seventh ACM symposium on access control models and techniques*, pages 74–83. ACM, June 2002.
- [10] D. Lee, M. Lim, and S. Han. ATLAS: a scalable network framework for distributed virtual environments. In *ACM Conference on COLLABORATIVE VIRTUAL ENVIRONMENTS*, pages 47–54. ACM, September 2002.
- [11] C. S. Pinhanes and A. F. Bobick. Interval scripts: a programming paradigm for interactive environments and agents. *Personal and Ubiquitous Computing*, 7(1):1–21, May 2003.
- [12] R. Qiu, F. Kuhns, and J. R. Cox. A conference control protocol for highly interactive videoconferencing. In *Global Telecommunications Conference, GLOBECOM '02*, volume 2, pages 2021–2025. IEEE, Nov 2002.