Next Steps for Mobile Entertainment Portals

Mobile entertainment portals are already appearing, but game designers lack tools for designing content and services. One approach is to use argumentation for hypothesizing the consequences of actions and interactions in a game world.

T.S. Raghu
Arizona State University

R. Ramesh
State University of New York at Buffalo

Andrew B. Whinston
University of Texas at Austin

Mobile communications and integrated data and voice services are ushering in a new era of personal, portable, and pervasive computing, in which entertainment services will figure prominently. A September 2000 Datamonitor study predicted that the mobile entertainment industry revenue will increase to $6 billion by 2005.[1]

Broadband wireless technologies are already enabling the development of intricate, multimedia-enabled gaming content. Several wireless entertainment companies—Codeonline, nGame, and Indiqu—offer mobile games that adopt simple text- and turn-based board strategies, such as those in *Trivial Pursuit* and tic-tac-toe. Role-playing online games like *EverQuest* (http://everquest.station.sony.com), as well as MUDs (multiuser dungeons) and MUSEs (multiuser simulated environments),[2] have become extremely popular. These games allow both one-to-one and one-to-many interactions, which often fosters community building that in turn helps portals retain players for a longer term.

Although mobile entertainment offers unprecedented flexibility, it also poses formidable challenges. Wireless games are constrained by the unpredictability and length of player interactions, traffic volume, uptime, and computational bottlenecks. As with other e-commerce applications, wireless games must be reliable, scalable, and secure. Guaranteeing correct processing under all anticipated and unexpected scenarios is a particularly thorny problem.[3]

Perhaps the greatest challenge is how to provide services and content rapidly while also satisfying continually changing customer requirements. Given the community orientation in entertainment portals, it is also natural to expect conflicts among players, designers, and game administrators.

To meet these challenges, designers must have tools for analyzing game design and verifying design objectives. Just as software engineering tools revolutionized information systems design, tools that aid design analysis and verification are likely to transform wireless entertainment portals.

DESIGN DIFFERENCES

The issues in mobile multiplayer games are distinctly different from traditional system design concerns, however. Multiplayer games elicit specific actions from players, and those actions alter the state of game-world entities and their behavior. Because players' knowledge of the game world is often incomplete, their actions and interactions could be indeterminate. Consequently, most online and wireless games are stochastic, imperfect information scenarios. The concurrency and sequence dependence in player actions could yield highly unstructured interactions. Design theory in disciplines such as artificial intelligence, in contrast, has long focused on perfect information games, which are more straightforward to model.

In traditional game design, such as for computer chess or Go, the focus has not been on developing a social business model. In contrast, *EverQuest* clearly focuses on interactions among players as an integral part of its business model. Therefore, multiplayer games must depend heavily on effective cus-

0018-9162/02/$17.00 © 2002 IEEE

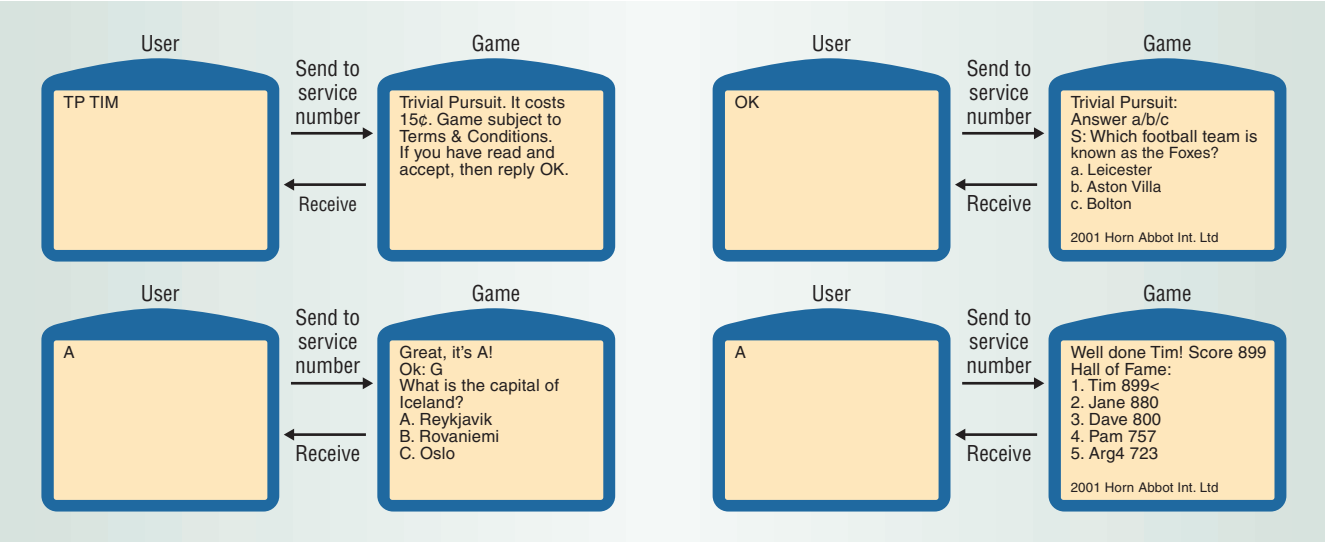

Figure 1. Codeonline's wireless* Trivial Pursuit*, which uses turn-based message exchanges. Players answer trivia questions to earn money.

tomer relationship management (CRM). Loyal gaming communities evolve from portals that can guarantee timely game closure, allocate game outcomes to players in a quantifiable manner, and avoid game states that create deadlocks. To meet these requirements, the wireless gaming industry uses several unique mechanisms such as nondeterministic payoffs, script-based moves for countering opponent moves while offline, and role playing.

Although these techniques make game design complex, they also produce game worlds with unique attributes for developing game-design tools. For example, games may allow only a finite set of actions, which means it is easier to formally describe the effects of those actions. Further, a mapping of game moves over program components could be a foundation for component-based game specification, which in turn would let designers reason about hypothetical player actions in logically verifiable terms.

This mapping serves as the basis for analyzing both the intended and unintended consequences of game content. Designers can use common-sense reasoning to hypothesize the consequences of actions and interactions in a game world. The reasoning may involve symbolic, probabilistic, and quantitative formalisms. The design process involves collaborative decision making, in which game content evolves from either formal or informal deliberations among game designers on design options. A model based on this strategy could lead to tools for the rapid design, development, and deployment of wireless games.

INTERACTION TYPES

Central to game design is the interactivity among players and portal services. Interactions can be either synchronous, as in a chat room, or asynchronous, as in some multiplayer networked games. In the wireless arena, network latency and unsynchronized system clocks can be restrictive, since action-oriented multiplayer games require real-time responses and clock synchronizations. As a result, most widely available entertainment services are based on synchronous interactions and turn-based games, such as tic-tac-toe.

Asynchronous interactions are becoming more popular, however, and in some games, such as nGame's *Alien Fish Exchange*, players can advance game states without engaging other players in real time. In this game, players breed, feed, and care for fish using asynchronous message exchange. A much simpler example of turn-based or server-mediated simultaneous action games is *Trivial Pursuit*, in which players answer trivia questions and earn points or money. Figure 1 shows Codeonline's wireless version of this game. Games like nGame's combat-oriented *Carrier Force* involve simultaneous actions, where the game server mediates between players.

Multiplayer networked games like these succeed in part because communities of like-minded people interact on a regular basis both inside and outside game sessions. Quality of service is a primary concern in managing multiple game sessions with many players. In the wireless domain, handheld devices still have limited capabilities, so game servers handle most game-related workflows.

MULTIPLAYER PORTAL ARCHITECTURE

An entertainment portal will require a scalable distributed architecture for providing full-blown entertainment services to consumers. The architecture should be robust and

- handle transactions and interactions with several thousand customers at a time,
- provide secure billing and payments services,
- allocate and manage internal resources among the various workflows,
- provide customers with a consistent interface and behavior regardless of the communication and computing platforms used,

- provide effective information collection mechanisms for efficient CRM and targeted marketing and advertising campaigns, and
- provide flexible and consistent interfaces to content developers so that they can quickly integrate new services.

As Figure 2 shows, game portals consist of the game world and game setup environment. The game world represents the contents of various games and their internal management. The gaming setup environment provides game setup, player setup, and portal management services.

Game setup services

Providing reliable and fault-tolerant wireless gaming environments is a major challenge. Until 3G technologies and more powerful handheld devices become commonplace, portals must contend with limited power on the client side and high latency in interactions. This clearly puts the onus on portal servers to manage all game sessions and track game progression. Hence, portal scalability is critical.

Distributed server architecture with a central proxy server is a logical solution to the scalability problem. As available bandwidth improves and mobile client devices become more powerful, greater client processing, animation, and real-time interactions will become feasible. These capabilities will impose additional requirements on portal and game content design. To provide an acceptable quality-of-service level, content designers will have to use communication protocols (broadcast, multicast, peer-to-peer) appropriately. For mobile users, designers could also use geographical information and connection speed in determining which server to assign.

Player setup services

Figure 3 shows the components of player setup services—the set of functions that collectively manage the user's experience during a session. To a large extent, the user's experience should be adaptive and respond adequately to changes in context, geographic location, and playing preferences. Player intent (type of game interested in, time in the session, and so on), availability of other players and their profiles (including skill levels), and connection characteristics collectively determine the context for an incoming player. Player setup services would use this information to provide matching services and set up game sessions. Other player setup services include the ability to resume previously stopped game sessions and conduct and manage tournaments.

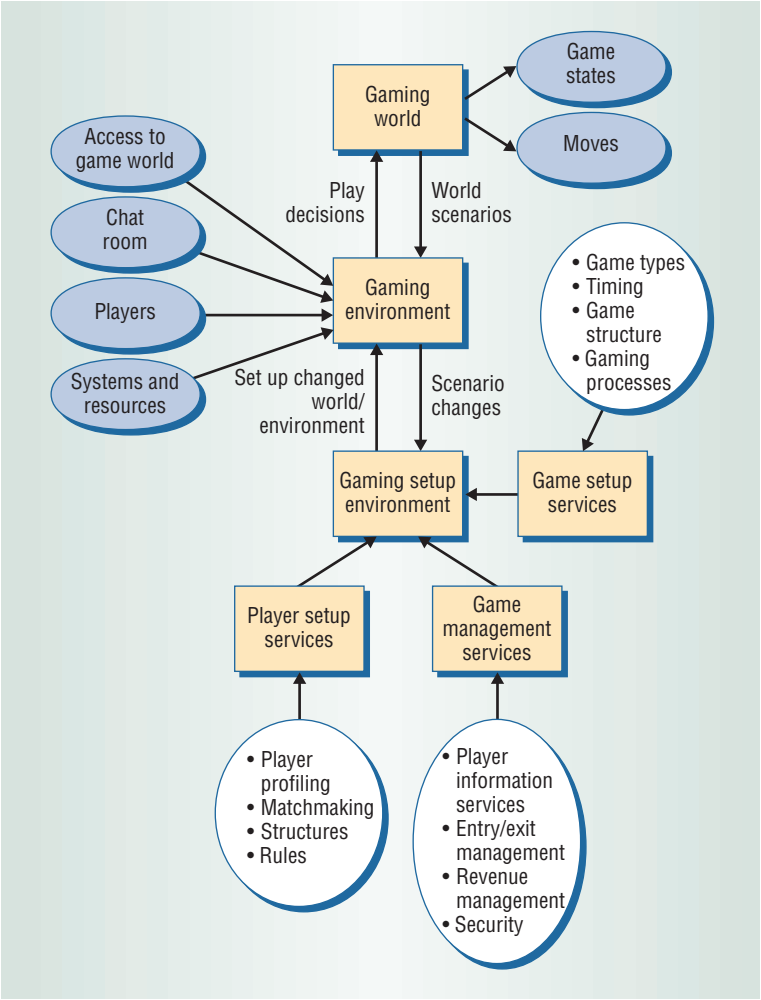

Figure 2. A conceptual model of an entertainment portal. The game world represents the contents of various games and their internal management. The gaming setup environment provides game setup, player setup, and portal management services.

Portal management services

As Figure 4 shows, an entertainment portal embodies the idea of electronic communities that interact by discussing, participating, and competing in games of mutual interest. Portals can augment information gathered during interactions with users' demographic, contextual, and geographical information. Content providers, carriers, and advertisers can share this rich set of information within the constraints of the users' desired privacy levels.

The e-community concept and the advertiser-supported model are already in vogue in wireless gaming. Codeonline, for example, builds tailored online games for advertising campaigns. While a community-based business model is stereotypical in online entertainment, designers could also augment such a model by combining advertising, infomediary, subscriptions, and pay-per-use models for revenue generation and business sustenance that exploit the information flows and customer behavior.

MULTIPLAYER GAMING

Players in network games interact with complex worlds constructed from simple object types such as rooms, exits, or representations of themselves. In addition to object attributes, designers commonly

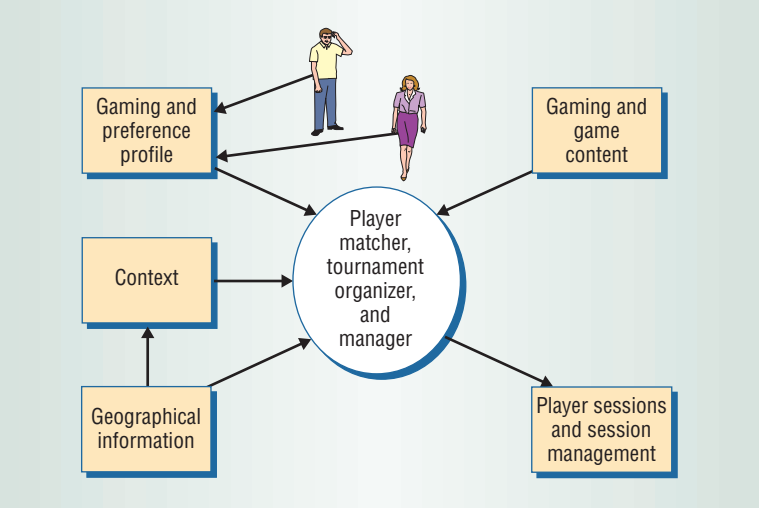

Figure 3. Player setup services in an entertainment portal. These services collectively manage a user's experience during a session.

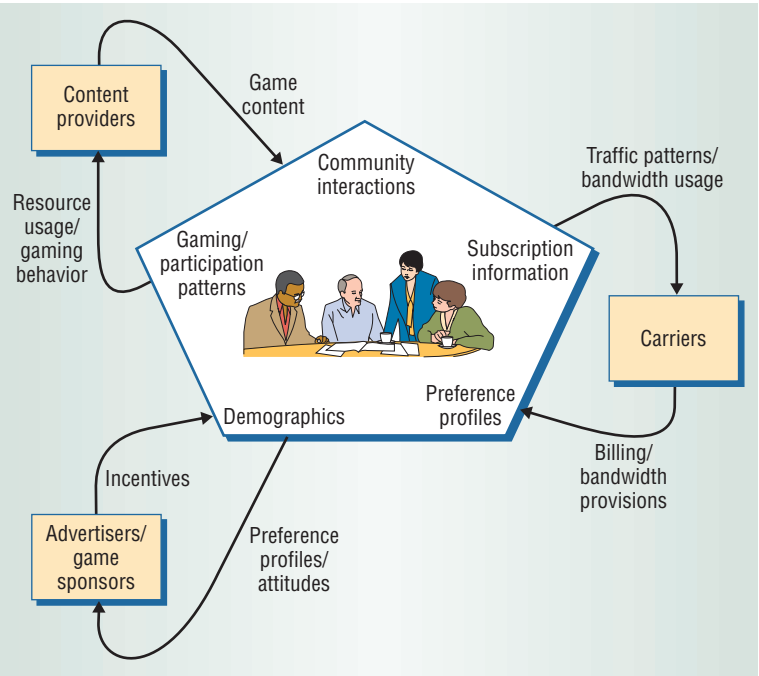

Figure 4. Information flows in entertainment portals. Electronic communities use the portal to interact by discussing, participating, and competing in games of mutual interest.

specify programmable object behaviors in these games. Player representations, or *avatars*, are projections of the human players onto the virtual world. The avatars can have possessions—money (tokens), weapons, antiques, and other game objects. Since avatars are persistent, players learn to interact with other players, explore, and create a virtual world. The games are deliberately open-ended and can evolve completely from the players' actions. For example, in the popular television game, *The Weakest Link*, players can vote during the game to eliminate another player, causing conflicts among players, designers, and game administrators.

In *MicroMUSE*,[4] players collaborate to build a virtual world featuring adventures and puzzles that combine recreational and educational content. In one incident, a participant in the game began harassing another player by spamming her and then completely destroying all that she owned in the game. Eventually game managers detected the offending player and disconnected him from the system. In another instance, a conflict arose between a player and an administrator who removed the player's avatars and properties from the game database because the player had made frivolous announcements. Other players protested, however, and the administrator eventually reinstated the player.

Such conflicts commonly involve mediation, power relations, fact finding, and arbitration. This strongly implies the need for a pragmatic approach to game content design—an approach characterized by extensive deliberative processes.

On the one hand, change is an integral aspect of entertainment portals, and it sustains customer interest. On the other, change can create unforeseen consequences.[5] Clearly, complex game worlds are characterized by unpredictable player behavior, information asymmetries among players, nondeterministic outcomes, and concurrency of plays.

Because of these characteristics, multiplayer game design must be a collaborative process, in which design decisions evolve from deliberations in design groups. The deliberations center on the possible consequences of game states and moves and might involve both strict reasoning, based purely on logic, and defeasible reasoning[4] that is based on beliefs and thus subject to argument.

SUPPORTING ARGUMENTATION-BASED DESIGN

To support these deliberations, designers will need some tool that aids logical validations and evidential and argumentative reasoning. Such a tool should include a language with expressive power to describe game entities, their behavior, states, and actions.

Figure 5 shows how we envision designers' evolving a game world through argumentation. Deliberations about game design can include assertions about hypothesized player actions that shape the game world's events and resources (proposed game content). Designers can support or oppose these assertions, leading to argumentation, which follows a set of dialectic rules. Eventually, designers accept certain design components and reject or alter others. The resolution of argumentation leads

to the final game architecture and the consequential executable code (translator).

Our vision is based on the notion of computational support for collaborative decision making[6] in argument assessment and analysis, resolution, and facilitation of deliberations. This approach recommends partial formalization augmented by parametric and interactive analysis of game content.

Most problems that arise in multiplayer games are unintended consequences of the game design itself. Although an argumentation-based approach does not uncover all possibilities in every instance, it does provide a systematic forum through which designers can carefully discuss and analyze possible game states. The "Representing Argumentation in Game Domains" sidebar describes an application of argumentation for a two-player game, in which designers use an action description language to reason about game content.

The resolution of the differences in defeasible reasoning hinges on strengthening the support base or persuasive presentation. Therefore, an analytical approach to assessing design changes to game content is at best heuristic.

An alternative to structural representation formalisms uses a framework to resolve defeasible logic.[3,6,7] Most of these approaches build an argumentation network from the dialectics so that it is easier to slot arguments into accepted and rejected categories.

A more recent connectionist approach that extracts the behavior of argument structures[6] moves toward a system of argument analysis. In this analysis, designers are not necessarily constrained to resolving argumentation to discrete categories such as accept and reject. Rather, the goal is to establish a continuum of argument strength using a connectionist rendition of argument networks. Designers are thus guided and enabled by assessments of their propositions' strengths and weaknesses relative to those of other designers. With this level of computational support by a deductive engine, designers can more easily arrive at a compromise or consensus in analyzing large, complex argument networks.

The next steps for making mobile entertainment portals a solid reality are to derive integrated frameworks for wireless game design and to define corresponding business models. Our argumentation-based philosophy for analyzing game-content design complements recent efforts in mobile environment design that are based on contextual inquiry and scenario-based validation techniques.

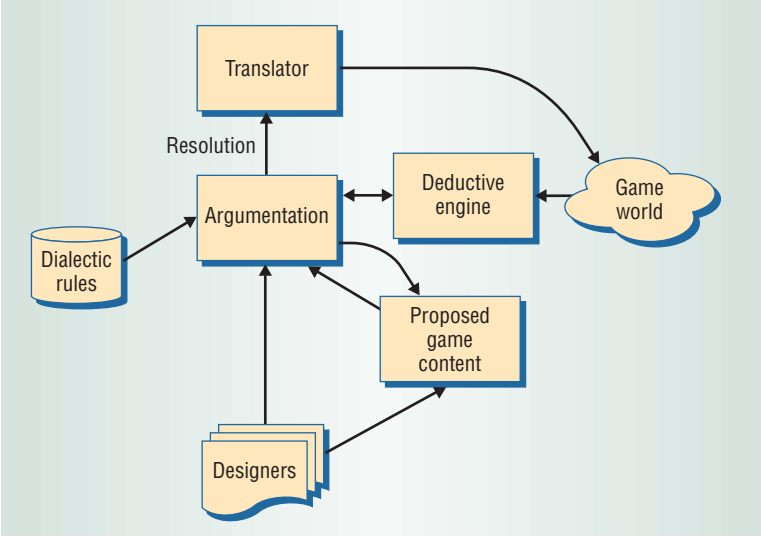

Figure 5. How a tool might support the argumentation-based evolution of a game world. The tool's backbone is a deductive nonmonotonic reasoning engine that facilitates dialectic analysis.

Our future work will focus on analyzing the more complex sociological phenomena that arise in multiplayer games—player community interactions, the creation and management of political entities and virtual societies, and their effect on game play, for example.

Several challenges remain in implementing a tool that supports argumentation-based design. We still need efficient computing and knowledge-representation mechanisms to capture domain-specific information. During argumentation, designers must be able to efficiently parse and execute queries about hypothetical player actions or game events.

Integrating the domain-specific and domain-independent components of the argumentation language is also a complex issue. Given the specialized nature of the action-oriented entertainment world, the development of high-level languages that are close to natural language with a restricted grammar is a plausible approach.

Regardless of the work remaining, however, we believe that design support tools based on argumentation and dialectic reasoning mechanisms have potential merit in augmenting traditional formal approaches to entertainment systems analysis and verification. ■

References

1. W. Knight, "Wireless: Game Explosion Forecast," ZD Net (UK), 5 Sept. 2000, http//www.zdnet.com/2100-11-523627.html.
2. H. Rheingold, *The Virtual Community: Homesteading on the Electronic Frontier*, Addison Wesley, New York, 1993.
3. R. Ramesh and A.B. Whinston, "Claims, Arguments, and Decisions: Formalisms for Representation, Gaming and Coordination," *Information Systems Research*, vol. 5, no. 3, 1994, pp. 294-325.

Game designers can use several representation formalisms to depict the game domain. Using an action description language,[1] for example, they can define the game domain as a tuple $\Gamma = <F, M, L>$, where F represents a set of *fluents*, M represents the set of *actions*, and L represents the set of causal laws. Fluents are propositions whose truth-values depend on the game state, and they can also be conditional or atomic. Actions represent possible player moves or game-world actions. The effect of actions depends on the game state and concurrencies or sequences in the action's occurrence. Designers define these effects using causal laws.

Dialectic reasoning in the form of logic sentences in an action language can be captured in a graphical user interface and translated into logic programming language statements.[2]

Consider a simple two-player game, in which each player begins with 100 tokens and can perform two actions, *shoot* and *jump*. The game begins when both players simultaneously choose one of the actions and convey it to the game server. Once the game server receives both players' moves, it evaluates them and updates the game state in each player's wireless device.

Figure A defines the domain for this game. The fluents and causal laws in Figure A initiate the design debate process. Assume that two designers (desgn1 and desgn2) reason about the game as in Figure B.

To some extent, designers can model games of this nature using game-theoretic methods. For example, a designer could construct the payoff matrix for two players and use that matrix to specify the game. However, when there are many player moves, more than two players in the game, or moves that lead to nondeterministic payoffs, specifying the payoff matrix is extremely difficult.

The linguistic, declarative approach is more closely connected with game implementation. Consequently, the resulting specifications may make it possible to build tools that generate game trees. Designers could then use these tools to investigate possible deadlock scenarios or repeated states. They could, for example, decide to limit game length by using a fluent of form

```
F5: Win(p1) if p1.tokens >
  p2.tokens at time(N)
```

where N is the limit on the number of moves in the game.

Of course, such finite horizons bring with it the usual problem that the game may end earlier than N because players gain exact knowledge of the payoffs. Designers may have to introduce nondeterminacy in the payoffs to the players to prevent such states. We can specify nondeterministic causal laws for the game in Figures A and B using these statements:

```
L6: {Jump(p1), jump(p2)}
  causes lucky(p1, p2)
  % Causes the fluent "lucky"
  to become true
F6: p1.tokens -=20 |
  p2.tokens -= 20 if lucky
  (p1, p2)
 % Nondeterministic effect
  of the fluent "lucky"
```

Once they implement game design as a program, designers face a host of other issues that stem from the distributed nature of computing and latency and errors in the communication medium. Designers can handle verifications of this nature by using either formal program verification tools such as Verisoft and Spin or dynamic program verification methods.[3]

References

1. E. Giunchiglia et al., "Nonomonotonic Causal Theories," http://www.cs.utexas.edu/users/v1/papers/nmct.ps, current Apr. 2002.
2. C. Kakas, R.S. Miller, and F. Toni, "E-RES—A System for Reasoning about Actions, Events, and Observations," *Proc. 8th Int'l Symp. Nonmonotonic Reasoning*, Routledge Press, London, 2000, pp. 134-165.
3. W. Wang et al., "E-Process Design and Assurance Using Model Checking," *Computer*, Oct. 2000, pp. 48-53.


```
Γ = {
 F = {
 F1: ¬Dead(P) at time(0),
 F2: P.tokens = 100 at time(0),
        %Each player has 100 tokens at the beginning of the
          game
 F3: Dead(P) if P.tokens <= 0,
          %Player is dead if he/she does not possess any
            tokens
 F4: Win(p1) if Dead(p2),
          %Player wins if the other player is dead.
 }

 M = { M1: Jump, M2: Shoot }

 L = {
 L1: Shoot(p1, p2) causes Dead(p2),
          %Explains individual consequence of a player
            action
 L2: {Shoot(p1, p2), Jump(p2)} causes p1.tokens -= 20,
          %Concurrent choice of shoot and jump causes
          % the player who shoots to lose 20 tokens.
 }
}
```

Figure A. Defining the sets in a game domain (Γ). Boldface words indicate domain-independent language constructs, which have commonly accepted meanings (such as* causes, at, if, time*). P stands for any player, p for a specific instance of a player. Words in regular fonts indicate domain-specific fluents or player actions within the game. Action or action sequences enclosed in [ ] indicate sequential ordering; in { } they indicate concurrency.

```
Desgn1:  A1: (S0: Win(P) after time(n))
         %Asserts a design goal that some player should win after an arbitrary number of moves.

Desgn2:  A2a: Execute [        % Execute instructs the game engine to simulate the moves.
           α1: Create p1, p2   %Create two players.
           α2: {Shoot(p1, p2), Shoot(p2, p1)}   % Both players shoot.
           ]

         A2: [α1, α2] ↑ (S1: Dead(p1) ∧ Dead(p2))  % This follows from the causal laws of the game.
         A3: A2 ⇓ A1          %This defeasibly defeats the design objective.

Desgn1:  A3a: L3: {Shoot(p1, p2), Shoot(p2, p1)} causes φ
                % A new more specific causal law overrides L2 when
                % both players shoot.
A4: Γ↓ A2       % This strictly defeats the previous argument.

Desgn2:  A5a: [α1, α2, α2] ↑ (S2: (P.tokens at time(1) = P.tokens at time(2)))
                % A verifiable statement indicating that the sequence of actions
                % does not alter the number of tokens.
A5: A5a ⇓ A1    % The deadlock in the game may defeat the game objective.

Desgn1:  A6a: L3: {Shoot(p1, p2), Shoot(p2, p1)} causes p1.tokens -= 10 ∧ p2.tokens -= 10
                % This overrides the previous causal law.

A6: Γ↓ A5a      % This strictly defeats the previous argument.

Desgn2:  A7: ?{Jump(p1, p2), Jump(p2, p1)}
                % Designer2 brings into focus the incomplete specification of
                % the game when both players choose to jump.

Desgn1:  A8: L4: {Jump(p1), Jump(p2)} causes p1.tokens -= 10 ∧ p2.tokens -= 10
                % A new causal law to evaluate two concurrent jumps.

Desgn2:  A9a: Γ ⇑ (S3: {Jump(p1), Jump(p2)})
                % Designer 2 analyzes the causal laws and points out that the dominant
                % strategy for both the players is to jump.
                % Alternately, a deductive engine could analyze the payoff
                % implications to indicate possible dominant strategies.

A10: A9a ⇓ A1   %This defeasibly defeats the design objective.

Desgn1:  A11: L1: Shoot(p1, p2) causes p2.tokens -=20 ^ p1.tokens +=20
                % Designer 1 redefines causal law 1.
           M3: Pass
           L5: Pass(P) causes P.tokens += 10
                % A new move is introduced and causal laws surrounding this move are defined.
           L6: {Pass(p1), Pass(p2)} causes p1.tokens -= 10 ∧ p2.tokens -= 10
       Γ ⇓ (S3: {Jump(p1), Jump(p2)})
```

Figure B. Two designers reason about game strategy using the domain specification defined in Figure A. S_i denotes defeasible statements. A_i denotes assertions, and α_i denotes an action or action sequence. The debate ends when Designer 1 redefines causal law 1 (L_1 in Figure A) and the designers agree to allow mixed strategies from the players.

4. A. Smith, "Problems of Conflict Management in Virtual Communities," *Communities in Cyberspace*, P. Kollock and M. Smith, eds., Routledge Press, London, 1998, pp. 189-211.
5. J. Dibbell, "MUD Money: A Talk on Virtual Value and, Incidentally, the Value of the Virtual," Apr. 1995; http://www.levity.com/julian/mudmoney.html.
6. T.S. Raghu et al., "Collaborative Decision-Making: A Connectionist Paradigm for Dialectic Support," *Information Systems Research*, vol. 12, no. 4, 2001, pp. 363-383.
7. G. Vreeswijk, "Reasoning with Defeasible Arguments: Example and Applications," *Proc. European Workshop Logic in AI*, Springer-Verlag, Berlin, 1992, pp. 189-211.

T.S. Raghu *is an assistant professor of information systems at Arizona State University. His research interests include e-commerce, collaborative decision making, and business process modeling. Raghu received a PhD in management information systems from the State University of New York at Buffalo. Contact him at Raghu.Santanam@asu.edu.*

R. Ramesh *is a professor of management science and systems in the School of Management at the State University of New York at Buffalo. His research interests include online gaming communities, design of economic mechanisms for coordination in virtual organizations, ontologies for multiagent architectures, and collaborative decision making. He also serves as an editor in chief of* Information Systems Frontiers *(Kluwer). Ramesh received a PhD in industrial engineering from SUNY at Buffalo. Contact him at rramesh@acsu.buffalo.edu*

Andrew B. Whinston *is a professor of information systems, economics, and computer science; the Hugh Roy Cullen Centennial Chair in business administration; and the director of the Center for Research in Electronic Commerce at the University of Texas at Austin. His research interests are all aspects of e-commerce, including resource allocation, bundle markets, trust, assurance, and market design. Whinston received a PhD in management from Carnegie Mellon University. Contact him at abw@uts.cc.utexas.edu.*