

# A Computer Participant in Musical Improvisation

William F. Walker

Apple Research Labs

1 Infinite Loop, MS: 301-4B

Cupertino, CA 95014 USA

+1 408 974 0187

walker@apple.com

[http://www.research.apple.com/personal/Bill\\_Walker/](http://www.research.apple.com/personal/Bill_Walker/)

## ABSTRACT

Musical improvisation is a collaborative activity analogous to conversation. Both are sequences of spontaneous utterances constructed within a collaborative structure that is interactively managed by the participants. Based on results from conversation analysis, I have constructed a computer improviser that participates in small group improvisation. Using conversation analysis rules for turn-taking, the computer tracks the roles of the other musicians and follows a structural model of the improvisation to determine its own role as the improvisation unfolds. User-centered design was crucial to the successful development and deployment of the system.

## KEYWORDS

Musical Improvisation, Conversation Analysis, CSCW, Object-Oriented Frameworks for Collaboration, User-Centered Design

## INTRODUCTION

The computer's ability to collaborate with humans is constrained by its inability to parse continuous streams of natural data and determine the functions of human gestures. For example, limitations in speech recognition technology impede participation in spoken conversation. Fortunately, the ideas of Conversation Analysis seem to apply to other collaboration domains in which computers can begin to parse the streams of natural data. Jazz improvisation is a collaboration which offers computers three opportunities for processing the utterances made by other participants.

First, an accessible representation of human musical gestures is available through electronic keyboards and musical controllers. Second, the improvisations being studied here feature a large-scale structure that provides both a nominal sequence of roles for the computer and a basis for evaluating the utterances of others. Third, a repeating harmonic structure governs the construction of syntactically correct musical utterances. Furthermore, an examination of literature about group improvisation yields numerous references to its conversational qualities. I believe results from Conversation Analysis can inform the design of an interactive music system that will both track

the changing roles of participants in a musical improvisation and synchronously generate appropriate contributions of its own.

Many human-computer interaction researchers have concluded that people regard interactive computer systems as social actors [17, 22]. In Lucy Suchman's study of human interactions with photocopiers [21], copier users analyzed failures in their interaction with the copier in terms of a turn-taking dialogue. Users became frustrated when their expectation that the copier would "take its turn" was not fulfilled. If a photocopier, with its limited sensors and simple interactions, provokes these kinds of attitudes, it stands to reason that users will expect as much from a computer system that parses a continuous stream of musical data and responds in real time.

Because they respond to stimuli, interactive music systems exhibit a kind of conversation. Furthermore, the collaboration they are attempting to engage in has many conversational qualities. In this context, it is important to design systems which fulfill users' expectation of turn-taking. To improve the way these improvisations conform to human social expectations, interactive music systems can be informed by the sociology of conversation.

Critics of the use of Conversation Analysis in CSCW (see, for example, [2]) assert: (1) that rules derived from observing conversation should serve as a resource for interaction, not an implementation of it; and, (2) that the asymmetry in interactional capability between humans and machines eliminates the possibility of real conversation or interaction. Even in the face of these complaints, lessons learned from conversation analysis yield important design insights for interactive music systems. As Graham Button says, "...I do not want to suggest that a computational model of conversation cannot reproduce something that might appear to be ... a simulacrum of conversation." [2]

Conversation analysis suggests several requirements for successful conversation participation. Each of these requirements contributes to the design of a musical improvisation system. In both improvisation and conversation, a participant must:

- Perform the distinct tasks of listening to other speakers, composing new utterances, and realizing the new utterances. The different tasks involved in spoken conversation (listening, composing, realizing) provide a functional decomposition of the task of musical improvisation.
- Synchronize the performance of its utterances with the ongoing utterances of other participants. Conversation



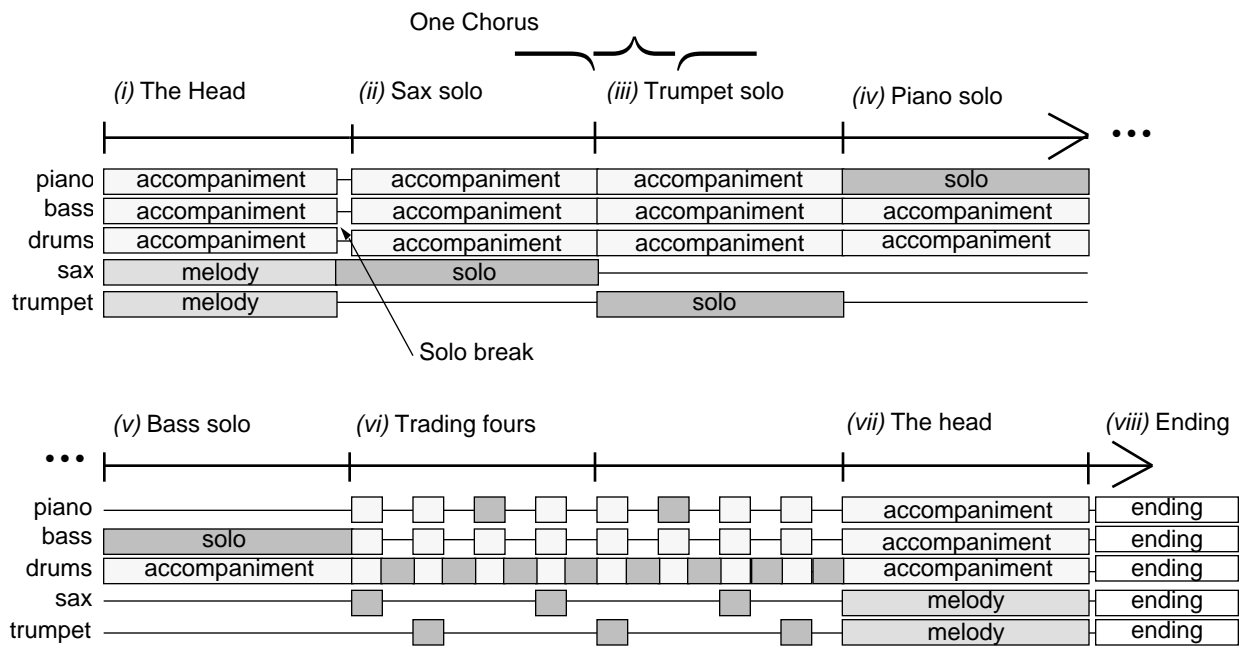


Figure 2. Time line of jazz improvisation structure. Jazz players use this large-scale structure to help them to coordinate their activities [8]. The salient features are: (i) To play the song's melody all the way through, so the audience will recognize what is being performed; (ii–v) To have the performers take turns playing solos of unpredictable length, repeating the harmonic structure of the song as a basis for improvisation; (vi) To have the performers trade short solos back and forth (optional); (vii) To play the song's melody all the way through at the end so the audience will know the performance is ending.

### Collaborative Techniques

Numerous tactics are used to negotiate the order of soloists, depending on the number of performers and their familiarity with each other. If there is only one musician playing the melody, he is often the first soloist as well. Once the choice of first soloist is made, subsequent transitions can be made by having the current soloist select the next one. When one of the musicians is the band leader (e. g., when Miles Davis performed with his band), he often assumes the responsibility of ordering the soloists. Sometimes, a band leader will determine the duration of the solos as well.

Throughout the solos (ii)–(v), verbal and visual cues are used to coordinate accompaniment and solo. Each solo usually takes several choruses, allowing the soloist time to explore several musical ideas or elaborate on the melody. At some point during the solos, one of the players may raise four fingers to propose “trading fours,” a standard procedure for exchanging shorter improvisations amongst the musicians.

If the proposal is accepted, the trading begins immediately following the last solo (v). Soloists reappear in the same order they soloed in, each playing a four measure fragment. When the band contains a drummer, the drummer injects a four measure phrase after each soloist. The four measure solos are accompanied, but the drummer's contributions are not. Deciding when to stop trading fours is complicated by the fact that, depending on the number of soloists, the end of the chorus may not coincide with each soloist having gotten an equal number

of chances (Figure 2 shows an uneven allocation of phrases during trading fours). Eventually, a musician will pat the top of his head to propose playing the melody of the song one last time (vii) followed by an ending (viii).

Some musicians make these kinds of negotiations an explicit part of the performance. In James Brown's performances of “Sex Machine,” he engages in several shouting matches with the rest of the band, such as “Can I take it to the bridge?” “Yeah!” and “Can I hit it and quit?” “Yeah!” to mark transitions between various parts of the song [1]. While trading fours, musicians often respond musically to each other's playing. Hartmann [9] wrote:

“A four measure phrase leaves the player enough room (say, three to six seconds) to develop one idea, to make one statement; yet there is no mistaking the dialogue within which each statement takes its place, and often the musicians answer each other directly. The resemblance to conversation is uncanny.”

The analogy proposed here is central to the design of my system. I view jazz improvisation as a conversation. It is a sequence of spontaneous utterances constructed within a collaborative structure that is interactively managed by the group. This collaboration fosters the interchange of ideas and encourages participants to express themselves. I will further develop this analogy by examining conversation analysis and considering its relevance to musical improvisation.

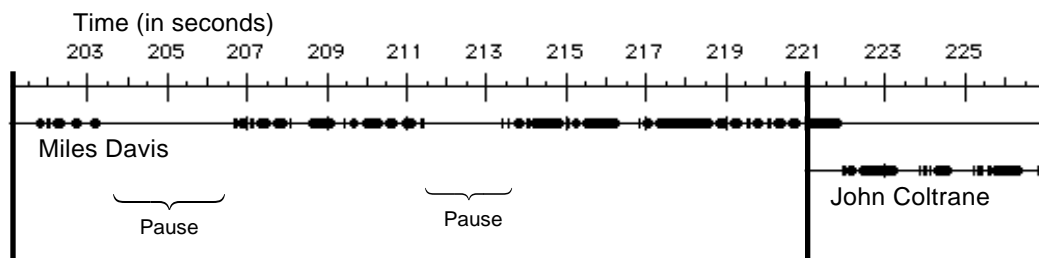


Figure 4. Time line showing solo phrases and pauses between them

### CONVERSATION ANALYSIS

Conversation Analysis regards everyday conversation as a highly ordered, collaborative social activity (see *Computers and Conversation* [15] for an excellent introduction to Conversation Analysis in the context of human-computer interaction). Conversation Analysis proceeds by empirical analysis of ordinary talk. Ordinary conversations are recorded and transcribed into a textual format that preserves information about pauses and overlaps. Conversation Analysis researchers consider various utterances within these transcripts and study how conversational participants design and use language to structure conversational flow.

Conversation Analysis findings are descriptions of behaviors derived from the researcher's analysis of recorded ordinary talk. These descriptions, often formulated as rules, are resources in maintaining the smooth flow of conversation.

One example of Conversation Analysis is Gail Jefferson's study [11] of list-making in conversation and how it relates to turn-taking. The study reports on the predominance of three part lists in a corpus of transcribed conversations. This rule of three manifests itself in both speaker and listener behavior. Speakers produce lists of three items, and can rely on listeners to expect the end of a turn after the third list item. Listeners will wait for the expected third list item, and can even cooperate in completing the list if the speaker appears to be having difficulty doing so. This is an example of the fundamental premise of Conversation Analysis that participants employ these rules as resources for speaking and listening and expect others to do likewise.

### Improvisation as Conversation

Proceeding from the premise that improvisation is a conversation, I assert that group musical improvisations employ rules like the one described above. Musicians use these rules as resources for playing and listening. Indeed, familiarity with jazz performance structure so defines the task of listening to jazz that one must divide a jazz audience into two groups, musicians and non-musicians. Approaching this from a semiotics perspective, Perlman and Greenblatt [19] wrote:

“When we say that the inside audience has structural competence, we mean that they can recognize basic elements in what is being played *as* it is being played.... The rest of the listeners—the outside audience—really do not hear or understand

improvised solos... it may be as meaningful in some way for us to hear an impassioned—but untranslated—speech by the prime minister of Japan, especially if we see and hear it in its original context... we'll never know exactly what the prime minister said.” (p. 181)

The following section presents some preliminary rules for musical improvisation by analogy with results from conversation analysis.

### Jazz turn-taking

One rule of musical conversation is that solos are allocated using a system like the one proposed for conversational turn taking by Sacks, Schegloff, and Jefferson [20]. In their system, speakers construct turns in units of phrases or sentences. These units are constructed in such a way that listeners can project when a unit will end while the speaker is uttering it. These projectable end points are called transition relevance places because they are moments when a transition between speakers is likely. At the end of each unit, conversational participants consider the form and content of the units presented by the current speaker and determine whether a new turn should begin.

In the jazz context, musicians build melodic solos from phrases. The end of each phrase is thus a transition relevance place. Musicians assess these places by consulting the harmonic structure of the tune being performed. Solos are almost always an integral number of choruses. Thus, if a gap occurs in at the end of a chorus, it is much more likely to signal the end of a solo than if it occurs somewhere in the middle.

Figure 4 shows a time line representing musical activity in an excerpt from Miles Davis' "Freddie Freeloader" [4]. As Davis' solo draws to a close it contains a number of gaps that might suggest he is finished. Coltrane does not begin playing during those gaps since they occur in the middle of a chorus. The real end of the solo comes slightly after the chorus boundary at 221 seconds, and Coltrane begins immediately thereafter at time 222 seconds. While the two musicians may also have employed visual cues, the smoothness of this transition is based in part on Coltrane's use of tune's harmonic structure to project the end of the solo. Group agreement on the location of transition relevance places allows for smooth transitions between speakers or soloists without accidental overlap or gap.

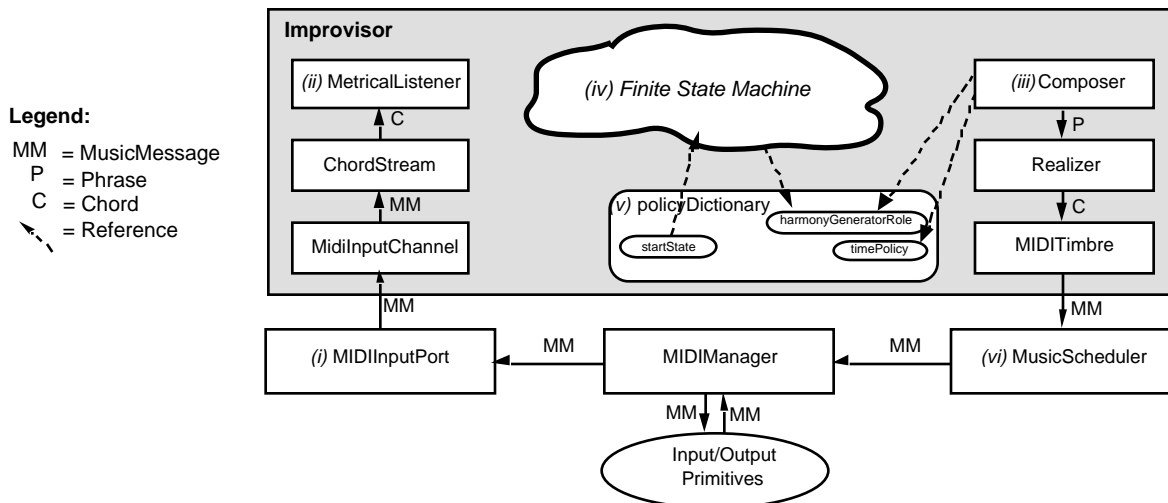


Figure 5. Configuration of ImprovisationBuilder components

Speakers employ two strategies to allocate the next turn. The speaker-selects-next strategy requires the speaker to specifically indicate which listener should speak next. For example, the speaker may direct a question to a specific listener. The self-selection strategy requires a listener to begin speaking at a transition relevance place. In the context of jazz improvisation, musicians use the same two strategies. Sometimes soloists use a visual or verbal cue to indicate who should follow them, while other soloists simply end their turn and leave allocation of the next solo to their colleagues.

In the next section I will discuss the implementation of a jazz improvisation system within the IB framework. In addition to providing an overall philosophical grounding for writing interactive music software, the preceding analysis of conversation and improvisation yields three rules that directly inform the implementation of a jazz improviser:

- Jazz solo turn-taking occurs at chorus boundaries. The computer should examine the soloist's behavior as each chorus boundary approaches to project whether the solo is about to end.
- Jazz performances are governed by a well-known structure. The computer should model this structure and track progress through it during the performance.
- Backchannels can be important to group performance. The computer should be prepared to follow explicit cues, when necessary.

### IMPROVISATIONBUILDER

ImprovisationBuilder is a framework for building computer improvisors. It is written in ParcPlace ObjectWorks/Smalltalk-80 on the Apple Macintosh. Smalltalk-80 primitives written in MPW C connect ImprovisationBuilder to input and output devices. Connections to MIDI devices are handled by the Apple MIDI Manager, which sends data through the Macintosh serial port to a MIDI interface.

The ImprovisationBuilder framework provides Smalltalk-80 classes corresponding to the tasks performed by improvising systems. *Listeners* process the incoming music, parsing it into phrases and focusing the system's attention. *Transformers* and *Composers* create new phrases, either by transforming phrases captured by the listener or by some compositional algorithm. *Realizers* express musical ideas appropriately, both through real-time presentation and by controlling *Timbres* that represent sound generating hardware. *Improvisors* monitor the information gathered by *Listeners* and use it to track progress through a map of the shared musical structure.

By providing a class hierarchy of components for improvising systems, ImprovisationBuilder facilitates rapid testing and prototyping through the combination of existing components [12]. Users can extend the framework by adding their own components to it. These new components reuse code provided by their superclasses and only implement the unique aspects of their behavior. ImprovisationBuilder also provides a standard representation of musical events for use by all components.

ImprovisorComponents are combined in a linear chain, with a Listener at one end and a Timbre at the other. This chain of components is contained within an Improvisor, along with all the information the computer uses to improvise. A typical configuration of components is shown in Figure 5. This figure also shows the MusicScheduler (vi) and other support components. They connect any number of Improvisors to input and output devices and handle scheduling details. The following sections describe the components of the framework.

### Listener

The *Listener's* task is to determine whether the other musicians are soloing, accompanying, or not playing. This determination can be made by assessing the pitch range and density of notes being played. Figure 6 shows a typical walking bass and chord accompaniment that a

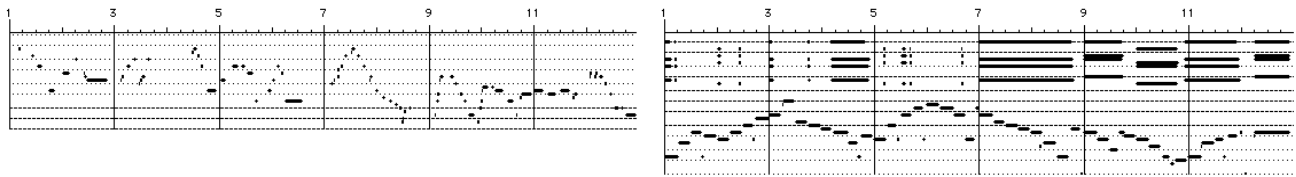


Figure 6. Typical walking bass and chord accompaniment (right) versus typical solo (left), as seen by Metrical Listener

pianist would play on the right, compared to one chorus of a piano solo on the left. The simplest way to distinguish between them is to find the lowest pitch in the last few seconds and see if it falls in the pitch range typical for walking bass. A pause in the musical input may denote the end of a solo, or merely a gap within the solo.

The *Listener* can also compare the incoming musical stream to the harmonic structure of the tune being performed. This allows the *Listener* to translate specific phrases into generalized material, independent of particular keys or scales. The *Composer* uses such generalized material as one basis for its improvisation.

### Composer

*HarmonyGenerator* is a subclass of *Composer* that constructs musical phrases in the manner of a jazz pianist. It draws on two sources of information (see Figure 5 (iii)). One is the *ChordChanges*, which encodes the underlying harmony for each musical beat. The other is an entry in the *PolicyDictionary* that describes the computer's current role as prescribed by its model of the large scale improvisational structure. This information is used to generate one measure of music, which is sent to the *Realizer* for playback through a MIDI synthesizer.

The *HarmonyGenerator* provides variety by combining several strategies for generating melodies. One strategy involves concatenating short melodic fragments ("riffs") to produce a solo. Beginning human improvisors often employ this tactic as they develop their own style, and even the most experienced players tend to rely more on their favorite riffs when playing at rapid tempos. The other strategy for generating melodies is to simply perform a random walk on the scale appropriate to the current chord structure. While this strategy alone is inadequate, it helps to separate the riffs and highlight them. Walking bass lines are also generated by fitting together short patterns. The *HarmonyGenerator* constructs chords using rules found in the piano jazz instruction materials of Tony Caramia [3].

### Representing Structured Improvisation

IB represents improvisational structure by a finite state machine, with each state corresponding to a specific musical activity—soloing or accompanying, for example. The state machine represents the current state of the improvisation and possible future states, in much the same manner as the conversation protocols of *ConversationBuilder* [13]. The current state monitors statistics about the input, as computed by the *Listener*.

Each state specifies a criterion for whether improvisational roles are about to change. An *Accompany* state interprets the appearance of bass notes in the input to mean that the other player is assuming the role of accompanist. Conversely, a *Solo* state interprets the appearance of treble notes to mean that the other player is beginning a solo (see Figure 6). When the current state satisfies its transition criterion, it passes control to the next state, which will give new instructions to the *HarmonyGenerator* and specify its own transition criterion. In accordance with the placement of transition relevance places, most transition criteria are tested at the end of each chorus, since roles usually change on chorus boundaries.

Figure 7 shows a sample finite state machine for a two piano performance. The state machine is based on a simplified version of the structure in Figure 2. With only two players, the choices about who plays the melody and the order of soloing are greatly simplified. One of the players plays the melody, followed by one of the players taking the first solo. The other player then takes the second solo. The process of trading fours in the absence of a drummer means the two performers exchange four bar phrases directly, enhancing their conversational quality. This special case of group jazz improvisation is easily represented in a small number of states.

### SITUATED PROTOTYPING

Much of software development for interactive computer music follows one of two scenarios. In the first scenario the programmer is also the composer. Software development is guided by composer's own aesthetics, and software design feedback is largely introspective. Such systems are likely to fit very closely to the composer's conceptual model and working style, offering high productivity and utility. However, such systems rarely become sufficiently general to serve a larger audience.

In the second scenario, the composer builds an environment from off-the-shelf software. Off-the-shelf systems offer a difficult trade-off: they either remain "aesthetically neutral" by offering low-level constructs and services (requiring additional effort from the composer), or they contain some significant musical assumptions (which may not coincide with the composer's vision). I believe the ideal process for developing interactive music software is one in which composer and programmer engage in an iterative, collaborative design process.

*ImprovisationBuilder* succeeds largely because of an iterative, collaborative design process known in the Apple Advanced Prototyping Laboratory as situated prototyping. Believing that contrived, sterile laboratory conditions offer

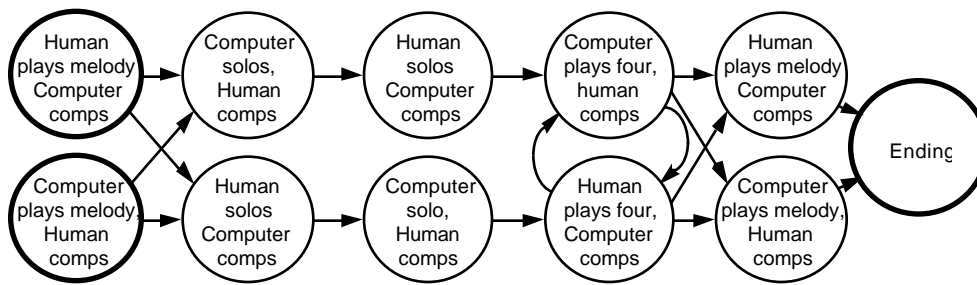


Figure 7. Finite State Machine for two piano jazz performance

no real context for testing the usability or utility of an artifact, our goal is to get working prototypes into the hands of real users as early and as frequently as possible. By observing how these prototypes succeed or fail, we learn valuable lessons that inform the next prototype. Two composers have used ImprovisationBuilder extensively, and have had considerable impact on its present state.

### Sal Martirano

Professor Martirano was an early pioneer in the construction of and performance with interactive musical systems [16]. His research with Sergio Franco culminated in the SalMar Construction, a sound synthesizer coupled to an algorithmic composition engine [6]. While the system could operate autonomously, the user could intervene in the compositional and synthesis processes at many different levels of algorithmic abstraction.

Martirano's second generation system was the YahaSALmaMAC orchestra, consisting of Yamaha synthesizers driven by a LISP program called Sound and Logic (SAL). SAL processed the MIDI data from Martirano's music keyboard and created new musical gestures from his.

Martirano's working style was one of constant rehearsal and experimentation in his home studio. Thus, while I was developing IB and customizing it for his needs, I tried to ensure that a working version was installed in his studio at all times. This meant I was getting a steady stream of bug reports and feature requests during development. While some of the requests were largely cosmetic (such as the specifics of which keyboard keys were assigned to which commands), others had important structural implications for the implementation (such as the need to reconfigure software components during a musical performance).

Martirano's willingness to endure user-hostile interfaces and arcane initialization procedures made him the ideal user for situated prototyping.

### Brian Belet

Professor Belet and I first collaborated on ImprovisationBuilder during a computer music workshop in 1993, a collaboration we resumed in 1994. Our initial efforts were concentrated into an intense full-time two week period. Belet had composed some material for a pianist and wanted to design an IB configuration that transform the material in interesting and appropriate ways.

We worked together to create some Transformer components, such as one for creating trills from sustained single notes and another for adding short ornamentation notes to the attack of single notes. As we wrote and debugged these algorithms together, he gained a greater understanding of IB's musical representation, while I came to understand how he intended these musical transformations to relate to the music he had composed for the pianist.

This foundation helped us to communicate when we resumed work in 1994, culminating in our premiere of "Cross-Town Traffic," a piece for two improvising pianists and IB in November of 1995. Each pianist's improvisation is transformed by a copy of IB. The transformed material is realized on the other pianists' piano while he is playing. While each of us used a distinct IB configuration, our mutual understanding of the implementation and musical consequences of each other's IB configuration greatly aided our improvisation[24].

### CONCLUSIONS AND FUTURE WORK

I believe that musical improvisation is a collaboration with many conversational qualities. To test this hypothesis, I have used ideas from conversation analysis to construct a system that interacts with human performers in a standard jazz performance. In particular, I used conversation analysis ideas about turn-taking to manage transitions between soloing and accompanying.

In practice, ImprovisationBuilder's performance is quite respectable, and has great promise as a jazz teaching tool. Where traditional "music minus one" recordings allow students to practice improvising over a fixed accompaniment, IB lets them practice accompanying, soloing, and when to switch between them. While the bass lines and melodies generated by the system are rather formulaic, its ability to detect the beginnings and endings of solos and borrow from the human performer's solo allow much more interaction than playing along with a prefabricated accompaniment.

Collaboration in the domain of jazz improvisation appears to benefit from the insights of Conversation Analysis. This result confirms the writings of Paul McIlvenny, Paul Luff, and Robin Wooffitt in *Computers and Conversation* [15]. They argue that conversation analysis has important insights for all of human-computer interaction because it proceeds in an empirical manner and produces results which can inform software design.

In the future, I plan to continue improving the generality and reusability of ImprovisationBuilder through continuing design iterations and musical experiments. I hope the architecture presented here will prove sufficiently general to explore the issues raised by computer participation in various aspects of computer-supported collaborative work, such as structural flexibility for workflow systems or better representation of cooperation in shared editing.

#### BIBLIOGRAPHY

1. Brown, J., *Sex Machine* 1970, Polygram Records (CD #314 517 984-2): New York.
2. Button, G., *Going Up a Blind Alley: Conflating Conversation Analysis and Computational Modelling*, in *Computers and Conversation*, P. Luff, N. Gilbert, and D. Frohlich, Editors. 1990, Academic Press Limited: London. p. 67–90.
3. Caramia, T., *A Guide for Jazz Piano Harmonization*. 1983, San Diego: Neil A. Kjos Music Company.
4. Davis, M., *Kind of Blue* 1959, Columbia: New York.
5. Ellinger, J. and M. Baker, *MiBAC™ Jazz Improvisation Software* 1990, MiBAC Music Software: Northfield, Minnesota.
6. Franco, S., *Hardware Design of a Real-Time Musical System*, 1974. Doctoral Dissertation, University of Illinois at Urbana-Champaign.
7. Giomi, F. and M. Ligabue. *An Interactive System for Music Improvisation in Proceedings of the International Computer Music Conference (Cologne, Germany, 1988)*, ICMA, 47-63.
8. Gridley, M., *Jazz Styles*. 1985, Englewood Cliffs: Prentice-Hall.
9. Hartmann, C., *Jazz Text: Voice and Improvisation in Poetry, Jazz and Song*. 1991, Princeton: Princeton University Press.
10. Heath, C. and P. Luff. *Collaborative Activity and Technological Design in Proceedings of ECSCW '91* (1991), 65–80.
11. Jefferson, G., *List construction as a Task and Resource*, in *Interaction Competence*, G. Psathas, Editor. 1990, University Press of America: Lanham.
12. Johnson, R. and B. Foote, *Designing Reusable Classes*. *Journal of Object-Oriented Programming*, 1988. 1(2): p. 22—35.
13. Kaplan, S., A. Carroll, and K. MacGregor, *Supporting Collaborative Processes with ConversationBuilder*. ACM Special Interest Group in Office Information Systems, 1991. 12(2/3).
14. Levitt, D., *A Melody Description System for Jazz Improvisation*, 1981. Master's Thesis, Massachusetts Institute of Technology.
15. Luff, P., N. Gilbert, and D. Frohlich, ed. *Computers and Conversation*. *Computers and People Series*, eds. B. Gaines and A. Monk. 1990, Academic Press: London. 284.
16. Martirano, S., *A Salvatore Martirano Retrospective: 1962–1992* 1995, Centaur Records: Baton Rouge.
17. Nass, C., *et al.*, *Can computer personalities be human personalities?* *International Journal of Human-Computer Studies*, 1995. 43: p. 223-239.
18. Pachet, F. *Representing Knowledge Used by Jazz Musicians in Proceedings of the International Computer Music Conference (Montréal, 1991)*, ICMA, 285–288.
19. Perlman, A. and D. Greenblatt, *Miles Davis Meets Noam Chomsky: Some Observations on Jazz Improvisation and Language Structure*, in *The Sign in Music and Language*, W. Steiner, Editor. 1981, University of Texas Press: Austin.
20. Sacks, H., E. Schegloff, and G. Jefferson, *A Simplest Systematics for the Organization of Turn-taking for Conversation*. *Language: Journal of the Linguistic Society of America*, 1974. 50(4): p. 39.
21. Suchmann, L., *Plans and Situated Actions*. 1987, Cambridge: Cambridge University Press.
22. Turkle, S., *The Second Self: Computer and the Human Spirit*. 1984, New York: Simon and Schuster.
23. Ulrich, J. *The analysis and synthesis of jazz by computer in Fifth International Joint Conference on Artificial Intelligence (Cambridge, MA, 1977)*, William Kaufmann, 865–872.
24. Walker, W. and B. Belet. *Applying ImprovisationBuilder to Interactive Composition with MIDI Piano in Proceedings of the International Computer Music Conference (Hong Kong, 1996)*, ICMA, 386–389.
25. Walker, W., K. Hebel, S. Martirano, and C. Scaletti. *ImprovisationBuilder: Improvisation as Conversation in Proceedings of the International Computer Music Conference (San Jose, 1992)*, ICMA, 190–193.