Multimedia and Live Performance

Ian Willcock

Submitted in partial fulfilment of the requirements for the award of PhD

Institute of Creative Technologies

De Montfort University

December 2011

Volume One of Two.

Abstract

The use of interactive multimedia within live performance is now well established and a significant body of exciting and sophisticated work has been produced. However, almost all work in the field seems to start by creating at least some of the software and hardware systems that will provide the infrastructure for the project, an approachwhich might involve significant duplication of effort. The research described in this thesis sets out to discover if there are common features in the practice of artists from a range of performance backgrounds and, if so, whether the features of a system which might support these common aspects could be established.

Based on evidence from a set of interviews, it is shown that there are indeed common factors in work in this field, especially the intensive linking of elements in performances and the use of triggering or cuing. A statement of requirements for a generic system to support work in digital performance is then established based on interview analysis and personal creative work.Ageneral model of live performance, based on set theory, is described which provides a rationale for the integration of digital technologywithin live performance.A computational model outlining the formal requirements of a general system for use in live performance is then presented.

The thesis then describes the creation of a domain specific language specifically for controlling live performance and the development of a prototype reference implementation of a generic system, the Live Interactive Multimedia

Performance Toolkit (LIMPT). The system is then evaluated from a number of standpoints including a set of criteria established earlier in the study. It is concluded that, while there are many resources currently used by artists working in digital performance (a comprehensive survey of current resources is presented), none offer the combination of functionality, usability and scalability offered by the prototype LIMPT system. The thesis concludes with a discussion of possible future work and the potential for increased creative activity in multimedia and live performance.

Declaration

This thesis presented here isoriginal and solely my own work. The activity it describes was carried out between December 2004 and November 2011. It is submitted for the degree of Doctor of Philosophy at De Montfort University.

Acknowledgements

I would firstly like to thank Jane Turner, Nick Rothwell and Daniel Biro fortheir creative inspiration and companionship throughout the eMerge project which began so many things.

Next, I would like to thank Professor Andrew Hugill and Professor Hussein Zedan for their professionalism and for their many valuable, creative contributions and hours of generous support with this project.

I’m very grateful to Lucy Moffat, Paul Obermayer and Sally Parkinson for their help at those crucial points just when it was needed.

I would like to thank my interview subjects, all of whom responded to my requests for their time and thoughts with great generosity and honesty.

At De Montfort University, I would like to thank my Masters students for their enthusiasm and patience and my friends and colleagues there for their encouragement, advice and support. Similarly, my Westminster Kingsway College, and latterly, University of Hertfordshire, colleagues have been highly supportive.

Lastly, I want to say how grateful I am to Molly and Dora, who have put up with a dad who has been less available and active than he would have liked.

Publications

Willcock, I. (2005) Pinning Down the Artists: Developing a Methodology for Examining and Developing Creative Practice. In *Proceedings of the 2006 DMU Humanities Graduate Conference* [available online] http://www.tech.dmu.ac.uk/STRL/research/publications/pubs/2006/2006-17.pdf

Willcock, I. (2006) Composing without Composers: Creation, control and individuality in computer-based algorithmic composition. In *Electronics in New Music*, C.-S. Mahnkopf, F. Cox, and W.Schurig, (Eds.) (2006) Hofheim: WolkeVerlag.

Willcock, I. (2010) Words Of Power: eMerge, A Language for the Dynamic Control of Live Performance. In *International Journal of Humanities and Arts Computing* 4.1–2 (2010): pp67–80 Edinburgh University Press

Table of contents

Volume One

1 Introduction

> "One of the things that impressed me about this world is that there are an awful lot of people doing fantastic work, but an awful lot of them are doing it on their own and a lot of the time they're reinventing the wheel." Martyn Ware

The incorporation of interactive digital technology within live performance has become a feature of a significant and extensive body of creative activity. It is often accepted without further comment that this activity, like most creative activity, is highly heterogeneous in nature; it consists of a wide range of differing individual and organisational practices whose difference is an important, even necessary, part of their creative identity. Indeed, when examining creative practice, including that in digital performance, a commonly used academic approach is to identify those features and approaches which make a given work or a particular artist's oeuvre unique. This study, however, attempts to turn this approach on its head; instead of asking what is unique, it seeks to discover what, if anything, is common. Are there aspects of creative practice and the needs of creative practitioners across a wide range of work with interactive multimedia in live performance that are similar and, if so, what are they? It seeks to test the hypothesis that there might be features of work in digital performance which could be legitimately called generic and that, if these are identified, it may be possible to develop a system which could support a wide range of work in this area through facilitating these generic features, enabling artists to concentrate on those aspects of their work which are unique to themselves; their own individual creative visions.

This first chapter sets out in detail the context and rationale for the study, the starting points and impetus for both the investigation into practices within multimedia and live performance and the adoption of a generic approach. It then summarises the outcomes and anticipated contributions to knowledge produced by the project and presents the overall research goals and describes how these will be used to evaluate the success of the study. The chapter concludes with an explanation of the structure of the thesis.

1.1 Context

The research project described in this thesis emerged following the researcher's involvement as a musician and technologist in a number of projects[1] seeking to incorporate interactive multimedia within live performance. These experiences led to reflections on the particular processes and difficulties involved in such work. Artists have a tendency (often a very strong one) to be completely focussed on the particular piece being created and in developing the resources needed for that particular work, yet on looking back over this work, and after sharing and reflecting upon experiences with other practitioners, it became less certain whether this approach is, for the field of multimedia and live performance as a whole, the most efficient way to support experimentation and creativity. In particular, the project-focus of much of the work and the bespoke nature of many of the resources used[2] can be seen as hurdles that make experimentation and the development and honing of performance techniques difficult in ways that do not apply (at least not to the same extent) in other fields; For example, when working with text, there are a range of text packages available, some designed for specific purposes (script writing, blogging, programming etc.) and others more generalised. The user selects the tool which seems most appropriate and then customises it to suit their own individual

[1] Particularly influential was the 'eMerge' project led by Jane Turner and Daniel Biro with Nick Rothwell. I am enormously grateful to all for their support and inspiration and for the chance to collaborate in a genuinely experimental and exciting project. The controller in the prototype system described in this thesis is very heavily based on the server Nick wrote for the project and his programming has set a quality benchmark for the work done since.

[2] While there are some tools which are very widely used (Isadora and Max/Jitter are perhaps the most common), almost all projects seem to involve bespoke software to some extent. Interview subjects often spoke of some approaches that they wanted to investigate but which were unavailable to them because they didn't have a programmer or collaborator with specialist knowledge.

preferences; it is not generally expected that a writer will need to create a text-processing system as an initial step to developing ideas but, in complete contrast, programming is an expected part of much work in live performance which incorporates digital technology (although it is not always the artists themselves who carry out this element of the creative activity).

> "…the making of the art-work is the making and remaking of those lines of code until they are right." (Downie 2005:347)

It is the case that specialised software packages are available for the field of live performance; Max[3] and Isadora are widely used and there are also collaborative music tools such as Peersynth, Atlantic Waves[4] and Auracle and real-time synchronisation systems such as Aura[5] and a number of commercial packages designed for controlling theme park attractions from companies including Alcorn McBride, Gilderfluke and MediaMation(How Theme Parks Work, no date). However, as the range of media and devices used in live performance grows, all these packages either become less useful in terms of their applicability, or require more and more advanced technical skills to apply[6] which renders them unavailable to many practitioners focussed on their own creative projects. Additionally, while these systems have been used to support a range (even a wide range) of work in live performance, they all carry within

[3] Throughout this thesis, titles of texts and art-works have been italicised, but names of software have not

[4] Developed by Monolake (Robert Henke) and Deadbeat (Scott Monteith) using Max, the system allows networked audio with complex visuals through an interface that permits the building up of audio structures. At present, it is not scalable to large numbers of collaborators.

[5] Aura is a distributed system originally aimed at real-time audio generation and presentation which now encompasses a range of media types. It also has a scripting language, Serpent.

[6] Max is potentially the most generic, particularly with the Jitter digital video processing extension. However, it is highly complex and not easy to use for a non-programmer. See also section 2.2 where I discuss Downie's criticism of its underlying assumptions about the nature of technology use in performance.

them a set of assumptions about ways of working and about how software can integrate with performance that can limit the work that artists can do since their design constraints are those of the artist or project(s) for which they were originally produced. As Downie has observed,

> "With few exceptions, these popular graphical environments (they are controversially sometimes referred to as visual programming languages) are based on a common small reservoir of ideas: a few visual metaphors, a few structuring concepts. They each possess a surprisingly similar flavour and set of capabilities. So similar, in fact, that one might suspect that we are suffering from a digital art tools monoculture." (Downie 2005:347)[7]

Finally, thehypothesisoutlined in this thesis about the possibility of a collaborative, less specific approach to the provision of software tools for artists working in digital performance is not, in itself,entirely new; the *Software for Dancers* project[8]of 2001 had a similar starting point, although it confined itself almost entirely to the rehearsal phase of producing performance (Software for Dancers: Description 2001). Indeed, the increasingly important Open Source

[7] While agreeing with the general line of Downie's argument, I will argue below (see section2.2) that his view of the similarities of current applications designed for supporting the integration of multimedia and live performance is, at least to some extent, overstated, particularly if one takes the experiences of users into account.

[8]The *Software for Dancers* project took place in London between June and October 2001 and had the following stated aims:

1. to develop concepts for a software rehearsal tool(s) for choreographers and those practitioners for whom the body in motion is a primary material.
2. use the first aim as an opportunity to engage discursively with a range of questions relative to interdisciplinary and collaborative practice involving live performance and digital technologies, for example:
    - what are the processes and products of coding and choreography;
    - to what extent and to what ends can coding and choreography be described as different or similar;
    - what level and type of shared understanding is necessary to facilitate interdisciplinary collaborative practice;
    - what are the possible range and outcomes of this form of practice;
    - how and where can an articulation of these and other issues function dynamically and generatively within a diverse community of practitioners." (Software for Dancers: Description 2001)

movement may be seen as another, wider, manifestation of the desires of practitioners to support practice in their chosen field through collaborative, altruistic endeavour.

1.2 Rationale

This research project sets out to test the hypothesis that a generic system could be, at least in principle, appropriate and beneficial to a wide range of work in digital performance, if it is, can the features of such a system be identified and a prototype system be developed? ‘Generic’ is used here to label features of the use of digital technology in live performance which are not specific to any particular performance, individual artist or performer or to any single parent tradition of live performance practice or particular group of users. It should be noted at the outset that there are some software and hardware packages that are widely used (Max and Isadora in particular) and consequently have serious claims to be considered ‘generic’. In recognition of this, Max and Isadora are evaluated in detail below (see section 2.2.3) and also are comparatively assessed in terms of features and affordances with the generic system developed for this study (see section 7.3).

Even before any specific features of any such system were identified, it was clear that it would have to be flexible enough to allow for all the individual freedom to make choices of method and technical means associated with creative work, yet be useful enough to provide real support for artists – so that they wouldn’t have to start from scratch each time a project was begun. Just as with software designed for other areas of application (e.g. text processing), some projects will always need a dedicated system due to their particular demands; a search for a generic system does not imply any requirement for universal application. Any generic system would also have to be as transparent

as possible in use and require little or no special equipment or training to use; indeed it would have to work seamlessly with all the technical resources and learnt skills that were already being used by artists in making and presenting their performance-based work. It would have to divert them from their primary activity for as short a time as possible while they integrated it with their work and thought about making performance.

The benefits of a study examining this area will arise both from the new knowledge produced as well as from the factual and technical resources which could support and stimulate work with multimedia in live performance. If, as my results from interviews with practitioners suggest (see section 4.1), some of the development of this area is constrained by access to specialist skills and technological resources by artists who are engaged in making performance, the experience of working with graduate students and the responses of workshop participants (see section 7.1) suggest that another major constraint is the difficulty of learning about and developing a practice with multimedia and live performance. The access to opportunities for rehearsal and experimentation, which are a large part of developing facility in all performance traditions, is often limited or absent during the period when creative artists should be finding their individual identities within multimedia and live performance.

Difficulties of access and knowledge cannot be solved just by producing another software system[9], part of the difficulty faced by both practitioners and those seeking to begin work in this area is the very large number of software and hardware systems available. Many of these were originally designed for other purposes but have subsequently found applications within live performance. Finding out what is available, what functionality is offered and what the technical requirements and licensing regimes are for particular packages can be difficult[10]. The outcomes this study has produced include both a comprehensive and detailed survey of applications currently used in live performance (section 2.2.2) and a database of resources, both of which will be disseminated to a wider audience after the conclusion of this project.

It is perhaps useful at this point to consider the range of practice being examined and also to explain the choice of factors left outside the scope of the present study. Multimedia and live performance is, unsurprisingly, a highly heterogeneous field of activity. On the one hand, it is informed by traditions of artistic practice that historically have placed great emphasis on individual differentiation through the production of artefacts using traits and stylistic habits unique to the individual artist. On the other, it is underpinned by technological and theoretical structures that promote the interpenetration and hybridisation of gesture and genre with a corresponding enlargement of the field of activity

[9] Importantly, this study does not seek to develop a system which will replace other resources and does not suggest that adoption of the prototype by other practitioners is a suitable metric for judging the success of the overall study (see section 1.4).

[10] This is particularly so at the present when new controllers (e.g. the Wii Remote, Microsoft's Kinect system) and mobile devices (iPad, iPhone etc.) are appearing and are being used in live performance.

available to the practitioner. The present study takes as a focus that area of digital performance practice which has a clearly identifiable relationship with one or more precursor live-performance traditions in the fields of dance, music and theatre. Some reference is also made to installation-based work in which the audience act, to some extent as both performer(s) and audience. The identification and extent of 'digital performance' is discussed in detail below (see section 2.1.2).

All of this artistic practice is obviously located within particular social and economic structures of power and control – and will, just as obviously, be influenced by and will reflect back and develop meaning through their culturally-located origins. While I would not want to suggest that these aspects of multimedia and live performance practice are unimportant, I am largely excluding them from the present study. Despite this caveat, it is clearly significant that much art that uses digital technology has as one of its many possible readings, a criticism of, and engagement with, the means of production and commoditisation of technologically-based cultural practice. Examples include the strong tradition of subverting control structures (the cracking movement, echelon hacks etc.) and the widespread use of open source software and licensing models (the jSyn[11] project, Audacity[12], the Creative Commons[13] movement). Clearly, any research that attempted to separate the interests of the researcher from those of practitioners might run the risk of

[11] *JSyn audio synthesis API for Java* (2011)
[12] http://audacity.sourceforge.net/about/?lang=en
[13] http://creativecommons.org/

transforming the research from an enabling process into another instance of the restrictive patterns of ownership and control that are widely seen as restrictive or even counterproductive. While the present study is not designed to consider the social context within which live performance involving multimedia occurs as a part of either its evidence base[14] or analytic criteria, for it to be successful, these factors will have to be taken into account when making decisions about the ownership and conduct of the research.

[14] Several of my survey subjects made interesting observations on the effects of technology on the nature of 'the work' and the changing ways their creative activity operates.

1.3 Contribution to Knowledge

The original contributions to knowledge provided by this study are in several areas: an account of the type, disposition and extent of multimedia-use by practitioners working across a range of traditions of live performance (see section 2.2.2); the development of a novel approach to defining requirements for creative arts software; and an innovative theoretical approach to modelling live performance.

From the review of the literature, it is clear that current accounts overwhelmingly concentrate on single genres (Traub 2005, Lycouris 2009, Duckworth 2005), individual projects or on the use of specific tools (Peng& Al 2004, Guedes 2007, Hamel 2006) without being able to describe and compare usage in terms of actual functionality employed across a wide, diverse range of current performance practice. It will be suggested below that, because of the assumptions implicit in these limited foci compared to the design of the current study, it has been able to identify patterns of usage inaccessible to earlier studies.

Examining previous attempts to develop software for general use in performance (deLahunta 2002b, Dixon 2007:206), it is clear that where projects have been structured by a set of defined functionality rather than through the gradual accretion of additional features (e.g. Composers' Desktop Project, Max), successful completion has been less frequent; the current study presents

an improved methodology for defining requirements for widely differing target users.

The current study has included the development of a novel generic approach to modelling performance based on set theory (see section 5.1). This approach, which is not based on taxonomic approaches to the content of digital performance but on modelling the transactions and decision-making of performers, may suggest fresh procedural or theoretical approaches to live performance.

Another implication of the results of my survey is that due either to the bespoke nature of many software resources, which makes them unavailable for other artists, or the assumptions about practice or performance built in to publicly available tools, there will be possibilities for using multimedia in live performances which may become available using a generic system which could not have been explored hitherto.

1.3.1 Research Outcomes

The study's research outcomes include a range of different types of artefact including structured interview and survey instruments, accounts of current theoretical debates and technical practices, formal models and a number of software systems. They are listed below in summary; please see the sections indicated for detailed descriptions and section 3.1 for a detailed description of the overall structure and methodology of the study.

- A structured, flexible interview framework for investigating the use of multimedia by practitioners in Live Performance (section 3.2.2).
- A set of interviews with practitioners using multimedia from each of dance, music and theatre performance traditions (section 4.1).
- A comprehensive survey of applications and software systems currently used with live performance (section 2.2).
- A set of interactive software systems for gallery-based installations (sections 4.2.2 and 4.2.3).
- A statement of requirements for a generic system to support work with multimedia in live performance (section 4.3)
- A set-theory influenced model of live performance (section 5.1).
- A formal computational model (section 5.2) and a formal (BNF) grammar for an associated control language (section 6.1).
- The prototype Live Interactive Multimedia Performance Toolkit (LIMPT) system (section 6.2).
- A survey of users' responses to the system (section 7.1).

1.4 Measuring Success

The outcomes listed in section 1.3 reflect and shape a structured set of activities designed to meet the overall goals of the research project. The extent to which these goals are met will allow judgements to be made about the overall success of the project (see section 8.1). However, it is important to remember the overall aims while considering the evaluation of specific planned research outcomes, particularly the prototype system. Success in creative work necessarily requires that artists can follow their own visions and use (and develop) their own criteria and working methods. Given this, adoption of the system by artists was not considered a viable metric for measuring success. While system development and wider use of that system are important, it will be the development of understanding, the stimulation of interest and the production of exciting, innovative performances that incorporate multimedia in ways that have not even been considered yet that are the real long-term benchmark of success.

1.4.1 Research Goals

The overriding goal of the programme of research described by this thesis is to facilitate and enable a range of different types of creative activity (making, thinking about, exploring and appreciating) in, and about, the incorporation of multimedia within live performance of all types. However, this overall goal is both too ambitious and too general to serve as an appropriate focus for a research project of the scale and duration described here. A set of more limited and specific goals derived from the overriding aim can be identified which are

appropriate for a doctoral research project and which allow for detailed and measured evaluation.

The first goal is the establishment of an authoritative account of current practices in digital performance which can provide satisfactory evidence for an answer to the main hypothesis underlying the project: are there features of practice that are generic and, if so, what are they? This alone could provide a sufficiently novel framework for the consideration and invention of live performance involving multimedia that it could stimulate activity and interest. Alongside this, the study aims to establish a comprehensive, detailed, catalogue of software tools currently used in live performance (see section 2.2.2). While this will support the current study, it could also function as a resource for the wider community.

A third goal, which is dependent on the first being realised, is the development of a prototype generic system which provides specific functionality (established in the statement of requirements) to enable work in live digital performance and which might provide new or improved facilities and opportunities for users. While future adoption and use of the system among creative practitioners will hopefully form part of future activity, the current study adopts an evaluative methodology which is more manageable and which is only partly based on user feedback.

1.4.2 Evaluation of Prototype

Software development processes usually have a well defined set of evaluation strategies applied at a number of different levels – from unit test compliance for classes and packages through to user and functional testing of complete (or at least partly functional) systems against metrics or heuristics derived from users' own experiences and needs or from specifications drawn up earlier in development processes. The evaluation of the software produced in the current study has been approached in three different ways to seek some measure of validity through triangulation:

- A survey of practitioners who used the LIMPT prototype system in a workshop (section 7.1).
- An evaluation of the prototype LIMPT system's capabilities against the requirements established during the study (section 7.2).
- A comparative study of functionality against currently used applications (section 7.3).

1.5 Thesis Structure

The thesis is structured so as to lead the reader from the initial context and starting points for the research project, through an examination of the work of other researchers, software developers and artists to a detailed explanation of the activity carried out by the researcher. It then presents an evaluation of the LIMPT system software developed during the project and concludes with an evaluation of the overall research and with suggestions for further activity.

In writing this document, a choice has had to be made between the conventions of humanities-based research and those of computer science. Generally, practices drawn from computer science have been adopted, in particular, the Harvard referencing system and chapter layouts.

Chapter 1 deals with the processes that led to the research being undertaken, both the wider creative and practical contexts and the individual experiences and preoccupations of the author. It then describes the contributions to knowledge and the research outcomes produced and goes on to explain how the overall evaluation of these outcomes will be approached, both in terms of evidence and a justification of the metrics employed. Chapter 1 concludes with an explanation of the structuring of the thesis together with a map of the thesis.

Chapter 2 deals with the conceptual underpinnings of the research and with previous activity which is related to the topic and approach being considered. The concept of ‘multimedia’ and the validity of identifying a body of performance

practice solely through its incorporation of interactive multimedia are examined and existing accounts and approaches are critically discussed. Existing approaches to modelling performance are evaluated, both explicit formulations and those implicit codifications, which are expressed through the assumptions made by software and systems designers who produce the tools and applications used by performers. The chapter then describes and critically evaluates the principal applications and other software resources currently used by practitioners in live performance and multimedia together with a brief survey of generic approaches to software development. The results of the interviews with practitioners are briefly summarised, concentrating on what they suggest about current practice in dance, theatre and music performance that incorporates multimedia.

Chapter 3 discusses the methodologies adopted for the various kinds of research activity encompassed by the overall project. One of the features of the research is that it draws on traditions and practices from distinctly different subject areas; some are drawn from a humanities background, others from computer science and creative arts. The discussion of methodologies has therefore to explore and follow these different threads, showing how they are combined in places and, where they remain distinct, how they can support each other in arriving at valid conclusions. The conduct of the interviews and subsequent qualitative analysis is explained and justified. The chapter then shows how results derived from the survey, combined with information from other sources, then formed the basis of a set of formal requirements for

subsequent software development. The structuring, conduct and evaluationprocedures of this development are then explained in detail. The coverage of methodologies concludes with a description of the processes used for initial creative collaborative activity in a workshop setting and of how participants' experiences and opinions were collected and analysed for future development.

Chapter 4 begins with an analysis of the survey materials and, from this, suggests a positive answer to the first research question: is it possible to identify the characteristics of a generic system which would support work with interactive multimedia in live performance? From the results of the survey, an initial set of formal requirements for such a system is suggested. This specification is further refined through consideration of two other software development projects undertaken within the project's time span which influenced the development of the statement of requirement; the 'Curious Listener' (an interactive site-specific installation sound piece) and 'Melbourne' and 'The Street', two interactive installations created with Martin Rieser which were shown at HEAT in Melbourne and ISEA09 in Belfast respectively.

Chapter 5 describes the steps in the development of the formal computational model which underpins the prototype LIMPT system developed as a part of the research project. It begins with the approach taken to modelling generic live performance which is based on set theory and Boolean logic rather than an approach involving any attempt at a taxonomy of the materials of performance.

This modelling then becomes the basis for the construction of a formal computational model. Some limitations of the model are discussed and the decisions taken in adopting particular solutions are justified.

Chapter 6 begins with a description of the design process and implementation of the eMerge language for dynamic control of live performance. This is the way users control the LIMPT system and it is crucial for delivering a combination of flexibility with a satisfactory user experience. A formal grammar is presented together with a discussion of the choices made about vocabulary and structural complexity. The LIMPT system's development process is then described in detail, beginning with the design of the XML message formats and then looking in turn at the system's discrete software components: the central server written in Java and the client which is a Director/Flash hybrid.

Chapter 7 evaluates the LIMPT system using three different approaches for triangulation. First, the system's use in a creative workshop whose participants were mostly familiar with the field of digital performance is discussed. The system is then considered and evaluated against the list of requirements for a generic system established in chapter 4. Lastly, the system is compared to other current applications and systems used in current multimedia and live performance practice. This comparison takes account of differences in intent and functionality, but also considers how easily and effectively different software tools can be integrated into artists' working patterns.

The final chapter presents a series of concluding thoughts about the research project as a whole. The original research questions are revisited to provide a context for an evaluation of the overall research activity. The original contributions to knowledge generated by the project are described and suggestions for future work are made. The proposed dissemination of the research outcomes both among practitioners and through pedagogic activities is also described.

2State of the Art

This chapter critically examines current practice and thought in the field of multimedia and live performance. It begins by establishing definitions for key concepts underlying the enquiry and then suggests a justification and rationale for the identification of live performance involving digital technology as a coherent and defined topic suitable for study. This reading enables me to propose and support through detailed reference a coherent framework of knowledge which provides a context for my research. I am able to confidently assert that my topic of enquiry is non-trivial and does not duplicate existing work but rather will complement the work of other authors and make connections between other theoretical accounts, particularly those positions exemplified by Broadhurst (2006, 2007), Dixon (2007) and Downie (2005). The chapter then examines approaches to modelling performance: existing accounts of activity and content which consciously or unconsciously attempt to frame definitions of what live performance can be in terms of its constituent elements. From these models, academic accounts of performance-practice are examined and critically discussed.

The chapter then proceeds to survey the actual resources currently used by practitioners. It proposes appropriate evaluative criteria for software used in live performance, and then presents a comprehensive and detailed overview of currently available software (section 2.2.2) together with a more detailed and critical account of the two most popular software packages used in live

performance,Max and Isadora. The chapter then looks at generic approaches to software development in the creative arts and concludes by summarising the results of my series of interviews with practitioners (see section 4.1 for a detailed account).

For this research, I have read and reviewed a wide range of sources located using a set of systematic strategies. These have included full index reviews of selected conference proceedings and journals for the past six years as well as university library and online database catalogue searches using a range of keywords relating to the topics of enquiry. My survey of multimedia resource-use by artists working in live performance and evaluations of software products and systems are based on interviews (both mine and those reported in secondary sources), accounts of performances and published specifications.

2.1 Review of Scholarly Activity

The literature dealing with the field of multimedia and live performance is extensive, but a significant proportion tends to be tightly focussed on a particular application or instance of use (e.g. Birchfield et al 2006, Martin et al 2010, Nagashima 2002, Naef&Collicot 2006, Dannenberg 2002 and 2004, Wang et al 2006). It thus reflects the pattern of theoretical, technical and creative activity in the field which has, for the most part, concentrated on providing resources for individual performances or research projects rather than on more generalised tools and on developing localised understanding of an individual's practice rather than frameworks for looking across broad ranges of work. There are some wider accounts (Broadhurst 2007, Dixon 2007, Birringer 2008, Chatzichristodoulou& Jeffries 2009) that do not follow this approach, and these, together with those software development projects which adopt a more generic approach, are discussed below.

In establishing the context for this enquiry, it is necessary to begin by looking at attempts to define and contextualise the term 'multimedia' itself (section 2.1.1) before critically evaluating the different ways multimedia use within the context of live performance has been conceptualised (section 2.1.2). This section will suggest that there is an emerging consensus that sees the field as a cognate area, related to performance studies, that is characterised by a preoccupation with using technology to extend the potentials of performance for creators, performers and audiences. In section 2.1.3, I examine the range of existing approaches, both explicit and implicit, to the modelling of performance which

incorporates interactive multimedia. The survey of academic activity concludes (section 2.1.4) by looking at attempts to survey the range of live multimedia performance-practice that are aligned with the generic viewpoint of the current study, in that they are not focussed on specific projects or genres.

2.1.1 Multimedia: Definitions and Context

'Multimedia' (originally multi-media) has seen a shift and expansion in its meaning from its earliest known[15] usage,

> "The first prong is a multi-media publicity campaign to encourage school children and others young enough for training to obtain adequate educational qualifications." (Greggains 1962:xviii/4))

where it indicated simply a performance or product that employed different types of information simultaneously towards a more complex field of meanings which mean that in contemporary usage its definition is sometimes problematic (Simpson &Polfremann 2003, Haskel 1996), sometimes closely related to its original meaning (Davison 2003) or, for some authors, more a property of the quality of communication than any features of the content involved (Chapman & Chapman 2004). There is currently a general consensus that the term necessarily indicates communication involving simultaneous use of more than one[16] type of information (text, sound, still and moving images). There is also an (often implicit) acceptance that digital technology or computers are essential

[15] As listed in the Oxford English Dictionary online edition, 16th November 2007. It is interesting that 'multimedia' is often suggested as a modern replacement for the term 'hypermedia', although the latter term was actually coined later, by Ted Nelson in his 1965 paper *Complex Information Processing* (Nelson 1965).

[16] The exact number of different types varies between authorities. Simpson and Polfremann (2003) note a usage applied to a project only including text-based information and clearly indicate that they do not consider this an accurate usage. Haskel (1996) requires two or more media types.

factors in mediating this communication. The main area of disagreement is over interaction, the ability of the user to influence and shape the communication process which results in profoundly different roles for both producers and users. Haskel, writing in 1996, makes the distinction explicit:

> "Multimedia can be understood to mean the combination of any two of the following elements mediated by computer: images (still or moving); sound (including music); the written or spoken word; and lastly, in interactive multimedia, some form of determination by the audience or user." (Haskel 1996:1.2.1)

Packer and Jordan (2001:xxxv) suggest a highly sophisticated definition[17] involving five separate factors: -

Integration:	the combining of artistic forms and technology into a hybrid form of expression.
Interactivity:	the ability of the user to manipulate and affect her experience of media directly, and to communicate with other through media.
Hypermedia:	the linking of separate media elements to one another to create a trail of personal association.
Immersion:	the experience of entering into the simulation or suggestion of a three-dimensional environment.
Narrativity:	aesthetic and formal strategies that derive from the above concepts, which result in nonlinear story forms and media presentation.

Figure 1: Packer & Jordan, Features of Multimedia

They clearly agree with other authors and see interaction and a multiplicity of media types as important constituents. However, some of their additional factors, particularly their insistence on immersion as an essential feature of

[17] They use the terms 'multimedia', 'digital media', 'digital multimedia' and 'new media' interchangeably in their discussion of definitions and defining characteristics (Packer & Jordan 2001:xxxiv-xxxv). I see this usage as a deliberate attempt to gather together and elide a wider synonymous usage rather than any failure of precision on the authors' part.

multimedia, are less convincing. It is difficult to sustain an argument for a definition of multimedia that excludes such canonical instances as (most of) the Internet – which with its browser-framed display, fixed focus and limited range of interaction-methods (largely confined to the mouse and keyboard) cannot hope to do more than hint in an inevitably unsatisfactory way at the more genuinely immersive experiences available using specialist equipment. It can of course be argued that many websites do attempt to 'suggest' a three dimensional environment and this has been a continuing pre-occupation of web systems engineers thorough systems such as Shockwave[18] and QTVR[19]. However, it also must be admitted that many do not and that there is no widely held categorisation of websites according to the degree of immersion they suggest. In the end it is theWittgensteinian notion of definition through usage (Wittgenstein 1953: Proposition 43) which suggests that requiring immersion is unjustified.

For Chapman and Chapman, writing in *Digital Multimedia* (2004), a book designed as a core teaching text for undergraduate students studying multimedia, the distinction has disappeared and interactivity has become an essential aspect of 'multimedia' which they explicitly distinguish from (non-interactive) 'combined media':

> "What is it then, if not the combination of media, that distinguishes digital multimedia from previous forms of combined media? It is the

[18] Shockwave is a browser plugin technology targeted by the Director authoring IDE. It has now largely been superseded by Flash, but provided one of the most powerful interactive 3D systems available for web delivery.

[19] QuickTime Virtual Reality (QTVR) from Apple is a combination of specialist software and image processing which allows the production of immersive navigable images where the user can choose and control their viewpoint.

> fact that the bits that represent text, sound, pictures and so on can be treated as data by computer programs… A program can control the order in which various components are presented and combined, and can do so in response to input from a computer user. In other words, digital multimedia can be *interactive*, in a way that, for example, a TV news bulletin is not, and that goes far beyond the simple control afforded by a VCR or DVD player." (Chapman & Chapman 2004:4)

Siegel and Jacobsen echo and expand this definition to relate it to potential applications in performance.

> "Interaction on the other hand, implies interplay between two equal parties… Interaction may of course be present in varying degrees from work to work or within a single work, but our main focus here is on rule-based software composition that allows a dancer's movements to influence musical processes, not simply start and stop fixed events or sequences of events." (Siegel & Jacobsen 1998:29)

While it might be argued that these on their own are inadequate, or at least polemical definitions, ignoring as they do so much of the format and structure of the experience of multimedia or multimedia and performance, it is important to remember that the context of these arguments has shifted massively from that of earlier definitions despite the comparatively short time between them.

I believe that the series of definitions given above, if taken together, do provide evidence of a shift of both meaning and context and that, while these shifts will have had many 'causes'[20], one factor will certainly have been the rapid development in computing power (Kurzweil 1999:104), both in terms of raw

[20] Indeed, I would not want to suggest that semantic changes in language are necessarily explicable in terms of a causal relationship between event(s) and subsequent phenomena; I suspect that, at least sometimes, a more emergent mechanism may operate.

processing power and also in terms of the consequential information handling and representational capabilities of personal computers.

> "For the magic of the digital revolution to take place, a computer must also *represent itself* to the user, in a language that the user understands." (Johnson 1997:14)[21]

The development of the nature of the human-computer interface, its growing graphical sophistication and adoption of the now-standard input devices and metaphors together with the shifting representations, affordances and cultural significances produced by this development has been extensively covered from many perspectives both as a discipline in its own right (Tognazzini 1992, Nielsen 1993, Shneidermann 1997; see also Prior 2011) and as an adjunct to other disciplines such as cultural theory (Manovich 2002). Given its widespread effects on users' social and cultural relationships with technology, the influence of these changes on accepted definitions of multimedia is perhaps more understandable.

Another factor in the shift of meaning is that users themselves are changing (Turkle 1995, Willcock 2002) in terms of their attitudes to digital technology. There is a continuing paradigm shift in terms of what computers are considered to be *for*; just because computers could represent themselves to the user and could handle high bandwidth information-types easily, did not mean that these tasks would be one of the purposes they were used for. This required a

[21] This view of the importance of computers' self-representation through the interface does have its own critics – see Stephenson 1999.

willingness to alter the relationships between people and computers with a consequent reallocation of the resources those computers provided.

> "it used to be considered wasteful and frivolous to devote time and money to the user interface, because computer cycles were so precious and had to be expended on the problem, not the person." (Negroponte 1995:89)

For authors in the 21st century the computer, the delivery platform for multimedia, was sufficiently powerful and ubiquitous that it had become a given; an assumed adjunct to any experience of multimedia which renders the explicit inclusion of simultaneous multiple asset types in definitions largely redundant since this capacity was part of the specification of even low-end systems.

2.1.2 Multimedia and Live Performance

In looking at the field in general, there are a number of widely accepted observations that can be made. Although the precise starting point of the process is contested[22], it is widely recognised (Kurzweil 2005, Gershenfeld 1999, Negroponte 1995, Schiller 1981, Lévy 1999, Turkle 1995, Broadhurst 2007) that we are in the midst of a transformation of our environment, society and selves through a cumulative, incremental and increasingly pervasive application of digital technology. That this transformation is highly uneven in its delivery of both 'progress' and 'opportunity' has also to be acknowledged. Of course, the application of technology is not a new phenomenon and this tool-

[22] The starting point for the digital infiltration of our culture has been variously identified as being the publication of the paper *As We May Think* (Bush 1945), the development of the mouse and bitmapped interface (Englebart 1968) and the widespread availability of personal computers beginning in the early 1980s.

building trait has been suggested as the defining characteristic of humans (Leakey 1981). It is also clear that using technology has always had implications for the context of human activity; a flint arrow-head or a mobile phone will both affect the society that adopts them. However, the extent of reliance on technology, particularly its intervention in terms of perceiving and creating the exterior, out of body, world suggests that technology may challenge both our paradigms of understanding human activity and the extent to which existing frameworks of explanation may be altered.

> "Perception is an active process of meaning construction involving not only visual perception, but all the senses together with the total physical environment in which the body is situated… by learning new skills and utilising technology, humans change the world they live in." (Broadhurst 2007:3)

If, on the one hand, this degree of intervention suggests that we may legitimately re-examine existing activities such as performance, one might also point to existence of a considerable body of work involving the combination of multimedia with live performance as an indication of a coherent domain suitable for (and even requiring) academic reflection.

> "The existing field of "dance technology" is one with many problems. It is a domain with many practitioners, few techniques and almost no theory; a field that is generating "experimental" productions with every passing week, has literally hundreds of citable pieces and no canonical works; a field that is oddly disconnected from modern dance's history, pulled between the practical realities of the body and those of computer art and has no influence on the prevailing digital art paradigms that it consumes." (Downie 2005:2)

It is not only in dance that creative artists working in performance traditions have been using digital technologies to extend and transform their individual palettes of expressive resources with inevitable consequences both for their own, individualised practice and for the identity and conceptual boundaries of the performance tradition(s) they work within. This process is perhaps most marked, and probably occurred earliest, in music[23] (Dixon 2007:x). Laurie Speigel's*Music Mouse* (1986) was the first serious attempt to use the newly available[24] GUI-based computers as an instrument. Before this, John Bischoff and others linked their computers to develop the musical performance traditions of the *League of Automatic Music Composers*[25]*,* "the first network band in history" (Jordà 2005:1) and later formed *The Hub* (1986-1993) to develop and expand these ideas of placing interactive networked digital technology at the heart of their activity. The process has developed tremendously since these early beginnings and digital technology is now firmly established in all areas of musical practice (Duckworth 2005, Willcock 2006, Magnusson &Mendieta 2007, Hugill 2007).

[23] Possibly because musical practice has always involved a strong element of technological innovation and intervention channelled through the search for more effective instruments. However, it has not stopped there; there has been considerable interest since antiquity in musical automata, systems for the production of musical performances without the involvement of human agency through the use of water and wind power, clockwork and, latterly, digital technology. See Riskin (2003) for discussion of Jacques Vaucanson's celebrated 18th century musical automata.

[24] Apple introduced the Macintosh in 1984, Microsoft's Windows 3 appeared a year later. There were earlier machines in universities and research centres (Xerox's Paulo Alto) which used interfaces based on windows, icons and a mouse-driven pointer (e.g. the Xerox Star of 1981) but these were not widely available commercially (the Star was very expensive and aimed at large business users).

[25]*The League of Automatic Music Composers* was formed around 1976-7 by John Bischoff, Rich Gold, Bill Hopkin and Jim Horton. It was based at the Center for Contemporary Music, Mills College, California.

Multimedia usage in performance was initially seen as part of a domain that was delineated by genre or performance tradition rather than one drawing on principles and practices that were common across a range of different types of performance. Indeed, this perspective is still the dominant mode, or at least starting point, for many authors due perhaps to their having backgrounds rooted in a particular performance specialism such as music (Nagashima 1998, Duckworth 2005), dance (Laziza 1996, Birringer 1998a,b), performance art (Hill & Paris 2001) or theatre (Dixon 2007), rather than an outlook grounded on factors like technology (Dannenberg 2002), digital innovation (Gershenfeld 1999) or critical theory (Goulden 1998) which might be more universal across the range of activity. However, as the range of media types involved in individual artists' performance practice expanded, there were obvious areas of overlap and increased opportunities for collaborative working; artists and commentators with different backgrounds and specialisms gradually began to identify common aspects to their practice. There is also the factor noted by Packer and Jordan that,

> "Because computer output can mimic traditional media, it lends itself to artworks that blur the lines between media and between disciplines, just as in consciousness the distinctions between different media forms (image, text, sound, movement) are less than absolute." (Packer and Jordan 2001:xxxvi)

Lévy (1999) goes further and suggests that the promotion of collaborative working is one of the side effects of networked digital technology itself. Broadhurst (2007) takes an approach grounded in aesthetic theorising and identifies digital performance as a coherent and legitimate domain of enquiry in

its own right precisely due to the effects of combining technology and the physical body:

> "It is also my belief that tensions exist within the spaces created by the interface of body and technology and these spaces are 'liminal' inasmuch as they are located on the 'threshold' of the physical and virtual. I am suggesting that it is within these tension filled spaces that opportunities arise for new experimental forms and practices." (Broadhurst 2007:186)

For Broadhurst, these 'new experimental forms' are still entirely connected with the performance and theoretical practices that they developed from; they are perhaps a technological augmentation or prosthetic enhancement of parent genres rather than an entirely novel field:

> "in analysing digital practices it is not my intention to coin a new genre of art and performance; rather, I aim to focus on a variety of practices that utilize sophisticated new and existing technologies." (Broadhurst 2007:2)

Dixon (2007:x) defines "digital performance" carefully and takes perhaps the broadest view of the field; his definition includes gallery-based installations, performance art and even games in which performance is a central aspect of their content or form as well as those performance modes[26] which are generally accepted by other authors approaching the field from the viewpoint of the parent performance traditions. However, Dixon's conception of the field also appears to include digital recordings of performance on optical media where the content constitutes,

[26] Digital Performance is subtitled 'A History of New Media in Theater, Dance, Performance Art and Installation, the author explains that music "has arguably been more radically revolutionised by the 'digital revolution' than the other performance arts we explore" (Dixon 2007:x) but has not been included in their study because of a lack of expertise and an already wide focus.

> "…a focus on a moving, speaking or otherwise 'performing' human figure". (Dixon 2007:x)

In this view, he is diverging sharply from that of most authors writing about technology who (often unconsciously) adopt an 'in the moment', live performative standpoint, and moves closer[27] to theorists such as Philip Auslander, who, separating the ontological from the phenomenological, argues that,

> "the playback of a recorded performance should be understood as a performance itself" (Auslander 2009:81)

At the end of the discussion of their subject area, Dixon appears to again depart (or perhaps prepare to accept the future possibility for departure) from the consensus-view (as articulated by Broadhurst above) that multimedia and live performance is necessarily an extension of performance. He concludes the preface with a quotation[28] about digital performance from the (online) Digital Performance Institute;

> "And we would argue that it is potentially a new paradigm in theater and performance. Not yet, certainly. We don't claim to have produced or witnessed particular examples of digital performance that deserve to be more than interesting experiments in production techniques. In that sense, digital performance as a real transformation of the field has not yet happened, and may never happen." (Reaves 2003)

Dixon's aspiration for a field that may go so far as to cut itself off from its roots in human performance to bring about "new paradigms, genres, aesthetics and

[27] I wouldn't want to overstate this affinity, Dixon later explicitly distances himself from Auslanders' ontology of performance – see *Digital Performance* pages 127/8.
[28] Only the first sentence is quoted in Dixon 2007:xii. Given the range and vitality of the work surveyed by Dixon, Reaves' subsequent qualification seems rather timid; by 2007, the body of performance work which incorporated digital technology was certainly more than merely 'interesting'.

interactive experiences" (Dixon 2007:xii) should be seen as an extreme view (at the present time). However, what is clearly and increasingly shared by all authors writing about this area is a conviction that the combination of technology and performance is a coherent and cohesive discipline which requires both further creative exploration and intensive study. Most authors identify the root-practices of this discipline as arising from dance, music and theatre, although increasingly, gallery installation is identified as a closely related field of activity[29].

2.1.3 Modelling Performance

If the rationale for a dedicated scrutiny of the combination of digital technology and live performance has been at least provisionally established, one can then proceed to ask questions about its nature and features; what is the epistemology of such an area and how can its connections, mechanisms and manifestations be understood? In approaching a 'theory' of digital performance, some authors adopt a broadly materialist approach based on clearly identified elements while other authors use an approach based on theories of embodiment, cultural theory or nueraesthetics. Some accounts arise from considering a single project or series of projects, while others have a wider evidence-base where performance is delineated by genre. General, explicit attempts are rare (i.e. where authors propose a theory of performance or of performance ontology) but implicit models, where the assumptions about the

[29] So, for example, the Digital Stages festival, held in London in April 2011 (http://digitalstagesfestival.co.uk/) included an installation alongside works which took conventional performance disciplines as their starting points.

domain are encoded in tools and other resources designed for use in that domain are more widespread. This encoding, the extent and depth of underlying assumptions made about the nature of performance, is particularly important when approaching the evaluation of software systems designed for live performance user (see section 2.4).

Nagashima (1998, 2002) describes his PEGASUS system, the design of which is clearly based on an implicit model of multimedia and live performance, “The best account for the concept of multimedia art…” (Nagashima 1998:1). However, this is an account that is heavily influenced by the author’s creative preoccupations in terms of the taxonomy of performance:

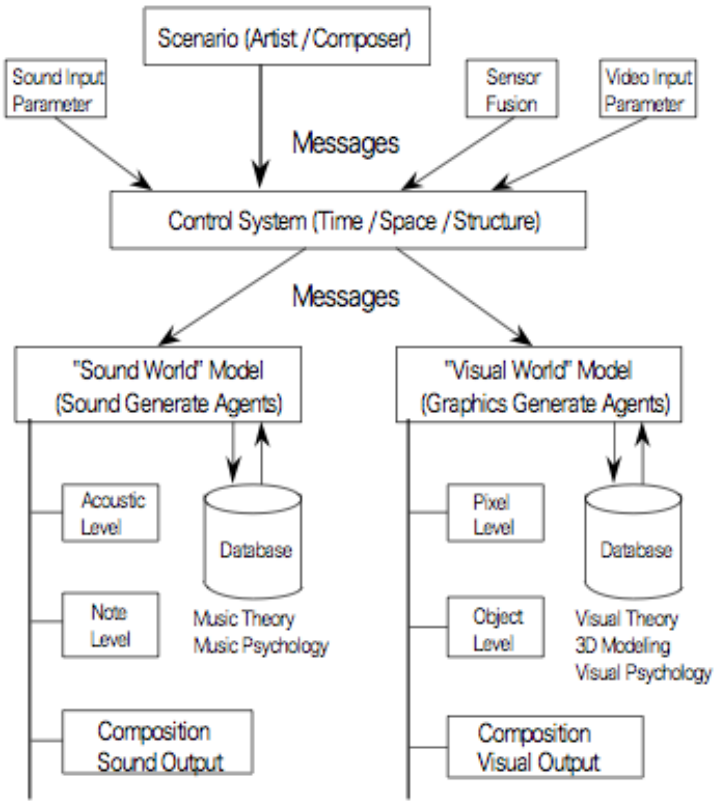


Figure 2 (Nagashima 1998:2)

This limitation is shared by almost all accounts that are based on technical development (Dannenberg 2004); the quality and potential universality of the performance-model is dependent on the design-parameters of the system being developed:

> "The intelligent stage is being equipped with a matrix of floor sensors for object localization, microphone arrays for sound localization, beam forming and motion capture system. ARIA system provides an interface for specifying intended mappings of the sensory inputs to audio-visual responses. Based on the specifications, the sensory inputs are streamed, filtered and fused, actuate a controllable projection system, sound surround and lighting system." (Peng& Al 2004: Abstract)

Camurri and Ferrentino (1999) have a broader and more ambitious idea of the constituent elements of performance that a multimedia system needs to engage with:

> "A typical sample scenario regards an integrated system which is driven, tuned and molded by the movements and by the sounds produced by the user(s) (actors, dancers, players), using specific metaphors for reaching, grasping, turning, pushing, navigating, playing, communicating their internal state and emoting potential, etc." (Camurri&Ferrentino 1999:33)

Their 'Multimodal Environments' (MEs) are clearly highly similar to Peng and Al's (later) 'intelligent stage'.

A much more open and general account, although confined to one aspect of musical performance and far less detailed in terms of technology, is Jordà 2005, particularly section 2.1 where different taxonomic approaches (principally those based on physical characteristics: time, distance) are discussed before specific examples of works are described very briefly (so briefly indeed that the major

contribution of the paper remains its summary of taxonomic approaches). This 'theory-exemplar' approach is followed by one of the best sources currently available on multimedia and live performance, Broadhurst 2007. She begins by setting out the theoretical frameworks available for situating and analysing digital performance and then moves on to examine particular instances of applications through a series of in-depth case studies on individual artists (STELARC) or groups (Troika Ranch). She very methodically (and persuasively[30]) begins by setting out what she feels are the defining characteristics of digital performance:

> "An important trait of the following digital practices is the centrality of non-linguistic modes of signification… The sublime is also central to such an analysis… There is also an accentuation of the chthonic or primordial, though this is not a feature shared by all digital practices. Other aesthetic features are heterogeneity, experimentation, indeterminacy, fragmentation, a certain 'shift-shape style' and repetition. Also there is the free use of 'defamiliarizing' devices, such as the juxtaposition of disparate elements, that… cause the audience to actively participate in the activity of producing meaning." (Broadhurst 2007:10)

While many of the features Broadhurst identifies could be said to be common features of much cultural production in late modernism (primordial energy, experimentation) or post-modernism (fragmentation, repetition), there are also clear resonances with Packer and Jordan's definition of multimedia discussed

[30] I do, however, take issue with part of the supporting arguments where she states (possibly following Merleau-Ponty's theories of embodied perception),

> "language without the body does not 'mean' at all, as corporeality provides language with meaning under sociocultural and thus temporal constraints" (Broadhurst 2007:1)

This, to me, seems probably more convincing to a dancer than it might be to a musician – and certainly would be to a mathematician. It is, of course, clearly true that a great deal of language (and perception) does achieve meaning through corporeality – but not formulae and possibly not (at least some) music.

above (juxtaposition, the active construction of meaning by the user, multiple modes of communication). The latter half of Broadhurst's book (the case studies) successfully combine broad aesthetic and cultural analyses with occasionally detailed technical discussions (e.g. Broadhurst 2007:109), a breadth of coverage which most sources focusing on multimedia use do not achieve (Dixon 2007 is a notable exception). While not explicitly dealing with digital performance as such, Corness (2008) proposes a model of performance with obvious implications for the integration of technology through concentrating on the qualities of (human to human) interaction and consequent event-production and decision-making within performance.

As well as these explicit accounts of performance, there are possibly an even greater number of cases where a model of live performance has underpinned a technological or creative development process, but has not been explicitly articulated; the domain's ontology has been assumed within the functional specifications of the work, device or application being developed.Between these two positions, algorithmic approaches to creativity (particularly prevalent in the domain of musical composition, but also applied to performance by composers from a range of backgrounds including IannisXenakis, KarlheinzEssel, Barney Childs and Brian Eno) occupy a middle ground where the intention is to produce a structure that generates a live performance, but which, in some cases at least, also articulate and codify their creator's epistemological views of live

performance, what it is important or necessary to specify and how these specifications should be seen as relating to one another[31].

In systems designed to facilitate collaborative networked performance such as Auricle (Neuhaus 2004), Quintet.net (Hajdu 1999) and Sensemble (Aylward and Paradiso 2006), there is typically a strong emphasis on the material phenomenology of performance; an underlying model of performance as comprising strands of gestural, sonic, MIDI or visual material which, when considered by other performers (possibly with reference to a score), will influence their future individual activity.However, the ontologies of their implicit models tend to be partial; the mechanisms and principles of the decision-making about the moment-to-moment onward progress of a performance are mainly assigned to the performers and are not articulated formally.A more complete formal model, the 'KTH rule system for musical performance' (Friberg, Bresin, and Sundberg2006) has a rule-based model which is able to predict the ways performers and a score interact to produce both the material and formal aspects of an individual performance[32], Hähnel (2010) describes a stylistically limited rule based model with very clear links between parameters (epistemologies) and model.

[31] This tendency is of course variable; performance-producing pieces such as Stockhausen's *Aus den siebenTagen* (1968) could perhaps be described as cultural or psychological (to the extent these differ) algorithms, but they mainly articulate more of the composer's model of *preparation* for performances rather than suggest the constituent (sonic) elements of the performance.

[32] The system does also have an impressive associated range of digital applications so that researchers are able to validate particular model configurations through producing performances and influence the progress of a performance in real time.

Implicit models of performance may also be codified in software and hardware designed for use by performers (MIDI), or for the production of performance-tools (Isadora, Max) through their enumeration of data or input types or the structural and functional flexibility (or otherwise) they allow in a performance. Marc Downie, of the performance and technology ensemble Open Ended Group, has suggested (Downie 2005:347-354) that some software and hardware systems can have a negative, narrowing effect on the variability of live performance, as they may channel artists to work in particular ways (and especially ways that are similar to those employed by the original designers of the systems) and also (particularly earlier versions of Max) can make changes to configuration during performances difficult.In a similar way, but on a smaller scale, the designers of MIDI clearly saw timbral transformation within an individual note as an unimportant part of their model of musical performance, perhaps a keyboard rather than string viewpoint.

Almost all the performance systems mentioned above have static configurations whose operating parameters are varied, but only within the limits set by the particular patch or configuration being used. It is important to recognise that this is, in itself, a result of implicit assumptions about the structural nature of performance. An interesting and powerful exception is ‘Field’ developed by Marc Downie, which can be reconfigured during performance.

2.1.4 Performance Practice

As has been suggested, most of the existing surveys of practice fall into two categories: those using a small number of detailed case studies, often based around individual artists' work (e.g. Broadhurst 2007, Hill and Paris 2001 or the more 'multimedia' Föllmer 2001 – the 'artists layer') or those that adopt a longitudinal approach, following a single artist or project over an extended period of time through a number of creative iterations or project versions (Downie 2005, Nagashima 1998, 2002, Dannenberg 2002, 2004, 2007). Both these methodologies have the clear flaw that conclusions drawn from any single study cannot be easily generalised with any claim for validity; they may be useful and insightful about the particular situation they examine, but will tend not to allow the scholar to apply that knowledge to other creative situations (unless they are very similar or, as in the case of longitudinal studies, part of the same ongoing creative process). There can also be a reluctance to critically evaluate software or hardware resources used by practitioners (although see Broadhurst 2007:109, 112) or to compare alternative ways of achieving the functionality used (although see Downie 2005:347-354 for a particularly thought-provoking critique). Other than these examples, current artist or practice centred accounts are generally unsatisfactory in giving any authoritative or objective analysis of resources; the tools used are treated as a 'given', part of the paradigm within which practice takes place and outside that portion of experience that the researcher can consider. Two important exceptions are the work of Johannes Birringer, particularly *Performance, Technology, and Science* (2008) and Steve Dixon's magisterial and

comprehensive*Digital Performance* (2007), particularly chapter 9. Here, Dixon follows a number of significant software products through successive development and applications within performance. The advantages of this approach include the ability to provide a coherent account of a time-based process involving many participants:

> "The development of original software and hardware systems by dance artists has been somewhat fragmentary and individual" (Dixon 2007:206)

Dixon's approach to resources is primarily phenomenological; he discusses software mainly in terms of the effects it provides in performance[33], but because his range of reference is wide, this approach provides perhaps the best account that attempts to comparatively evaluate individual practices. Importantly for this study, Dixon's is the only academic account which includes a (relatively short) discussion of open source and collaborative software development for use by artists in the performance field (Dixon 2007:206).

Birringer looks at artists from a range of creative disciplines, but filters study of their work through a categorisation that is, initially at least, primarily functional.

> "This book… investigates mediator technologies and probes their results in the realm of cultural production… of the practical consequences for the ways in which we work as performers, designers, programmers and composers…" (Birringer 2008:xiii)

Birringer divides his account using five categories (movement through technologies, the interactive paradigm, digital environments and wearable

[33] Dixon does additionally discus some of the metaphors behind particular applications – particularly Max and Isadora and also makes some important points about the technical skills required for artists working in digital performance which are discussed in section 4.3.1.

spaces, artificial intelligence and biotechnologies) and these could be seen as a means to group artists through the types of technology they are primarily associated with[34]. Making sense of the complex domain that is labelled performance practice incorporating digital technology in this way might be seen as attractive since there are a clear rationale and methodology for the taxonomic process and, further, the problems of making comparative studies across a range of essentially heterogeneous phenomena are reduced by looking at that one certain common factor (digital technology) and using that as the organising principle. However, as I have shown, this system on the face of it, would be a poor fit with commonly accepted definitions of multimedia – which see the term as necessarily involving multiple features. Because of the multi-faceted, overlapping and pervasive nature of digital technologies applied to performance, any account of the field that relies on cataloguing technical resources alone cannot be sufficient; there is rather a need for taking account of technology, but at the same time allowing the preoccupations and purposeful activity of practitioners to be accounted meaningful and to take their place in a survey of the field. The following three sections (2.2-2.4) take this approach at examining previous practice. Section 2.2 looks at software tools using in live performance while 2.3 discusses some previous approaches which are not focussed on a specific tool, but rather on generic approaches to development. Section 2.4 then presents a summary of the results of a series of interviews with

[34]Birringer's project is both more subtle and much more ambitious than this might suggest; his book is actually an exploration of a philosophical framework for understanding digital performance and its significances, and his categories (and their sub-divisions in individual chapters) are thus part of a sustained argument about the aesthetics of technology and 'human sensory activity' (where specific artists and works are used as exemplars) rather than a survey of practice.

practitioners about their practice based on both resource-categorisation and examinations of purpose.

2.2 Applications Used in Live Performance

The survey presented below covers the applications which are currently used by practitioners working with multimedia in live performance. It aims to be extensive, but not exhaustive since it does not include bespoke systems, i.e. those created and used exclusively for a single performance or project. It also does not include enabling low and middle level frameworks (e.g. TransJam, MidiShare) except where these have been designed specifically for live performance (e.g. JunXion, Mrmr, OSCulator) or applications with other significant areas of application and whose use in live performance is limited (e.g. most sequencing and audio workstation applications) or packages involved in out-of-performance activity such as preparing assets or documentation[35]. It also does not cover systems which have fallen out of widespread use (e.g. Life Forms[36]) or which have been discontinued (Image/ine[37]). I have also not included tools designed for the analysis or preparation of performances where the application is not used in the actual live presentation itself (e.g. Whatever Dance Toolbox[38], Life Forms).

---

[35] However systems used to produce applications for live performance (Max, Isadora, vvvv) are included.

[36] Life Forms is still commercially available (http://www.charactermotion.com/index.html) and is at version 5, and appears to still be in active (if slow) development.

[37] Image/ine was a Mac-only, real-time visual mixing system produced by STEIM (http://www.steim.org/steim/index.php) from 1997 onwards which could use a wide range of input types including live video streams. Its last available version (1.3.4) appeared in early 2002. Tom Demeyer, the lead developer, announced a port to OS X (ImX) in early 2005, but based on evidence from pages stored by the Internet Archive, development appears to have stopped at version 0.96 in November 2006 and doesn't now appear to be available anywhere online (see http://web.archive.org/web/20090621062902/http://www.image-ine.org/download.html).

[38] The Whatever Dance Toolbox (Turing 2011) has been developed by BADco and Daniel Turing and was publicly released in April 2011 as a free download. It is designed for use in rehearsal so that,

> "dancers can manipulate the image of movement and work with an 'active mirror' to produce qualities that they cannot produce on their own. Body is placed inside a different relation to its environment, which, in turn, determines and changes its expressiveness.

2.2.1 Criteria for Evaluation

In trying to develop an account of software and hardware resources used in live performance, establishing the criteria for evaluation are key. Downie, as might be expected given his rigorous and detailed approach, is explicit about the criteria he will use:

> "This section will begin with a brief survey of the common principles behind the graphical environments. This will not be a critique of their implementation details, their stability, their processing power, but rather of what it is they set out to do and the affordances they offer to artists who come to them."
> (Downie 2005:348)

and:

> "While they may be sold (and even taught) on the basis of how rapidly they create potential what I will ask of these tools here is how they interact with, navigate and manage the potential of interactive media." (Downie 2005:349)

However, despite the apparent clarity and laudable ambition, it can be argued that Downie is not being entirely consistent here in applying his chosen criteria. From even a preliminary, informal consideration of the experience of working with technology in live performance, it is clear that ease of use, platform requirements, stability and capacity to perform at reasonable speeds are also very much part of the list of desired 'affordances' that artists seek in a performance resource. While Downie does rise far above most of the current literature, he is still (understandably) focussed on his own development process with Field[39] and on his own needs and skills as an artist.

Tools employ visual analysis, tasks and temporally manipulated reproduction of captured images to allow dancers and choreographers to study and complexify their movement and composition." (http://badco.hr/works/whatever-toolbox/)

[39] Previously known as 'Fluid'.

Another attempt to set out explicit criteria for evaluating software tools was that of the *Software for Dancers* project (see section 2.3). Scott deLahunta describes the participants' initial discussion and approach:

> "…we were working together across a range of different approaches, the discussions were initially dominated by top-down procedures that generated questions such as - what do we want the software to do… what would its purpose be? What problems could we come up with that a piece of software could solve? Alongside this, we worked through a process of selection and elimination of various possibilities. We established some additional parameters at the outset such as the software should run on a standard (off the shelf) portable computer with only mouse, keyboard and audio video input." (deLahunta 2002a)

In examining the available software tools for artists working in live performance, the current study uses as criteria those themes of concern and interest which have arisen from the survey of practitioners (see section 4.1). They are recast (refactored?) here in the forms more usually adopted for software evaluation, but this is a semantic change, not one designed to remove them from their context in live performance. The criteria are:

Functionality – what can a program do and how well does it meet the needs of its target user(s)? These factors are not absolutes, but are rather complex functions both of asset-types and functionalities supported together with contributions from other factors discussed below.

Flexibility – ease and range of configuration and also the extent of the constraining effect of design assumptions on users (see Downie 2005:347).

These are not the same thing, although they are clearly inter-related. Flexibility also needs to be considered in different dimensions (cfJorda 2005); one needs to know about the range of possible applications for which a program may be used and how prescriptive it is in allowing a range of working methods. It is also important to consider the extent to which it may be recast during use – how easy it is to dynamically reconfigure the system while using it.

Extendibility – to what extent is the user locked into the available capabilities of the system as provided? Some applications allow no additional features to be added other than through upgrading to a more fully featured version, others have publicly available Software Development Kits (SDKs) which give details of how to write new system components which will integrate with existing code and allow anyone with sufficient programming skills to extend the program to meet their individual needs. Many programmes adopt a middle way where new functionality can be added through plug-ins, self contained software modules which are re-loaded whenever the application is started up and can be installed simply by placing them in a certain location on the user's computer.

Availability – how easy is it for artists to have access to the system? This clearly is influenced by such factors as price, licensing restrictions and the specifications of the equipment needed to run the software involved, but it is also highly dependent on the skills needed to use the package and how easily and quickly they can be obtained.

These qualities are clearly interrelated, so, for example, a package will inevitably tend to lose easy assimilation as it amasses complexity (particularly if the active assistance with the learning of a system is not made part of its design goals). The reverse is also true; as programs become easier to use, particularly through the adoption of invariant metaphors for working with performance, they allow less freedom to the artist.

> "With few exceptions, these popular graphical environments… are based on a common small reservoir of ideas: a few visual metaphors, a few structuring concepts. They each possess a surprisingly similar flavor and set of capabilities. So similar, in fact, that one might suspect that we are suffering from a digital art tools monoculture." (Downie 2005:347)

2.2.2 Software Tools Used in Live Performance

Application name	Main purpose	Media types[40] S – sound V – video I – images T – text M – MIDI	Interface, control inputs[41]	Scripting/ configurable?	Multiple performers?	Extendable	Supported platforms[42]	Connectivity	Licence type,availability	URL
Ableton Live	Live music performance and creation system based on samples.	S, M	GUI, control surfaces	yes[43]	No	yes	Mac, PC	Midi	Com	http://www.ableton.com/
Arkaos	Video sampler for live VJing	I, V, swf	GUI	No	No		Mac, PC	Midi, dmx	Com	http://www.arkaos.net/
Atlantic Waves	Network music performance and improvisation system	S, I, T	GUI		yes	no	?	tcp/ip	?	http://monolake.de/concerts/atlantic_waves.html
Aura (also AuraRT)	Real-time system for creating interactive multimedia	S, M (+I, swf?)		Yes	Yes		linux PC	tcp/ip MIDI	Free s/w	http://www.cs.cmu.edu/~auraRT/

[40] In general, no attempt is made to distinguish media file formats but 'swf' indicates Flash format.
[41] GUI means that there is a visual interface which can be manipulated by the mouse and keyboard. Control surfaces are sets of physical controls which output MIDI information mapped to elements of a software interface. Human Interface Devices (HID) are controllers such as joysticks that typically connect via USB (although older devices might use another serial connection such as RS232). Audio here means that an audio signal can actually be used to control the operation of the system (through pitch tracking or gesture control), not that the system can produce or project audio data. Wii remotes often require the host machine to have Bluetooth connectivity.
[42] The latest versions of operating systems are generally assumed (although Vista/Windows 8 support is not universal on PC packages). Java indicates that the package should run with a recent version (1.42+) of a JVM. 'Mob' indicates that at least some mobile computing devices are supported – check supported devices in each case.
[43] Ableton can be made configurable through scripting by using Max patches in its processing loop(s).

Application name	**Main purpose**	**Media types**[40] S – sound V – video I – images T – text M – MIDI	**Interface, control inputs**[41]	**Scripting/ configurable?**	**Multiple performers?**	**Extendable**	**Supported platforms**[42]	**Connectivity**	**Licence type,availability**	**URL**
Auracle	Voice controlled group instrument for real-time, interactive, distributed music making	S, T	GUI, mic	no	Yes	no	Browser-based Java	tcp/ip	free	http://www.auracle.org/
Chuck/Audicle/ miniAudicle	Audio programming language for real-time composition, coding & performance	S, M	command line, GUI, HID	yes	yes	yes	Mac, PC, linux	MIDI, OSC	Open source	http://chuck.cs.princeton.edu/
Director	Authoring interactive multimedia	S, V, I, T, swf, 3D	GUI	yes	no	Yes, plug-ins	PC, Mac	tcp/ip	com	http://www.adobe.com/products/director/
Elektronika (Aestesis)	Live VJ application	V, S, M, swf	GUI	no	no	Yes, plug-ins	PC	MIDI, tcp/ip	Open source	http://aestesis.eu/elektronika/
EyeCon	Live motion and gesture recognition	V, S, M, swf	GUI, Video	no	no	no	PC	MIDI, OSC	com	http://eyecon.palindrome.de/
EyesWeb	Supporting research on multimodal expressive interfaces and multimedia interactive systems.	S, M, V, I	Video, GUI, control surfaces, HID	yes	yes	yes	PC	MIDI tcp/ip OSC DCOM serial	Open source	http://www.infomus.dist.unige.it/EyesWeb/EywPlatform.html

Application name	**Main purpose**	**Media types**[40] S – sound V – video I – images T – text M – MIDI	**Interface, control inputs**[41]	**Scripting/ configurable?**	**Multiple performers?**	**Extendable**	**Supported platforms**[42]	**Connectivity**	**Licence type,availability**	**URL**
Field	Authoring system creation package for live performance applications	S, M, I, V, T	GUI, others	yes	yes	yes	Java	Dmx MIDI tcp/ip OSC	Open source	http://openendedgroup.com/field/wiki/BetaBinaryRelease
Flash	Authoring interactive multimedia	S, V, I, T	GUI, Video, Sound	yes	no	yes	Mac, Pc, mob	Tcp/IP	com	http://www.adobe.com/products/flash/
Isadora	Authoring interactive live performance systems.	V, S, I, T	GUI, Wii, Control surfaces	yes	no	yes	Mac, PC	MIDI, OSC, serial	com	http://www.troikatronix.com/isadora.html
JunXion	Interface Data Routing	M,OSC, V, S	HID, Wii, control surfaces, Arduino	yes	no	no	Mac	MIDI. OSC usb serial	com	http://www.steim.org/steim/junxion_v4.html
LanBox LCX	Network DMX interface/controller	M, various data formats	Control surfaces	Yes[44]	n/a	no	Mac, PC	MIDI tcp/ip udpu sb DMX	com	http://www.lanbox.com/lcx.php
LiSa	Real-time audio manipulation environment	S	GUI, Control surfaces	yes	no	Plug-ins	Mac	MIDI	com	http://www.steim.org/steim/lisa.html

[44] The unit comes with a custom editor application, LCedit+.

Application name	**Main purpose**	**Media types**[40] S – sound V – video I – images T – text M – MIDI	**Interface, control inputs**[41]	**Scripting/ configurable?**	**Multiple performers?**	**Extendable**	**Supported platforms**[42]	**Connectivity**	**Licence type,availability**	**URL**
Max/Jitter	Graphical programming/ authoring environment for multimedia performance tools	S, M, V, I	GUI, control surfaces, HID, serial	yes	yes	yes	Mac, PC	MIDI, tcp/ip OSC	com	http://www.cycling74.com/products/maxmsp
Module8	Video mixing and compositing for VJ and performers	V, S, I, T, swf	GUI, control surfaces	yes	yes	yes	Mac	MIDI DMX	com	http://www.garagecube.com/modul8/index.php?c=home
Motion Dive	Video sampler/VJ system	V, S, T, swf?,	GUI	no	no	Plug-ins	Mac, PC	none	com	http://www.digitalstage.net/en
Mrmr	Live performance control system for mobile devices		GUI	yes	n/a	yes	Mob	OSC	Open source	http://mrmr.noisepages.com/4-2/
OSCulator	Interface Data Routing	M	GUI, Wii, HID, iPhone, Control surfaces	yes	no	no (?)	Mac	MIDI OSC	com	http://www.osculator.net/
Pd (Pure data)	Real-time graphical programming environment for multimedia	S, M, V, I T	GUI, control surfaces, HID	yes	no	yes	Mac, PC, Linux	MIDI tcp/ip OSC serial	Open source	http://puredata.info/

Application name	**Main purpose**	**Media types**[40] S – sound V – video I – images T – text M – MIDI	**Interface, control inputs**[41]	**Scripting/ configurable?**	**Multiple performers?**	**Extendable**	**Supported platforms**[42]	**Connectivity**	**Licence type,availability**	**URL**
Peersynth	Interactive, networked multimedia performance environment	S, M, V	GUI, control surfaces, HID, audio	no	yes	Plug-ins	Mac, PC.	MIDI OSC tcp/ip	free	http://www.quintet-net.org/info-frames.html
Processing/Wiring/Arduino	Programming language and environment for people who want to program images, animation, and interactions	S, I, M, V, T	GUI, HID. serial	yes	no	yes	Mac, PC, Linux	tcp/ip MIDI OSC serial	Open source	http://www.processing.org/ http://www.wiring.org/
Qunitet.net	interactive, networked multimedia performance environment	S, M, T V	GUI, MIDI	yes	yes	yes	Mac, PC	tcp/ip OSC	free	http://www.quintet-net.org/dl-frames.html
ReacTIVision	computer vision framework	V	Video, multitouch	yes	yes	yes	Mac, PC, Linux	tcp/ip MIDI	Open source	http://reactivision.sourceforge.net/
UpStage	Interactive, Multimedia,web-based performance environment	S, I, T, V, swf	GUI	yes	yes	no	Browser (Linux for server)	tcp/ip	Open source	http://upstage.org.nz/blog/?page_id=2

Application name	**Main purpose**	**Media types**[40] S – sound V – video I – images T – text M – MIDI	**Interface, control inputs**[41]	**Scripting/ configurable?**	**Multiple performers?**	**Extendable**	**Supported platforms**[42]	**Connectivity**	**Licence type,availability**	**URL**
Visual Jockey	VJ and real-time animation system	V, S, T?, I?	GUI, audio	no	no	Plug-ins	PC	none	com	http://www.visualjockey.com/index.html
vvvv	Toolkit for real-time interactive DV systems	V, S, T, I, M	GUI	yes	no	yes	PC	MIDI tcp/ip OSC DMX	Free, com	http://vvvv.org/tiki-index.php

2.2.3 Max and Isadora

There are two software packages which are so widely used in live performance and so influential and pervasive in their metaphors and workflows that they require a more detailed coverage in any study of the area.

Max(originally known as Max/MSP) was developed at IRCAM[45] in the mid-1980s for composers and musicians and then released commercially with a real-time signal processing capacity from 1997 onwards (Dechelle 1999, 2006). It is arguably the most influential and popular software resource currently used within live performance[46] and is probably one of the first applications specifically targeted at the live performance sector[47]. It has a range of competitor packages, some of which (Pd, JMax) began (to some extent) as direct clones, albeit with different licensing or platform requirements and others which have taken the 'visual programming' (more strictly data-flow programming) metaphor but adapted the functional content for a different domains (vvvv, Isadora and, to some extent, EyesWeb) or combined it with a different philosophy of programming and performance (Field).

Isadora(Coniglio 2008) was originally developed for use by choreographers in 1999 by Mark Coniglio (deLahunta 2002b). Developed after Max, it shared a programming metaphor (that of the patch-cord to make connections between functional blocks) and a concern with the manipulation of real-time data. Unlike

[45]Institut de Rechercheet Coordination Acoustique/Musique, located in Paris, France.
[46]Downie calls it 'canonical' (2005:350)
[47] Even before it had real-time processing capabilities, it was capable of controlling devices in real time.

Max (or Downie's Field[48]), Coniglio's target users were not programmers, rather he attempted to build a system that allowed choreographers to work with functional programme blocks in the same way they would work with dancers in rehearsal: in an experimental, iterative process.

The visual programming approach of many systems is designed to provide non-programmers with access to flexible processing systems. However, as Downie notes (see quote above), this facility comes at the price of flexibility. I would also argue that Max does not even do visual programming well from this standpoint – it is considered notoriously difficult for non-programmers, although once users have mastered it, they are extremely enthusiastic[49]. The system is undeniably extremely powerful, particularly when combined with the Jitter video processing extension[50]. However, it does have problems with its interface and with temporal flexibility. The interface is typically limited to a representation of the system components[51] (see Downie 2005:350-353) and the data-flow through them. For those struggling to understand complex programs, this can be a liberating experience. However, it does also have severe limitations; there is little immediate sense of hierarchy or of modularity (despite the apparent limitations of the visual programming metaphor, these can easily be implemented – and indeed recent versions provide the ability to deploy Max

[48]Downie sees his target user as necessarily an 'artist-programmer'. (Downie 2005:399)
[49] While Max is a commercial product, it has a thriving open source community existing with the support of the manufacturer, Cycling 74. Thus while the program itself is not free, there is a huge amount of free software and advice available.
[50] This graphical power is extended in Max 6 (the latest version at the time of writing) through 3D and animation extensions.
[51] Experienced users can hide complexity in modules so that just user interface elements appear – but this is just a hack to counter an inherent weakness of the program design philosophy.

patches as modules within the Ableton Live processing/performance environment) or ability to separate the program data and logic from the interface controls[52]. Downie makes the further criticism that Max imposes a static view of performance upon what is essentially a dynamic medium (recent updates have to some extent answered these criticisms). However, while his warnings of a 'visual programming monoculture' do need to be taken seriously, for musicians used to applying (and exploring) instruments in and through performance, this criticism loses some of its force.

Having said this, Isadora does still manage temporal flow in performance better than Max; arranging the programmed 'instruments' in a series of 'scenes' is at once an accessible metaphor and at the same time an efficient and practical way of using multimedia in live performance situations[53]. Isadora has also always (unlike Max) allowed the programmer to split the view of the program between a functional, data-flow representation and a control-surface, a custom built GUI. While this is definitely very much better than not having a split as in early version of Max, it is hindered by the constraints of the input devices available – the mouse and keyboard are poor sensors for multi-channel gestural input[54]. Both Isadora and Max have sought to explicitly address the increasing demands of digital performance through additions designed to facilitate networked applications (e.g. via OSC) and working with specialist output

[52] This can be done by more advanced users, the point is that the program metaphor makes it difficult.
[53] Max now provides a patch mixer meaning that transitions can be managed; this feature was not available in early versions.
[54] This can be overcome to some extent through the use of the OSC or MIDI connection actors which permit the use of control surfaces. On the other hand, this then means that much of the customisation of the GUI is lost.

controllers (e.g.Lan Box's LCX[55] or Max's recent emphasis on physical computing[56]).

[55] http://www.lanbox.com/lcx.php
[56] http://cycling74.com/products/max/physical-computing/

2.3 Generic Application Development

In addition to the range of specific software tools considered in section 2.2, there are important projects whose aim has been to produce generic tools for use with performance. These need to be included in this survey of prior art since the current study positions the production of a generic tool as a central research goal. In this context, it is particularly important to define what a generic approach might be and to establish that such approaches to software design are legitimate in that they are an approach that is capable of meeting some of the overall research outcomes of the current study, particularly that of supporting further work in the field of multimedia and live performance.

While it has been suggested above that the current approach to considering creative work by searching for common features rather than unique characteristics is, if not entirely novel, uncommon, similar claims cannot be made for software development. Given that modern computers can be considered (Bolter 1984:114, Deutsch 1998, Davies 2000:189) as implementations of the completely generic Universal Turing Machine (Turing 1936:241), it could be argued that all software development represents a narrowing of computational effort so that it focuses on particular tasks in ways that suit the user. While different software development methodologies all place defining the range of specific requirements for a particular application as central to producing an efficient development process[57], there is an expectation that

[57] The articulation of this significance and its placement and repetition represent some of the major differences between methodologies but formally establishing needs in some way is common.

most software tools designed for use by non-specialists[58] will be used for a range of purposes often linked by a common media type or social or economic context. While this range may be broader or narrower in specific cases, the design process is of a fundamentally different nature to that of physical tools, where the starting point was a requirement to perform a specific task as well as possible. While there are more, and less, flexible (in terms of range of application) physical tools, it is reasonable to suggest that the nature of digital tools produces a digital design culture which is essentially different to those paradigms of physical design.

The characteristic features of this emerging digital design culture include wide design goals: a flexibility in intended applications of software where features may be added that are not immediately needed, but provided because they can be implemented or could be useful at a later point. Examples of this tendency include the many framework development projects (e.g. Away 3D[59], Processing (Fry &Reas 2001), ReacTIVision and Isadora). Another significant feature is an altruistic attitude to users; software is seen as empowering both those who will use it and other programmers, enabling them to develop their individual practice and contribute to the common good. The most significant example of this is perhaps the Open Source movement.

[58] Interestingly, programmers often code software tools for other programmers which do have a single or limited set of functionality. For example, the management and configuring of the Linux operating system is carried out using a standard suite of many small programmes, each of which does (usually) only one thing.
[59] Away 3D is an open source 3D engine for Flash, see http://away3d.com/.

Programmers have been identifying themselves as a social group, sharing slang, computer code and supporting each other ever since programming became an activity involving significant numbers of people.

> "From Eckert and Mauchly's first ENIAC computer onward there was a more or less continuous and self-conscious technical culture of enthusiast programmers, people who built and played with software for fun." (Raymond 2001:3)

The establishment of the Open Source Initiative in 1998 (History of the OSI, no date) represented a labelling and codification of an already well-established process for efficiently developing software. It had its roots in hacker[60] culture and the Free Software movement which had emerged during the early 1980s, with the development of personal computers helping the movement to grow outside of the professional programmer demographic[61]. While the motivations and business models associated with the Open Source movement are too complex to be considered here, two features are important for the current study as they validate central aspects of the approach and methodology related to the wider influences of digital technology and particularly to those two aspects of digital culture singled out above. Given the widespread success of the Open Source movement and the wide range of activities, both creative and commercial, that it supports, an approach to supporting creative work through the provision of a generic system can be related to a well-established, successful model of creative technology development and utilisation.

[60] Steven Levy identified this culture and popularised hackers in 'Hackers: Heroes of the Computer Revolution' (1984).
[61] See Raymond 2001, chapter 1, *A Brief history of Hackerdom*, for a detailed coverage of the events over this period.

There are two development projects that have taken place in the UK which transfer aspects of the Open Source ethos to digital performance; the Composers Desktop Project (CDP) begun in 1986 (and still active) and the Software For Dancers (SFD) research project of 2001. Both brought together artists who were working with technology, although importantly the CDP collaborators were mainly artist-programmers whereas those invited[62] to join the SFD project comprised dancers who used technology together with programmers who worked with dancers. Both projects began with the aim of providing software tools to support work in their respective fields.

> "We…seek to create a mutually supportive environment in which practising composers can help each other, both in designing software tools and in using them effectively." (Composers Desktop Project 2010)

The SFD project confined themselves to the choreographic and rehearsal phases of dance while although the CDP began with an emphasis on non-real-time composition (and this still remains a highly important focus of their software suite), it rapidly moved to include good support for live performance use involving MIDIGRID[63] (Hunt & Kirk 2003) and (even) real-time use of CSound. Both projects aimed to produce a set of generic software tools that would not be tied to a particular project or way of working, but the development methodologies were markedly different. The CDP provided a hardware/software

[62] The Software For Dancers project was grant aided by the Arts Council of England and was led by Scott deLahunta who invited four choreographers and five 'digital artists' and two writer/researchers to join in a series of research seminars (*Software for Dancers: Participants* 2001).

[63] The adoption of the Atari ST with its integrated MIDI In and Out was an important driver for the whole project, although until CSound was added in 1989, real-time synthesis was difficult. MidGrid was a particularly innovative programme; by 1988, it could map mouse events to multiple MIDI event streams including controller data. It is still available for PC, although active development seems to have ceased in 2007, see http://midigrid.com/.

framework based around a set of sound processing tools and the CMusic/CSound synthesis packages. Participants were invited to submit software they had written to the project[64] and, if it was considered good enough, it was added to the project's software toolkit and distributed. SFD took a different approach; a provisional specification for a prototype system (provisionally titled *Emergent Moves*) for use in rehearsal was drawn up based on:

> "discussion about dance making process between digital artists and choreographers to articulate requirements for Emergent Moves" (Software for Dancers: Basic Description)

This latter approach to software development was ineffective; the project stalled in disagreements,

> "In one breath, someone would identify a potential use for a certain software application in the dance studio; in the next moment this would be negated by arguments both pragmatic (e.g. the problem was solvable in a more efficient way) and artistic (e.g. fundamental formal contradictions could be named)." (deLahunta 2002a)

No consensus could be reached on what the features of a generic system might be and hence the plans for a prototype were abandoned[65]. In contrast, the CDP has produced a substantial set of software tools, some of which were originally developed by particular individuals for specific projects, but all of which has found applications across a range of practices and styles. It is not that an

[64] For example, I wrote Linedraw, a graphic tool for producing numerical data files for complex CSound synthesis operations. While it was submitted to the project and used extensively in my own work, it didn't make it into an official system release.

[65] This is not to suggest that the project did not have valuable outcomes. There were two interesting reflections on the relationship between creative art and programming by deLahunta (2002) and Sanjit Roy (2002). The project also established a number of key questions about considering the relationships with software used by non-specialists.

individual composer will find all of the suite useful or stimulating, but that there is such a range of applications, many only providing low level functionality (such as a synthesis or sound transformation environment) that many different creative artists have found something of value while being able to maintain their individual creative vision and characteristic practice. By trying to establish specific functionality at the outset, the SFD project were not able to establish the requirements for a tool useable by a range of practitioners, but by concentrating on frameworks rather than specific, tightly-defined functionality, the CDP has managed to have wide appeal. Indeed, as the summary of my practitioner surveys below suggests, specific usages of tools appears to be where the differentiation essential for the establishment of an individual style of digital performance resides, making it the least promising approach to establishing the requirements of a generic tool.

2.4 Summary of Practitioner Survey

Six in-depth interviews were conducted with subjects chosen carefully to represent a range of performance traditions;two each from music, dance and theatre. The interviews were conducted using the interview framework developed in an initial phase of the research (see section 3.2) and the average duration was about an hour. Only a partial summary of results, concentrating of the specifics of practice is presented here as a contribution to evidence for establishing current practices involving multimedia and live performance. For a more detailed presentation and analysis of results, see section 4.1.

All the artists surveyed from all performance traditions used video, either live or pre-recorded, in their work. However, it was also clear that how this video was used, particularly the specific manipulations and transformations employed, was an important part of what gave identity and specificity to each artist's work; the technological means were ubiquitous, but the detail of use was highly individual and characteristic. Analysis of all the mentions of specific tools by subjects identified Max as the most frequently cited and used with Isadora as the next most frequent. Analysis also identified a large number of resources which were specific to the practice of individual practitioners and which were not used by others, even from the same performance tradition. It appeared that artists had at least some resources that were significant for their practice because of specific functionality or processes which fitted with an artist's wider creative preoccupations or methodologies. Indeed, most subjects reported a marked reluctance to acquire new technical skills where these were not required for

their creative explorations. Despite all my subjects being enthusiastic adopters of digital technology, they all felt there was a difference in the roles of artists and technologists.

In examining the control and connectivity used by practitioners in digital performance, it appeared that binary triggering was much more common across all strands of practice than analogue, continuous control; things were overwhelmingly stopped and started rather than directed and controlled on a moment-to-moment basis. Much performance across all disciplines appeared to consist of overlapping, parallel segments of material, each of which was presentedonce initiated, more or less autonomously by performers or devices. While this was to be expected in theatre (it corresponds to the cue-effect model used for almost all theatrical performance), it appeared to be equally marked in other practices.

A consistent pattern of digital technology involvement related to parent performance discipline was tentatively identified with technology-use appearing to be most intense and thorough-going in dance and least integrated in theatre.

2.5 Chapter Summary

This chapter has summarised previous work in the field using a range of approaches. A systematic survey of academic accounts has established key concepts underpinning the enquiry and the digital performance domain as a coherent, appropriate field for enquiry. It went on to present and critically evaluate a range of approaches to considering practice and modelling performance. A set of criteria were established for evaluating existing software tools and a comprehensive survey of those currently available was presented together with more detailed discussions of the two most prominent software packages, Max and Isadora, which are used across the digital performance field. The examination of the current state of the art then considered collaborative software development, particularly that aimed at producing generic tools. The chapter concluded with a brief summary of the results of the practitioner survey which identified in outline some characteristics of current practice and provided an overview of the shared and individual pre-occupations of practitioners identified in the survey.

3Methodology

The rational (section 1.2) above established the generic theme and central hypothesis of my enquiry. In setting out to answer this and my other research questions, it was clear that a mixed methodology was required, one that included elements of enquiry practices drawn from both the humanities and from computer science. This chapter sets out the structure of the enquiry and then looks in detail at the construction of appropriate methodologies for the different phases of research. Some material in this chapter (particularly section 3.2.1) is based on the paper *Pinning Down the Artists: Developing a methodology for examining and developing creative practice,* presented at the 2005 De Montfort University Humanities Graduate Student conference and published in the conference proceedings the following year.

3.1 Project Overview

My research activity can be divided into two strands: ‘academic’ activity and software development which included a number of creative projects. Each of these types of activity interacted with the other and informed their progress and objectives at several different points in the overall process. This resulted in a fairly complex project structure (see Figure 3 below)

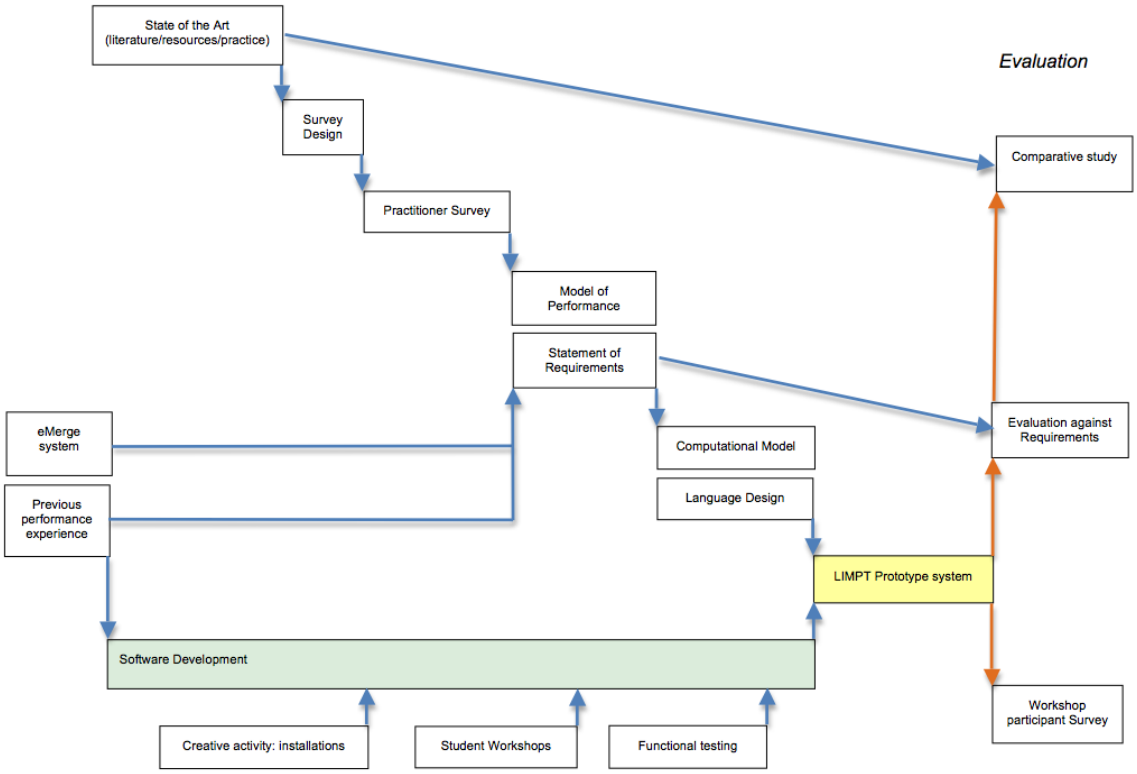


Figure 3: Enquiry structure

While Figure 3 does to some extent simplify the scheduling and interior detail of activities (e.g. the Practitioner Survey item includes both administering the survey and analysing results), it does indicate dependencies (the influencing of

later stages of enquiry by the results of intermediate outcomes) reasonably accurately.

There were three main areas where methodology was crucially important in ensuring high quality research: the construction, administration and analysis of the initial survey (since these underpin the formulation of the definitive statement of requirements for the system); the management of the software development process which ran throughout and actually implements the requirements; and the conduct of systematic and comprehensive surveys into previous academic work and currently used resources. Each of these areas are treated in detail below alongside justifications of the approaches taken to creative activity and the evaluative processes applied towards the end of the project.

3.2 Practitioner Survey

The rationale for the current study (section 1.2), with its emphasis on a generic approachand its scoping of the extent of the current study, underpinned the design of the survey. There is a wide variety of work being done by artists working with multimedia, both in conventional genres such as music, dance and theatre, as well as in newly emergent hybrid and innovative live-performance forms such as real-time animation and physically distributed and telematic performance. For the practitioner survey, I decided to sample practice based on the topic-identification derived from previous academic work in the field (section 2.12). This suggested that it would be necessary to interview practitioners working within performance traditions from each of dance, music and theatre if a representative picture of practice was to be established[66]. Given the highly individual nature of artistic practice, any claim for a fully representative sample of creative artists is going to be suspect and any findings require triangulation for conclusions to be defendable. I decided that choosing two contrasting practitioners from each performance background, thus talking to six interview subjects overall would, given the possibility of comparing findings from other sources, be adequate.

I was concerned to build up a picture both of which technical resources artists were employing and to what effect technology was being used, the technologies

[66] It could be argued that artists working on public, gallery-based installations should also have been included in the survey. I decided against this in the original study for two reasons, partly the time constraints on getting interviews finished so that results could be fed into subsequent stages of enquiry and also because my own creative activity throughout the project was concentrated almost exclusively in this area.

subjects were using and the purposes it was being used for. From the results, I wanted to be able to attempt to identify common features of usage: overlapping or congruent combinations of technology and intention that can be used to firstly establish the possibility for, and then propose the feature set of, a generic multimedia tool for use in live performance. Given the highly diverse nature of both the subject domain and of individuals' thoughts about their practice, I needed to devise an interview structure and associated epistemology which might capture the information I needed; simply asking, 'what do you do?' would be unlikely to produce reliably comparable data from individuals whose main focus was on producing individually characteristic creative work.

3.2.1 Survey Design and Epistemology

I chose to interview subjects so as to have a diverse sample of artistic practice and use of technology. However, this deliberate courting of diversity necessarily raises fundamental methodological problems for a study that seeks to compare practice and identify common ground. On the other hand, any attempt to restrict the sample (to make comparisons between works and practice easier) would actually weaken the study by making it impossible to identify genuinely generic features of multimedia usage, since any patterns which emerged might be features of a particular genre or style. The problem is further complicated, since if the study is to seek acceptable levels of authority and objectivity, meaningful combinations of practice and intention cannot be based solely on an epistemic, preconceived taxonomy, but the survey and its analysis must allow for

unforeseen categories of meaning to emerge from the evidence[67]. A suitable methodology must enable criteria for analysis to be based on connections or correspondences, made between divergent areas of practice, that were unknown (or even unknowable) at the point the study was originally designed. Another important requirement for an appropriate survey methodology was the ability to work with a range of data which may exhibit high degrees of variability in both extent and coherence.

While some data may be amenable to some pre-structuring (e.g. interviews will have prepared sets of main questions and some follow-up questions, see below), when looking at performances and their documentation, the structural rationale is usually derived from the work itself rather than from the data-validation requirements of an external study. Since the evidence will be drawn from 'real-life' rather than from carefully planned and tightly controlled experiments, there will be a requirement to be able to search for, identify and describe meaningful elements which, at the start of the study, are unknown and cannot be planned for.

As far as my research is concerned, 'meaning' is a quality of an item of evidence which renders it significant for my research aims, specifically, identifying what the features of a generic multimedia tool for use in live performance might be. A significant item then is a particular use of digital

[67] The framework actually changed very little during the course of the interviews themselves (some specific examples were added, but no structural changes were made), but as analysis proceeded, additional codings and categorisations emerged some of which suggested interesting avenues for future, further study, for example into relationships between working practices and use of technology and conceptualisations of 'space'.

multimedia in live performance which is identifiably different from, or is similar to, other uses of digital multimedia in live performance. However, the specificity of the evidence item is also important; noting that MIDI[68] is used by a particular artist is not, in itself, necessarily significant since, because MIDI can be used for such a wide variety of things, the simple fact of MIDI usage cannot be compared to any other usage of MIDI without more detail about each of the specific instances being recorded. This requirement for specificity is true for both the technical aspects of usage (i.e. floor pads using MIDI to connect to a control system) and for the purposes the technology is put to (i.e. MIDI used to allow performers to interactively control stage lights). In building up a picture of what artists are doing with multimedia in live performance, I need to understand both what technical resources practitioners are using and what creative needs are being facilitated by that technology. Clearly, the relationship between these categories needs to be both fluid and non-judgemental. One technological resource may be used for many different purposes; a single creative requirement can be realised using different technical means[69].

When looking at artistic practice, it is impossible, because of the nature of the work and the priorities of those carrying out the work (the artists), to expect to

[68]Musical Instrument Digital Interface. An early networking system specifying hardware, software and protocols which was introduced in 1983 to allow consumers to interconnect the new digital synthesisers that were becoming available. It is now widely used for other purposes as well – position sensors, stage lighting control and digital sound effects control. See http://www.midi.org/

[69] Crucially, the reasons for making choices about the most appropriate technology to serve a particular purpose may be based on a (partially) different set of reasons from those usually employed in systems development. Factors such as the familiarity of the artist with available choices or the importance of unintended implementation features (such as the importance of slow network connections in some works that derive performance parameters from network latencies) to an artist's concept may outweigh more common considerations of cost, elegance and reliability.

use study methods that are commonly applied to laboratory-based investigation or to those where subjects either have to, or are prepared to, change what they do to accommodate the needs of a study for consistently obtained, uniformly formatted data. A successful approach, must engage with the work and the practitioners in their own terms and be able to use those paradigms as frameworks for analysis and conjecture. An epistemology upon which a system of significance and hence an appropriate data collection and analysis methodology can be based needs to be dynamic, working from a tentative, preliminary set of significant conceptualisations about the use of technology while being open to extension and amendment as further data is obtained.

3.2.2 Survey Framework

The approach adopted for the survey-design was designed to retain the strengths associated with ‘open’ interviewing styles:

> “An open interview can be characterised as a form of interview in which ‘control’ is very much in the hands of the interviewee. The advantage of using this style is that, if it can be properly mastered, issues which are of *real* importance to the interviewee will be addressed.” (Berndtssonet al 2002:61)

However, it was also important to be able to provide some measure of coherence to the data obtained and accommodate the extensible set of meaning-categories described above. Each interview thus compriseda set of main and follow-up questions intended to enable subjects to provide detailed information about areas they have particular experience or knowledge of (Rubin & Rubin 2005:64), while not having to spend time on aspects that were not

relevant to their practice. Questions were focussed on a conceptual framework of nine use-categories, ways of thinking about multimedia use in live performance which subjects were invited to consider in relation to their own work. The list of categories was not intended to be authoritative or exhaustive. Both the structure and content were dynamic and were continuously redesigned based on responses to completed interviews (Rubin & Rubin 2005:62).The initial set of categories was intended only as a preliminary taxonomy to enable the study to commence in a structured and coherent way facilitating both data collection and analysis (see section 2.2.3).

The nine initial themes were:

- Gathering input from performers/users
- Gathering information about the environment
- Outputting and presenting information
- Controlling devices or systems
- Processing information
- Planning
- Remembering
- Generating material
- Connecting performers and/or audiences

The construction of this preliminary list of meaningful categories of data followed the approach suggested by Miles and Huberman (1994) for qualitative analysis. They recommend researchers draw up a provisional “start list” of codes created prior to analysis or even field work and suggest a basis for this:

> “That list comes from the conceptual framework, list of research questions, hypotheses, problem areas, and/or key variables that the researcher brings to the study.” (Miles &Huberman 1994:58)

My initial themes are a preliminary set of viewpoints rather than codes, ways of organising and conceptualising data as it is collected and as starting points for a more complex code-field hierarchy as it is analysed. See Appendix 1 for the materials used in surveys.

Each of these themes were then further expanded diagrammatically to serve as stimuli for the structured discussion:

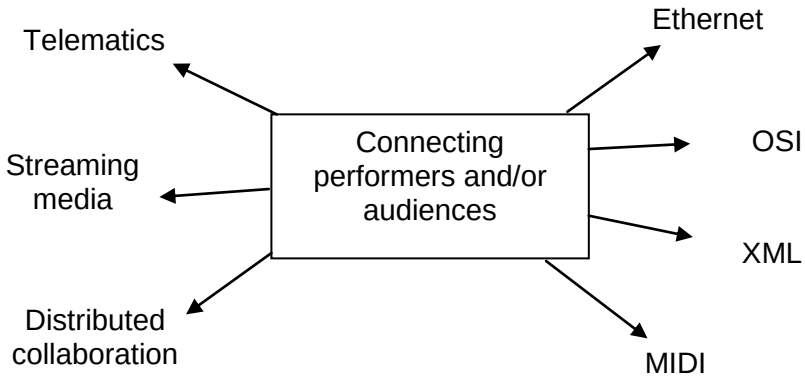


Figure 4 – Interview theme expansion

Each expansion contains a mix of example uses and specific technologies, designed to get subjects to talk about any incorporation in their practice from both viewpoints. This initial expansion served mainly to clarify the categorisation and to focus discussion on specific relevant points. It was expected that interview subjects would add novel exemplars as well as reproducing existing items[70].

[70] Subjects did add new items, although the extent of this augmentation was not as great as had been originally expected. One unforeseen effect was that almost all subjects found the suggested amplifications valuable as pointers for applications or technologies they might consider adding to their own practice.

Interviews began with a short description of the project and a formal request for participation (see section 2.3.4 for a more detailed description of ethics). The structured part of the interview began by asking the subject to describe briefly the work they had been doing. This was intended to relax the subject and also to allow them to apply their own descriptors to their experience which through comparison with that generated by the structured interview, could provide a secondary validation mechanism.The researcher then took the subject through the proposed list of areas for discussion. This was both to familiarise them with the themes and also to provide the opportunity for discussion about the basis for selecting and identifying elements as foci for consideration. As a part of this discussion, subjects were invited to identify additions or amendments they felt were required. Any additionswere added to the framework and used to help structure subsequent interviews. The final version of the framework was carried through to the analysis phase, underpinning the structured search for patterns of meaning in what the subjects said.

3.2.3 Analysis

Analysis of the interviews was based on a fairly standard qualitative methodology: detailed readings of transcriptions against an evolving set of significances, meaningful expressions that are looked for across the range of collected data. Each significant idea is represented by a *code*, a label which organises and identifies data for the researcher and which also permits

significant conceptualisation and manipulation of the patterns of meaning and linkage within data.

> "It is worth stressing here that codes are organizing principles that are not set in stone. They are our own creations, in that we identify and select them ourselves. They are tools to think with. They can be expanded, changed or scrapped altogether as our ideas develop through repeated interactions with the data. Starting to create categories is a way of beginning to read and think about the data in a systematic and organized way"(Coffey & Atkinson 1996:32)

There is a clear danger here that, in choosing the codes, the researcher may be choosing what they will find; that the meanings extracted from the data are artefacts of the researcher's projections upon the data rather than 'genuine' features of the data itself. There are two ways this possible subjectivity can be countered; the awareness and acknowledgement of the researcher of their own biases[71] and the reliance on meaning as derived not primarily from the codes themselves (which are chosen by the researcher) but rather from the patterns in the data that those codes allow the researcher to discern.

> "The point is not search for the 'right' set of codes but to recognize them for what they are: links between particular segments of data and the categories we want to use in order to conceptualize those segments." (Coffey & Atkinson 1996:45)

The framework of themes (section 3.2.2) used to structure the interviews was also used to derive a preliminary structured code-set. This was dynamic and evolved from very early on in the analysis, both in terms of low-level addition of

[71] The concept of the 'interpretive researcher' (Helen Wood, Principle Lecturer in Media Studies at De Montfort University) is important in qualitative analysis. Through immersion in the data and guided by previous academic work, one should build up confidence in ascribing meaning to patterns observed in the current data set.

additional codes to particular themes but also new high-level themes emerged as patterns and correspondences were followed and evaluated. This evolving process is covered in detail in section 4.1.

3.2.4 Ethics

As all the participants were well-known professionals, it was important to establish a basis for trust so that they would feel able to be candid about their use of technology; I was not only asking about successes, but also about failures, difficulties and intimate details of working processes that artists might wish to be hidden from a wider public. The first part of each interview started with an explanation of what the research project was and why the subject has been approached for an interview. The researcher detailedhow the interview would be conducted,the use to which data would be put and confirmed it would be stored securely. It was further explained that subjects were free to withdraw at any time and that no direct quotations would be used in the thesis without express permission. The recording was started and the subject then asked formally to agree to take part in the survey. Recordings were stored as audio files on a limited number of personal computers secured by passwords. They were transcribed and transcriptions were sent to the interview subjects. Transcription files were stored in the same way as the original audio files and their contents were not divulged to anyone apart from my supervisors.

In addition to the duties a researcher has towards those who help with their research, there are also wider duties to other researchers and to society in

general. One important underlying rationale for this project is a desire, based both on enthusiasm for the artistic practice(s) of others and a search for enhanced and novel personal creative resources, to promote and enable activity and creativity in the field of multimedia and live performance. In contrast to earlier models of artistic practice, which saw individual artists as necessarily in competition with each other, contemporary artists who use digital technology may have other models of activity and social relationships prompted by the tools they are using. Digital networked technology is widely seen as a force which, by its very nature, promotes novel and essentially collaborative patterns of working and consuming (Lévy 1999, Duckworth 2005)[72]. These factors place important constraints on the design of this particular research project, particularly on how its outcomes will be used and disseminated. Arguably, the emergence of a playful, experimental, non-goal directed exploration of digital creative practice is best supported by the widespread availability of open source software and licensing models (e.g. Processing[73], Audacity[74], the Creative Commons[75] movement etc.). Clearly, any research that attempted to separate the interests of the researcher from those of practitioners through commercialisation might run the risk of transforming the research from an enabling process into another

[72] Of course, other commentators (Schiller 1981), suggest that digital technology necessarily encapsulates and promulgates those economic structures which produced it. It is in this sense that we can consider the deliberate utilisation of technologies for purposes and situations other than those they were originally intended for as a continuation of the artist/subversive narrative.
[73] Processing (http://www.processing.org/) is a programming environment widely used by digital artists. It is relatively easy to learn, but is scalable and capable of producing highly complex and creative artefacts using a wide range of media-types.
[74] Audacity (http://audacity.sourceforge.net/about/) is an audio recorder and editor which is available for most platforms and is capable of fairly sophisticated audio processing. The interviews for this project were all recorded using Audacity.
[75] The Creative Commons movement (http://creativecommons.org/) produces and promotes licences which allow creators to make their work available to others while retaining their rights to control commercial exploitation.

instance of the restrictive patterns of ownership and control that can be seen as limiting or even counterproductive. For this reason, the LIMPT software and the knowledge about practice produced during the project will be distributed (section 8.4) using open source or creative commons licences.

3.3 Software Development

The prototype LIMPT system has an architecture (section 6.2) which divides into two components; the central controller and the client application, multiple instances of which are run when the system is deployed. The components communicate using XML messages passed over any standard Ethernet network. The central controller was implemented in Java while the client was a Director/Flash hybrid (see chapter 6 for a discussion of the rationale behind these decisions). The software development process consequently used a range of tools, each a development environment suited to a particular programming language or authoring system. To provide efficiency and quality, an approach to development management was adopted which operated at system-level and brought all the different component development activities within a single process.

3.3.1 Development Methodology

The development methodology for the LIMPT system was heavily influenced by agile development practices (particularly XP[76]) adapted to a first-person situation. The process extended throughout most of the timescale of the research, taking as its requirements a changing set of specifications derived, initially from personal experience and then gradually incorporating features and functionalities suggested by results from the interviews with practitioners, the

[76] XP or Extreme Programming is one of the four main models of agile development: efficient, responsive and flexible models of software production which are well suited to dynamic technical and economic contexts.

formalisation of requirements and language-design and from activities (public demonstrations and workshops) carried out using the system.

Agile development processes are iterative, but provide a much faster and more responsive realisation to users' requirements than 'traditional' waterfall development methodologies (Wells 2009a, Shore & Warden 2002) although the degree of functionality actually implemented on each iteration is relatively small. They manage development processes through the prioritisation of desired features which then form a set of staged, near-term project goals (Wells 2009b) and take working software as "the primary measure of progress" (Agile Alliance 2001:Principle 7). This is in contrast to waterfall processes where project goals are established and fixed at an early stage and activity then 'falls through' the development and production stages until the working artefact emerges at the end of the process. Checking that the product actually meets users' requirements is delayed until the system is substantially complete. Alteration to requirements (as often happens in contexts involving rapid change such as commercial or creative practice) are particularly difficult to deal with using this traditional model (e.g. see Larman 2003:87). Another significant weakness is that even partially working systems are not typically delivered until towards the end of the process, which in the context of a long-term enquiry would restrict the opportunity to use and learn from the system while it was under development. In contrast, the first principle of agile development according to the Agile Alliance is,

> "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software." (Agile Alliance 2001)

Modification of established agile methodologies was necessary since agile development is normally a team-activity and emphasises a high degree of (human to human) interaction; programmers typically work in teams and have daily, face-to-face access to the end-users or client's managers. In contrast, the LIMPT system was developed in a 'first-person' environment, one involving a single individual fulfilling a number of roles. This is a very common scenario in early-stage non-commercial software projects where the 'client' is also the programmer and (often) the end-user, but it does have the potential to weaken controls on quality and decision-making during the software production process. These factors were addressed by adopting a structured approach to testing (see below) and by using formal descriptions of both language grammar and logical process (see chapters 5 and 6) to underpin the iterative identification of functional goals which drove the development cycles.

A key feature of agile development is the use of 'user-stories' (Wells 1999, Shore & Warden 2002:19) to structure the development process. Typically, a future user writes a short description, known as a user-story, of something they want to be able to do using the system. This is then given to the programming team who estimate how long it will take to implement. They then present a 'menu' of current functional requests and projected timeframes to the end-user for prioritisation. Those functionalities that the user considers the best use of the team are then implemented.

For this project, the desirable features, were provided by a range of sources; initially by the need to refactor the code base used as the starting point to take account of new language (Java) and component (Jess[77]) versions, the migration of the database from PostgreSQL[78] to mySQL and the requirement for a more consistent and scalable implementation than one driven by the need to have a working system by a deadline. As the project progressed, functionalities identified from comparisons with other resources and from survey results, creative work and workshops were used to steer the development process.

3.3.2 Tools

In a highly iterative process of software development, keeping track of file versions is generally considered crucial for the efficiency of the overall process. Because incremental progress requires a continually improved, but always working codebase, having managed access to a central repository of the latest working code is necessary. This process is known as version control and it is an area where tool development has been fairly rapid and competitive. There were a number of systems available when the project was initiated, but because of the need to operate with both Eclipse[79], the Integrated Development Environment (IDE) used for Java development, as well as Director and Flash

[77] Jess (http://www.jessrules.com/jess/) is an Java Inference Engine which is used to provide the complex rule-based decision-making functionality of the LIMPT system (see section 6.2.2).
[78] http://www.postgresql.org/
[79] Eclipse is an open source, development environment which was originally deigned for use with java application and applet development, but has since grown to allow development in a wide (and growing) range of languages. It is freely available from http://www.eclipse.org for a wide range of platforms.

across both Windows and Mac platforms, the Subversion (Apache Software Foundation 2011) system was used as it appeared to offer significant advantages (Neary 2005, Eliasson 2005) over the (longer-established) CVS system and was the most popular well supported open source choice[80].

The Java development was undertaken using the Eclipse IDE for a number of reasons. It is a very powerful system offering automated building and testing using the Ant[81] extension. It can also be used to edit and validate XML and has plug-ins for working with the Subversion version control system. It was also my Java environment of choice and was available for all the machines I used for project development work. The commercial Director and Flash authoring packages were used although open source alternatives to the Flash Authoring package were examined. At this time, open-source support is limited to ActionScript compilers[82] which, given the importance of interface design to the project, was not considered adequate.

The documentation produced by an extended development process is extensive including architectural and class diagrams, formal API specifications, feature requests and issues (or bugs). The central controller and client were treated separately in terms of documentation storage, but a similar strategy was

[80] It has subsequently (for a number of reasons) been largely overtaken by Mercurial (http://mercurial.selenic.com/) as the most popular fully featured version control system, see Auvray (2008) for a discussion of approaches to version control and a survey of current software tools.
[81] http://ant.apache.org/
[82] For example, MTASC (http://www.mtasc.org/). Since Adobe released an open source Flex SDK (http://opensource.adobe.com/wiki/display/flexsdk/Flex+SDK), which includes a compiler, most projects seem to be graphical front-ends which rely on that to produce compiled Actionscript.

adopted for both. Development documentation, feature requests and the running lists of issues were paper-based. While this does require careful management of paper, it also means that rapidly evolving development can be supported through amendments and annotations on diagrams and lists in a more flexible and less structured way than that imposed by commercial-level tracking systems[83] such as Bugzilla (Bugzilla.org 2010).

3.3.3 Functional Testing

The testing strategy involved testing each component separately as well as the overall system integration. Because of the size and complexity of the system, a testing regime based on whole-system operation rather than class-based unit testing was used alongside more conventional class level unit testing. The first step was to compose a setof reference messages that a client might send in response to a range of performance events and user commands, based on the schemas specifying the XML message formats:

```
<?xml version="1.0" encoding="UTF-8"?>
<server:psmessage xmlns:server="http://www-edge.cassiel.com/p-
server/ns" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
ID="0.358027125708759">
<genTime>2004-04-26T22:19:44</genTime>
<originator>
<identifier>Ian1</identifier>
<protocol>TCP/IP</protocol>
<IPaddress>129.0.0.1</IPaddress>
</originator>
<target>
<identifier>server</identifier>
<protocol>TCP/IP</protocol>
<IPaddress>127.0.0.1</IPaddress>
```

[83] The balance in favour of paper-based development documentation was only possible because of the single user; a development team would not be able to adopt such a system reliably and consistently.

```
</target>
<projectName>ian_Test1</projectName><itemType>perfData</itemType
>
<pData>
<dataSource>userEvent/mouse</dataSource>
<dataItem>#value,double</dataItem>
</pData>
</server:psmessage>
```

Code Example 1: reference XML message

Java testing was done using an Ant build file[84] which compiled and ran the central controller and then sent one or more test XML message files (see Code Example 1) to the application. Responses were tracked both through a variable-level comprehensive activity logging capability and by monitoring database additions and changes manually using phpMyAdmin[85].

The development of the client application was carried using a similar approach to testing, although in this case the unit testing involved a number of Flash movies that tested subsets of the class-libraries[86] in isolation. For higher-level tests (i.e. those involving the client as a complete or near-complete system), standardised user interaction sequences were followed with deviations from expected behaviour noted. These deviations were then listed, allocated an informal severity rating and then resolved sequentially according to the available time and issue severity.

[84] The build file is an XML configuration file which can perform a sequence of compilation and file deployment operations. A typical test sequence would delete old application files, compile the Java classes and then, if successful, run the application.
[85] phpMyAdmin (http://www.phpmyadmin.net/home_page/) is an open source software tool for administering and optimising mySQL databases.
[86] The Component Manager, ProjectDB and MessageBuilder packages were all tested in this way, with successive frames in the test movie checking and reporting on expected operation. There are some unit-testing frameworks available for use with Flash (e.g. ASUnit – http://asunit.org), but they were considered an unnecessary complexity given the speed with which timeline-based test suites can be coded.

3.3.4 User Experience

As a part of the development process, two workshops with early prototype versions of the LIMPT system were presented to different cohorts of MA/MSc Creative Technology students at De Montfort University taking the Performance Technology module[87]. These sessions were deliberately fairly unstructured; the intention was to provide students with direct personal experience of using a collaborative performance technology system in development (including setting up, familiarisation and experimentation) rather than to use them as a part of a formalised testing regime. The value for my research was that while both the sessions were quite short, the database on the central server stored a record[88] in the form of materials students had created (see section 6.2.2) which show the extent to which they had been able to assimilate the metaphors behind the LIMPT system and the eMerge language. This evidence, together with notes taken about software issues students encountered, was fed into the development process.

[87] The workshops were on 6th March 2009 and the 29th February 2008, both in the Institute of Creative Technologies at DMU.
[88] These records in turn were saved in SQL exports of the state of the database which were made every few months when the system was in active development.

3.4 Creative Activity

Over the period covered by this research, I have engaged in a number of individual and collaborative creative projects (see sections 4.3 and 4.4). These have been public gallery-based installation projects rather than performances, but have provided useful data on relationships between the complexity of projects and the applicability of tools; in particular, how simple a particular use of technology has to be before a generic solution is inappropriate and a bespoke solution becomes most efficient. None of these projects used the LIMPT system, although this was actively considered for one of the installations as the complexity of the bespoke system increased (section 4.4). Reflection on these activities was fed into the final statement of requirements in the form of a lower limit for the scale of project the system would be designed to support.

3.5 Evaluative Strategies

The evaluation of the final LIMPT prototype has been approached in three different ways:

- Use in a public workshop with digital performance specialists
- An evaluation against the formal statement of requirements produced as a result of the survey.
- A comparative study which assesses the functionality of the LIMPT system against that offered by currently available software resources used in digital performance (section 2.2).

For the public workshop at the Digital Humanities for the Humanities and Arts 2010 conference, the LIMPT system was installed with a number of laptops running the client software, some connected to data projectors and input collected from floor pads as well as keyboards and computer mice. The participants in the public workshop were taken through the system including participating in a short performance exercise. They were then invited to experiment with the system and think about how their own working processes used technology. Towards the end of the session, participants were given a short presentation about the research project and asked to complete a short questionnaire if they were willing to be involved. The questionnaire was structured to ask questions about the system, its underlying model of performance and how the outcomes of the research project might fit with the individual's view of their own practice (see appendix 4 for the questionnaire). The survey results are presented and discussed in section 7.1.

3.6 Chapter Summary

This chapter has provided an overview of the project and described in detail how methodologies which were appropriate to each of the research activities were established. The practitioner survey required the creating of an epistemological framework expressed initially (and provisionally) through a set of suggestions about the ways interactive multimedia could be used in live performance. Interview subjects were invited to talk about how these applied in their work as well as suggesting any other ways they thought about the integration of digital technology and performance. The framework also served as a basis for analysing what subjects said through the establishment of an initial set of codes: labels for meaningful data which allow the researcher to explore patterns of significance. Software was developed through an adaption of established agile development methodologies. The development process was driven by a statement of requirements derived from the survey results as well as a number of other sources. The evaluation of the software artefact was carried out in three different ways to ensure validity. This chapter has established a solid basis for producing valid research results arising from a range of research activities and evidence-types. In the next chapter, I shall present and explore the results obtained by my primary research activity and show how these can provide an answer to one of my research questions and form the basis for a formal statement of requirements for a generic system.

4Requirements

This chapter begins by presenting and analysing the results of the survey of practitioners in the field of digital performance and, from these, establishes answers to the first of my research questions: that it is possible to identify the features of a generic system for use in multimedia and live performance and that some of the possible features of a generic system can be identified.

The chapter uses a detailed analysis of the survey results (section 4.1) and reflections on a range of other creative activity (section 4.2) including two collaborative works made with Martin Rieser, presented at the RMIT Gallery, Australia in the 2008 exhibition *HEAT: Art and Climate Change* and at the ISEA 2009 conference in Belfast, to identify and specify the features of a realisable generic system. These functional and operational constraints are then brought together in a formal statement of requirement (section 4.3), a specification of the required functionality and modes of operation of a computing system capable of providing support to artists with a wide range of backgrounds and interests working in digital performance.

4.1 Analysis of Survey Results

The analysis of the survey results sought to answer the two linked questions posed in the rationale (section 1.2); 'can any generic use of technology be identified?' and 'to what extent are creators' works characterised by particular uses of technology'? Implied in these questions is the recognition that individual artists' use of technology is a part of an individual (or collaborative) practice that tries to produce characteristic difference. Producing artefacts that are, to some extent, unique is a central part of creative practice and thus even if a particular tool or use of technology is common to all working in digital performance, if it is a highly differentiated area of use, one of the most significant vectors for producing identity in performances, it would not be a good candidate for including as a feature of a generic system because as an area of particular concern, artists will have already found tools to meet their needs and, further, their choices about these tools will themselves be an important part of the production of identity. For example, all the artists interviewed use projected video[89] in their work which involves some processing of the live or recorded images. However, the affective intent and the particular processing involved are highly individual and artists have already found digital resources to use for these purposes. In seeking answers, the analysis had three main foci: the identification of which technology digital performers are using, the categorisation of what they are using it for and, because adoption of tools is not just a matter of functionality, to investigate how those working in digital performance relate to the technology they use.

[89] Projecting video was the most common use of digital technology identified in my survey.

The survey (see section 3.2) was administered to six subjects with extensive experience of integrating technology in their work, drawn from a range of performance disciplines and deliberately chosen to have divergent practices both in terms of the content of their work and, where possible, in terms of the genres and ranges of reference they employ[90] (see appendix 2 for brief biographical notes on interview subjects).

Subject	Original performance tradition	Date of interview	Mode of interview	Duration (h,mm)
Jane Turner	Dance	05/06/2007	Telephone	0,31
Joby Burgess	Music	11/06/2007	Telephone	0,53
Sarah Rubidge	Dance	15/06/2007	In person	1,12
Andy Lavender	Theatre	14/12/2007	In person	0,56
Martyn Ware	Music	02/05/2008	In person	1,27
Steve Dixon	Theatre	20/01/2010	In person	1,06

Figure 5: Details of Interviews

The interviews, conducted over two and a half years, provided just over six hours of material which was transcribed[91] and then imported into the qualitative analysis tool, TAMS Analyzer (Weinstein 2011). Each interview was then coded (see section 3.2.3) using an system of labels (codes) which were initially

[90] So, for example, the two practitioners from a musical performance background, Joby Burgess and Martyn Ware, come from (modern) 'classical' and 'popular' backgrounds at least initially (see appendix 2). This diverse pattern of original generic influence is replicated in the other performance fields I surveyed, although as has been suggested (Chatzichristodoulou et al 2009:1) the integration of technology in performance tends to "disturb boundaries of traditional performance and create new paradigms of emergent practice and discourse" which might suggest that digital performance can embody a mixture of stylistic features which mask or even negate the original specialisms of practitioners.

[91] Transcription for this study concentrated on semantic features rather than on recording the detailed patterns of speakers' vocal performance such as hesitation, error or repetitions. While these can be highly significant for some studies, they were not felt to be appropriate for this particular enquiry where individual artists and their practice were not the primary focus.

derived from the interview framework used to collect data together with some additional codes to collect data about subjects' experiences of the interview itself as well as the ways they chose technology and the effects it had on their practice. As analysis progressed, changes to the structure and numbers of codes were made reflecting the developing understanding of the epistemological patterns within the data itself.

In looking at which tools practitioners are currently using, the survey used two approaches; one was based on identifying specific tools artists had used in their practice while the other sought information on the collection and presentation of different sorts of data[92]. Figure 6 shows the digital tools subjects reported using in their practice based upon the number of mentions each tool received across *all* the interviews[93]. It should be borne in mind that Jitter is always used with Max, thus these two categories should be seen as reflecting nuances of Max use (i.e. most use of Max seems to involve video). 'Human operator' here means that cuing or controlling of a digital effect was carried out by hand.

[92] The two approaches were adopted to ensure validity, it is entirely possible, given the highly collaborative nature of the field, that practitioners might not know which tools had been used. However they are more likely to know which sorts of information were being used as inputs and outputs in their projects.

[93] The mentions each item received are based on a count of the number of times each was tagged with the relevant code and is thus a measure of how prominent it was in the conversations with interview subjects (each subject was taken through the same interview structure). Isadora received 11 separate mentions, Max 9 and human operator, 7. Those tools which were not mentioned by any subject have been omitted.

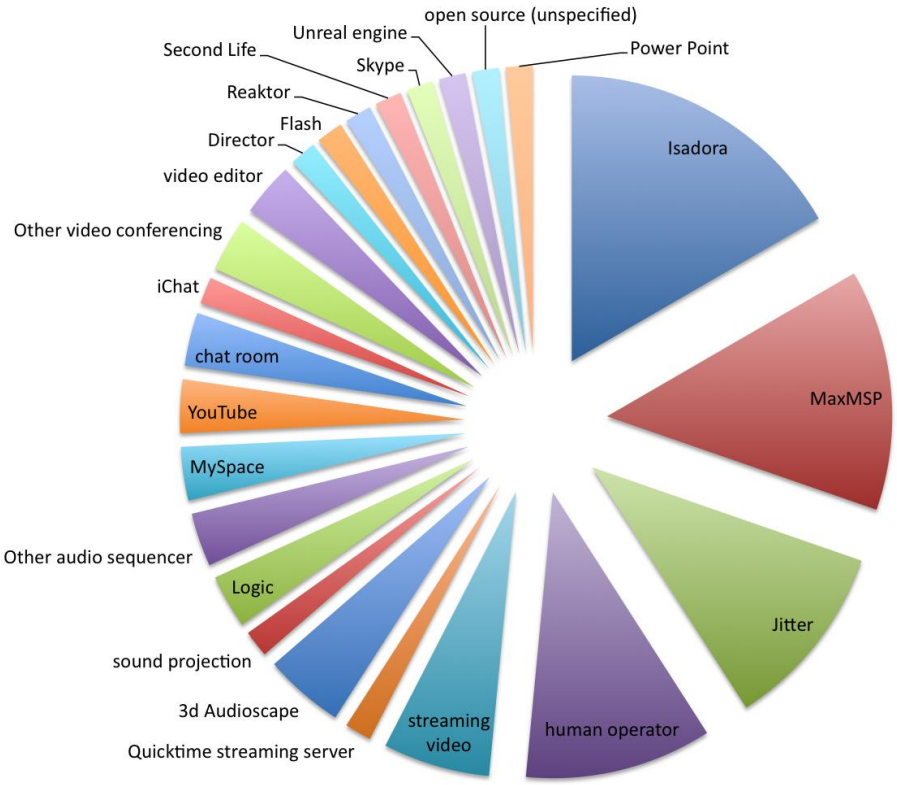


Figure 6 – Digital tool use as mentioned by interview subjects

Figure 7 shows another presentation of the same data, showing how often the most popular tools (n ≥ 2) were mentioned by each interview subject. There is a large difference in the mentions of specific tools, some of which can be explained by the different working patterns and creative processes of different performance practices[94] and some of which can also be attributed to the extent of bespoke system use. If Figure 7 is read, looking at pairs of practitioners from dance, music and theatre (i.e. from left to right), a pattern of gradually decreasing use-mentions can be observed mirroring that identified during our interview by Steve Dixon, that real-time technology use was most prominent in dance.

[94] The removal of those tools mentioned by only one practitioner (for clarity) has a significant relative effect on Andy Lavender's total, he actually mentioned only one fewer (7) than Martyn Ware and Steve Dixon.

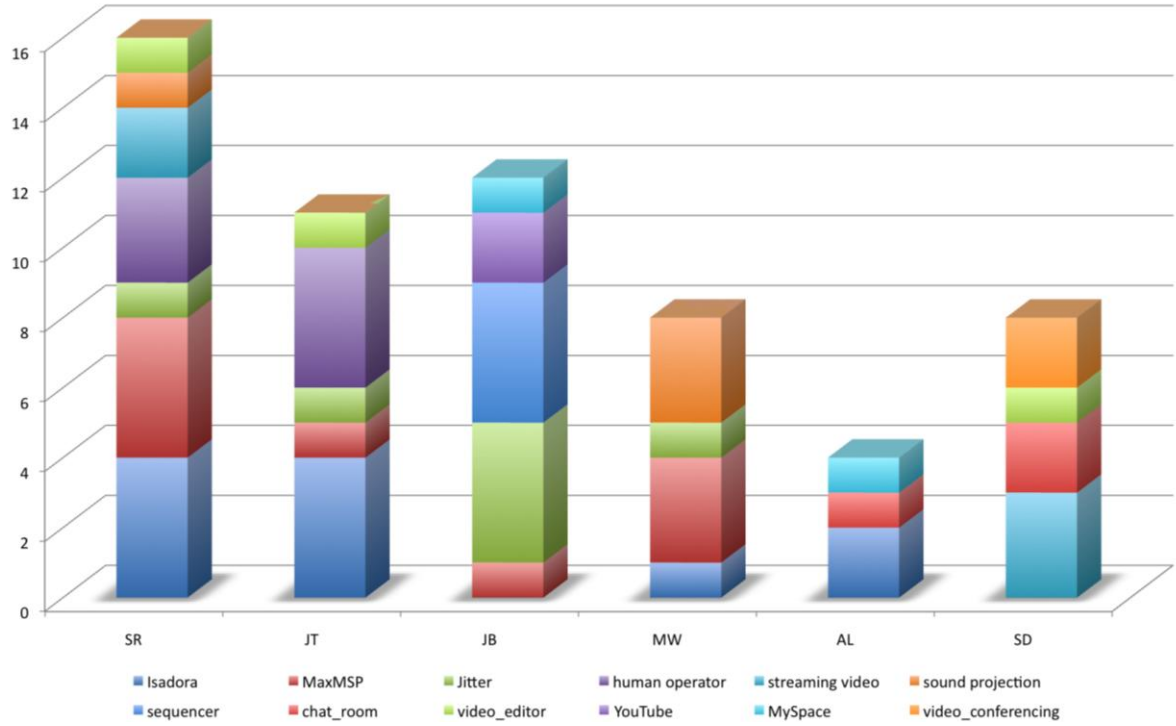


Figure 7 – Tool-use mentions by each interview subject

In interviews, some subjects also talked about why they *hadn't* used tools; there were suggestions that, for some artists at least, this was not solely due to choice,

> "Frankly, I would use anything that was in accord with the theme of the piece, what it was 'about'." (Sarah Rubidge)
>
> "Well, fundamentally we use all and everything we can." (Martyn Ware)

Looking at the sensor types used to collect information for digital performance, both overall (Figure 8) and by artist (Figure 9), some similarities can be noted with the use of tools data although the patterns are more complex. The importance of video for current digital performance practice is confirmed; not

only were the most frequently mentioned tools (Isadora and Max with Jitter) those for processing video, but video cameras were the most widely used sensor-type (16 mentions) and were used by every interview subject.

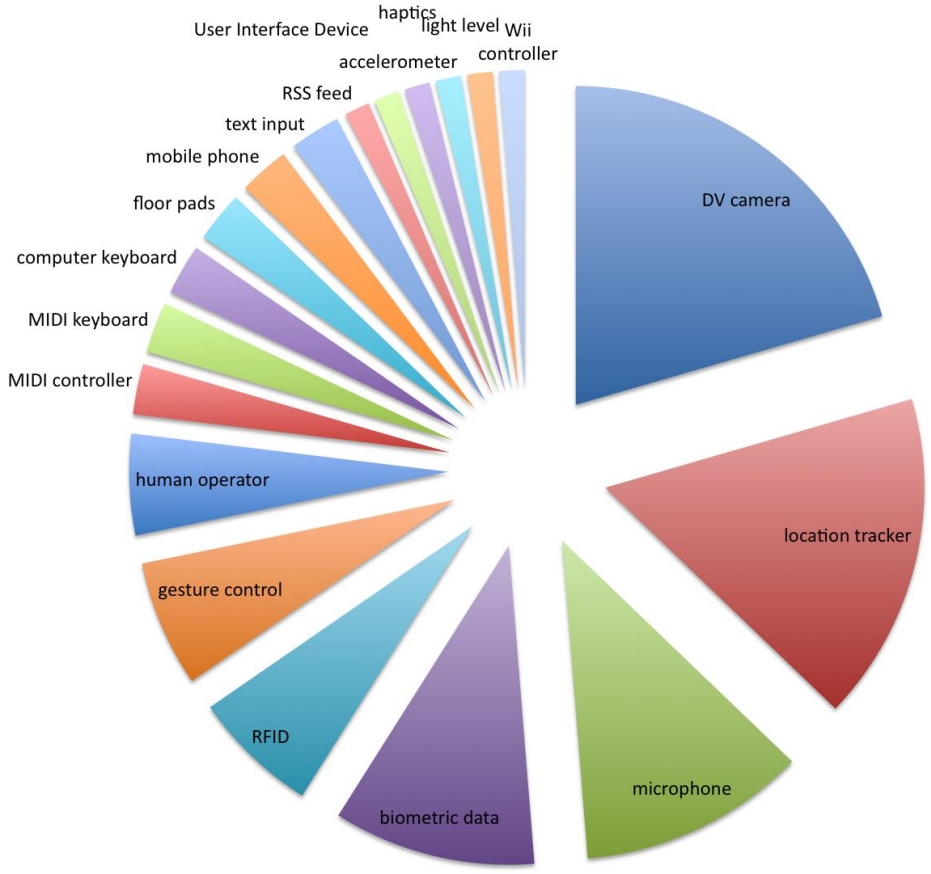


Figure 8 – Sensor-types used to collect information in digital performance

However, the second most commonly mentioned sensor is a location tracker[95], which featured prominently in those two artists[96] who work mainly with bespoke performance systems and yet not at all in the other artists' interviews.

[95] Note that no distinction was made between the various techniques for tracking location or limb-position. The data presented here includes video-tracking, ultrasound and infra-red approaches.
[96] Sarah Rubidge and Martyn Ware.

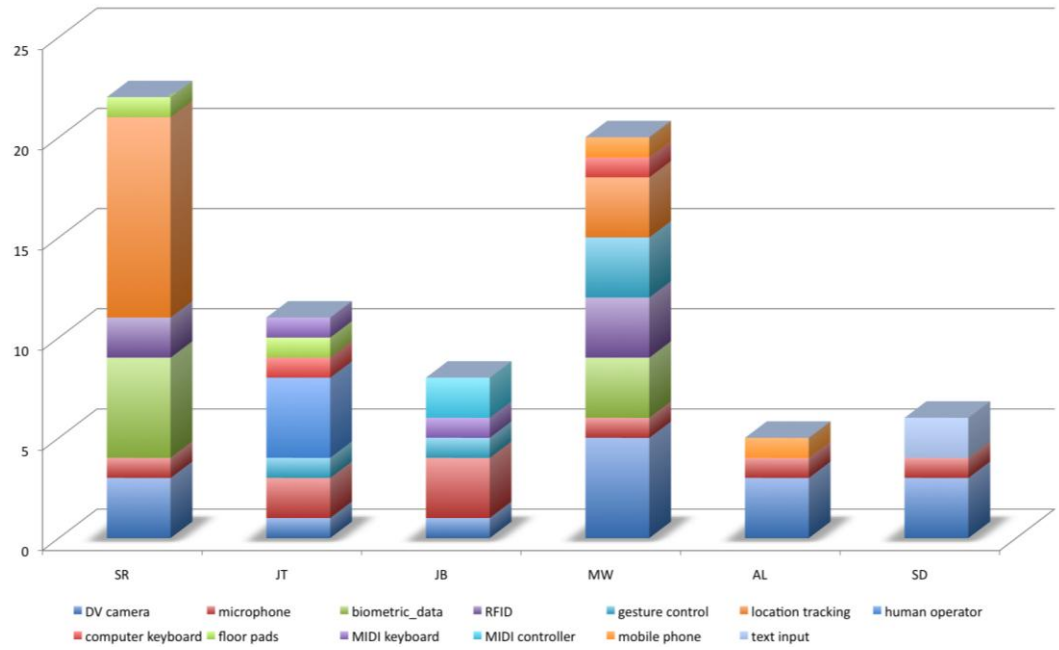


Figure 9 – Use of sensor types by interview subject

The third most common sensor type was the microphone (nine mentions), which was mentioned by all artists. There appears to be a pattern here of common sensor types (video camera and microphone) used to some extent across most artists' practice and other, more specialised sensors which are very prominent in just a few artists' work, but may not figure at all in that of others[97].

The results for the use of technology to present material (Figure 10) largely confirm the pattern of media use suggested by the data on tool and sensor-type use. Video (both generated images[98] and playback) is overwhelmingly the most popular usage with sound coming second:

[97] This pattern is repeated for biometric sensors and RFID – both used intensively by just the two artists who use bespoke systems and not at all by the other interviewees.
[98] 'Generating images' is used to identify usage which involves at least some representational material as opposed to abstract shapes (visualisations) or non-realist figurative material (animations).

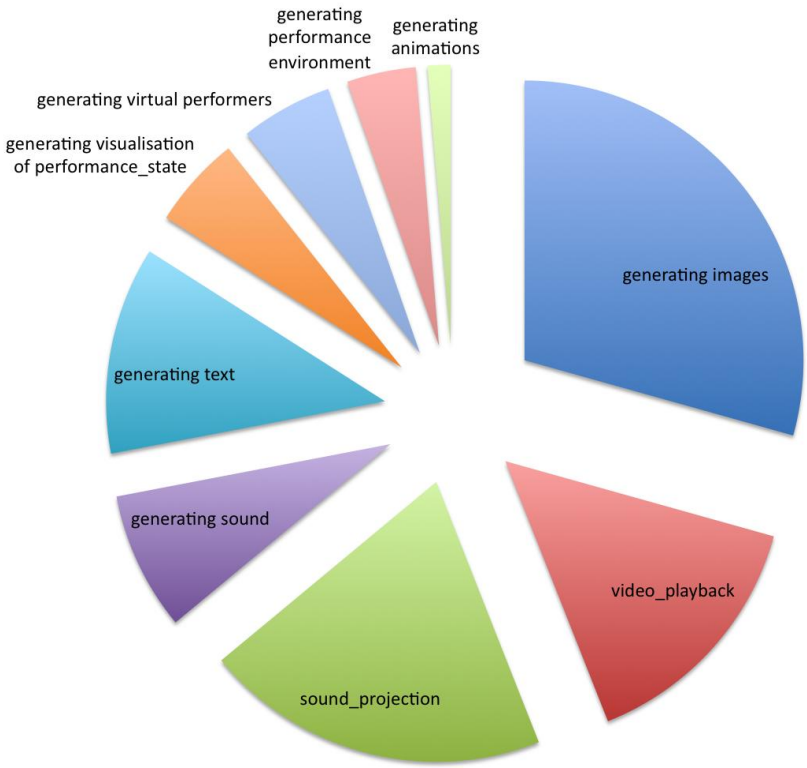


Figure 10 – use of digital tools to present material

Figure 11 shows that video playback (either generated live or using pre-recorded material) is a feature of the work of all the interviewees. Sound use (either generated or played back, is almost as ubiquitous, only one interview subject did not mention sound playback[99].

[99] Although they did not talk about sound on its own, they did discuss sound-use in detail when talking about the way their video material was constructed.

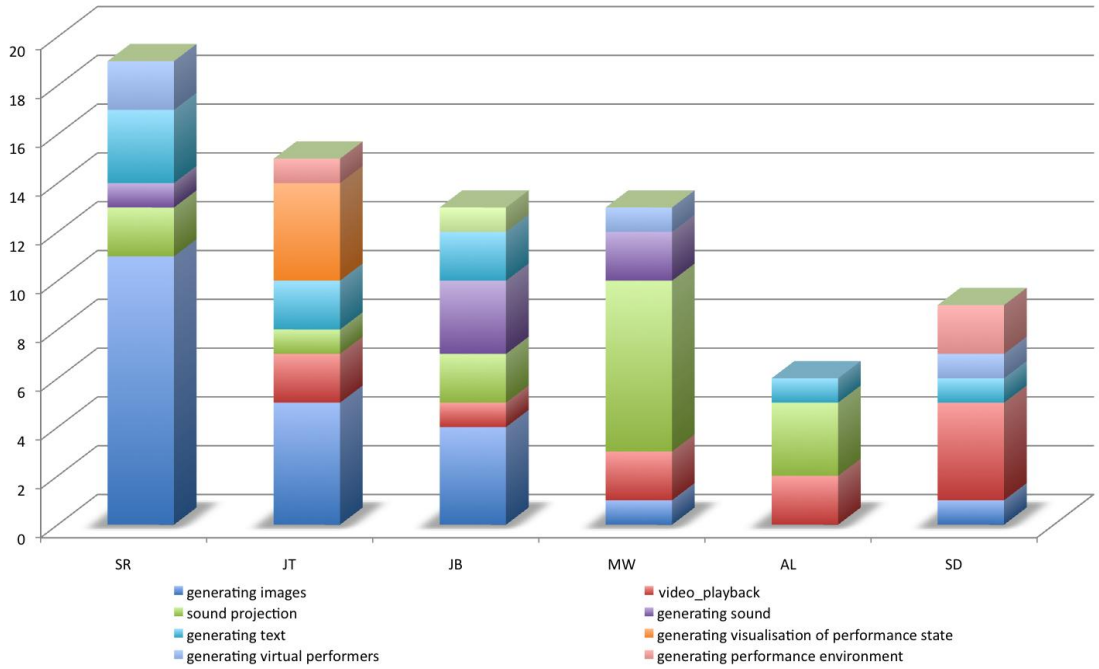


Figure 11 – Use of tools to present material grouped by interview subject.

It might be supposed that the data in Figures 10 and 11 suggests that sound and video presentation (of all types) is a generic aspect of practice. However, qualitative data completely contradicts this interpretation. Artists did mention presenting images and sound a lot, but this was because this material seemed to be a highly important means for artists to characterise their performance work.

> “Now this [points to screen]is a ‘critter’, which is generating the image as it moves. You didn’t see this on the (actual) image on the screen. You can see [from the flow of the motion] that it’s very much out of New Dance..., but that’s the kind of movement I’m interested in. The programmer was trying to get that kind of motion from a critter: this is an 'evolving' critter, there are all sorts of things built into the programming that drives it motionto give it its quality. This results in a sensibility in the computer graphics which I feel comes from human movement and dance.” (Sarah Rubidge)

> "…one of the things we like to do, particularly with high-pitched sounds, sometimes real recordings, for instance rain or in non-real, high-hats for instance, is to randomise location within three dimensional space. It works incredibly well, sounds amazing." (Martyn Ware)

> "…one of my interests really is to find ways of utilising properties of screen-space within a theatrical mise-en-scène. Where you can't just take the video of the video and watch it on your DVD at home because it wouldn't make sense or it would be incomplete and probably meaningless. It can only ever be fully meaningful in relation to a theatrical mise-en-scène." (Andy Lavender)

The detail of descriptions and evident care taken over the content and placing of sound and video material suggests that while the use of video and sound is ubiquitous, it is not generic, but rather the opposite; an area of activity in which the majority of digital performance work is maximally differentiated both from other works and from works by other practitioners.

Chatzichristodoulou et al (2009) approach their examination of the "technologization" of performance through asking,

> "…how do technologies expand, extend, (re)present, dislocate, disperse or invade bodies in performance?" (Chatzichristodoulou et al (2009:2)

Figure 12 shows the relative frequency with which various possible technology uses to separate, link or connect within performance were mentioned during interviews. 'Linking Venues'[100] (12 mentions) and 'Connecting Performers'[101] (11 mentions), were both major themes in almost all artists' work (Figure 13)

[100] Physically distinct performance spaces being connected together for a particular performance.
[101] Performers within the same performance being connected in ways that enhance those connections of perception and sensitivity which are part of non-digital performance-practice.

and do not show the same pattern of technology use following performance tradition[102] as in the data discussed above. Apart from one subject, connectivity appears to be a feature of practice across all performance traditions to a roughly similar extent.

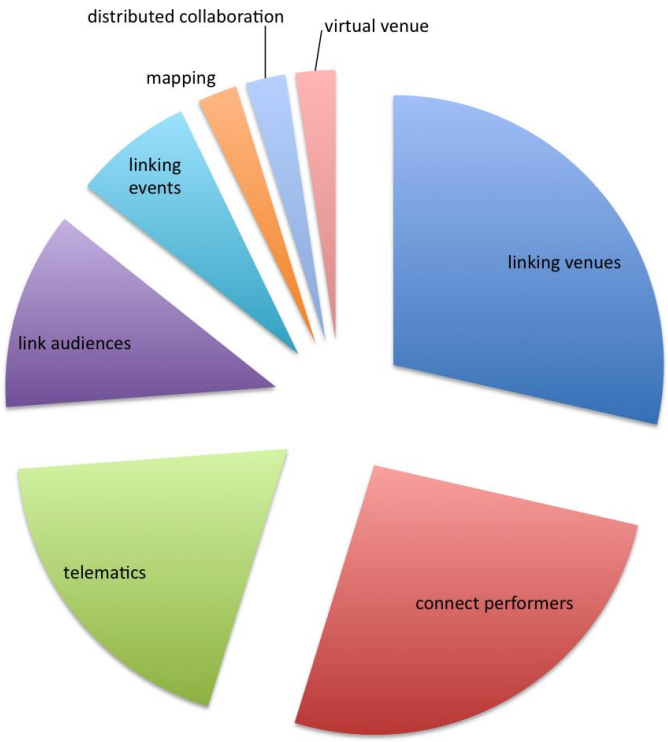


Figure 12 – Use of connectivity by intention

[102] I.e. that dance is the most intensively technologised, followed by music and then theatre.

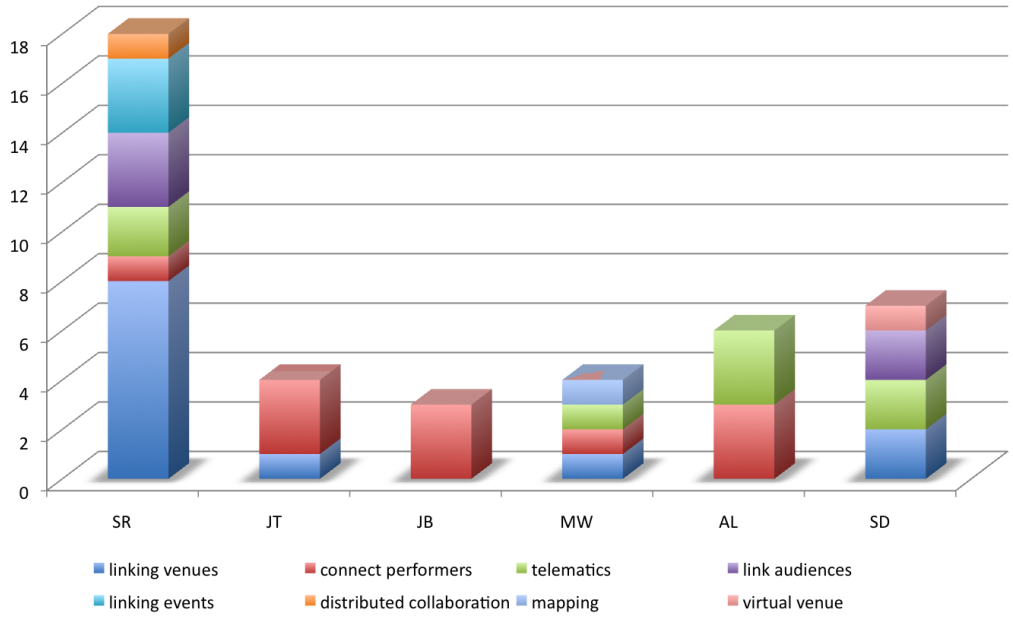


Figure 13 – Use of connectivity grouped by interview subject

Data for controlling, another lower-level form of connection, are shown in Figure 14. While I asked about uses involving control in some detail (relative frequencies of mentions are shown in figure 14), one striking result needs to be highlighted: out of all the uses of technology to control, most (70%) involve some form of triggering. The distinction is one of signal type; controlling is a continuous process often involving analogue data[103], triggering is a discontinuous, episodic process involving binary or pulse data.

[103] In a digital system such signals are sampled and converted to a stream of numbers; strictly speaking they are no longer continuous but *quantised*. They do still retain the requirement for a higher bandwidth in signal processing and networking compared to binary (on/off) triggers.

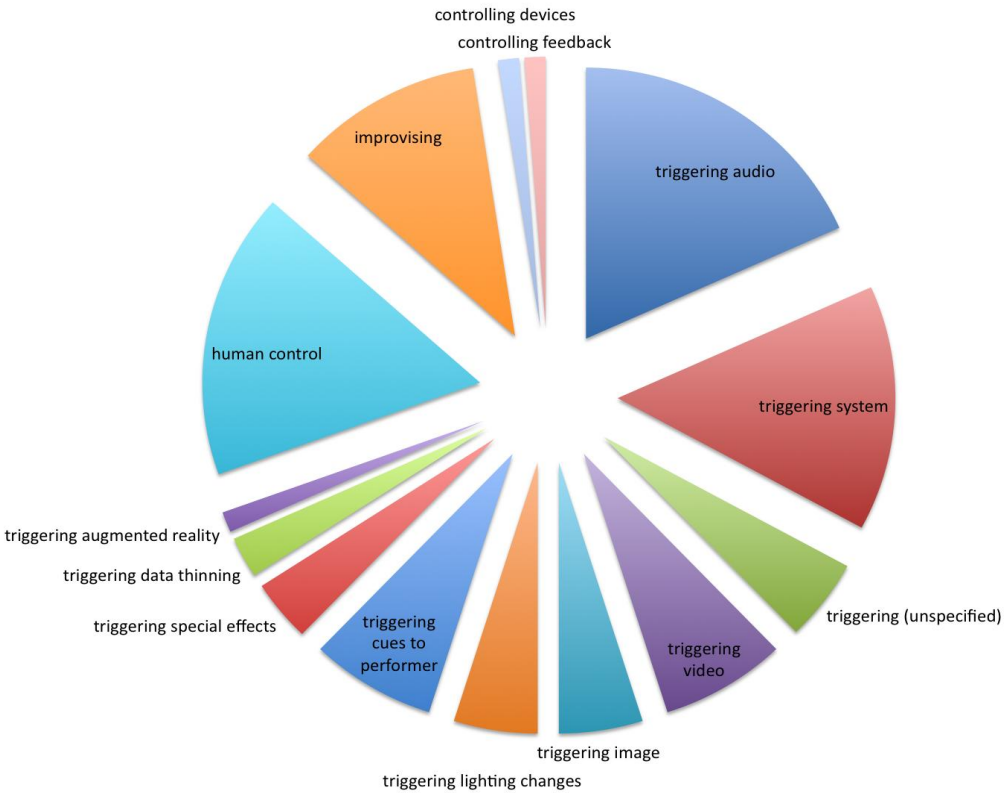


Figure 14 – Controlling in digital performance grouped by purpose

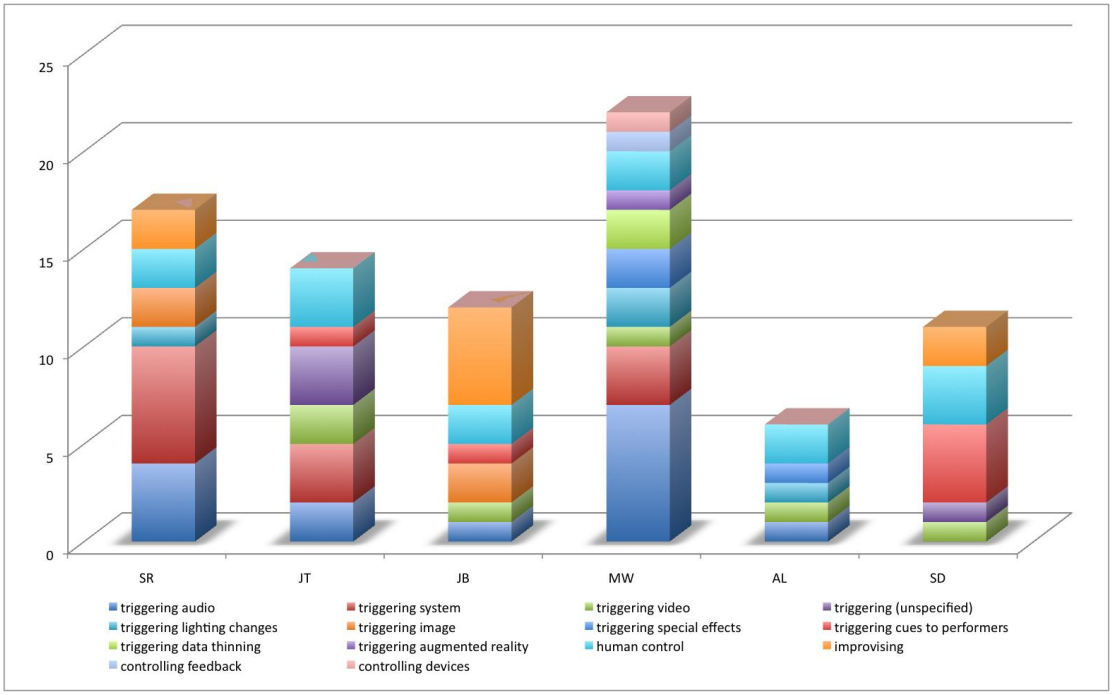


Figure 15 – Control use and purpose grouped by interview subject

In looking at how control was used across my interview subjects (Figure 15), two further patterns are evident; all practitioners used triggering or cuing and all also mentioned human operator involvement in their practice to about the same extent (improvising, which here means intervening in performances through triggering or directing its constituent elements, was not universal but was clearly important to some practitioners in forming their practice). The use of non-improvisatory human control does not reflect the pattern of increasing use of technology from theatre to music to dance observed in some data presented above. Triggering was talked about by subjects in a qualitatively different way to controlling; it was mentioned often as a feature of performance practice, but its detail (beyond what was providing the trigger or cue and which device or performer reacted) was clearly less important than the content of the action that resulted. Controlling, where it was used, was in contrast a highly considered part of practice: -

> "Processing information, for me, tends to be on the basis of a continuum rather than on-off triggers. So it's the continua [*knocks repeatedly on table*] I'm interested in. Hence the use ofmovement, hence the use ofbiometric systems..., if this continuum of data input is going fast or it's going slow – these are very human things which gives the imageryits breath [*breathes and moves*]" (Sarah Rubidge)

Subjects were also asked about where in their performance production processes they used technology[104]. It was clear that most (although not all) used an iterative process, working at projects over several cycles of presentation and reflection. Four out of the six subjects reported that they

[104] The interview framework identified three phases in the preparation of performances: preparation, performance and documentation.

publicly presented work in progress and made audience's reaction a central part of their creative development:

> "…where possible, we show work in process to audiences in order to get feedback. Not for that feedback to make us slavishly respond in a knee-jerk way but on the principle of the sneak preview in silent cinema; we're making things that eventually audiences should take pleasure in." (Andy Lavender)

> "And so all that stuff, once you accept that as a kind of honest exposure of the working process it encourages, and this is something I couldn't really have predicted beforehand, it encourages people to, by definition, to continue with their experimentation. They're not looking for an end…" (Martyn Ware)

As well as the suggestion that incorporating digital technologies within performance might privilege this way of working, it was also clear that almost all subjects either had, or wanted to have, their digital tools available throughout the rehearsal period.

> "But the preparation of most of these things is crucial, anything where you're dealing with interactivity. The reason that a lot of interactive performance work doesn't happen as well as it could because you never get a chance to do test runs of the show." (Sarah Rubidge)

> "I think because of the nature of the cost implications in terms of using technology, it means that one doesn't work with it enough to be able to know whether the system might become more…" (Jane Turner)

> "That's a really nice way to work because the studio is such a different environment to working live. We've actually a couple of times said, 'this is what we're going to use', and we've taken it to an empty place and just set it all up and went from there." (Joby Burgess)

This tendency was less marked in those practitioners from a theatrical performance tradition, although even here there was clear evidence of exploration of new ways of working which involved technology in a more sustained way with the process of making performance:

> "…the pieces tend to be content-driven although we do do explorations with cameras or with video… frameworks if you like, to see what 'juice' that yields that we might exploit narratively or thematically…as far as possible, certainly the aspiration is for everybody to be in the room when the work is being made. And then to make it, sometimes very slowly and sometimes without all elements being in play, but to evolve it organically and collaboratively." (Andy Lavender)

Within their evident enthusiasm for integrating technology into their practices, many interview subjects did place clear limits on their personal involvement, particularly where specialist learning might become involved.

> "I'm only really interested in what it can do. I'm not interested in coding. I'm not particularly interested in how it works. I'm interested in the end result." (Martyn Ware)

> "I just want the effect – give it to me. It's like the lighting or the sound. I know enough about it and I'm no fool, and I'm really interested, but I'm interested in the effect and the end result. I'm not that interested in the process. I'm very interested in theatrical processes and filmic processes." (Steve Dixon)

To some extent, this is a reflection of a field in which collaborative working is the norm, but it is also a recognition that making performances is always a highly skilled activity, digital or not and that additional complexity needs to relate strongly to performance (and known practice) rather than asking users to open up a new area of expertise. My survey suggests that Downie's 'artist-

programmer' (2005) is likely to be in a minority among those working in interactive multimedia and live performance.

4.1.1 Generic Features of Practice

The results of my survey suggest that just because a particular use of technology is widespread, it does not follow that it can be considered as generic; use of Isadora and Max and the projection of sound and video were features of almost all the practice discussed by subjects, but they were also areas where the characteristic individuality of artefacts tended to be located. Artists have found ways of working and appropriate tools that produce the results they want and which complement and enhance their performance practice.

However, my survey results have also suggested that there are genuinely generic areas of practice: operations and processes of digital connection and control that are features of almost all the work that was mentioned by interview subjects and which is both currently problematic and only peripherally involved in those features of creative work that is at the observed surfaces of digital performance.Rather, these generic features underpin the structures and processes which produce those characterising phenomena. It is clear that one of the features of digital performance is connectivity; the mutual influencing of performers and devices through networks of control that are characterised by triggering and cuing more than continuous data streams (although these are present).

In addition to these structural dispositions of technology, it can also be suggested that the processes of making digital performance have been changed from those of their parent performance traditions by sustained use of technology in the preparation phases of work, although there was a widespread feeling that, often for funding reasons, this access was less than ideal. The feature of digital tools to promote collaborative working identified by Lévy (1999) appears to be conformed in this sector with all practitioners mentioning collaboration as a central feature of their working processes well before actual performances were given.

4.1.2 Potential for a Generic System

Based on the findings summarised above, it can be established that the features of a generic system can be identified. The system would primarily address issues of connectivity and triggering; it would be a low-level system that sought to enhance the functionality of the devices and systems that artists are already using rather than attempting to persuade them to change their personal digital resources. It would have to be adaptable to a very wide variety of practice and material and permit performances to be shaped during preparation and performance.

A possible advantage of such a system might be a partial end to feelings of over-reliance on specialists:

> "…theproblematics with using technology is that you are blind to its workings often and I am dependent on other people." (Jane Turner)

As suggested in the rationale (section 1.2), a generic system would have to be avowedly non-specialist, speaking the 'language' of practitioners while enabling them to continue to form collaborative partnerships with specialists who would be free to concentrate on those aspects of digital performance which differentiate one project from another. It would need to be available on a longer term basis than is usual with bespoke systems or with specialist resources which are part of a the infrastructure of a specific institution or venue. It should also attempt to avoid the continuing duplication of effort and resources that one of my interview subjects identified as characteristic of digital performance: -

> "One of the things that impressed me about this world is that there are an awful lot of people doing fantastic work, but an awful lot of them are doing it on their own and a lot of the time they're reinventing the wheel." (Martyn Ware)

4.2 Creative Activity

In addition to the findings presented above, the preparation of the statement of requirements for a generic system to support work in digital performance is informed by reflections on, and evaluations of, my individual creative practice in three areas: previous experience of working in live performance and multimedia; the site-specific sound installation *The Curious Listener* (2007); and two interactive gallery installations, *Melbourne* (2008) and *The Street* (2009), made with Martin Rieser. However, the contributions of these projects to the formulation of the requirements lie in different areas to those aspects derived from the interviews; they are concerned far more with defining constraints on the specifications in terms of the scale of project for which a generic solution might be appropriate and also with the accommodation of human factors and the demands of system deployment under 'real-world' conditions. The desire to support work by practitioners without disrupting or constraining their existing working patterns requires that both the deployment methodology and the documentation of the system become important to a greater extent than might be the case for a resource designed for technical specialists or to meet a single, well-specified need.

4.2.1 Previous Experience

My experience on two projects in particular, *Fugue* and *eMerge*, carried out before the current study, have so strongly influenced the identification of the requirements of a generic system, especially in the areas of deployment and

integration with (and within) artists and performers' working patterns, that they require inclusion here.

Fugue (2002) was created in collaboration with Ruth Torr at Middlesex University; we were working with BA Performing Arts Students from drama, dance and music specialisms and put together an interdisciplinary performance work in which bespoke interactive multimedia tools were a central part of the work's development process and final content. The two main software tools were created[105] for the project and comprised a sound player which played overlapping random sections from a set of sound files and a text projector which presented animated characters and words on a black background designed to be projected as an overlay onto the performance; the performer could control how the text was formatted (there were four different choices – see Figure 7) and where the image was sited in the projector's overall display field allowing for direct interactive participation in the overall performance[106].

[105] I created both using Macromedia's Director authoring system, chosen because of the rapid development times and wide range of supported media types it afforded. The programme has since been acquired by Adobe (http://www.adobe.com/products/director/) and has been eclipsed by other packages, notably Adobe's Flash.

[106] This functionality is now very easy to implement using (for example) Isadora; however, at the time, there were no similar tools available (Isadora came out of beta towards the end of 2002 – deLahunta 2002).

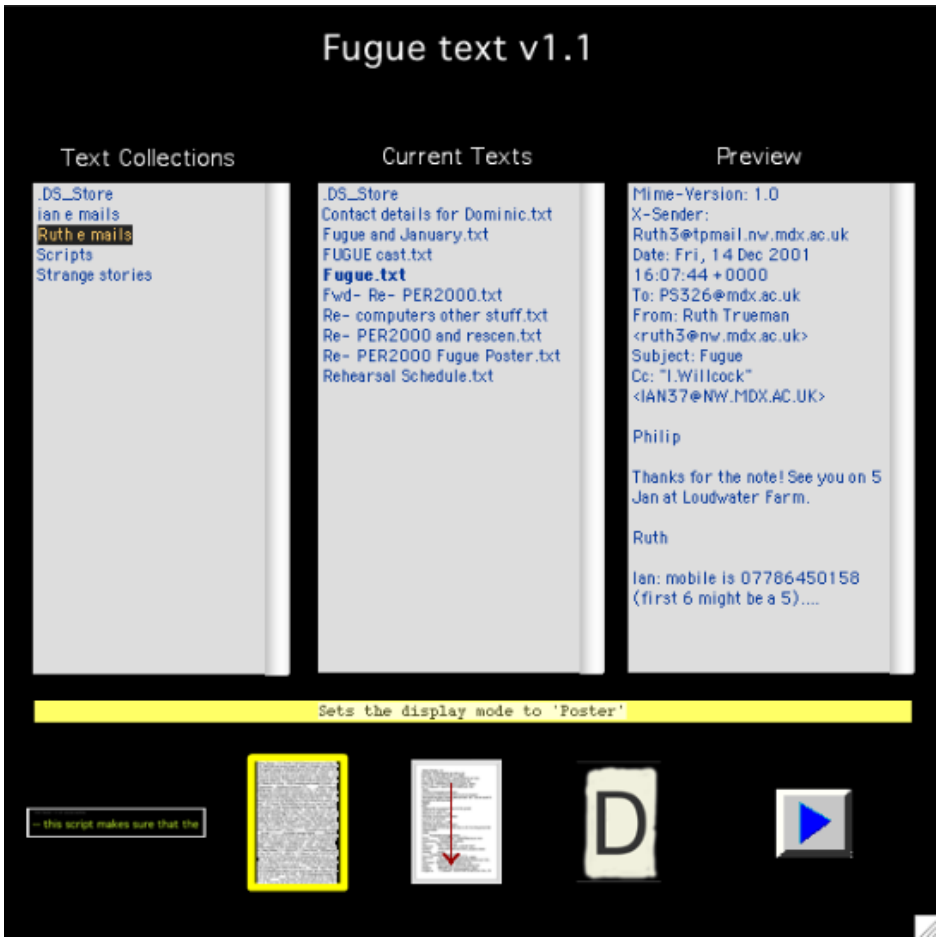


Figure 16 – Fugue text v1.1 interface (2002)

The performers' preparation included choosing and preparing their own materials, sounds and texts, which would then be used in the final piece. Both pieces of software automatically scanned a segment of their local file system when they were started and built up lists of available materials.

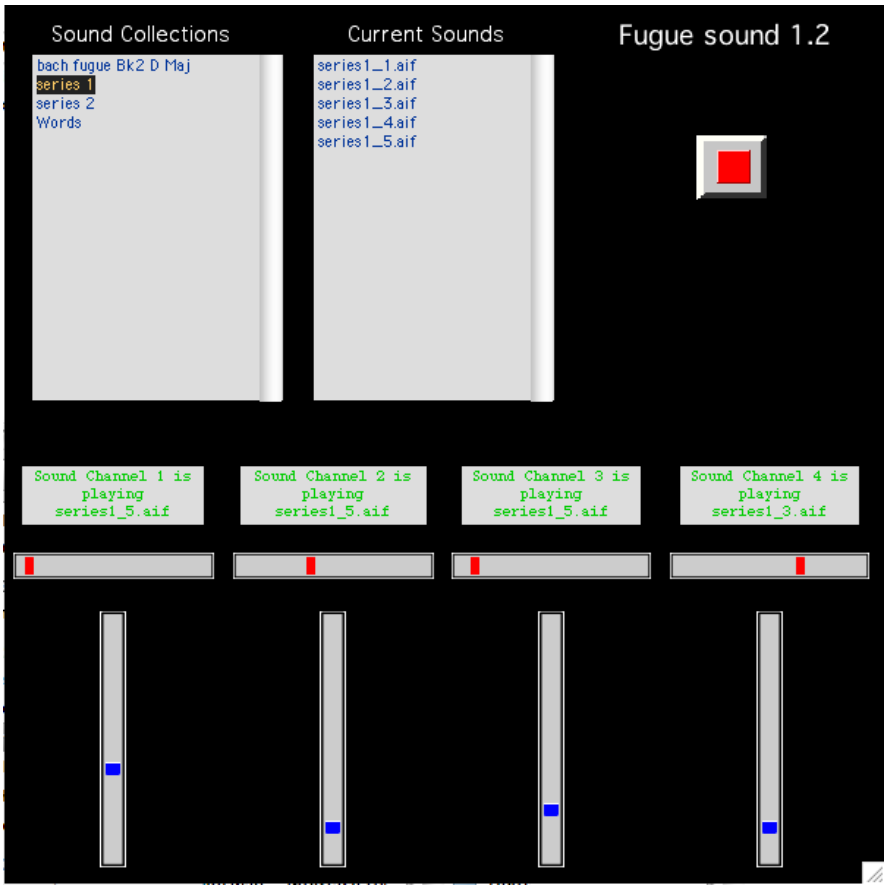


Figure 17 - Fugue sound 1.2 interface (2002)

They then allowed the user to dynamically select which sound-collection[107] or texts were presented in response to the moment-to-moment performance situation. The performers were integrating interactive technology with live performance in the same way that an improvising musician might contribute to an ensemble performance; they were choosing both the materials and timing of their interventions based on their reactions to the overall situation or, if they were in a temporary sub-ensemble, the particular part of it they were

[107] A 'sound-collection' was a set of sound files in a folder. Participants built up one or more collections, sets of sounds that they felt would work well together and had an overall affective quality, over the preparation phase of the project.

concentrating on[108]. This positioning of technology-use at points of decision-making was one of the important lessons drawn from the experience; to specify a system that supports work in digital performance the use-methodologies are as important to consider as are the actual functionalities provided. A generic system would need to not only provide 'generic functionality', a shared subset of those specific services provided by digital resources but also to map onto the linear unfolding and proliferation processes of live performance in a generic way; to be capable of integrating with the most commonly used structures of performance across as wide a range of practice as possible. A specification for a generic system to support the use of interactive multimedia in live performance necessarily requires a generalised model of performance as the basis for its implicit definition and taxonomy of performance (see section 5.1).

eMerge (2004) was devised and led by Jane Turner and Daniel Biro and was a large-scale project involving around 18 participants to investigate performance as emergent behaviour. Performers were a mix of musicians and dancers and, to support exploration of rules-based techniques, for parts of the performance they worked with a software system developed by a programming team of four[109].

[108] These reactions would include their intuitive evaluations and expectations about the future unfolding of the performance (Corness 2008).
[109] I managed the development process and provided the overall system and information architecture. I also authored the performer-interfacing client while Nick Rothwell programmed the central server and database. A visualisation component was produced by Alex Wilkie and Ian Moore.

Figure 18– *eMerge* (2004): lit squares denote number and position of floor sensors

The system was based around a number of client instances, each of which interfaced with a performer (a musician or dancer) and connected to a central hub. This architecture follows an approach to integrating technology into the performance which has been validated in a significant body of prior work (Fléty 2005, Aylward&Paradiso 2006[110], Wozniewski&Boulliot 2008). The clients collected cues from performers (these might be key presses or sound-signals) and from the performance space (MIDI trigger signals when a performer steps on a floor-pad). These cues were sent to a central server which included an inference engine and a database of rules: conditional associations and consequent actions. Rules were entered before a performance by a separate system component.

[110]Aylward and Paradiso actually use several (typically two to four) 'micro'-clients for each performer, each of which sends data back to the central processing unit.

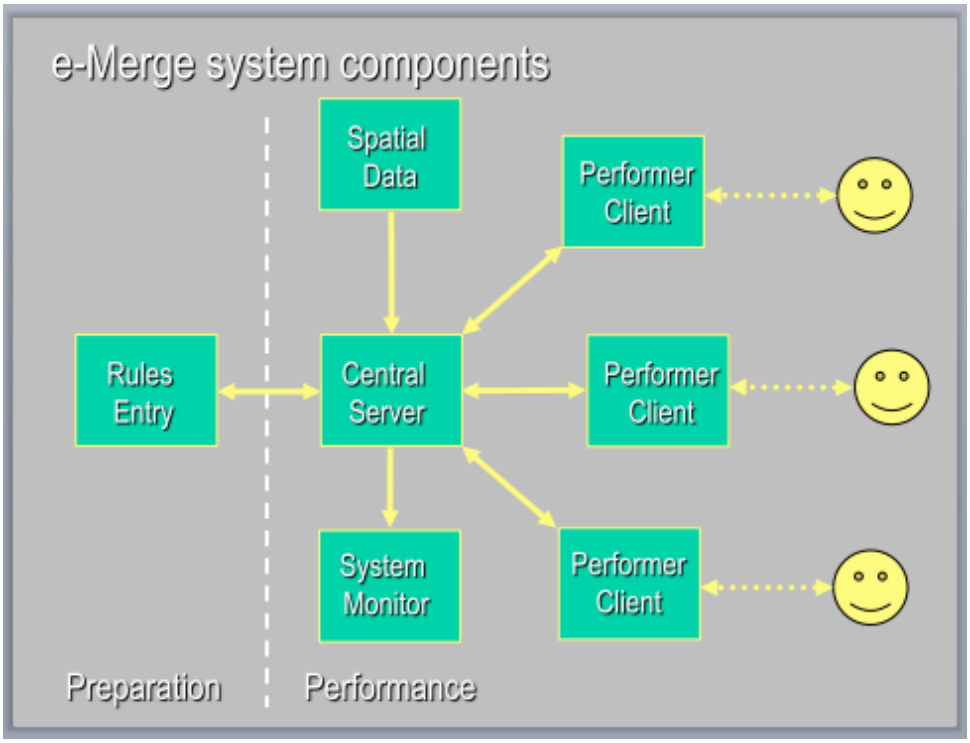


Figure 19 – *eMerge* system architecture (slide from 2004 presentation)

If the conditions for a particular rule were met, the actions were carried out – which meant that a signal of some sort was sent to one or more performers to initiate or alter some sort of activity. A system monitor allowed the audience to see something of the system's internal state, either literally (rules could be displayed as they were triggered) or symbolically through dynamic animations. For example, in the example below (Figure 20), which is taken from a piece of music I scored for five vocalists for one of the sections of the May 2004 performance, the performer has to press the 'd' and 'c' keys at the points indicated.

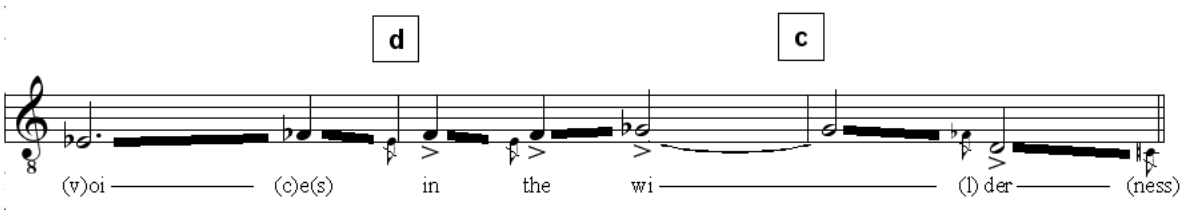


Figure 20 – Section of vocal part showing cuing instructions

When these keys are pressed signals were sent to certain other musicians and dancers to indicate they should change to a new set of materials[111].

The system was developed very quickly, but the development process allowed for the presentation of work in progress on two occasions[112] before the main performances at the ICA, London on the 24th and 25th May 2004. This allowed the development team to progressively rework software based on feedback from those devising the performance as well as on experiences of using the system in limited configurations before the main performances. It also provided numerous opportunities for observing participants as they learnt how to work with the system, a process which had implications for the way I approached documentation and interface design. The comparatively long period of time (February to May) over which the system was available meant that facility with the system could be developed[113]. It was clear that the project leaders found this very valuable both in terms of their creative ambitions for the

[111] Musicians were shown a new page of their part while dancers were shown a letter character indicating which of their blocks of choreographic material they should move to next. The whole ensemble moved through materials in a way that was consistent and highly complex but not tied to a common metric framework and not directed by any single performer.
[112] The system was presented at CAMAC, Marnaysur Seine, France (http://www.camac.org/english/intro.htm) and at London Metropolitan University during February 2004.
[113] As can be seen from the discussion of interview results above, prolonged use of systems is not the general case in digital performance.

project and from the increasing sophistication and confidence of the demands they made on performers and technology. From comparatively simple sets of rules based on simple binary tests designed to produce emergent performance behaviour, Turner eventually produced performance rule-sets distributed between technology and human performers whose decision-making criteria included memories of previous group and individual behaviour as well as current states.

> "Having established that the *eMerge* system worked satisfactorily in the preliminary research experiments, I moved on from simple action commands to the use of symbolic commands. These were used for the main *eMerge*presentations that took place in London in May 2004…" (Turner 2011:Chapter 3)

The desire of artists for prolonged availability of digital performance systems during their preparations for performances was established above in the consideration of interview results (section 4.1). However, reflection on the *eMerge* project suggests that the effects of this would not be confined to increased familiarity with systems and accuracy or confidence on the part of performers (i.e. *better* realisations), but might also tend to affect the content and nature of those performances through greater complexity and/or more developed articulations of creative preoccupations.

Another effect of this prolonged use of the eMerge system as both it and the project developed, was the iterative experience of the deployment of a complex, collaborative system for performances. This helped suggest some principles for domain-specific evaluations of usability that are independent of those

considerations of communication or affordancerelated to a specific interface design. The developing system was used by both ‘amateur’ and professional performers over the course of the project; both groups found the actual performances challenging in that the demands being made of them[114] purely in terms of movement or musical performance were such that it was apparent that the technological aspects of the system should not increase the demands made upon the performers. The effects of this on the programming team included the need for auto-configuration and retention of user settings as early development goals[115].

Similar requirements were present when looking at the *eMerge*project’s system from the perspective of the directors although, given the timescale available, their needs were never part of the development goals. Entering configurations or rules into the system required direct editing of the underlying mySQL database, a task which required a computing specialist. This meant that changes to a performance’s rule-set were cumbersome and development and experimentation were hindered. Different performances required wholesale loading of different database data; again this was a slow process that required a specialist. While these limitations were manageable within the context of a single project where technical specialists were part of the project team, they are not paradigms that could be adopted for general use; flexibility and

[114] The material and ‘performance-processing’ required were different for different groups. Turner discusses the abilities and limitations of dancers extensively in relation to the demands of emergent performance in chapter 3 of Turner 2011.
[115] These are, of course, features of much consumer software, but they are often functions that are actually implemented (as opposed to being planned for) comparatively late in a development process (where core functionality may be concentrated on first).

management of performances by those making them are essential. Further, as has been discussed above (see sections 2.1.3, 2.2.1) widespread adoption of digital performance tools is strongly influenced on ease of use and the coherence of a piece of software's structuring metaphors with the ways artists already think about their practice.

The technical specification of the platform for any proposed generic system was also influenced by experience with the eMerge project confirming the evidence of my survey; any generic system must be capable of running on readily available computer platforms. Specialist equipment generally appears to limit availability of systems to artists either through it being physically located in a specific location or too expensive. Where performance is not the intended use-case for specialist equipment, it can also be, at least to some extent, inflexible (see, for example, Miklavcic&Miklavcic 2007 for a discussion of the logistics of using the AccessGrid academic networking system in a distributed performance project).The eMerge system used a small form-factor desktop cpu running a Linux distribution for its central server which connected to clients using a switch-based local Ethernet network. As a specialist was part of the team, this arrangement worked well[116], but it would not be appropriate for the target users of a generic system; the analysis of interviews as well as experience, strongly suggests that this kind of system configuration and administration would place a system outside of the zone of acceptable difficulty for most of those involved in

[116] It did have the additional limitation that wireless connections were not possible for clients. In practice, this was less of a limitation than might be expected since many academic institutions lock down wireless access to network ports other than that (80) used for html (see section 6.2) while their management of wired connections tends to be more flexible.

making digital performance[117]. To be maximally available, the system must be useable without requiring non-standard computing resources.

The final significance of the eMerge project for the present enquiry is that it supplied a substantial codebase from which the LIMPT system was developed. While some software elements were discarded completely, almost no code was used unaltered and many aspects of the internal architecture of both central controller and client were radically changed as development proceeded (see section 6.2.2), it should be noted that the work done by the original team, particularly by Nick Rothwell, provided a very valuable resource as well as a reference point for subsequent development.

4.2.2 The *Curious Listener*

The *Curious Listener* was a site-specific installation project which, through its design and implementation, suggested principles for organising and delivering systems designed for use by non-specialists when separate system components with different functionalities are involved. The prototype project was installed in the Institute of Creative Technologies (IOCT) at De Montfort University in June 2007. There were three components to the system: a 'listener' which was sited inside the main room of the IOCT, a 'player' which was situated in the foyer and a database system hosted on the public-facing IOCT

[117] All my subjects mentioned collaboration with technical specialists and programmers as the way they managed requirements for in-depth technical knowledge. This is especially significant since several had an advanced facility with digital tools (Isadora, Logic, Final Cut Pro); they were not averse to using technology as such.

server, Bartleby[118]. The components connect with each other over a standard TCP/IP network: -

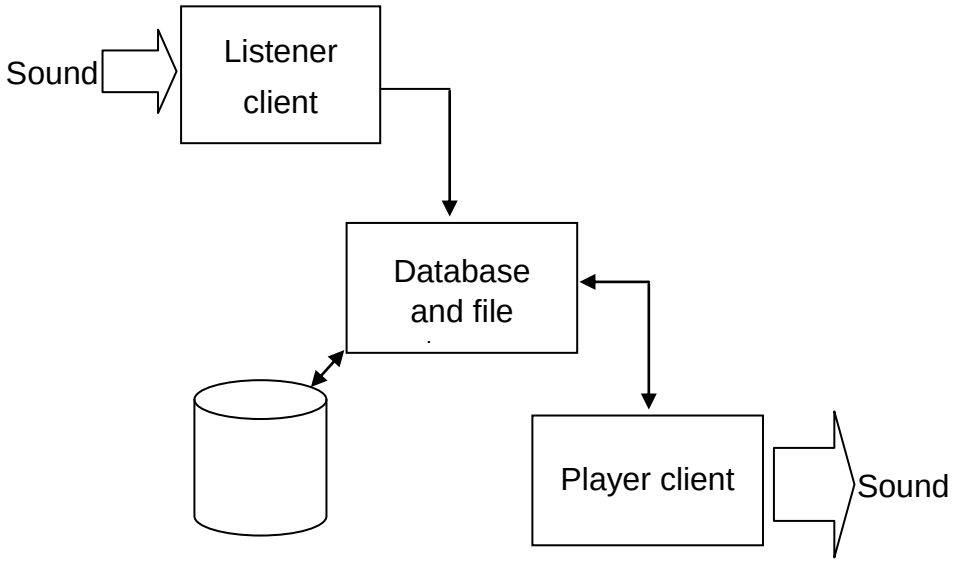
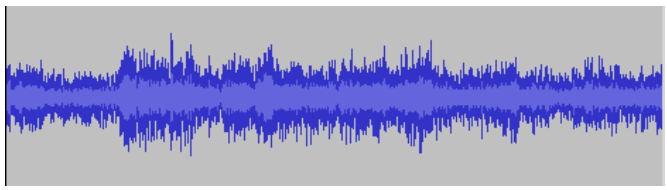


Figure 21 – *Curious Listener* components

The listener client automatically monitored the audio activity in the main IOCT area and recorded anything interesting[119]. When sound activity ceased and the recording was finished, the recording was analysed[120] and the listener then

[118] Available at http://bartleby.ioct.dmu.ac.uk
[119] The listener client continually monitors the audio input level. If it rises rapidly over a set sensitivity threshold, it triggers the recording component which logs the start time and begins recording. The recording continues until a significant period of non-activity has occurred (1.0 seconds in the current configuration).
[120] The input level is monitored throughout the recording and off-on and on-off transient times are logged. For example, here is a sample recording (note this has been normalised to -0.3 dB so that transients can be identified visually).



The recording shown here is 2.77 seconds long and the listener client identified four transient pairs (on + off) within this time:

uploaded the digital sound file and information[121] about the audio to a (remote) database component.

The database system stored information about the recording, such as when it was made and what sort of sound activity it contains and maintained a catalogue of recordings which grew over time. The database was (is) accessible over the Internet via a phplayer which provided the network interface for listener and player clients.

The player interrogated the database and retrieved any recordings made within a specific timeframe (i.e. the day before, the same time a week before etc.). These files were then sampled using a randomising algorithm and a customised dynamic envelope and played back through multiple sound channels (mixed down to stereo in the prototype system). The effect was of a continuously changing impressionistic sound collage in which odd, isolated words and phrases were heard fading in and out, but anything longer than a couple of seconds was truncated and fragmented.

0	0.04
0.524	0.566
1.125	1.165
1.73	1.77

As soon as the recording is ended, the transient log is written to a text file and then the analysis component runs a simple rule-based algorithm taking as inputs the transit density, the average length of transients and the duration of the shortest transient. This process returns a probable audio-type for the recording – currently, the system attempts to distinguish between speech, music and 'unusual noise'.

[121] Eventually it is envisaged that the server system will attempt to further analyse recordings based on their formant signature, the patterns of unique sonic resonances present in an individual's speech – from which the system would attempt to identify who was speaking.

The significance of the *Curious Listener* project for the development of requirements for a generic system for use in Live Performance lies in the implementation of the *Curious Listener* as a set of components with different functionalities connected using standard, public Internet IP addresses and using a standard transport protocol. This is in contrast to approaches based on maximally complex solutions (a single software/hardware instance that implements all system functionality) or to approaches involving components with different functionalities interconnected using a dedicated local sub-net (the *eMerge* project), or other form of networking such as MIDI (Nagashima 1998) or specialist protocols (Ramakrishnan et al 2004, see also Friaetta 2008).

4.2.3 Installations

As a part of my research activity, I worked on two linked installation projects, *Melbourne* (2008) and *The Street* (2009) devised and led by Martin Rieser. Both were interactive installations designed for exhibition in gallery spaces and both included a significant performative aspect in that part of the intention was that the audience should watch an audience member interacting with the work; it was both a shared (group) and individual experience, it

> "…allowed for both active and passive modes of audience consumption. The random allocation of a house to a particular visitor enhanced both the curiosity of the 'active' viewer and an observing 'passive' audience by encouraging them to explore the associated narratives for each house." (Rieser, The Street)

In considering how such work might be relevant to the main research project, it needs to be remembered that many artists working with technology (Sarah Rubidge, Jem Finer[122], Brian Eno, STELARC) work across and between both live performance and installation and indeed some authors (Dixon 2007) specifically include installations within the digital performance field (see section 2.1.2).

The project involved a large, long image of a residential street which could be scrolled smoothly across a composite display created by three linked data projectors – only a small section of the image was visible at any one time. When a member of the audience entered a zone close to the image, a particular house was chosen by the system at random and the image of this house then moved into view and tracked the movement of the audience member as they moved along the projected image. If the person stopped moving, their house blossomed into a short, poetic video exploring the themes of the exhibition.

[122] See, for example, Finer 2011 for an overview of *Longplayer*, a sound piece/installation/occasional performance with a planned overall duration of 1,000 years.

Figure 22 – Melbourne at RMIT, Australia 2008 (image: M. Rieser)

The tracking was carried out by analysis of a video stream from an overhead camera; however, given the lack of control over the environment (e.g. numbers of people, clothing colour) and the variable light levels as the projected image scrolled through lighter and darker areas, it was a fragile process and the system had to be able to cope with a situation when the sensor could not identify an audience member's position with confidence.

In original discussions, it was clear that a modular approach to the application design would need to be used; I did not have time to do all the coding and so other programmers were creating the motion tracking functionality. The development time available for the 2008 exhibition was such that development had to proceed in parallel, all parts being testable before the final system was

assembled. The possibility of using the LIMPT system, which was in early development, was discussed initially, but it was felt to be unnecessarily complex approach, so for the first showing, a single application[123] was produced which combined sensing, video processing, application logic and display driving.

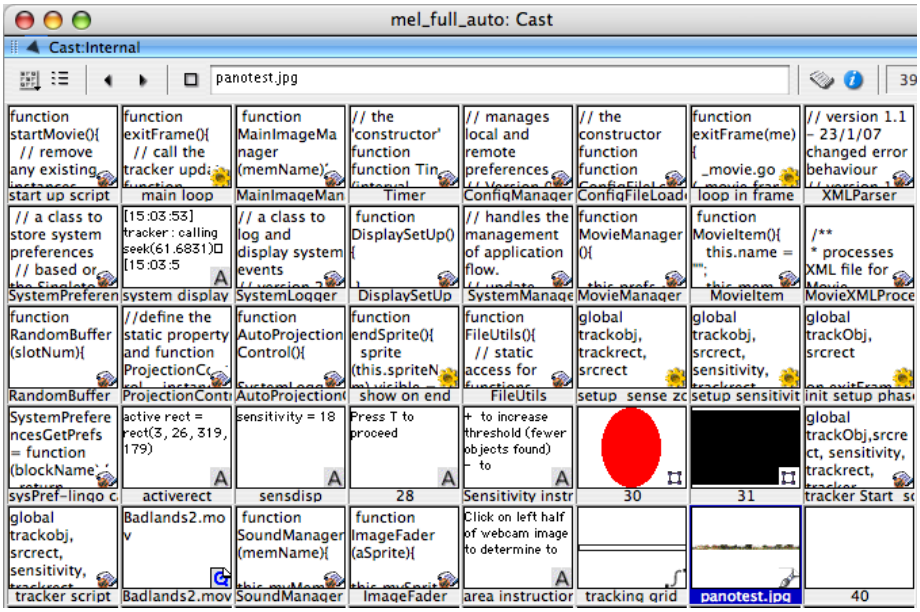


Figure 23 – Melbourne application (2008 version): internal software components

This application eventually involved 29 separate script elements, mainly class files (see Figure 23). Due to the lack of access to equipment and the requirement for setting up in Australia, testing of the whole assembled system was not carried out over long periods (the installation was intended to run continuously) and crashes due to a memory leak in the (commercial) video driver plagued the system. Video input and processing were a particular problem; video processing plug-ins for Director were not cross-platform and recent Windows updates combined with new chip-sets in available web cams

[123] Director was used as the development platform, partly because of its speed of development, but also because plug-ins were available which could do video processing and it could cope well with animating very large bitmap images overlaid with Quicktime videos.

had severely limited the functionality and stability of those that were available. Matters were made worse since the display-splitting system[124] only worked with certain manufacturer's machines. The result of these difficulties meant that the system was not fully operational during the first showing in 2008.

For the more developed version of the project shown at ISEA 2009 in Belfast, *The Street*, the team decided that a fundamental rethink of the architecture and engineering approach were required; in particular, display and sensing needed to be separated as finding a single machine able to do both with good accuracy and stability seemed highly unlikely. Computation was thus split between two machines, one concentrating on video tracking and the other on application logic and display. Connecting the two machines was achieved by using an OSC plug-in[125] for Director over a small Ethernet network (Martin, Forster&Cormick2010 adopt a similar approach to connection). Location data along the projected image was sent as an integer parameter while other messages were adopted for acquisition and loss of a subject to be tracked.

[124] We were using the Matrox Analogue TripleHead system (http://www.matrox.com/graphics/en/products/gxm/th2go/).
[125] We used the free OSCXtra from the Interactive institute, Sweden (http://www.tii.se/node/2520). This is available for both Mac and Windows machines and is very straightforward to use although it does not implement the whole OSC message-set.

Figure 24 – *The Street* set-up (two computers, overhead webcam and three projectors), Ormeau Baths Gallery, Belfast 2009

The second iteration of the project was more successful; while tracking was still sensitive to environmental factors and could be upset, the system did also work effectively for long periods. Even when location-tracking data was temporarily unavailable, the display was able to maintain in automatic mode, switching to tracked mode seamlessly when data flow was resumed. The significance of these installations for the current project is that taken together they suggest a lower limit for the scale of project where a generic solution might be appropriate. The ISEA09 version of the installation had moved some way towards the architecture of the LIMPT system; indeed, much of the complexity in the display application concerned the complex condition-based decision-making that drove the panning of the output image and showing of individual videos. Despite this, the bespoke solution remained probably the best use of available resources,

although any added complexity in terms of system elements or processes (i.e. interfacing with additional sensors, dealing with more complex data or controlling a wider range of devices) would almost certainly mean that using a generic framework as the basis for the system architecture would become increasingly attractive.

4.3 Statement of Requirement

From the results presented above, it is now possible to propose a formal statement of requirement which outlines the features and functionalities of a system which might support the work of practitioners from a range of performance backgrounds in working with interactive media in live performance. The presentation of these requirements follows standard software development practice[126] in categorising requirements under the following headings: Data, Usability, Technical Requirements and Functional Requirements.

The broad topics underlying the proposed requirements are concerned with enabling flexibility, connectivity and ease of use (the major shared pre-occupations of interview subjects[127]).The requirements presented here describe the benchmarks for a generic system, one which aims to give artists ownership of the epistemology and methodology of the creative and performance processes involved in live performance and to allow them to establish contextually-determined dynamic links between their chosen performance participants (both human and machine) without disrupting their established working patterns.

[126] JISC infoNet– Creating the Statement of Requirements (2011) was also particularly useful in developing the format used here.

[127] Another important theme that emerged from the study was system accessibility; artists of all backgrounds wanted to be able to use technology without requiring specialist resources over an extended period so they could build up a technique and familiarity with systems without the pressure of having to ensure a performance was ready within a very short time-span.

4.3.1 Data

Category	Function	Description	Purpose	Priority
Management	Storage	System configuration and settings	The operational settings for the system's components need to be stored locally with a reasonable set of default values provided.	2
		Performance configuration	The specific connections, mappings between different devices and any assets required need to be retained in a secure and reliable way throughout preparation and between performances.	1
		Performance archiving	The system should retain a record of what happens in a performance for archiving and review.	5
	Access	Physical location	Access to data needs to be independent of location so that the system is useable from rehearsal rooms and a range of performance spaces.	1
		Time and duration of access, availability	Access to data needs to be at times, and over time-periods, that are not constrained by assumptions about working patterns.	3
		User access to data	The system manages access to performance and system data based on a system of user privileges.	5
	Internal communication	Speed, bandwidth	Data needs to be processed and transferred between devices at speeds compatible with human performers and to be able to cope with multiple overlapping demands for mapping and connection.	1
Input	Sensing performer activity	Simple digital inputs	The system needs to accept simple inputs such as pressure pads, computer-key presses, mouse clicks.	1
		MIDI performance data	The system can accept a range of MIDI input data	2/3
		Other complex data	The system can use gestures or positional data as inputs	4

Category	Function	Description	Purpose	Priority
	Input from devices	Interface boards as inputs	The system can accept input from interface boards such as Arduino or iPac.	5
		Stand-alone digital systems or devices	The system can accept input from other digital systems such as Max patches or bespoke applications.	4
	User control	During preparation of a performance	A user should be able to enter the information required to create and edit the configuration and settings associated with a specific performance as it is devised.	1
		During performance	A user should be able to engage with the progress of a performance in real time.	2
Output	Connecting to devices	Triggering video, sounds, lighting changes or other effects via MIDI	The system should be able to connect to and initiate activity in output devices connected via MIDI.	3
		Complex continuous control of systems or devices	The system should be able to control the operation of a device (e.g. a lighting board or robot) or system (e.g. an application generating an avatar or performance environment) at a level beyond that of starting and stopping activity.	4
		Interface boards as outputs	The system should be able to use interface boards to control actuators such a servo motors and other low power devices.	5
		Live monitoring of system	The system should allow its operation to be monitored, remotely or within a performance.	4
	Communicating with performers or audiences	Cuing performers (visual/audio)	The system should be able to provide a simple cue to a performer in different ways that are appropriate to a range of performance situations.	1
		Presenting text, image or sound material to performers or audiences	The system should be able to present pre-selected text, images or sounds. These could be instructions, scores or other materials associated with a performance.	2-3

4.3.2 Usability

Category	Function	Description	Purpose	Priority
Target users	Performance makers	Simple connection and set-up	Setting up the system should not be different to normal computer usage (so may involve connecting to the Internet and running an application, but not more complex operations).	1
		Performance preparation: skills and knowledge	The preparation and configuration of the system for a performance should rely on performers' established concepts about performance and performers.	2
		Technical skills requirement	Technical or programming knowledge and skills should not be required as far as is possible.	2
		Use in performance	Creators should be able to intervene in and contribute to performances as they are taking place.	2
	Performers	Performance preparation: skills and knowledge	Minimal active preparation should be needed for performers and technical knowledge should not be required. Any configuration settings should be retained locally.	2
		Use in performance	The system should demonstrate clearly that it is working, but should operate in a way that integrates with the performer's specific discipline as transparently as possible.	1
	Developers	Source code available	Non-commercial parts of the source code should be available through a standard repository web portal.	4
		Source code documentation available online	The source code should be documented in a standard format (e.g. javadoc) and made available online.	5
		API and documentation for extensions	There should be a template for adding functionality to the system, particularly for adding new sensing or cuing capabilities.	3
Training	Community	General introduction to system	There needs to be a presentation which describes the system and its capabilities to those who are not users but who might be interested in working with it.	3

Category	Function	Description	Purpose	Priority
	Performance makers	Reference resources for preparing performances	There needs to be comprehensive, searchable reference documentation available with examples of how to prepare for a performance and implement specific functionality.	2
		Working demonstration	A interactive set of working examples showing how the system can be configured to operate in various ways.	4
	Performer	Short guide for performers	A limited guide which can be read or communicated quickly and which sets out just sufficient detail for performers who are not involved in performance preparation.	4
UI views	Navigation	Clear, task-based navigation structure with a small number of user interface (UI) views.	The interface should be based only on a small number of views of the system. which reflect phases of activity involving a set group of tasks. It should be immediately obvious which view is being presented.	1
	Configuring	System configuration options	The system must allow users to configure settings.	1
		Configuring extended functionality	The configuration view must allow configurations for a variable number of system components to be accommodated.	2
	Preparing	Performance preparation overview	The system allows users to prepare and save a performance configuration.	1
		Retrieve and store performance configurations.	The system should allow users to view and edit saved performance configurations.	2
		Feedback on outcome of performance configuration operations	The system should confirm that users' work or assets have been stored.	1
		Store performance assets	The system allows users to select and store assets (e.g. images, texts, sounds…) to be used in the performance.	4
		Manage asset storage	The system should allow users to manage and preview stored assets.	4
		Integrated help system	The system should assist creators, through interface design and contextually aware help-text, with the process of preparing performances.	3-4

Category	Function	Description	Purpose	Priority
	Performing	Optimised performance view	The system should present an interface which is focussed on the needs of performers and which is informed by the way they are relating to the system.	1
	Language	Vocabulary and structure to be based on natural language	The language employed by the system for feedback and configuration should be close to natural language and use a vocabulary drawn from performance where possible.	1
		Provision of versions of system localised for other languages	The system and its documentation are available in languages other than English.	5

4.3.3 Technical Requirements

Category	Function	Description	Purpose	Priority
Platform	Hardware	System should run on standard hardware that is readily available	The system should be usable with non-specialist hardware having performance characteristics and hardware configurations that reflecting those found in affordable systems.	1
		Required user interface devices	The system should be useable at some level with standard UI devices (mouse and keyboard).	1
		Cross-platform operation	The system should be available for appropriate combinations of operating systems and devices commonly used by digital performance practitioners.	2
		Standard network connection protocol(s) used for connections	The system should use connection methods and transport protocols which are supported widely.	1
	Availability	Stability	The system should be stable enough for live performance use.	1
		Usable at any location	The system should be usable for preparation and performance in any location.	2
		Usable across any physical scale of venue or performance.	The system should not make assumptions about the size of a performance venue or of the proximity of creator(s) and performers.	2
		Available over extended time periods	Users should have the opportunity to engage with the system over extended periods both during preparation and over the course of development projects.	2
	Configuration	Default configuration	The default configuration settings should allow users to begin to work with the system without requiring extensive customisation.	2
	Peripherals	MIDI interfaces	The system should be able to use MIDI interfaces where available.	2

Category	Function	Description	Purpose	Priority
		User Interface Devices (UID) as input	The system accepts input from UID.	4
		Use of controllers	The system can interface with controllers such as the Wii remote and Kinect.	5
Architecture	Scalability	Complexity appropriate to projects involving three or more devices	The system is applicable for projects that involve three or more devices or performers. For smaller-scale projects a single stand-alone application is probably preferable.	3
		Able to accommodate a wide range of performers or devices at a time	The system can accommodate between three and fifty performers or devices within a performance without degradation of response.	1
		Accommodation of intensive processing requirements	A requirement for intensive processing should not affect other parts of the system's operation.	1
		Manage and support multiple performances	The system should be able to accommodate multiple performance projects over the same time period.	3
extendibility	System development	Defined architecture, API and documentation for creating extensions	There should be a template for adding functionality to the system for adding new sensing or other capabilities to meet the specific needs of creators that are not already catered for.	2
	Interoperability	Protocol and format for interconnection between system components	System components should communicate using a bandwidth efficient, flexible, well documented and portable format.	1
		Interface with digital tools that artists are already using	The system should work with the digital tools and devices already used by artists in their work.	2
		Capability to interface with bespoke software and hardware	It should be comparatively easy for specialists to create interfaces in new or reworked bespoke systems that work with the system.	3
	Source code	Use of version control system	Development should use a version control system for effective management of the development process and for enabling collaboration in future stages.	1
		Open Source license	As much as possible of the code should be made available free, preferably under an open source licence.	3

4.3.4 Functional Requirements

Category	Function	Description	Purpose	Priority
Connecting	Routing	Specific routing	The system should facilitate connections between a specified performer or device and any, or all, other performers or devices.	1
		Multiple routings	The system should allow many specified routings simultaneously.	1
		Dynamic routing	Routings should be capable of being changed at any point during preparation or in performance.	2
	Contextuality	Conditional routing	Routings should be conditional, where connections are made, or not, depending on specific activity.	1
	Specification	Storage and validation	The system should accept, validate and store routing specifications.	2
		Formulation of routing specifications	The system should accept routing specifications couched in language and structures that are based as far as possible on those used in performance rather than computing.	2
		Variable level of detail for specifications	Routing specifications should involve only the level of detail required.	3
Performance	Tracking	Track the activity of performers and devices	The system should maintain a data structure representing the activity of all performers and devices involved in a performance.	1
		Track time	The system should enable timing to be part of a performance's properties.	2
		Track physical spaces or locations	The system should enable spaces or locations to be part of a performance's properties.	2
	Journaling	Storage of tracking data	The system should store its tracking data.	4
		Support reviewing performances	The system should allow users to review a performance's progress through its stored tracking data.	5

Category	Function	Description	Purpose	Priority
System and configuration	Components	Auto-connection and configuration	System components should be automatically connected and configured.	1
	Users	User settings should be saved automatically	Users' settings for any aspect of a performance should be saved automatically.	2
		User access management	Users access to performances is managed.	4
Input	Processing	Quantisation	The system should quantise performance activity with some user control over the scale.	1
		Mapping from one activity event type to another	The system should be capable of appropriate mapping of events between performers and devices.	1
		Passing activity data	The system can pass data from one device to another.	2
	Extensibility	Support for addition of extra sensor-types or new devices	The system should allow additional information types or formats or new devices to be used as inputs.	4
Output	Triggering	Conditional cues or triggers	The system should provide cues or triggers for performers or devices that are appropriate in format and medium and are conditional on specific activity.	1
	Extensibility	Support for addition of extra output-types	The system should allow additional information types or formats to be used as outputs.	4

4.4 Chapter Summary

This chapter has presented the results from two of the primary research strands of the study: the set of interviews with practitioners and personal creative activity focussed on a set of installation works created alone and in collaboration. Analysis of this data has established some important results for the study as a whole. The first of these is that it is possible to identify, with a high degree of confidence, certain generic features of practice across work in digital performance which is drawn from a range of performance traditions and genres. However, it is also clear that the preoccupations of artists with their own methodologies and practice require that the features of any generic system are comparatively low level, supportive at the level of infrastructure rather than at the higher level of a public-facing resource which shapes the form and content of users' experiences. Thus the main features of a generic system would focus on ubiquitous connectivity and contextually aware triggering or cuing of devices and performers. The control of such a system would need to avoid as far as possible language and ideas from computer science, but would rather relate to artists' existing knowledge and conceptualisations about their own performance traditions and practice.

The chapter then looked at the researcher's creative activity, looking at a number of projects and examining their relevance and significance for the current research. The chapter then drew all this evidence together and established a detailed formal set of requirements for a feasible generic system designed to support work with interactive multimedia in live performance.

5Computational Model

The statement of requirements established above sets out the functional, technical and usability features of the proposed system. To move from this to an actual implementation requires not only consideration of how the requirements may best be met in terms of engineering, but also a conceptual understanding of how a technological tool may engage with live performance. This in turn requires that a general model of performance is produced which describes (or at least accounts in a useful way for) the ways performers engage with and operate within performances. The requirement for a general model is particularly acute since the proposed generic nature of the system requires that it should be usable across many, heterogeneous performances rather than just one, specific exemplar. The approach taken in the current study proceeds in two stages; first a generic approach to modelling performance is described and then, proceeding from this, a formal computational model describing the operation of a generic system to support the use of multimedia in live performance is outlined. This latter model thus describes the formal and computational structures which underpin and inform the subsequent implementation of the prototype system described in Chapter 6. Some material in this chapter is drawn from the paper, *Words of Power*, published in the International Journal of Humanities and Arts Computing (Willcock 2010).

5.1 Modelling Live Performance

A system which seeks to integrate itself in a generalised way with live performance will require a generalised model of performance itself, a theory about what can happen and how different parts of a performance may be connected. This is not to suggest that these are the only or even the most important questions one can ask about performance, or that they alone are in any way sufficient for a 'Theory of Performance'[128].My goal is much more limited: looking at performance in a systemic way, can one identify classes of phenomena and the connections between them and, from this, can one identify ways in which technology can support and intervene in these operations? Underlying the proposed model is a view similar to that of Susan Broadhurstand others (see section 2.1.2) in seeing digital performance as an extension of existing performance traditions rather than as a completely, or mainly, new performative genre[129]. In developing a formal model of performance, I will consider digital performance as a subset of all performance activity, differentiated by the incorporation of technology with a consequent, and potentially radical, alteration to its means and articulations and creative processes, but with an underlying formal structure of moment-to-moment operation which is shared with non-digital performance.

[128] Indeed, for practitioners and audiences they might perhaps be among the least important questions, as they do not address 'purpose' or 'meaning'.
[129]Some authors do see the development of performance disciplines which seek to integrate new technologies as representing 'new paradigms of emergent practice and discourse' (Chatzichristodoulou et al 2009:1) and (at least to some extent) Dixon (2007).

In attempting to produce a single model able to account for the longitudinal connections of performance activity, a taxonomic, enumerative approach would be obviously futile due to the essentially heterogeneous nature of the phenomena being studied; creative acts are, by their nature, diverse and resist categorisation. Indeed, the major reason behind the limitations of most of the models discussed in section 2.1.3 is a result of their material basis; they attempted to model performance (explicitly or implicitly) by specifying a possible range of constituent materials. For the purposes of the current enquiry, and having a much more limited aim than a 'complete' theory of performance, I am proposing an approach which is influenced by set theory[130] and Boolean logic[131]. The model described below seeks only to provide an account of the moment-to-moment unfolding of live performance sufficient to allow the incorporation of technology in performance in ways that do not disrupt the performance production process.

5.1.1 A Generic Model of Live Performance

For the reasons identified above, in attempting a generic account, one cannot (and should not) attempt to enumerate the constituent elements of live performances, but one can identify the set of such entities and then attempt to

[130] Set theory developed from the middle of the 19th century and Georg Cantor is widely considered the most significant figure in its development (Ifrah 2000:262, Ferreirós 2011). Among many important and powerful mathematical operations, it allows manipulations to be conducted on sets or collections of items (numbers, infinities etc.) without necessarily having to engage with every detail of the collection's (set's) members' properties.

> "In what may be termed the *calculus of sets*, a highly abstract concept of algebra has been adopted, which is entirely independent of the nature of the elements involved..." (Ifrah 2000:258)

[131] George Boole developed an algebraic model for formal logic in the middle of the 19th century. This model underpins all computational logic (Ifrah 2000:84).

conduct operations on that set as a whole without trying to specify anything other than the criteria for inclusion in the set.For any sort of creative activity, those criteria are generated by the artist(s) and selection by the artist(s) is sufficient to place an element within the target set. A generic model must locate the specific taxonomy of a performance within the control of the creators of that performance.

Each individual live performance is a member of the set of live performances, which is a particular subset of the set of all creative products.I am going to suggest that the set of live performances is (at least partly) distinguished from other creative products by its members' constituent elements being activities, by being made up to a large extent of things happening.I am also going to suggest that the 'live' aspect of my set's name implies that a live performance's constituent activities are, to some extent, produced 'in the moment'[132] and are dependent for their final performed form on conditions and dispositions at that moment within the performance as a whole.They are, to a greater or lesser extent, unpredictable until they are produced, at which point the probabilities governing the composition of their nature collapse onto the actual performed gesture, a gesture that then becomes part of the ongoing fabric of the performance. I will further formalise this by stating that these happening

[132] Live performances also often have some elements that are not 'happenings';scenery, ambient qualities etc. These are not factored into the proposed model as a matter of course since they generally do not affect the moment-to-moment progress of a performance. If they did, however, the model is sufficiently general that they could be incorporated without needing to alter the model.

phenomena are a necessary condition for live performance[133], but not a sufficient one; there are obviously many situations where events occur that are produced in-the-moment and that are not (generally) considered performances[134].

Note that this construction is not making any assumptions about the nature of these 'live happenings'making up a performance. They can be in any medium, they can be quick, slow, repeating, random, numerous, rare, novel or entirely predictable.They may indeed be simply the 'start' and 'end' of a performance (John Cage's *4'33"*of 1952 perhaps) or subtle and not noticeable by some or all of any audience that may be present. However, whatever their nature, they are not definitively shaped until they are actually performed. In improvisation, a gesture may take many forms or may not be made at all; in a conventional theatre work, a word or physical movement may be inflected in a number of ways that reflect the creative processes of actor and director.However, even after extensive rehearsal, there is always some room for variability, even for surprise.The point of my proposed description is not to rigidly define what is and is not a performance, but rather to construct a set which will include the vast majority of activity which might be considered live performance. I am not seeking to pin artists down and restrict what can be done, but rather produce a model that can accommodate and welcome the wild imaginings and

[133] It might also be necessary (should one wish) to distinguish live performance from film or recorded performances through specifying something about the nature or origin of the activity involved, although some authors (Auslander 2009) would resist this.
[134] There may well also be other creative genres that have this condition and which are to some extent contiguous with live performance.

interdisciplinary nature of work that seeks to apply digital technology within live performance.

Given that we are able to identify a set containing the constituent gestures and actions that make up a live performance, it is now necessary to explore the connections between these, to derive a rationale for the progress of a performance from one moment to the next. In considering the connections between live actions, I am going to propose a quality belonging to them called *significance*.For my purposes, significant activity (i.e. an activity that has the quality of significance) is that which is meaningful in determining the unfolding of a performance; it is an *event*.Events are those actions or combinations of activity or qualities of performance-elements that identify points in a performance where its further progress becomes more fixed, where some of the potential variability of following activities is collapsed and a selective process operates.An event is a signal or gesture or a state that has implications for what happens next[135].Just as any activity becomes part of a performance through an artist including it in that performance, an activity or state becomes an event through its consideration in determining the future course of the performance.Performers follow this process many times during a performance, sometimes consciously ('I'm going to watch the soloist here for the downbeat', 'I'll start my speech as soon as the video reaches a certain point'), sometimes

[135] Events might include a performer reaching the end of their material or making a particular gesture or taking up a position on stage, lights fading to blackout, a particular image being projected or the end of a section.

atan instinctual, habitual or intuitive level[136] ('I felt that something disjointed was needed').

The structural or narrative determinations that are made when events occur can be highly complex and involve many terms and fine judgements (see, for example, the decision-making processes built in to the KTH model).However, it is possible to build up complex logical operations from just a few basic operations[137] and my deliberately simplified model is able to operate convincingly if the decision-making process is limited to applying a few simple binary conditional rules (e.g. 'If the dancers finish their phrase then start playing the next section').A binary conditional rule relates qualities of one or more events to a description (e.g. an actor says "Welcome", the harp plays a low B or the dancer stands still) and gives a yes or no result; one is waiting, watching out (perhaps expectantly) for something and it either does or doesn't happen.Based on this yes/no result, a performer can adopt a particular strategy for future activity which may have been decided in advance or may be 'spontaneous'. This strategy may include the forming of gestures in particular ways; by making decisions based on the situation within the performance, the future unfolding of the performance is affected.

[136]Corness (2008) proposes a similar model of performance but places particular (indeed I would argue too much) emphasis on trust and negotiation based on intuitive understanding of peers in reaching decisions about the future progress of a performance.

[137] Boolean logic is able to derive all the four basic arithmetic operations (add, subtract, multiply and divide) through combinations of just three logical operations, AND, OR and NOT. I wouldn't want to suggest that this implies that all operations are able to be built in this way, or indeed that the decision-making processes that form part of performance activity are exclusively algorithmic or computable (although, if pressed, I would say that most of them are, see Willcock 2006).

In my proposed generic model of performance, performers are seen as autonomous agents engaging with the performance by monitoring activity for significant events, making decisions based on pre-existing rulesand what they have noted as significant and then producing or inflecting subsequent activity (see Figure 25). The rules which frame performers' decisions may be explicitly codified in the form of a score or script or other format, or may be based on experience, implicit rules derived from stylistically defined performance practice. Often, there may be a combination of both types of rule-sets in use.

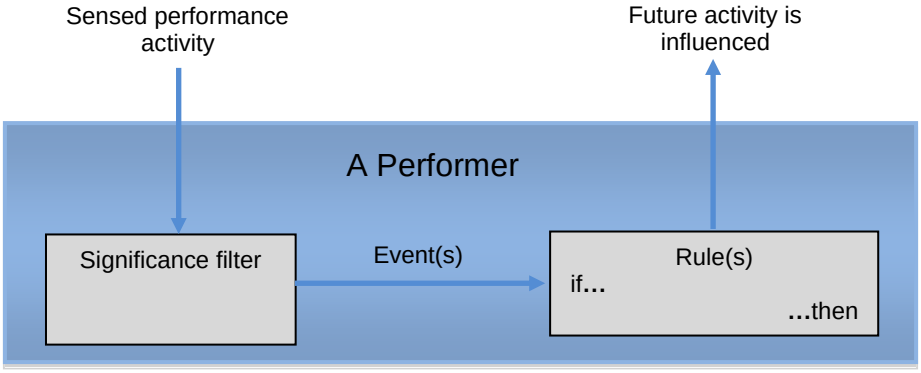


Figure 25 – Idealised performer model

The model presented below (Figure 26) represents performance as a process involving the activity of one or more simplified, idealised performers. These performers are influenced by each other and by other aspects of the overall state of a performance (elapsed time, a position in a score or script etc.). They individually make decisions on further activity based on these perceptions in conjunction with a pre-existing rule-set. The process of performance is characterised by a combined, co-ordinated decision-making; thesum of performers' actions represents the material of the performance itself.

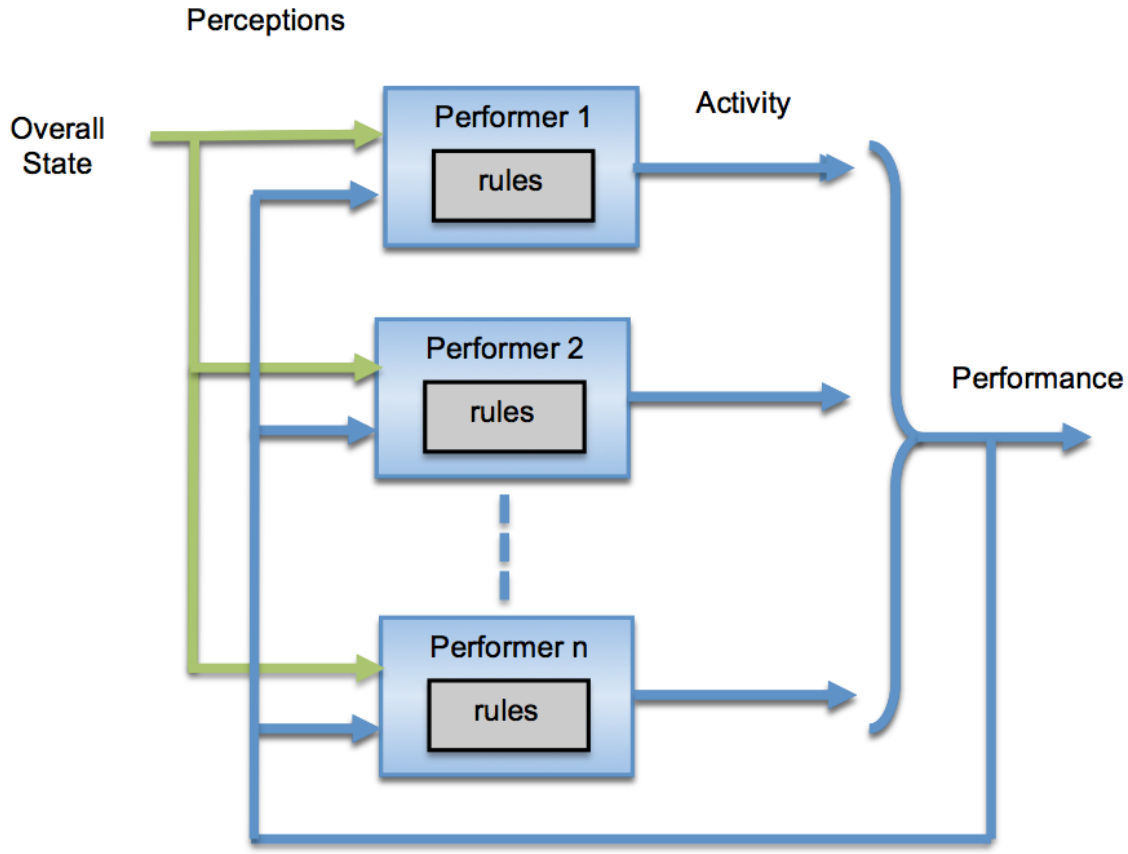


Figure 26 – Generic live performance model

For application to digital performance, the concept of a 'performer' is extended to include *any active element* involved in a performance. An active element is defined as a self-contained system which reacts to events according to a set of rules or strategies. So an element might be (indeed most often is) a human performer, but could also be a lighting desk with pre-configured settings or a software object such as a Max agent, but not a CD player or a non-automated mixing desk. One important difference that arises as a result of this extension of the definition of a performer concerns the sensing of activity; human performers have built-in perceptual abilities, they can sense a wide variety of activity in

performance and are then able to filter this material for significance. These capabilities are not general to most standard pieces of digital technology used in digital performance (lighting desks, etc.); these kinds of devices rely rather on being fed appropriately formatted, significant data which they then react to. Bespoke interactive systems designed for live performance or installations, on the other hand, almost always include sensing and filtering as necessary part of their functionality (Stelkens 2003, Rothwell 2005[138], Lozano-Hemmer 2005, Weiss 2009).

5.1.2 Integrating a Generic System

This 'technologised' model of live performance provides both a pattern and a rationale for a generic interactive system intended to integrate with live performance and which implements the features specified in the statement of requirements established in chapter 4. Flexible, universal connectivity with contextually aware cuing or triggering was the most important common functional feature identified; combining this with a sensing and significance-filtering 'service' suggests both an architecture and computational process which can be integrated successfully and sensitively with the general model of performance. Importantly, such a system is overlaid on the general model, it does not break or otherwise disturb the existing processes of performance. As the statement of requirements states, performers must be able to create and produce performances using their own performance practice; working from a

[138] Nick Rothwell's*Tryptychos* (2005) articulates these stages explicitly; it is a dynamic sound installation with three graphic displays, the left panel shows the video input, the middle panel the filtered and thresholded signal and the right panel shows the derived graphic score which drives the sonic engine.

generic model of performance should allow a technological intervention to be as transparent as far as is possible[139] since it does not disrupt the underlying generative structures which underpin live performance.

In operation, information about each elements' activity will be collected directly from them and, similarly, relevant information about the total activity is sent directly to each element. In addition to the activity of individual performance elements, the information-set that elements' decisions can be based upon will include time-based formal structures (sections) and spatial constructs (zones). These latter objects correspond to the overall state of performance that human performers may take into account when forming their activity. It is a requirement that any generic system be able to sense a wide range of activity types and also that information is sent in a timely way to each element in an appropriate format: perhaps images or a sound cue to a human performer, but probably MIDI to a lighting board or OSC messages to a Max object. Each performance element is thus likely to need an individual interface with the overall system, with a central device maintaining a canonical version of the current summed situation in terms of both elements' activity and overall state. Communication with the central device should use standard network protocols and a simple message format so that the provision of new sensing or messaging functionality is possible. The central device will provide a 'significance filtering' service, using the information it holds about what is going on in a performance to determine if

[139] The ideal of transparency as presented by Boulter and Grusan (2000) remains elusive, particularly if, as in the proposed system, users' attention is focussed, particularly during the making phases, on aspects of the performance process (cuing, significance etc.) which are often instinctual.

something significant has occurred via a set of rules. Then, when specified significant activity occurs that meets the triggering conditions of one or more rules, cues are sent to one or more of the elements involved in the performance. It should be stressed that, in line with the non-specific starting point to the construction of the generic model of performance, decisions about the sort of activity to be monitored, what events should be considered as significant (i.e. which activity constitutes an event) and the nature of the rules determining when any response should be produced, what form that response should take and which performance elements it should be communicated to, are left entirely to the artists.

5.2 Computational Model

The proposed system architecture is based on a slight modification of the real-time controller model:

Figure 27 - Real-time controller architecture

In this case the sensor and actuator components are combined into discrete autonomous software objects called *clients* (note that the use of the term client is not intended to imply any relation to the client-server model) – which may implement the functionality of one or both components and which may both send and receive messages from the controller: -

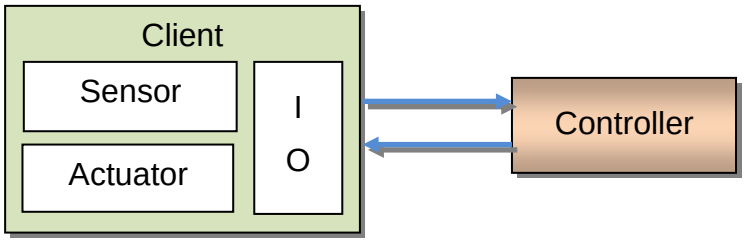


Figure 28 – Client-controller architecture

The architecture is then further elaborated by providing a persistent data store for the controller and a variable number of instances of the clients. The clients are connected to the Central Controller by a network capable of supporting the TCP/IP protocol. The proposed computational model is based on a 'message passing' model rather than a dataflow model (François & Chew 2006).

Each client instance is responsible for communicating with a participating *element* in a performance. An element may be a performer (a musician, a dancer, an actor…), a device (a lighting board, digital projection system, MIDI synthesiser…) or a dedicated software application (a Max program, avatar generator system, LiSa…). Each client instance can have different sensing and actuating functionalities to suit the needs (i.e. the required input and output information formats) of the element it is supporting.

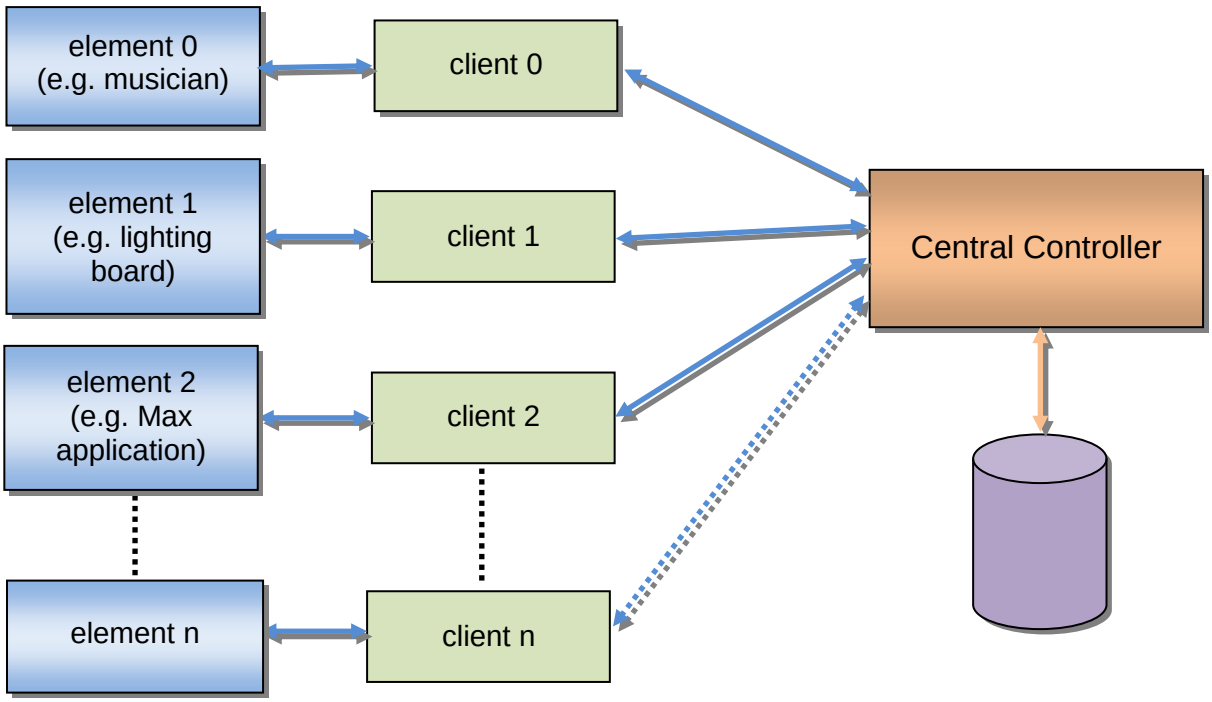


Figure 29 – System architecture

Each client functions as a media-translation layer between a performance element and the central controller. They convert sensed activity or input data (quantising if required) to a standardised message format and send this to the central controller. This quantising and vectorising functionality is necessary to

maintain scalability, given that standard networking infrastructure will be used rather than specialist protocols (Fléty 2005).

They also react to messages from the controller and transform these into an information or media-type appropriate for the performer or device they are interfacing with. For example:

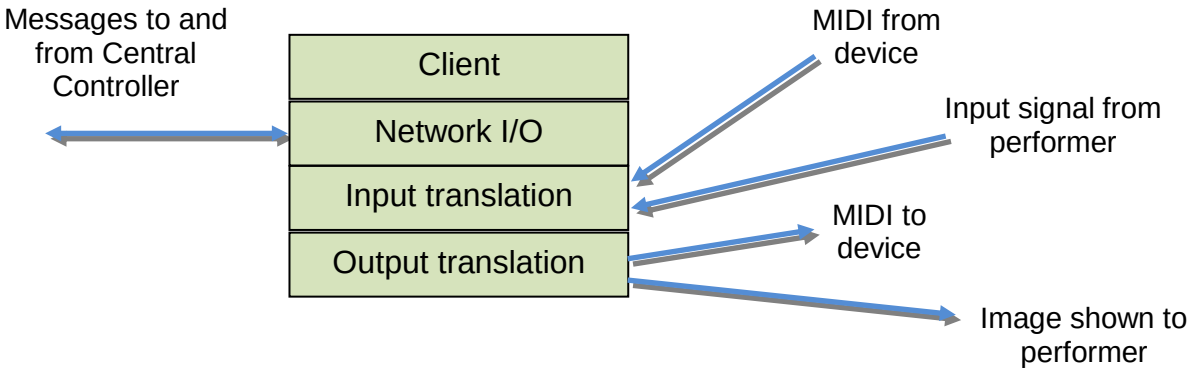


Figure 30 – Example client translations

The overall system architecture of a central controller connected to a variable (possibly large) number of clients, each of which has a primary data processing role, follows established models (Ramakrishnanet al 2004) that seek to balance the processing requirements for a scalable system against the bandwidth constraints of standard network infrastructures:

> "Another source of concern is that there is a central sever. To mitigate the probability of a performance bottleneck, Auracle's architecture is designed to minimize the work done by the server. The server is merely a conduit for data and does no processing itself. Work may be duplicated by the clients, but we preferred that to adding load on the server. Our benchmarking shows that we can support 100 simultaneous users, each sending one gesture per second." (Ramakrishnan et al 2004)

5.2.1 Central Controller

The unit of computation for the Central Controller is a self-contained object. As an abstract model located in a particular instant, the central controller may be considered as an object that receives and sends asynchronous messages. The connection between input and output is a complex function of the object's state variables and a set of *rules* for determining when an output should be produced. This rule-set is itself dynamic and part of the state of the object.

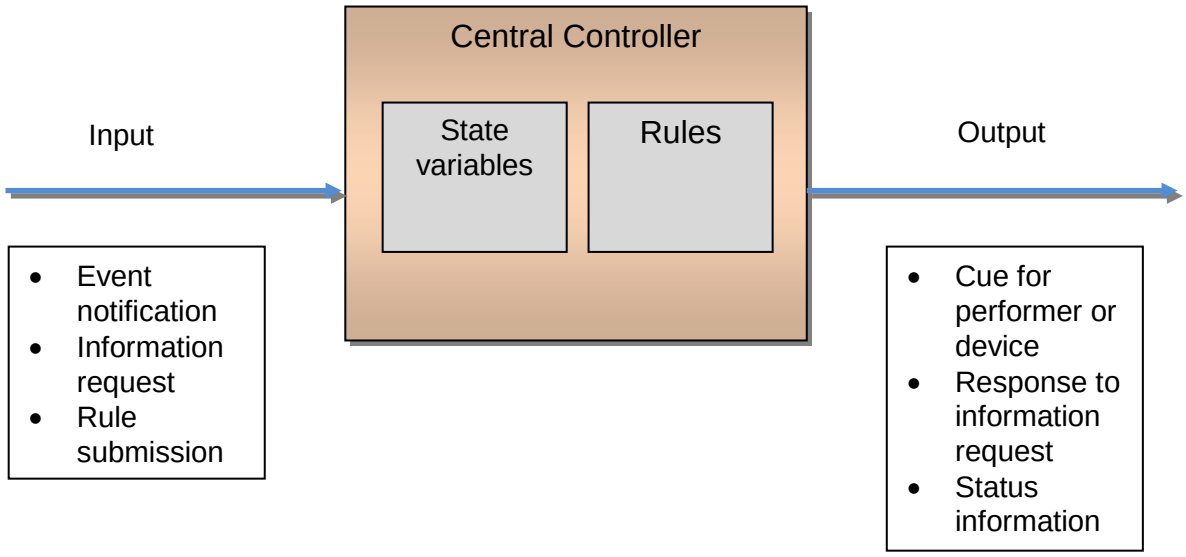


Figure 31 – Central Controller: messages and architecture

The *state* is a model of the disposition and activity comprising a performance at a specific point in time – as sensed and communicated to the controller by the clients. By state, I mean the mapping between state variables and the values:

$$\sigma : \text{var} \rightarrow \text{val}$$

Where the state variables may be considered as the union of many individual variables:

$$\text{var} = \bigcup_i \text{var}_i$$

The behaviour of the system may thus be considered as a series of states (which may or may not be different from each other) which defines a computation:

$$\text{å} = s_0 s_1 s_2 \cdots s_n \cdots$$

5.2.2 Responding to Events

The transition between states is driven by *activity-events.* Activity is sensed by clients through monitoring what is happening as the result of performance activity by the performer or device it is interfacing with (e.g. a MIDI note may have been played, a button might have been pressed or a floor-pad triggered) and which will result in a message being sent to the Central Controller. A client continually monitors this activity and generates activity-events, messages to the Central Controller that an activity-event which may or may not be significant[140] has occurred and specifying what the nature of the event was.

Activity-events thus arise as the result of *actions*. However, the relation between actions and events is complex, due to the differing characteristics of performance gestures. An action can be singular (e.g. a floor-pad is triggered),

[140] Significance is used here in the sense identified above, that an activity-event may have implications for the future unfolding of a performance; i.e. it may be an *event* in the terms of the generic performance model.

compound, composed of an action started and then finished (e.g. a MIDI note is played, an action-event comprising a MIDI noteOn action followed by a MIDI noteOff action) or a sequence of actions (e.g. a phrase of MIDI notes is played). Thus some sensed actions will lead to an event being generated while others will not.In formal terms, the permitted operations on actions, which could lead to an activity-event, are:

$A : Actions\ a,b \hat{I}\ A$	
$a \subset b,\ a<b$	Concatenation where the result is a totally ordered set: an event requires actions, having a specific relation to each other, to occur sequentially in a particular order (e.g. a MIDI note which is comprised of a note-on event followed by a note-off event).
$a \grave{E} b$	Simultaneity: an event requires actions to occur at the same time (e.g. a MIDI chord).
$a \mid b$	Alternative (or): an event is generated for different actions (e.g. sound intensity event triggered by a clap or a shout).
$a;b$	Sequential: a event requires a series of actions (e.g. a musical or spoken phrase).
$a \hat{I}\ Æ$	Empty or null: an event is generated by inactivity (e.g. if no one is moving).

Figure 32 – Permitted relationships between activity and activity-event generation

In the proposed computational model, the capacity for using this range of relationships between actions and events is preserved and made part of the flexibility of the system overall. How and when an event is generated is a function of the sensing component of an individual client and its settings. It is thus localised and open to contextualisation for particular applications and situations. As well as sensing and quantising activity, a client determines the logical relationship between sensed activity and events. So, following the event

descriptions given above, MIDI could be sensed at the lowest level of individual events (noteOn, noteOff) or at a higher level of individual notes (i.e. a noteOn+ noteOff pair) or as a phrase (a series of notes).

The Controller deals with activity only via the messages activity-events cause to be sent. No high-bandwidth data is involved and all communication is assumed to be asynchronous thus avoiding the problems which are involved in establishing and maintaining a synchronised, common timeframe across a distributed, networked system involving a highly variable mix of devices (Fober, Orlarey&Letz 2002, Lee &Borchers 2005). In the proposed computational model, an activity-event results in a message to the Central Controller which carries details of the event's origin and nature and may include additional qualities or attributes. These qualities provide further information about the event which can be mapped to the Controller's state. Figure 33 shows two examples of sensed activity-events showing how they might be communicated to the Controller.

Type	Attributes	Value(s)	Comments
A mouse click	type value	userEvent/mouse short|long|double	
A MIDI note	type note duration	midi/play integer integer	 0 ->126 (milliseconds)

Figure 33 – Example activity events showing attributes and values

The Controller is notified of an activity-event by a client *after* the action or actions that produce the event have finished and after the client has determined the event's nature and attributes and performed logical operations relating activity to action-events. The Controller thus receives a compressed (vectorised) summary of the triggering activity (rather than the raw sensed data) communicated within a standard message format. This has obvious advantages for the network bandwidth required, avoiding the technical challenges (Dannenberg 2004) and other problems associated with real-time data streams:

> "Focusing on interactive systems and their underlying models of computation separates on-line from off-line approaches. In the on-line group, synthesis-oriented efforts adopt models from Digital Signal Processing (DSP), and are often concerned with scheduling (because of the hard real-time requirement)." (François & Chew 2006)

as well as enabling a wide variety of different sensed activity to be incorporated into a uniform processing framework in the controller.

Each client is represented in the controller by a subset of the state variables. When an activity-event notification is received from a client, the state variables associated with that particular client are updated and hence the state space changes. Returning to the computational model, events trigger state transitions:

$$\sum : \sigma_0 \xrightarrow{e_0} \sigma_1 \xrightarrow{e_1} \sigma_2$$

The order of activity-events within a single state-scrutiny cycle may or may not be significant; the model assumes that an activity-event's creation time is the

same as the time that the message describing it is received by the central controller.

Each sensed activity event is mapped to a particular subset of the state variables that model the state of the device or performer (the performance element) that a client is interfacing with.

If:

$$E = \bigcup_{i,j} E_i^j$$

where E is all possible events and j identifies a particular type of event. Then for any E_i^j, there exists a subset of var, k, such that:

$$E_i^j \mapsto \mathrm{var}_k$$

Thus the ordering of events from *different* clients, mapped to different subsets of the state variables is *not* significant (although intermediate states may, or may not, differ) within a single state-scrutiny cycle:

$$\sigma_0 \xrightarrow{e_0} \sigma_1 \xrightarrow{e_1} \sigma_2$$

$$\sigma_0 \xrightarrow{e_1} \sigma_{1'} \xrightarrow{e_0} \sigma_2$$

However, the ordering of events from any individual client is significant since each different mapping of a given subset of the state variables changes the entire state of the controller:

$$\sigma_0^a \xrightarrow{e_0^a} \sigma_1^a$$

$$\sigma_0^a \xrightarrow{e_1^a} \sigma_2^a$$

We can summarise the above by stating that *activity* (subject to the permitted logical operations outlined above) define *activity-events* and hence *states*.

5.2.3 Generating Output

The input and output of the Controller are considered to be asynchronous. Messages from clients are sent to the controller, and the controller's state variables changed, as and when the sensor component of that client determines that significant information is available (i.e. that an activity-event has occurred). However, the examination of the Controller's state (against the currently active rule-set) to see if an output should be produced is triggered by a periodic, internal clock pulse. There is thus the possibility, particularly at low clock frequencies, that states might be missed by the Controller. As discussed above, the clients themselves also have a quantising function, although their sampling frequencies are typically much higher than that of the Central Controller's state scrutiny.

The relationship between input and output, where the input is confined to event notifications and the output to information for performance elements is:

> *An output will be produced if the state conditions of one or more of the currently active rules is met.*

A *rule* may be considered as a set of one or more conditional binary expressions concerning one or more state variables together with a specification of the output that should be produced if the conditions are met:

If (*condition*) then do (*output*)

In the proposed computational model, this form may be extended through the addition of an AND clause:

If (*condition*$_1$) AND (*condition*$_2$) then do (*output*)

Additionally, both conditions and output may be comprised of multiple items with implied OR operators between conditions:

If (*condition*$_1$[OR *condition*$_2$…]) AND (*condition*$_3$ [OR *condition*$_4$…])
then do (*output*$_1$ *output*$_2$…)

The output behaviour of the model may be summarised as follows: if a specified subset of the state space matches a stored logical expression, then a specified (stored) set of instructions are executed. The specification of these conditions and outputs uses a Domain-Specific Language (DSL) described in chapter 6.

5.3 Chapter Summary

This chapter has developed a novel approach to modelling the structure of live performance which is not based on a taxonomic approach, but which rather leaves the decisions about what the constituent elements of a particular performance might be to those creating that performance. It presents an idealised model of a performer as an agent who performs actions, the nature of which are determined at their point of production by the performer in response to perceptions about the state of the performance. A live performance is thus seen as the sum of activity produced by one or more performers. This general model is localised for performance involving interactive multimedia by extending the concept of a performer to include any device which can respond in a variety of ways to a range of inputs.

Given this generic model, the architecture and features of a computational model which will integrate with live performance without disrupting its structure are described based on a modification of the sensor-controller-actuator architecture. A formal computational model for the system components is then proposed which links the overall system state, the identification of significant activity and the generation of outputs. This computational model will be used in the following chapter to guide the implementation of the prototype system.

6Realisation

The previous chapter developed the generic model of live performance and the formal computational model which underpin the realisation of a generic system for supporting work with interactive multimedia in live performance. However, the software and hardware do not constitute a complete working (or workable) system; the statement of requirements clearly sets out the need for user control in both preparation and performance which is couched in language and structures drawn from performance practice rather than computer science. This chapter thus begins by describing the construction of a control language which specifically encapsulates the structures and elements of the general model of digital performance proposed in section 5.1.2; a Domain-Specific Language (DSL) for the control of live performance. It then considers the engineering decisions and discusses in detail the implementation of the messaging format and software components of the LIMPT prototype system, the Central Controller and reference implementation of the client. Some material in this chapter is drawn from the paper*Words of Power*, published in the International Journal of Humanities and Arts Computing (Willcock 2010).

6.1 Language Development

The design of Domain Specific Languages (DSLs), particularly when the proposed target domain is an area which is wholly or party non-technical, is necessarily problematic since there is on the one hand a requirement to encapsulate both an epistemology and the logical relations between the 'knowable' elements so identified, while at the same time this conceptual and taxonomic system has to be couched in a system of naming which is consonant with the experience and desires of the intended users.The language has to be a system for describing the world (albeit a 'world' that is usually an extremely small subset of Wittgenstein's "all that is the case"[141]) in ways that are easy for those who will use it to understand and engage with and which do not require them to become experts in a new field. This requirement, for the use of a vocabulary and grammar which are recognisably part of, or closely allied to, that of normal performance practice, is the central guiding principle used in designing the language[142] used to control the LIMPT system.

Underlying and preceding a language design should be a conceptual model of the target domain incorporating its capabilities for causality and manifestation.These models may be implicit or explicit; they may be expressed in a variety of ways ranging from a simple list of aspects of phenomena considered within the compass of the putative language right through to

[141] As Bertrand Russell points out in his introduction (Wittgenstein 1974:xvii), Wittgenstein suggests in the *Tractatus* that *all* languages can only address a subset of 'the world'.
[142] The language is named *eMerge* in recognition of the importance of the *eMerge* project, led by Jane Turner and Daniel Biro in developing my ideas.

language designs based on a comprehensively described formal logical model of the domain they address. Existing models of performance (both explicit and implicit models underlying performance resources) were considered in detail in section 2.1.3 but for the most part these models are not translated into system control structures approaching a DSL; van Deursen, Klintand Visser, give a useful definition:-

> "A domain-specific language (DSL) is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain." (vanDeursen, Klintand Visser, 2000:27)

Almost all the performance systems considered in the discussion of models of performance in section 2.1.3 provide control (or allow the programmer to provide control) in the form ofGUI elements: sliders, buttons etc.or MIDI/OSC control signals.Field (2005) developed by Marc Downie, is able to be reconfigured during performance using the same language[143] that was used to build the system software components and their configuration[144]. The system described here follows the same approach in supporting preparation and real-time intervention using the same semantic structures.

A generic model designed to support the incorporation of digital technology within live performance was presented in section 5.1.1. Working from this and

[143] Python is the 'language of choice', but others can be used.

[144] One of the design principles behind the development of the system is,

> "Field takes seriously the idea that its user — you — are a programmer / artist doing serious work and that you should be able to reconfigure your tools to suit your domain and style as closely as possible." (Downie 2005)

This is clearly very exciting in lots of ways, although my research did suggest that a system that started with the idea that the user was necessarily an artist/programmer might not be welcomed by at least some practitioners in digital performance.

from the computational model (especially section 5.2.3), the base-line decision-making element needs to encapsulate a conditional connection between two performance elements where a specified action of one triggers an activity or alteration of behaviour in another. Thus, the simplest general form for the definition of an individual decision-making element(a *rule*) is:

```
if (event) then do (command)
```

Where *event* is a significant action and *command* is a combination of a trigger and a recipient. So, for example, if a particular specified event or events occur during a performance, the specified cue or trigger will be sent to a performer or device: -

if
nick says “boo”
then
say “go to section 3” to jane

Figure 34 – A simple rule

This is extended in line with the computational model to allow multiple events (both alternatives and combinations) and multiple commands:

```
if (event0 [| event1]…) [&(eventa…)] then do
                    (command0 [,command1…])
```

if	
	ian gives a mouse signal daniel plays “F# 5”
and	
	performance time is more than 3 minutes
then	
	show image “score2.jpg” to zoe say “start transition” to joel give a “D 3” noteOn midi signal to lights

Figure 35 – A more complex rule with multiple commands

A decision was taken that the logical structure of rules should be presented to users through the GUI of the performance preparation panel of the client which means that the language design only needs to facilitate users in specifying events and commands, the ‘if’, ‘and’ and ‘then’ qualifying conditions.

Events and commands are both further structured in two different ways:

```
event = element + state + value
```

or

```
event = element + <gives a> + value + <signal>
```

And

```
command = keyword + value + <to> + element
```

or

```
command = keyword + element + <to> + value
```

Where *element* is an entity involved in producing the overall state and current activity of the performance. An element could be either a performance-element (a specific performer or device) or one of the system'srepresentations of the overall state of the performance: structures of time, space or the current rule-set. *State* is either a conditional test (equality, non-equality, greater than or less than) or a verb describing the type of activity (plays, says).*Value* is a constant specifying either the value which incoming events are to be tested against for event descriptions or the quantitative specification of the cue or trigger for commands. In commands, the keyword and the data-type of value indicate exactly what is to be done to produce the desired trigger or cue[145].

Alternative forms for both events and commands were felt necessary sothat they both could follow the natural language patterns associated with verb and element-type.Even though alternative forms could be seen as potentially confusing and hence damaging to the system's usability, allowing users to use phrases structured in ways that they were already familiar with (and dealing with the increased complexity in software) appears to be preferable as my survey results suggested that a perception by artists that a different knowledge domain is required to use a tool may be highly off-putting. Figure 36 shows part of the Backus Naur Form (BNF) grammar of the final version of the eMerge language (expansions of argument and signal types are omitted for brevity).

[145] E.g. '*show "start section3" to ian'* would display the text string and '*show image "section3.jpg" to ian'* would display the named image.

```
S:= Event|Command
Event   :=ObjectArgCondition
Command         :=<SAY>GeneralArg<TO><IDENTIFIER>
                |<SHOW>GeneralArg<TO><IDENTIFIER>
                |<PLAY>GeneralArg<TO><IDENTIFIER>
                | <GIVE><A>SigDataSigType<SIGNAL><TO><IDENTIFIER>
                |<SET>ObjectArg<TO>GeneralArg
                |<START>ObjectArg
          |<STOP>ObjectArg
GeneralArg:=ObjectArg
                |ConstArg
Condition       :=StateVerbGeneralArg
                |<GIVES><A>SigDataSigType<SIGNAL>
StateVerb       :=<SAYS>
                |<TYPES>
                |<PLAYS>
                |<MORE_THAN>
                |<IS><MORE><THAN>
                |<IS><LESS><THAN>
                |<IS>
                |<LESS_THAN>
```

Figure 36 – Partial BNF grammar of the eMerge language

The configuration and control of such a system requires that users are able to rapidly and easily describe performance activity that should be considered ‘meaningful’ (events) and specify commands to be executed either immediately or when a rule is triggered, sending cues to participating performers or devices or making changes in the state of the internal representations (time, position, rule-set) of the central controller. In addition to the ideals of structural consistency and coherence (which are addressed through the formal grammar), there is also a requirement that the language used needs to reference the vocabulary and structures of artists’ previous performance practice in ways that

support and empower creativity rather than exhibiting and amplifying the differences in approach between technology-centred priorities and individual artistic preoccupations.Performers are used to thinking and talking about what they do in ways that a specialist language must accommodate and draw upon.

One of the earliest decisions was that performers would be referred to simply by name. Participants enter their name the first time they run the client software (each performer has their own instance of the client software running on a dedicated machine) and this is then used to identify that client in writing events and commands that refer to the performer or device interfacing with that client (see Figures 34 and 35). Thus a name without any qualifier is always a participant (ANYONE, NOONE and EVERYONE are also allowed), other objects that can be used in events and commands are identified by putting their type before their name[146](see Figure 37).

Object	**Example event**
performer	ivan says “stop”
section	section transition1 time is less than 2 minutes
zone	zone downstage is occupied
rule	rule SysName_RULE_2 is active

Figure 37 – Example objects and associated events

Similarly, even though logically, all event specifications involve a conditional test, the language treats the activity of elements in two different ways, mirroring

[146] Each project has a section object created automatically called ‘performance’ which allows reference to the performance itself. The ‘section’ identifier is not required for this.

natural language usage: performers either do things (e.g. Joe types “stop”) or they give signals (e.g. Anne gives a mouse signal) (see Figure 38).

Activity	Types Says Plays
Signals	Key Mouse Sound (level) MIDI

Figure 38 – Sample performance-element activity types

The syntax for specifying signals in events is deliberately flexible, allowing users to specify only the level of detail required; from an event triggered if a performer gives *any* sort of signal (‘Jane gives a signal’), a specification detailing the type of the event (‘Paul gives a mouse signal’) and a specification which details both the type and quality of event (‘Danni gives a double mouse signal’). System objects have properties that are incorporated in event specifications using the form of *element* + *state* + *value* with a state verb indicating the conditional test to be used (see examples in Figure 37). The syntax for all events is described in detail in the eMerge Language Reference (Appendix 3).

In commands, the *set* keyword is used with form*keyword* + *element* + <to> + *value*[147]. Alternative structures are also available for referring to sections in commands since “start section intro” is much closer to natural language and

[147] For example, ‘*set zone downstage to occupied*’, ‘*set rule mouse_rule3 to inactive*’, ‘*set performance to active’ – although’ start performance*’ would be preferred for the latter.

hence more familiar for the target users than the (grammatically more consistent, but clumsy) “set section intro to active”.

Another early realisation was that, in addition to the variability of grammar discussed above, logically identical operators changed their natural-language form in ways that needed to be accommodated if the vocabulary of the eMerge language was not to appear ‘difficult’ and outside the experience of target users.So, for example, the character sequences: ‘>’, ‘is more than’ and ‘is higher than’ are all considered as synonyms for the same logical relationship (<MORE_THAN> in Figure 36); a test deciding if the occurring event’s value (elapsed time, position, MIDI note) is more than that of the value specified in the rule. This mapping of several different character strings onto the same token is handled at parser level.

The language described here is structured around event-producing objects: performance elements (performers and sensing devices) and abstract structural constructs which permit formal units based on time or space/place to participate in the structuring of the overall performance.The events generated by these objects are tested for significance through the application of comparative operators (is, isn’t, is more than etc.).Logical connections between events are established through a limited set of Boolean operators and, when the triggering conditions are satisfied, one or more commands are executed.All of these structures are encoded in a Domain-Specific Language (DSL), a near-natural language that relies heavily on abstractions and vocabulary drawn from live

performance practice.Each artist or group of artists is able to configure the system so that its connectivity and decision-making structures are responsive to their own creative demands in a particular performance context.

6.2 Prototype System

The development of the LIMPT system was carried out in parallel with that of the eMerge language described above. Many decisions (particularly on event types and formats) had consequences for information, software and hardware designs which needed to be implemented at the same time. In addition to the language, there were three foci for development: the network messaging format, the Central Controller and the client. An early decision was to host the Central Controller on a public-facing web server and to hide the complexities of the LIMPT system by operating it as a web service. For most users, use of the system is a matter of downloading and installing the client on as many machines as they wish[148]. Communication between clients and the Central Controller is by XML sockets, persistent connections that are available in almost all programming systems and on most hardware platforms. The client realisation (see section 6.2.2) is considered a reference implementation of a number of the possible client functionalities rather than a system component that must be used. It is envisaged that other applications could be produced, possibly with more limited functionality (sensing, preparation, interfacing with specific devices or systems) or using different development platforms. Underlying all interfacing with the Central Controller is the specification of the messaging system.

[148] Source code for the Central Controller is publicly available (apart from the few commercial components) and users could host their own controller if they wished. However, setting up the server and database etc. is definitely a task that requires a reasonable level of technical facility.

6.2.1 XML Message Format

There are a limited number of communications required in an implementation which follows the computational model presented in section 5.2. These can be categorised as follows in terms of inputs and outputs to the Central Controller: -

Inputs to controller	Outputs from controller
Sensed activity data	Cue or trigger to element
Command for execution	Response to information
Rule for storage	Status report

Figure 39 – Input and output message types

It was decided that a custom XML format was the most appropriate for messages between system components as it is highly portable, can be processed in all modern development environments and, crucially for a system which is intended to be publicly available and web-facing, has a well-defined set of validation facilities and tools available. This approach follows that of the now-standard SOAP protocol (Simple Object Access Protocol (SOAP) 1.1 2000). Potential disadvantages include the lack of compression compared to other possible formats, such as Java Script Object Notation (JSON, no date)[149]; however, due mainly to the excellent support for XML in both Java and ActionScript and the quantisation performed by the client's sensing of activity, these were not judged to be serious problems.

[149] See Introduction to JSON, http://www.json.org/; JSON is supported in both AS and Java, but the linkages between these languages and the XML data-types are much more powerful.

There are two ways that an XML document's format can be specified: Document Type Definitions (DTD) and XML Schema Definitions (XSD). The latter approach was chosen because of XSD's support for data types, name spaces and their capacity for self-checking since they are XML documents themselves (Bex, Neven& Van den Bussche 2004). The advantage of the schema approach is that an open source Java framework, Xerces[150], is available which can load a given schema and hold it in memory as a data structure for rapid validation of XML documents.

A set of XML schemas was produced using the Oxygen XML editor[151] which together specify the format of XML messages sent to and by the central controller and the formats of their constituent elements. These were simplified as the development process proceeded; final versions omit much of the detail that was originally included and concentrate on a limited number of message formats which specify just the detail required to meet the currently implemented functionality.

[150] http://xerces.apache.org/xerces2-j/ The framework is also available for Perl and C++.
[151] http://www.oxygenxml.com/.

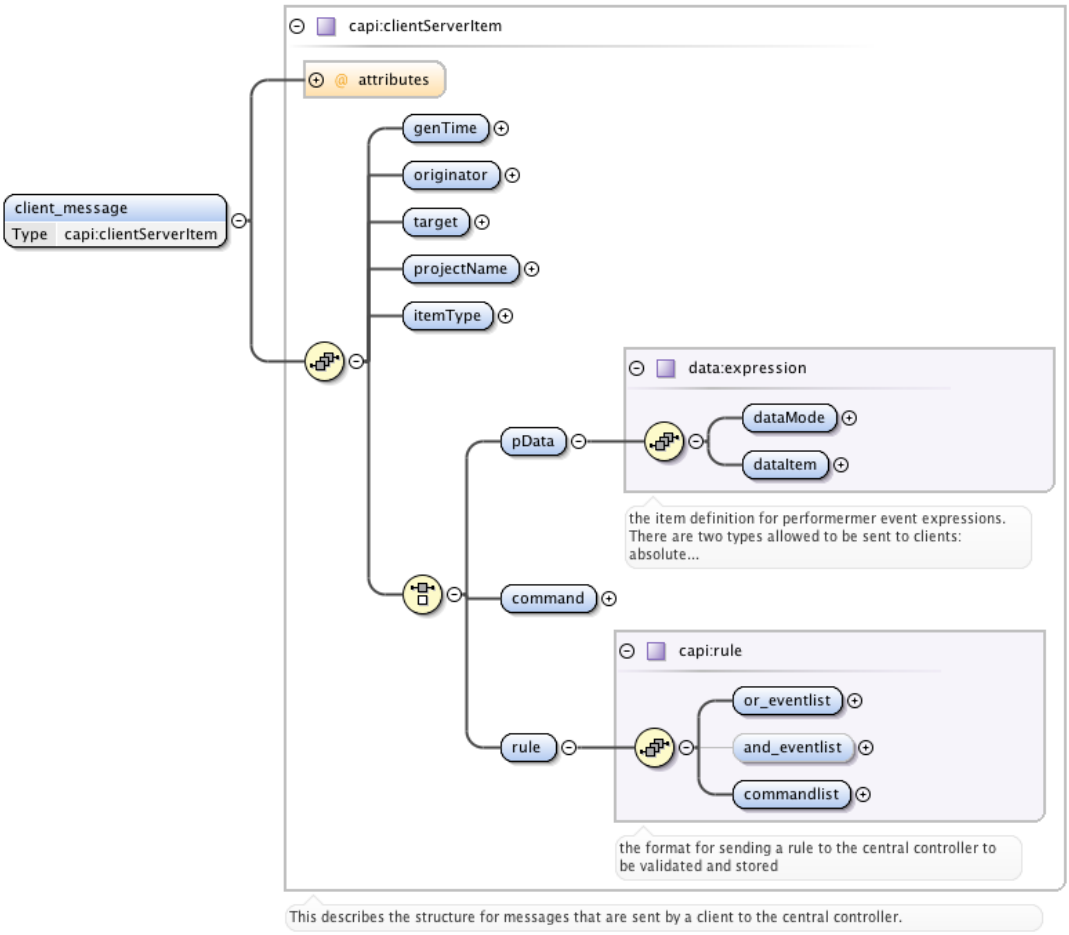


Figure 40 – Schema defining the structure of messages sent to the Central Controller

Messages sent to the Central Controller all have a root *client_message* element with a common header (the top five elements in Figure 40), an approach similar to the SOAP 'Envelope' (Simple Object Access Protocol (SOAP) 1.1 2000 – section 4) and then have one of the three alternative elements which carry information specific to each purpose:

- The *pData* element carries information about an activity-event that has been sensed by the client.
- The *command* element is a text string written in the eMerge language. Examples could include a direction to the central controller to return

information (a rule, a list of projects etc.) or a direction to send a trigger to a specific performer or device.

- The *rule* element contains two or three text lists which correspond to the event and command specifications for a rule (see Figures 34 and 35).

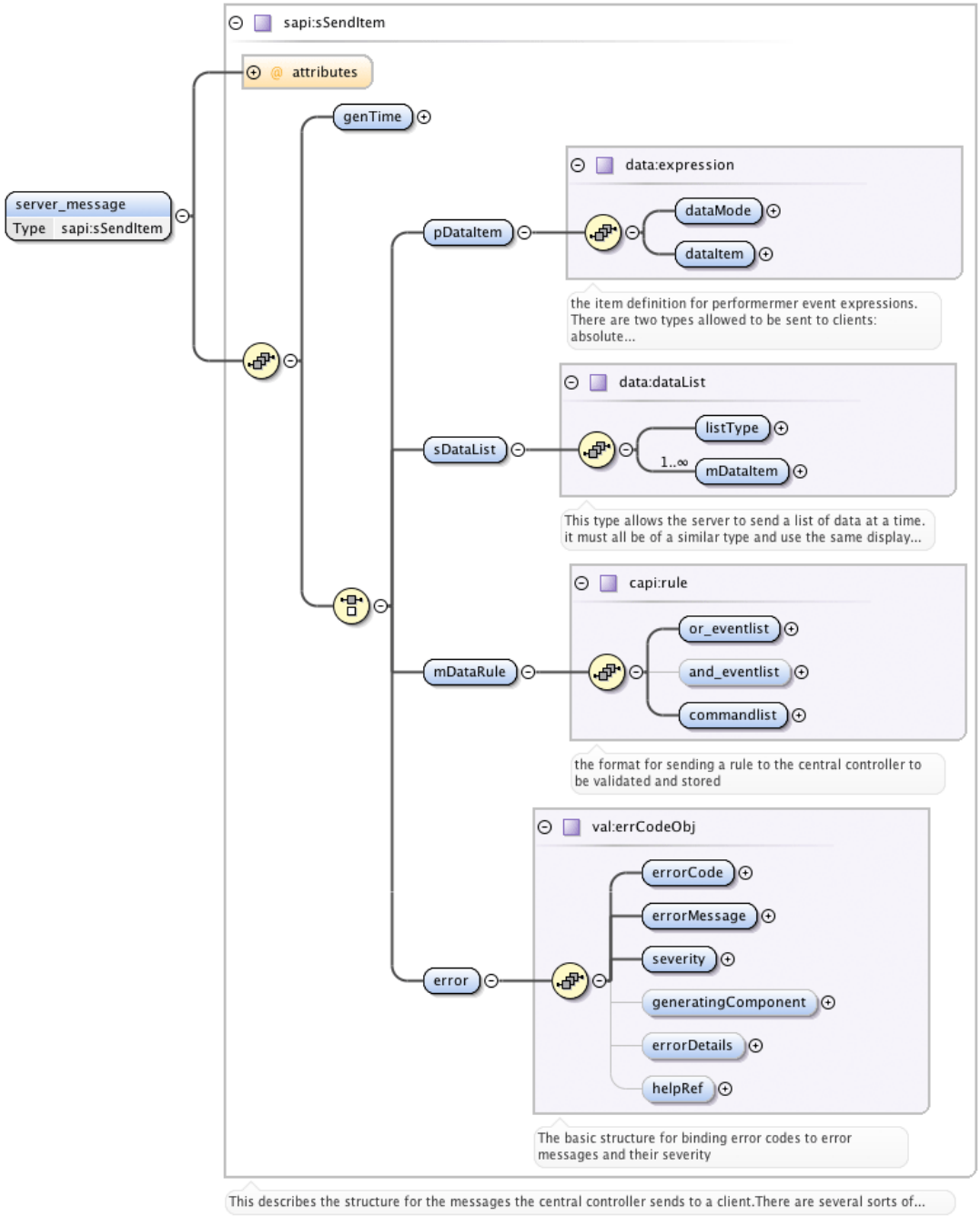


Figure 41 – Schema defining the structure of messages sent to clients

The format for messages sent to clients follows a similar model to that described above; there must be a root element called *server_message*, but because the Central Controller directs the message to a specific client instance, the header information is much simpler. Messages from the Central Controller then have one of four different elements:

- The *pDataItem* element specifies a cue or trigger for the performer or device a client is interfacing with. It has two parts: the *dataMode* which specifies both data type and how it should be presented (e.g. 'text/speak') and *mDataItem* which specifies the actual content as an absolute value ("hello") or an absolute file reference (http://www.willcock.org/assets/LIMPT_intro1.jpg).
- The *sDataList* element provides the client with a list of rule-names or performance-names. The *listType* element signals which information has been sent.
- *mDataRule* contains a rule definition for the client to store locally for editing and preparation.
- The *error* element carries information about the status of an operation executed by the Central Controller[152].

[152] 'Error' is used here in the usual computer science sense of a status report which has a graduated level of severity and which includes a nominal (OK) status in the range of states reported. In the current Central Controller implementation, unless a 'real' (i.e. non-OK) error occurs, only database storage operations generate a response for successful completion.

6.2.2 Central Controller

The Central Controller is the hub of the LIMPT system and has the following main areas of functionality:

- Managing connections with clients.
- Storing performance data, particularly rules.
- Validating rules and commands.
- Maintaining a representation of performance activity and acting upon any combinations of state that are significant.

The current implementation of the Central Controller started out using the code base from a system developed[153] for the *eMerge* project in 2003. At that point, the controller component used Java SDK1.4 (Lindsey, Tolliver, &Lindblad 2005:6) together with an outdated version of the Jess inference engine, several of whose API calls were deprecated in the current version. Its architecture and code structure reflected the extremely short development time available for the original project with numerous 'hacks' and short cuts taken to 'just get things working'.

The first decision was to retain the general approach already adopted for the earlier project: to retain Java as the development environment because of its development-speed, reliability, error handling and the numerous third party libraries which were available for specific required area of functionality but to

[153] The programming team for this project was led by the author who project managed the development process and designed the information architecture, language specification, XML schemas and the Flash/Director client. Nick Rothwell designed and programmed the controller system. Despite significant changes, the current application owes a great deal to his work.

refactor all code to current Java specifications. It was also decided to continue to use the Jess[154] Rules Engine (Thirumalainambi 2007) to provide the state-space and conditional triggering functionality required and to generate the parser for the eMerge language using the Java Compiler Compiler (JavaCC[155]).

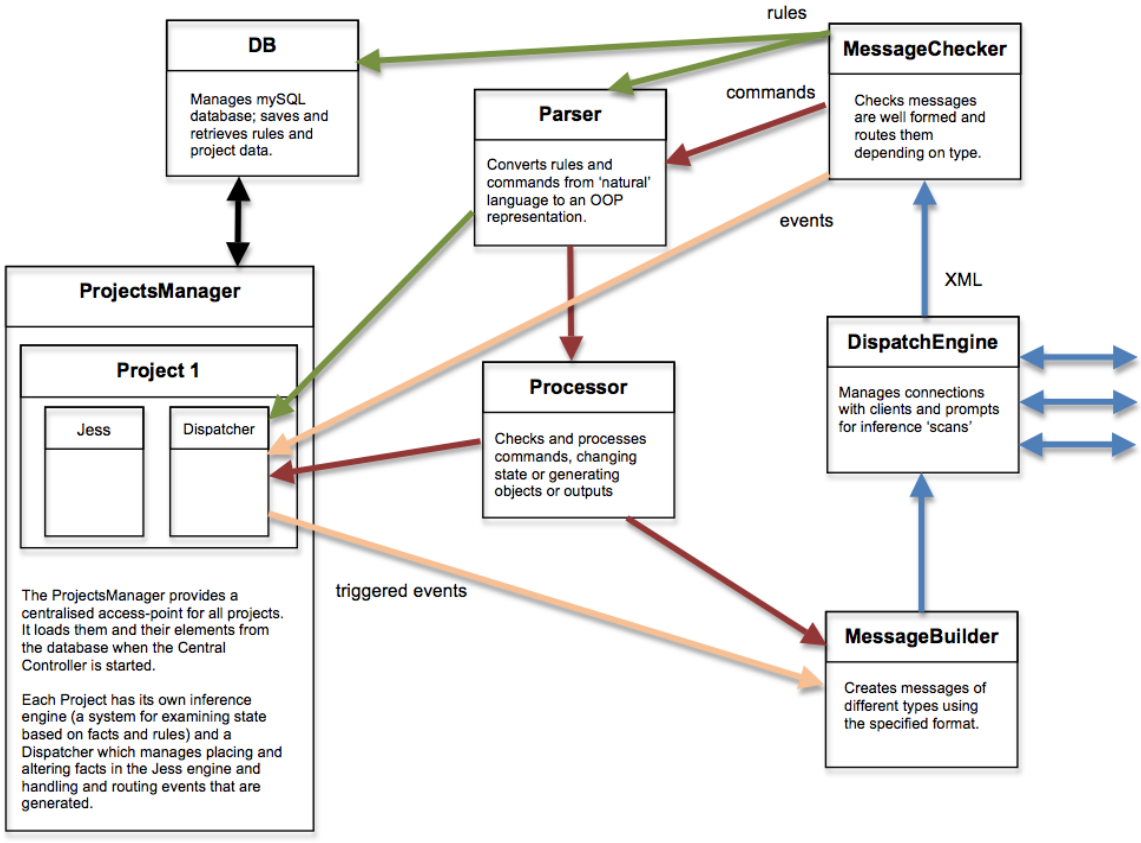


Figure 42 – Central Controller architecture showing signal flows

A simplified overview of the architecture of the Central Controller is shown in Figure 42. The system is constructed around a series of variable-spaces (only one is shown in Figure 42 for clarity), one for each project (a project can involve

[154] http://www.jessrules.com/jess/index.shtml Jess is a commercial package but is available free for academic use.
[155] http://javacc.java.net/.

repeated performances, so this terminology seemed more intuitive). In use, each of these variable-spaces (called 'working memory' in Jess) are implemented using Jess inference engine instances and are populated by rules and a number of agents (which must extend the Java Beans class), each of which represents one of the objects whose activity might influence the unfolding of a particular performance. This obviously includes any connected clients (represented as instances of Performer), but also includes software agents representing rule-states (Rule_State instances), spatial elements (Zone instances) and time-based formal units (FormalUnit instances). Instances are created dynamically whenever the controller first receives a reference to an object in a rule specification or command. They remain in memory (unless the system is restarted); this does not increase the processing load since the Rete algorithm caches computation results where possible; where objects are not receiving events updating their state, they cannot affect the outcome of the rule-engine[156]. Currently, the exception to this persistence is rules, which are removed from the working memory when they are no longer part of the decision-making structure for a performance (i.e. because they have been deleted).

Each agent has a series of properties which correspond exactly to the properties that can be specified about its real-world counterpart in event descriptions.These objects' properties are collectively known as 'facts' in the

[156] It is very important to remember that evaluation of the working memory to see if an output is required occurs when the overall state *changes*; where there is no change in some or all of the state-space, processing is correspondingly reduced.

inference engine; together they dynamically represent the current state of a performance. For example, a *section* in the eMerge language has the following specification:

section

e.g. section interlude1 is active

Sections are ways that time-based divisions of the performance can be dynamically represented by the system. They have two qualities; whether they are active or inactive and an elapsed time.

Sections are identified by a name:

section intro

Sections are created when a new section name is encountered by the system, but they are not started (set to active and their clock reset to 0 and started) unless the start command is used:

start section intro

Sections are stopped by the stop command:

stop section intro

When a section is started, its activity can be used to fire events in rules:

(if) *section intro is active*

Its elapsed time, counted in seconds, can also be used in rules (until a section is started, it remains at 0):

(if) *section intro time > 4 minutes*
(if) *section main time is less than 20* [NB seconds is assumed]

Figure 43 – Section specification (page 12 of the Language Guide, Appendix 3)

The UML diagram[157] for the FormalUnit class is shown in Figure 44.

[157] The diagram has been slightly simplified; it does not show access control or exceptions. The source of the class can be seen in Appendix 4 in the com.cassiel.emerge.inference package.

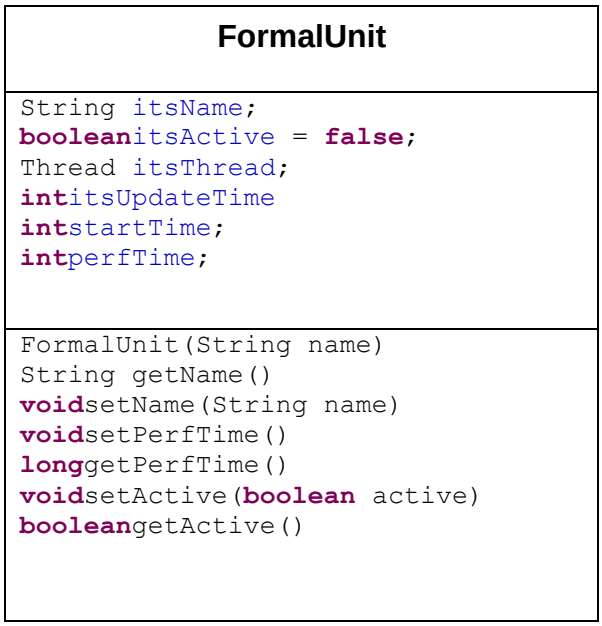


Figure 44 – Simplified UML diagram of the FormalUnit class

The class follows Java-Bean practice in having getter and setter functions for all the properties which are involved in decision-making. The thread is necessary so that, once activated, the object can change the value of its *perfTime* variable (and hence its state) autonomously (or until it is made inactive).

When an agent's property in a performance is changed because an event notification has been received, the rule engine for that project scans its working memory, matching rules and agent-states[158] to see if any output needs to be produced. There is also a tick pulse which prompts rescans of all the currently active projects' state-spaces at a set interval[159].

[158] The Jess rule engine was chosen as it has excellent integration with Java and is regarded as an efficient implementation (Thirumalainambi 2007)of the Rete algorithm developed by Charles Forgy (Forgy 1982).
[159] 100 ms is the current scan interval.

The data flow for different message types are also shown in Figure 42. Clients connect to the Central Controller using network sockets and communicate using null terminated XML character strings. When a client first connects, an instance of a Performer object is created and added to a project's Jess rule engine (a project name is required for messages from clients – see Figure 40). When a message arrives, it is sent to the message checker which identifies what sort of content it carries: performance data, a command or a rule. Events are sent straight to the appropriate project and the state of the object (and hence the overall state-space) which corresponds to the client is altered according to the information about the activity event that the client has sensed. At that point, an output may or may not be fired, depending upon the relationship of the current state-space and rule-set. If a rule (or several) are triggered (i.e. its conditions met), the specified action(s) in its command clause(s) are sent individually to the MessageBuilder where they are wrapped in the required message format and sent by the DispatchEngine to their intended recipient client(s).

Commands and rules are both sent to the parser (rules are also stored in the database in their raw text form). The parser splits up text strings into an object-orientated representation of events and commands, where each lexical token is converted into an instance of a class in the AbSyn package[160]. The parser itself is created using the JavaCC Java parser generator package which takes a grammar file produced by the programmer and generates the Java source code

[160]A contraction of 'abstract syntax'. The full package name is com.cassiel.emerge.AbSyn.

for a parser which follows the specified token definitions and grammar[161]. This grammar file is where the words making up the language (see page 22 of the eMerge Language Guide – Appendix 3) are defined and the allowed constructions and combinations of these words specified together with the java classes which will represent them. Figure 45 shows this process for a command, this could have been sent as a stand-alone directive from someone wishing to influence the course of a performance or it might be part of a rule definition. If it is the former, a command for execution right away, the command class processes the required activity (shown as the processor functional block in Figure 42) and, where required, causes the appropriate output message to be generated. If the command is part of a rule, the class generates a functional statement in the Jess language and sends it to be dispatched into the working memory[162] along with its qualifying conditions.

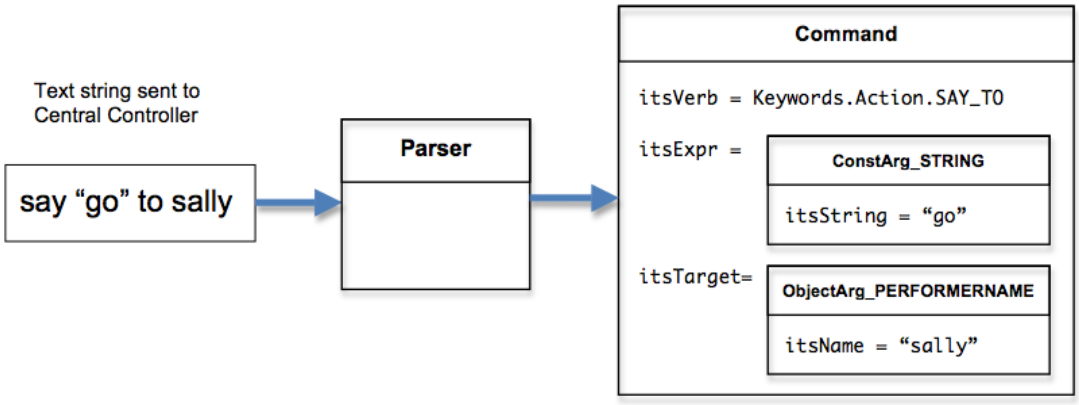


Figure 45 – Parsing process for a command

[161] See *JavaCC Grammar Files* for a detailed description of the JavaCC configuration file format. The eMerge.jj file corresponding to the BNC grammar shown in Figure 36 can be found in the com.cassiel.emerge.parser.gen package (Appendix 4).
[162] Because of this need to handle rules arising in two separate contexts, the AbSyn.Command class is very complex and a priority for refactoring in future system development.

Events are parsed in a similar way to commands except that they always then generate conditional statements in Jess language which are placed in the project's working memory.

Persistent storage for the Central Controller is implemented using the mySQL opensource database engine accessed from Java using the appropriate jdbc package[163]. The database schema is shown in Figure 46, slightly simplified for clarity:

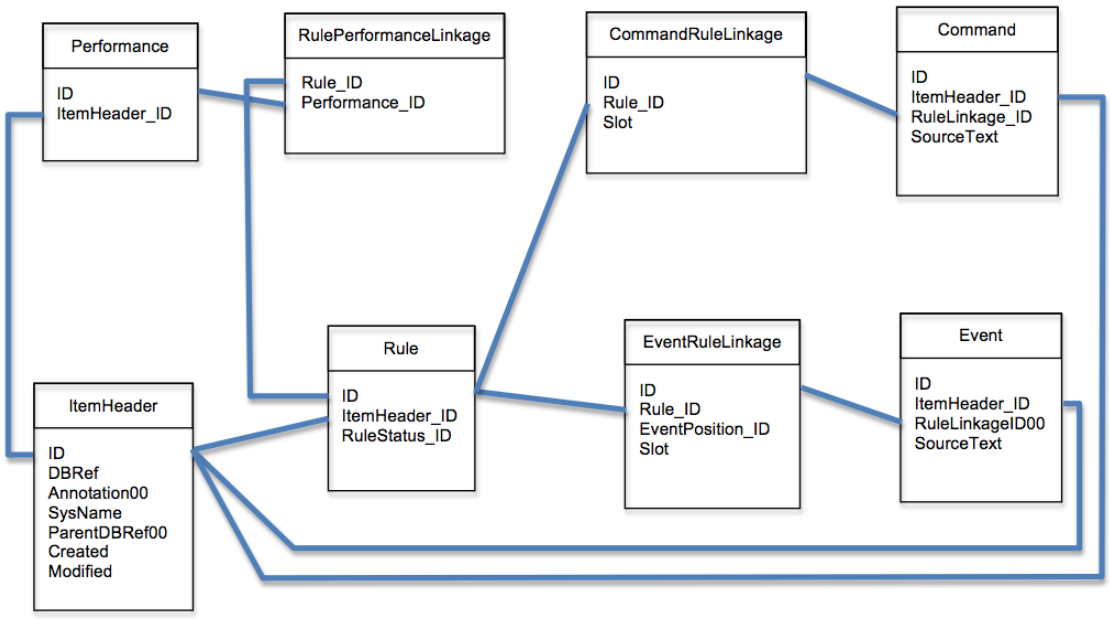


Figure 46 – LIMPT database schema (slightly simplified)

The linkage tables are necessary for commands and events since although there is a many-to-one relationship to a rule (i.e. there can be many commands in a single rule, but a single command can only relate to a single rule), there is

[163]Available from http://dev.mysql.com/downloads/connector/j/.

an ordering of commands or events within a rule and holding this in the link table makes for efficient retrieval of events or commands from the database.

When the Central Controller is started, the ProjectsManager loads each project's rules from the database, validates them and then causes them to generate their agents in their inference engine's working memory (the process is the same as when a new or modified rule is sent to the Central Controller by a client). At this point, it enters the loop listening for socket connections, ready to respond to clients. In practice, the Central Controller is started using SSH connection using the 'nohup' option so that it continues running when the terminal session is closed. For troubleshooting or testing it can be started and stopped from the terminal window, when it reports activity by logging to the standard output (this output is directed into a text file when run using 'nohup'). The stability of the Central Controller appears to be acceptable; it has been run on Bartleby for almost a full year without restarting. When not in active operation (i.e. no clients connected), the system load is minimal – the 'top' shell command reported it using 0.3% of cpu load every 4 or 5 seconds[164]when other usage was low enough for it to appear in the list of processes.

[164] The *top* command reports on currently running processes ranked in order of their processor-use. The Central Controller only appeared in the list every few seconds (the list is updated every second). Its use of virtual memory is high (although use of physical memory is low), this would need to be an area of focus for further development.

6.2.3 Client

Clients are the interfaces with performance elements and can also provide the interface for artists to prepare for performances by specifying the conditional connections and cues that will govern its unfolding. It is not required that all clients offer the full range of functionality, indeed it is likely that a performance might involve the use of a number of different applications, each of which offered a subset of the full client functionality (for example, performers would not need to be able to edit the rules which make up a performance's specification). It is also envisaged that the existing sensing or triggering capabilities of a client will probably require extending to meet the needs of specific projects; this extensibility also needed to be facilitated in deciding upon the approach to client implementation. This situation will be familiar to many programmers:

> "User interfaces are especially prone to change requests. When you extend the functionality of an application, you must modify menus to access these new functions… Different users place conflicting requirements on the user interface… Building a system with the required flexibility is expensive and error-prone if the user interface is tightly interwoven with the functional core. This can result in the need to develop and maintain several substantially different software systems, one for each user interface implementation." (Buschmann et al 1996:126)

The overall approach taken to the architecture of the client was informed by Software Patterns[165] and, in particular, the Model-View-Controller (MVC) pattern, where data structures and control logic are separated from the rendering of the user interface and from each other.

[165] Software Patterns are structural approaches to solving common programming problems that have been generalised from experience of solving similar specific problems. Gamma et al (1995) is considered the canonical text.

> “MVC decouples views and models by establishing a subscribe/notify protocol between them. A view must ensure that its appearance reflects the state of the model. Whenever the model’s data changes, the model notifies views that depend on it. In response, each view gets an opportunity to update itself. This approach lets you attach multiple views to a model to provide different presentations. You can also create new views for a model without rewriting it.” (Gamma et al 1995:4)

Rather than producing a number of applications, it was decided to produce a reference implementation which offered the full set of functionality, both performance-element interface with a wide range of connection options and a preparation view where rules could be created and edited. However, the underlying data structures and the logic for their manipulation were not tied to the interface. The intention was that future production of new client implementations (possibly with a more limited set of functionality) could incorporate the code libraries providing the data and networking/control services making development times faster. In practical terms, this was achieved by producing a series of structured packages, each of which provided data representations and control, but which, in most cases, are capable of running without any user input, freeing the developer from having to provide user interfaces for those features for which control is not needed.

The client was implemented as a hybrid Flash/Director application. Director was included because it runs on Macs and PCs, provides access to local system resources and has an extensive (and extensible) range of plug-ins while supporting a wide range of asset-types, including Flash which has full

functionality[166] when run inside Director. However, Director is not an effective object-orientated coding platform, particularly for a large project. It is also less and less widely used so that while it could provide an effective framework linking Flash to local system resources and providing hardware extensibility, the majority of coding, and capacity for extensibility, needed to be Flash-based. Flash has the advantages that code management is reasonable (class files and packages), it has a wide range of data structures and types and the development environment supports reasonably fastimplementation of high quality user interfaces[167]. Thus the data and control frameworks of the project were realised in ActionScript[168] as were all user interface elements.

The client data and logic code is split between nine packages shown in Figure 47, brief descriptions of each class are given in the outline documentation (Appendix 5). For the most part, code architecture follows a design-pattern led approach: a few centralised points of access implemented using the Singleton pattern (Gamma et al 1995:127); dynamic configuration of inter-object communication using the Observer pattern; and the use of server-proxy intermediate software objects to represent the Central Controller for objects that need to access its (remote) functionality (Völter, Kircher&Zdun 2005).

[166] There is a limitation in that only ActionScript 2 is allowed. Even though both Director and Flash have been updated since the project started, this limitation still applies.
[167] Flash was unable to access the local file system at the point when the project was started. It is still (even when run as an AIR application) highly problematic to interface with serial ports and other hardware extensions and interfaces such as MIDI and Arduino – although these are promised for future releases. When these functionalities are added, the use of Director will no longer be necessary.
[168] ActionScript is the scripting system used in Flash. It exists in several versions and has become increasingly powerful to the extent that it is increasingly used outside of Flash. It is syntactically very close to Java (strongly typed, class-based instantiation etc.).

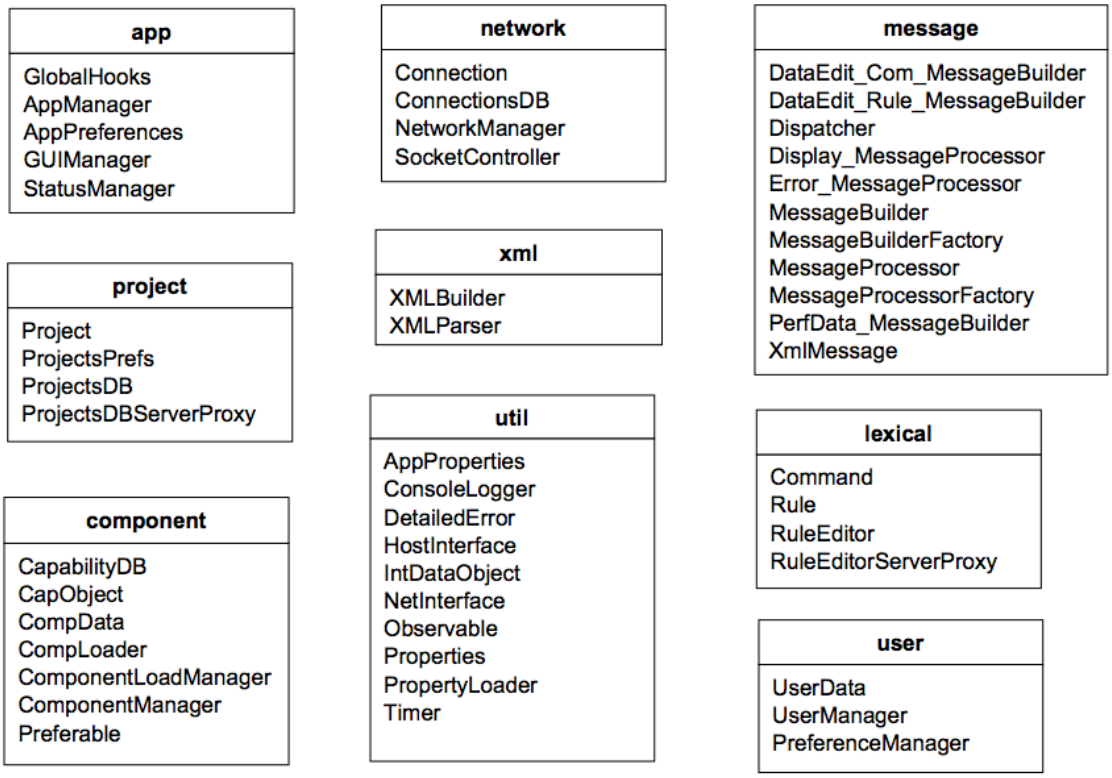


Figure 47 – LIMPT Client ActionScript packages

The place of these code libraries within the overall architecture of the client can be seen in Figure 48 and accounts for the difference in complexity between the Director and Flash layers. The Director component mainly serves as a wrapper for Flash providing extensible hardware interface services. Figure 48 only shows the built-in sensor and actuator functionalities, they are usually added to by loading additional components that utilise Director's wide range of plug-in extensions (called Xtras). A MIDI in/out component is included in the reference implementation.

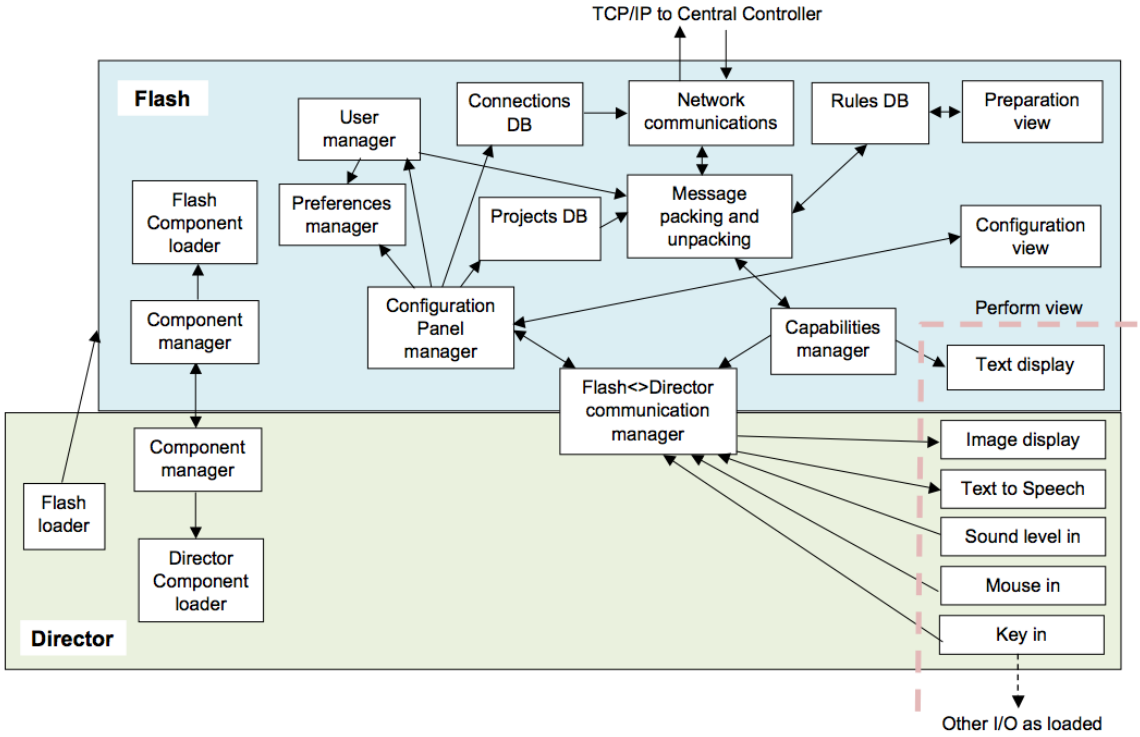


Figure 48 – LIMPT Client reference implementation architecture

When the client is started, the Director layer loads the Flash component and, after initialising the application framework, an XML configuration file is loaded which specifies which system components (configuration panels and in/out components) to load. This approach was adopted to maintain flexibility and extensibility; adding functionality is a matter of creating a Director object which interfaces with a given hardware device and the Flash panel which provides the required user controls. As long as both components[169] follow the client component API they will be loaded and operate just by being added to the XML component list. Once the component list has been loaded and parsed, Director and Flash components are loaded. When loading is complete, each component is initialised and input or output components notify the Capability Manager

[169] Components are not required to have both Director and Flash elements – either can be omitted if not needed.

about the data type and display mode they support. The Capability Manager uses this list to provide a fall-back for any cues and triggers for which a client instance does not have the required functionality; an alternative presentation is used (so if the server requests text/speak for a cue, but no text to speech component is available, it would be printed on the screen instead). This approach was adopted since the coupling between the Central Controller and clients is (deliberately) weak. Clients are seen as essentially mutable parts of the system; additional ways that individual artists can customise sensing and actuating to suit their own practices and projects. The only requirement for a client application is that it sends and receives messages formatted according to the schemas presented above. This does, however, force decisions about how to handle data that the client cannot process down to the client level[170].

Once the capability list has been created, the client loads the user's preferences (based on the last user if one exists; if not, a set of defaults are used) and requests all components to configure themselves accordingly. The client then auto-connects and shows the user interface view that was last used[171]. The client has three user interface views (see Figure 49): Client set-up, Performance preparation and Performance:

[170] It would be possible to make the capability tracking part of the Central Controller's services, but this was felt to be over-complex.
[171] Both these are fully configurable.

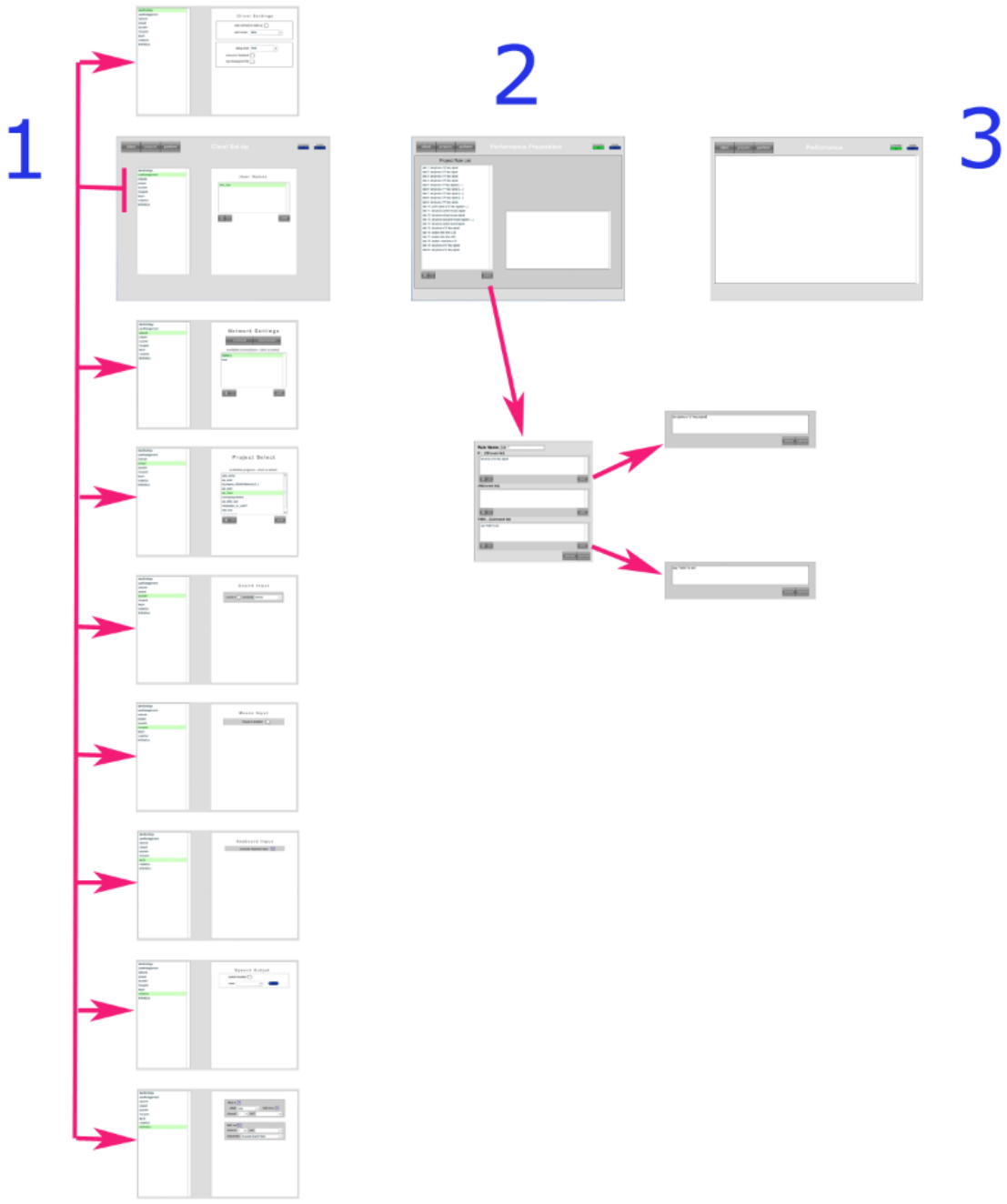


Figure 49 – LIMPT Client UI navigation map

The design of the client navigation is task-based; the three main screens each focus on a single aspect of the activities surrounding live performance. Users can switch between any of the main screens with a single click; however, it is envisaged that many users, particularly if they are performers, would spend

most of their time in the performance view (view 3 in Figure 49). A typical performer's use-story (Wells 1999) might be (NB this is invented):-

> The first time I used the system, I downloaded the programme onto my computer, then ran the app. I entered my name and selected the name of the project we were working on and connected. Ever since then, all I have to do is start the app and it just shows me the performance screen.

The performance screen is the simplest of the UI views:

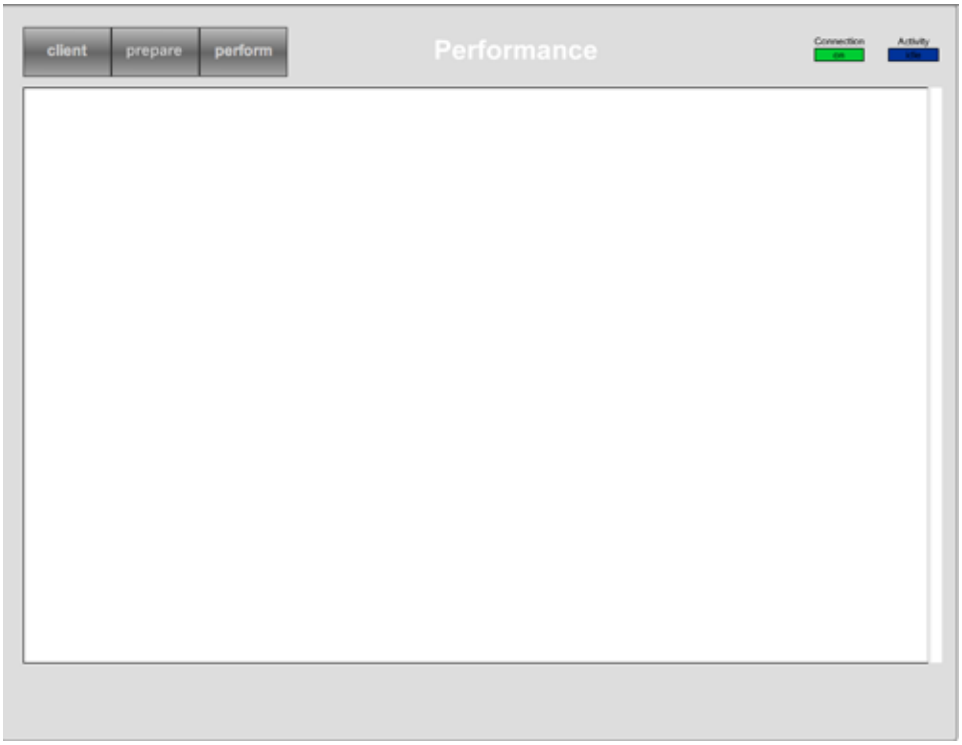


Figure 50 – LIMPT Client performance view

The design for this view deliberately removes all unnecessary distraction ("noise may or may not be sound; it is simply the part of a communication one does not want" – Prior 2011) from any communication with the performer; text and images are automatically scaled to fill the central white area without distortion of their aspect ratio since it is assumed that some performers (e.g. dancers) may be checking for cues etc. from some distance away. In this mode,

the client is receiving messages from the Central Controller and rendering them to suit the performer or device it is supporting.

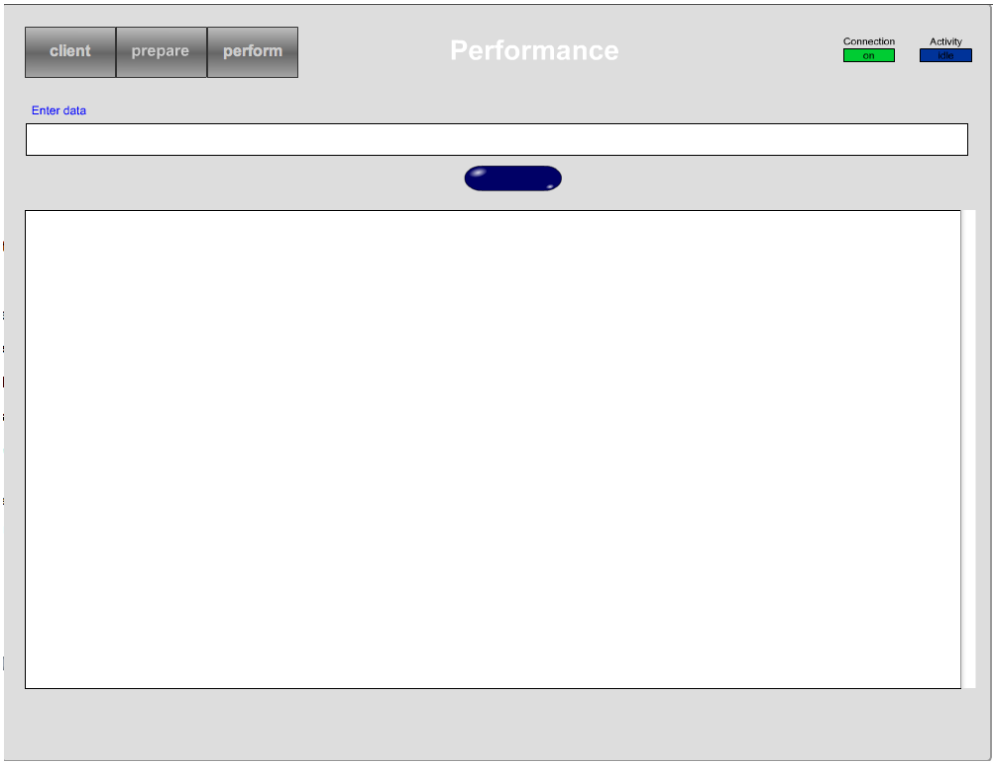


Figure 51 – LIMPT Client: Performance view with text entry field

If single-key input is turned off on the client set-up view (see Figure 52 below), the performance screen also provides a text field where text can be entered (rather than single key-presses).

While a client is showing the performance view, it may also be sensing performance activity and sending messages to the Central Controller about what has happened. The detail of the conversion of *activity* to *activity-event* (i.e. the identification of a complete gesture that needs to be sent to the Central Controller to be checked against the current rule-state to determine if it is

significant) is left to the particular software component that is monitoring a sensor. However, this operation always involves some element of quantisation, and, indeed, has to involve this if the problems of limited bandwidth and processing capability are to be avoided (e.g. Ramakrishnan et al 2004, Collins &Olofsson 2006). For many signal-type activities (e.g. mouse and key clicks, sound level inputs) the system currently only treats these as *compound events* (i.e. a sequential combination of a start and end action – see Figure 32) and the quantisation reduces all such activity to only three different temporal categories; *short* (<0.5 seconds), *long* (>0.5 seconds) and *double* (two click events within a second). MIDI events will be timed by the Xtra (Director plug-in) being used, but given that MIDI is already a quantised account of a performance (rather than a sampling of audio data itself) the monitoring frequency required to capture a full account of activity is still comparatively low compared to audio or video signal processing. Given the order of timescales involved in the range of sensor monitoring, sensor reporting frequencies of around 35-50Hz[172] are adequate to capture sensed events and produce cues or triggers without perceptible delays.

The only control elements in the performance view are the main navigation buttons in the top left-hand corner; these are common to all views and operate similarly throughout. Consistency was a central principle of both the information architecture and design of individual controls since this helps provide the following benefits:

[172] The actual rate will depend on the processing power of the machine running the client and on the nature of the sampling; Xtras impose differing processing loads.

"• Users require less training
• Users can work faster and more effectively on consistent systems" (Koritzinsky2001:102).

The client set-up view (view 1 in Figure 49 and also Figure 52) is the most complex and also, potentially, the least predictable part of the user interface. In this view, the client presents the available configuration panels to the user; any extra functionalities that are added will generally have a corresponding configuration panel which will be added to this view's left-hand menu on start-up (via the XML configuration file discussed above).The set of panel views shown in Figure 49 thus reflect only the current set of installed input/output and configuration panels. However, the interface controls are consistent in appearance and functionality across all the UI elements; for lists, there are three buttons and a list of items which can be selected[173] (Figure 52 shows the controls for the user name selection, but the design is common).

[173]The design was influenced by that of some Mac OSX UI elements, although the labelling and control sizes have been made much larger.

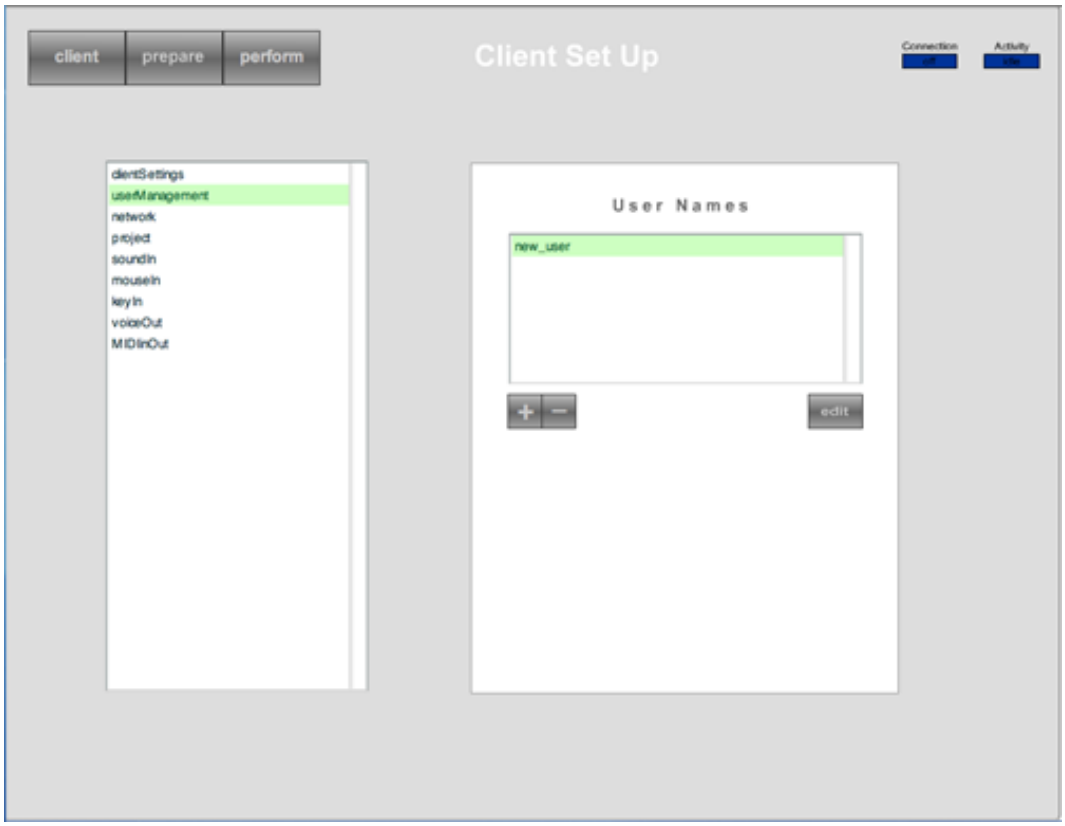


Figure 52 – LIMPT Client: Client set-up view

In use, a performer will only need to use three or four of the configuration panels, and these only on the first occasion that the client is run. On their first use of the client, they begin by entering their name[174], this is how they will be referred to in the project's rules and will be remembered by the client.

[174] Just as in the eMerge language, performers and devices are identified by the name entered into the client. System behaviour if there are duplicate names within the same project is unpredictable and will depend on the order of connection in any session. There is a facility to enter a password, but this is not actually used at present.

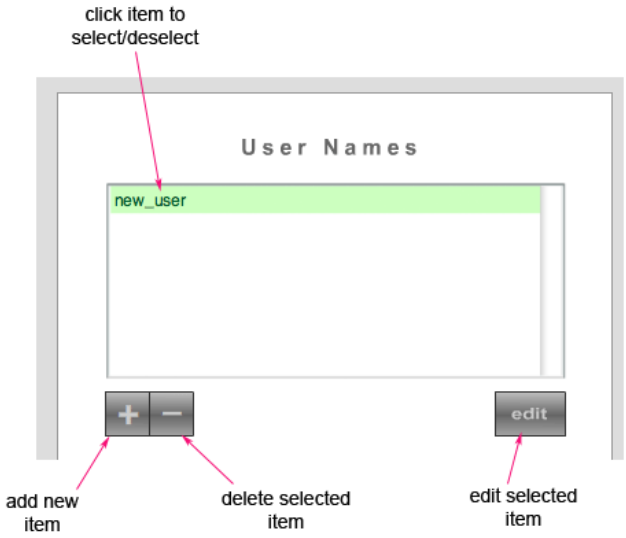


Figure 53 – LIMPT Client: User names list showing UI element functionality

The user can control if the client will auto-connect in future and which view will be shown (this is optional):

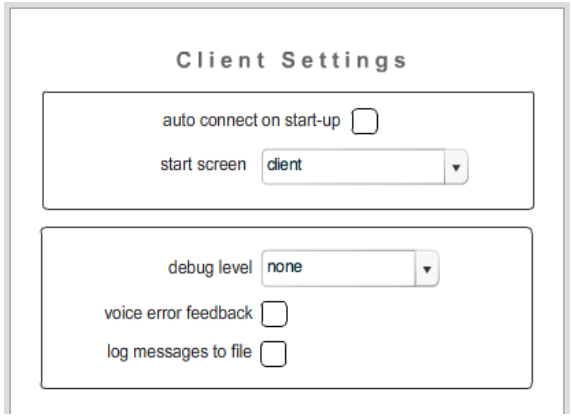


Figure 54 – LIMPT Client: Client settings panel

They then connect to the Central Controller in the network panel (Figure 55). The default setting here reflects the overall design decision to make the Central

Controller essentially a web service, although there is the option to add a new server if artists want to host the central controller on their own machine[175]:

Figure 55 – LIMPT Client: Central Controller selection

When a client connects, the Central Controller sends it a list of available projects, the user is then able to choose the project they are working on in the project panel (Figure 56), although the default on first use is a project, *Introduction_to_LIMPT*, which takes new users through the concepts and functionality of the system using the system itself.*The LIMPT Workshop Guide* (Appendix 6) contains detailed, step by step instructions on how to set up and use the system for those encountering it for the first time. As well as connecting, it assumes workshop participants will be interested in creating their own performances using the preparation view (Figure 57).

[175] The 'local' option is for testing. The combined processing load is such that both the Central Controller and a client instance can run without problem together, even on modestly specified or older machines.

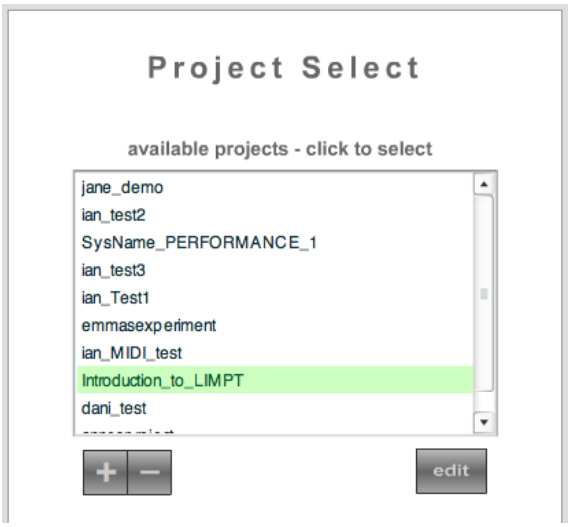


Figure 56 – LIMPT Client: Project select panel

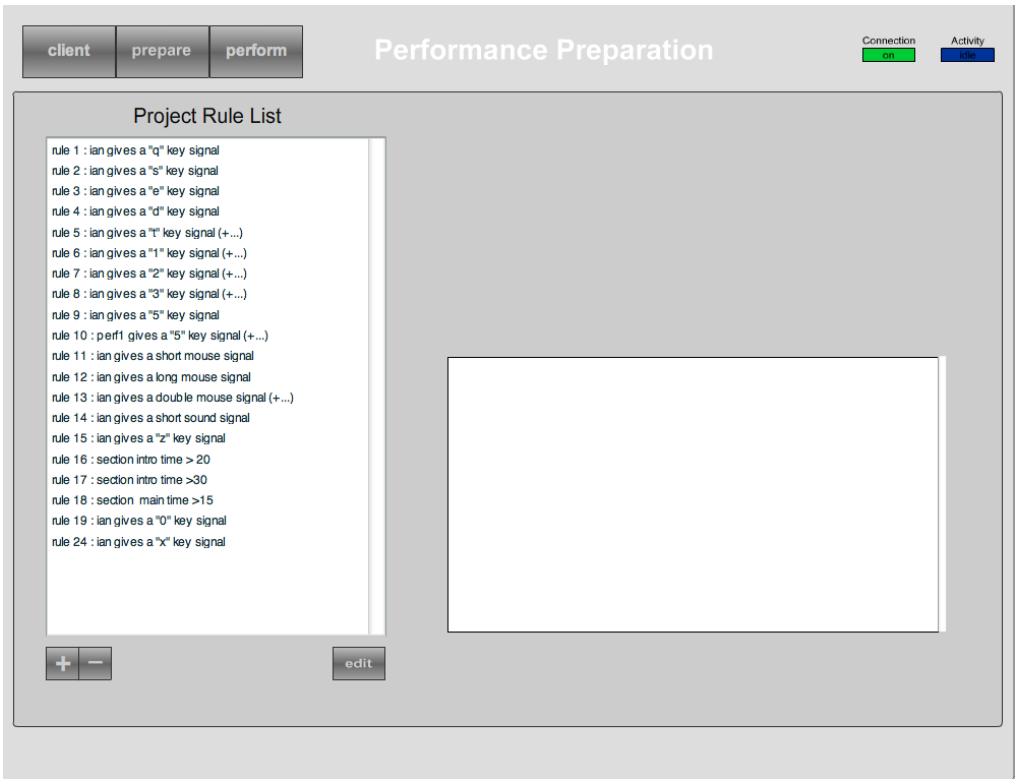


Figure 57 – LIMPT Client: Performance Preparation view

The preparation view is where the rules specifying the contextual connections and influences within a performance are listed and can be added to, deleted or

edited. The interface functions in a very similar way to the other list-based elements, except that there is an intermediate editing panel:

Figure 58 – LIMPT Client: rule-editing panel

The three list panels function in the expected way – allowing the user to add, edit or delete individual events and commands. When the store button is pressed, the rule is sent to the Central Controller and is validated and stored[176]; a notification of success or a message indicating the nature of the problem is returned and displayed in the panel on the right-hand side of the preparation view. Rules are active immediately once they are successfully saved.

[176] The nature of the storage depends on the rule name – if it already exists in the project, the rule is overwritten, otherwise a new rule entry is created.

6.3 Chapter Summary

The chapter began by discussing the construction of *eMerge*, a Domain-Specific Language to control live performance. The language is deliberately aimed at users with a performance background and this is reflected in its grammar and vocabulary. The implementation of the prototype LIMPT system was then described, beginning with the overall system architecture and moving on to explain the formats of the XML messages used for communication between clients and the Central Controller. The implementation and operation of the Java Central Controller was covered in detail for different types of data flow: activity events, commands and rules. The chapter concluded with a description of the LIMPT client reference implementation. It is anticipated that other clients might be produced to work with the system and so one of the goals of the client development was to produce, and validate through use, a set of ActionScript software libraries to support other developers. The architecture, task-centred navigation and user interface views of the client were then described alongside the user-experiences which it might support.

7Evaluation of Prototype

The previous chapter described the realisation of a prototype generic system for supporting work with interactive multimedia in live performance. This chapter systematically evaluates that prototype using the three distinct approaches (see section 3.5): user feedback from a small-scale workshop for specialists together with data gained from student workshops; an evaluation against the formal statement of requirements (see section 4.3); and a comparative study which assesses the functionality of the LIMPT system against that offered by currently available software resources used in digital performance (see section 2.2.2). Part of the material for this chapter is based upon a public workshop given at the Digital Resources for the Humanities and Arts 2010 (DRHA10) conference at Brunel University in September 2010 and on parts of the paper *Words of Power*, published in the International Journal of Humanities and Arts Computing (Willcock 2010).

7.1 Use in Workshop

The LIMPT system was presented in an immersive, participative public workshop lasting two hours presented at the DHRA10 conference on 7th September 2010. The workshop was aimed at practitioners already involved in working with technology within their live performance practice. All participants were asked if they would provide feedback on their experience of the LIMPT system for research purposes[177]. I was particularly interested in how intuitive and comfortable the system and its elements (hardware, software and language) seemed after a short-term experience and also whether participants felt they have been encouraged to extend their use of the system and/or develop their creative or critical practice further (see appendix 8 for the questionnaire used to collect data). The workshop was structured so that the system was introduced using a prepared online introduction and then a demonstration set of rules using a sensor set-up of a set of floor pads. Participants were then invited to experiment with creating and editing rules and asked to feed back ideas about any potential integration with their own practice.

The workshop had four participants[178], and three completed questionnaires were handed in. Taken alone, the number of subjects is far too small for any reliable conclusions to be drawn; the material presented here should be considered not as the result of a conventional set of user acceptance tests, but

[177] It was made clear that providing feedback was entirely optional and that no identifying data would be collected and that all responses would be kept confidential.
[178] This attendance was similar to that of other sessions; due to pressure of time, several conference events were scheduled in parallel. There was also a short fire alarm drill during the workshop!

rather as a set of provisional contextualisations for the criteria-based evaluations presented in the following sections[179].

The participants who gave feedback came from backgrounds they described at theatre/movement, multimedia theatre, body art and installation art; no one mentioned music or sound. All the respondents found the specification of triggering events and commands in the eMerge language easy to understand and all agreed that the model of live performance as a series of decision points which underpins the LIMPT system could integrate, at least partly, with the ways they thought about and worked with live performance.

All participants were at least possibly interested in working further with the LIMPT system and mostly felt confident enough after the workshop to explore it further[180]. Opinion on whether the workshop had suggested things that participants might want to develop further (either with or without the LIMPT system) was mixed, however; two respondents were positive, while one was strongly negative. The final question was open, asking about features or improvements which subjects would like to see; this elicited some interesting responses, even from such a small sample. One subject suggested ‘a simple hardware toolbox (an interface and some basic sensors)’ which was easy to use, with minimal set-up and which might make the system more attractive to

[179] Additionally, it should be remembered that this was a group of digital performance experts whose short exposure to the system and its underlying conceptualisation of performance was informed and facilitated by their enthusiasm and knowledge of working with technology in live performance.

[180] One respondent said they were not really confident to explore it, but specifically mentioned the fire alarm which occurred during the session as the cause.

users reluctant to get involved with the physical technologies involved in digital performance. Another mentioned better system stability (see below), while the third had the idea of a default set of rules in a new project which would then act as a starting point, being adapted for particular applications.

Despite the limited amount of information gained directly from respondents, the deployment of the LIMPT system was smooth and without problem once access to the port used to communicate with the Central Controller running on Bartleby had been provided. Physical set-up took about an hour which included taping down five pressure pads and setting up a data projector[181]. The system was installed without any problems on machines owned by Brunel University (who were hosting the conference) and on participants' own machines. All managed to connect and use the system and run the online introduction to the system (see appendix 7), although there were some issues with system stability on view change[182].

While the caveats detailed above concerning scale must be borne in mind, I think that it is significant that all those who expressed a view felt that the underlying model of live performance and its articulation and representation in the eMerge language was consonant with their conceptualisations and working methods. However, I'm less confident about ascribing similar meaning to their reported desire to use the system further; it seems to me that the identification

[181] The most time-consuming part of the set-up was untangling the wiring for the pressure pads; the sensing system used (which is entirely separate from the LIMPT system) required long cable runs of split ribbon cables.
[182]This is a known and long-standing problem. It has proved very difficult to trace due to the lack of low-level debugging support in Director and (to a lesser extent) Flash.

of the researcher with the system within the workshop context means that the possibility of social pressures affecting this particular result is high[183]. I feel that the remaining results are not clear cut enough to be significant; opinions varied, but no significant patterns could be observed. While the experience of presenting the workshop is useful for the research described here, it also reinforces the necessity for more extensive user-surveys (both formal functional user testing and more open user experience surveys) in future work (see section 8.3).

[183] I don't feel the same about the results to the questions about the model and language, while these obviously underpin the system, they are not 'present' in the same way that the software and hardware are.

7.2 Evaluation Against Requirements

Chapter 4 established a formal set of requirements for a system designed to provide generic support for the use of multimedia in live performance. These requirements were based on information gathered from analysis of in-depth interviews with experienced practitioners (section 4.1) and systematic reflection on personal creative experience (section 4.2). These requirements now form the basis for a structured evaluation of the LIMPT system against each of the specific criteria in the four categories: Data (section 4.3.1), Usability (section 4.3.2), Technical Requirements (section 4.3.3) and Functional Requirements (section 4.3.4). Each section of the requirements established in chapter 4 is presented below with detailed evaluations of the extent to which the prototype LIMPT system meets each requirement-specification. These detailed evaluations are then summarisedin section 7.2.5.

7.2.1 Data

Category	Function	Description	Purpose	Priority	LIMPT system implementation
Management	Storage	System configuration and settings	The operational settings for the system's components need to be stored locally with a reasonable set of default values provided.	2	Default values are provided in a configuration file which is automatically loaded and allows the user to connect and access the LIMPT Introduction with only a few clicks. Subsequent changes are stored locally automatically.
		Performance configuration	The specific connections, mappings between different devices and any assets required need to be retained in a secure and reliable way throughout preparation and between performances.	1	These are stored both in the Central Controller's inference engine and in the MySQL database which is located on Bartleby, the DMU server. Whenever the Central Controller is restarted, all data for all projects is retrieved from the database and reloaded into projects. Asset handling is only partially implemented; asset names are saved, but no manifest of assets for a project is produced or validated on controller restart.
		Performance archiving	The system should retain a record of what happens in a performance for archiving and review.	5	Operation is logged and in normal operation (i.e. using the Unix *nohup* option[184]) is written to a text file. This is unwieldy and currently only usable by specialists.
	Access	Physical location	Access to data needs to be independent of location so that the system is useable from rehearsal rooms and a range of performance spaces.	1	Provided a working Internet connection is available and access to port 7010 is allowed, the system can be used anywhere.

[184] The *nohup* option allows a process to be started by a user logged into a terminal session which is not then stopped (or terminated) when that user logs out, but keeps running. Output which would have appeared in the user's terminal window is instead written to a text log file – which can then be subsequently examined when the user logs in again.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
		Time and duration of access, availability	Access to data needs to be at times, and over time-periods, that are not constrained by assumptions about working patterns.	3	The Central Controller runs for months at a time without intervention; the system has high availability.
		User access to data	The system manages access to performance and system data based on a system of user privileges.	5	The system client has provision for name/password entry, but currently the Central Controller does not restrict access to projects on this basis and the database does not record who created items. Individual rules can be created and deleted by anyone. The underlying database is password-protected and currently whole projects can only be deleted through direct operation on the database.
	Internal communication	Speed, bandwidth	Data needs to be processed and transferred between devices at speeds compatible with human performers and to be able to cope with multiple overlapping demands for mapping and connection.	1	No latency noticeable to humans has been noted for event capture. For large assets such as some images, there can be a noticeable delay while they are downloaded depending on network speed. The way the system deals with assets probably needs to include an automatic local-caching facility; this is already allowed for in the XML message specification.
Input	Sensing performer activity	Simple digital inputs	The system needs to accept simple inputs such as pressure pads, computer-key presses, mouse clicks.	1	Key presses and mouse clicks are supported as is typing text. Pressure pads can be supported either using a USB interface such as iPac or via MIDI.
		MIDI performance data	The system can accept a range of MIDI input data.	2/3	MIDI input is supported for some common MIDI events (noteON, noteOff and Continuous Controller events) and MIDI can also be captured at the 'note played' level (i.e. a combination of a noteOn and noteOff). There are issues with cross-platform operation as MIDI is captured using a platform specific plug-in.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
		Other complex data	The system can use gestures or positional data as inputs	4	This is not currently implemented directly, although systems which map to MIDI notes would be supported.
	Input from devices	Interface boards as inputs	The system can accept input from interface boards such as Arduino or iPac.	5	The system cannot accept serial input as used in older Arduino boards, but the newer ones (e.g. the *Uno* 2010), which emulates a USB device could be supported at a basic level though treating events as key-presses. iPac interfaces can be supported in the same way.
		Stand-alone digital systems or devices	The system can accept input from other digital systems such as Max patches or bespoke applications.	4	Systems which map to MIDI notes would be supported and any system which could provide appropriate XML messages via a network socket could communicate with the Central Controller. However, OSC is not currently supported[185] which would be a useful and flexible way for Max patches (and other devices) to integrate.
	User control	During preparation for a performance	A user should be able to enter the information required to create and edit the configuration and settings associated with a specific performance as it is devised.	1	The rule editor allows complex rules to be written and stored. Stored rules can be edited or deleted. There is some feedback on syntax errors.

[185] OSC input and output was implemented in a Flash/Director hybrid in the second of the installation works produced during the period covered by this enquiry.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
		During performance	A user should be able to engage with the progress of a performance in real time.	2	The current version of the client does not provide the user with an opportunity to directly send commands to the Central Controller for immediate execution. However, the Central Controller does implement this functionality and does respond to commands generated by clients (e.g. get me the rule list).
Output	Connecting to devices	Triggering video, sounds, lighting changes or other effects via MIDI	The system should be able to connect to and initiate activity in output devices connected via MIDI	3	The system is able connect to devices or software systems and control them by sending MIDI events. This has been tested on self-contained MIDI synthesisers and also with a stage-lighting control system, which accepted MIDI input.
		Complex continuous control of systems or devices	The system should be able to control the operation of a device (e.g. a lighting board or robot) or system (e.g. an application generating an avatar or performance environment) at a level beyond that of starting and stopping activity.	4	The Central Controller and language definition currently only support some MIDI events (noteOn, noteOff, controlChange, programChange and pitchBend). The current client implementation is more limited and only supports noteOn and noteOffevents which, even though suited mainly to initiation or triggering, do have associated values which can be used for finer control.
		Interface boards as outputs	The system should be able to use interface boards to control actuators such a servo motors and other low power devices.	5	This is not currently supported.
		Live monitoring of system	The system should allow its operation to be monitored, remotely or within a performance.	4	This is supported only if the Central Controller is run in terminal mode, and is thus not available to most users.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
	Communicating with performers or audiences	Cuing performers (visual/audio)	The system should be able to provide a simple cue to a performer in different ways that are appropriate to a range of performance situations.	1	The system can provide a range of cues suited to different performance situations. These includeimages, text displayed on screen, spoken text and pitched notes.
		Presenting text, image or sound material to performers or audiences	The system should be able to present pre-selected text, images or sounds. These could be instructions, scores or other materials associated with a performance.	2-3	The current system cannot present text or sound files, although text for presentation can be written into rules. Image files can be presented.

7.2.2 Usability

Category	Function	Description	Purpose	Priority	LIMPT system implementation
Target users	Performance makers	Simple connection and set-up	Setting up the system should not be different to normal computer usage (so may involve connecting to the Internet and running an application, but not more complex operations).	1	Once a computer is connected to the Internet, the client is run as a standard application. On the first run, the user enters their name, clicks to connect to Bartleby and then selects the Introduction to LIMPT project; in subsequent usage, the client can be set to auto-connect and start in any desired UI view. This criterion was demonstrated in the workshop (see section 7.1).
		Performance preparation: skills and knowledge	The preparation and configuration of the system for a performance should rely on performers' established concepts about performance and performers.	2	The underlying metaphor, of performers who give signals to each other when significant activity occurs is well supported by the system and language. This was confirmed in the workshop where all participants agreed that the model of performance might integrate with their practice and that the language was easy to understand.
		Technical skills requirement	Technical or programming knowledge and skills should not be required as far as is possible.	2	The basic use of the system does not require specialist knowledge. However, using additional sensing or external output devices does require knowledge not just about the devices or sensors themselves, but also about the type of information required for input or output. For devices connecting using MIDI, this is still within the experience of those with a music background, but those from other disciplines may struggle.
		Use in performance	Creators should be able to intervene in and contribute to performances as they are taking place.	2	The client UI does not currently support this (although earlier versions did). As comparatively minor development effort would be involved, this would be an early target for future development.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
	Performers	Performance preparation: skills and knowledge	Minimal active preparation should be needed for performers and technical knowledge should not be required. Any configuration settings should be retained locally.	2	Once a computer is connected to the Internet, the client is run as a standard application. On the first run, the user enters their name, clicks to connect to Bartleby and then selects the Introduction to LIMPT project; in subsequent usage, the client can be set to auto-connect and start in any desired UI view, so a performer could set it to start up and show the performance pane without any further configuration.
		Use in performance	The system should demonstrate clearly that it is working, but should operate in a way that integrates with the performer's specific discipline as transparently as possible.	1	The performance view (see Figure 49) is almost entirely devoted to the task of displaying cues; there are two small indicators confirming connection and showing network activity, but otherwise the interface view is empty except where specific content is required by the performer.
	Developers	Source code available	Non-commercial parts of the source code should be available through a standard repository web portal.	4	The source code for the non-commercial elements of the system (i.e. everything except for the Jess inference engine and the MIDI Director plug-ins) are available on the public-facing repository hosted on Bartleby (http://bartleby.ioct.dmu.ac.uk/repos/mm_server/)
		Source code documentation available online	The source code should be documented in a standard format (e.g. Javadoc) and made available online.	5	Online documentation of the source code has not been publicly provided.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
		API and documentation for extensions	There should be a template for adding functionality to the system, particularly for adding new sensing or cuing capabilities.	3	There is an Interface[186] and API which has been used successfully during the development of the prototype implementation, but the documentation is not complete or appropriately formatted for public use.
Training	Community	General introduction to system	There needs to be a presentation which describes the system and its capabilities to those who are not users, but who might be interested in working with it.	3	There is an online introduction to the LIMPT system which is available to users through the system itself. To access the introduction, start the client, connect to Bartleby and then select the *Introduction to LIMPT* project (see appendix 6).
	Performance makers	Reference resources for preparing performances	There needs to be comprehensive, searchable reference documentation available with examples of how to prepare for a performance and implement specific functionality.	2	There is a workshop guide (see appendix 6) which is designed to get new users started with the system and to demonstrate its main architecture and features. There is also a detailed guide to the *eMerge* language which covers all the currently implemented features of the language with examples and descriptions of parameters and syntax variations (see appendix 3).
		Working demonstration	A interactive set of working examples showing how the system can be configured to operate in various ways.	4	There is a set of examples which have been used for workshops with students and at conferences, but these have not been drawn together into a definitive demonstration project.
	Performer	Short guide for performers	A limited guide which can be read or communicated quickly and which sets out just sufficient detail for performers who are not involved in performance preparation.	4	The workshop guide has proved effective in student workshops and at the DHRA10 conference in enabling users to connect to the system reasonably quickly and confidently. This guide is not specifically aimed at performers however and does cover some material that they would not need to know.

[186] *Interface* is used here in its OOP sense, meaning a set of required functionality that an implementing object should provide.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
UI views	Navigation	Clear, task-based navigation structure with a small number of user interface (UI) views.	The interface should be based only on a small number of views of the system which reflect phases of activity involving a set group of tasks. It should be immediately obvious which view is being presented.	1	There are only three main user interface views, each focussed on a single specific set of tasks (client set-up, performance preparation and performance). Each view is characterised by a different graphic identity (which are discernable under low light conditions or from a distance) and by labels.
	Configuring	System configuration options	The system must allow users to configure settings	1	The reference implementation of the client allows many settings to be altered. The client is designed following the Model-View-Controller pattern meaning that more or fewer user-configurable options could be provided easily.
		Configuring extended functionality	The configuration view must allow configurations for a variable number of system components to be accommodated.	2	Configuration panels are handled by a panel-manager which loads panels according to an XML configuration file at start up. There is no limit to the number of panels that may be loaded. The Central Controller is less configurable; system-wide options are set by a properties file loaded at start-up.
	Preparing	Performance preparation overview	The system allows users to prepare and save a performance configuration.	1	Performance configurations (rules) can be written and saved to the Central Controller's database.
		Retrieve and store performance configurations.	The system should allow users to view and edit saved performance configurations.	2	The current project's rules are listed and each can be edited or deleted. Editing a rule automatically updates the saved version. Validation errors are generated and displayed.
		Feedback on outcome of performance configuration operations	The system should confirm that users' work or assets have been stored.	1	The system confirms a successful rule save or update. If necessary, validation errors are generated and displayed to the user indicating the problem. Asset management is currently not available through the client.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
		Store performance assets	The system allows users to select and store assets (e.g. images, texts, sounds…) to be used in the performance.	4	Asset management is currently not available through the client.
		Manage asset storage	The system should allow users to manage and preview stored assets.	4	Asset management is currently not available through the client.
		Integrated help system	The system should assist creators, through interface design and contextually aware help-text, with the process of preparing performances.	3-4	The design of the rule entry interface panel reflects the structure of rules. However, predictive text and contextualised help are not implemented.
	Performing	Optimised performance view	The system should present an interface which is focussed on the needs of performers and which is informed by the way they are relating to the system.	1	The performance interface is effective and focussed on the needs of performers in difference situations; almost no unnecessary communication is employed.
	Language	Vocabulary and structure to be based on natural language	The language employed by the system for feedback and configuration should be close to natural language and use a vocabulary drawn from performance where possible.	1	The language is modelled on the concepts and vocabulary used by performers and appears effective for the currently implemented features. This evaluation has been evidenced by the responses in the workshop.
		Provision of versions of system localised for other languages	The system and its documentation are available in languages other than English.	5	The system and documentation have not been localised for any other languages.

7.2.3 Technical Requirements

Category	Function	Description	Purpose	Priority	LIMPT system implementation
Platform	Hardware	System should run on standard hardware that is readily available	The system should be usable with non-specialist hardware having performance characteristics and hardware configurations that reflecting those found in affordable systems.	1	The processing, memory and storage requirements are modest and hardware limitations on performance have never been observed.
		Required user interface devices	The system should be useable at some level with standard UI devices (mouse and keyboard).	1	The system is usable without any additional interface devices, although the opportunities for integration in performance are consequently limited.
		Cross-platform operation	The system should be available for appropriate combinations of operating systems and devices commonly used by digital performance practitioners.	2	The system runs successfully on both Macs and PCs (although not all input options are currently available on both platforms in the prototype system due to licensing constraints). Operating systems from Windows XP or OSX 10.4 onwards are supported.
		Standard network connection protocol(s) used for connections	The system should use connection methods and transport protocols which are supported widely.	1	The system components connect using standard Ethernet (both wired and wifi) and tcp/ip connections.
	Availability	Stability	The system should be stable enough for live performance use.	1	The system is highly stable in performance. However, there have been stability problems observed when switching repeatedly between UI views, but this does not occur in performance.
		Usable at any location	The system should be usable for preparation and performance in any location.	2	The system can be used with full functionality in any location where a live Internet connection can be made.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
		Usable across any physical scale of venue or performance	The system should not make assumptions about the size of a performance venue or of the proximity of creator(s) and performers.	2	The system uses the Internet to connect its components; their physical proximity does not affect operation, although over very large distances, network delays might be noticeable. Because the system is based around cues rather than live streams, latency is not the problem it is where audio or video material is shared between distant locations.
		Available over extended time periods	Users should have the opportunity to engage with the system over extended periods both during preparation and over the course of development projects.	2	The Central Controller is running continuously; users can work with the system whenever they wish, their projects will be saved ready for subsequent use.
	Configuration	Default configuration	The default configuration settings should allow users to begin to work with the system without requiring extensive customisation.	2	The default client configuration settings only require the user to enter their name and to initiate the connection to the default Central Controller. This has proved very easy for those new to the system.
	Peripherals	MIDI interfaces	The system should be able to use MIDI interfaces where available.	2	The system is believed to be able to work with all MIDI interfaces currently supported by the host operating system.
		User Interface Devices (UID) as input	The system accepts input from UID.	4	Input from UIDs may be supported if their controls are mapped to standard key-presses and mouse clicks, but the system does not provide a mapping system.
		Use of controllers	The system can interface with controllers such as the Wii remote and Kinect.	5	Direct input from these devices is not supported.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
Architecture	Scalability	Complexity appropriate to projects involving three or more devices	The system is applicable for projects that involve three or more devices or performers. For smaller-scale projects a single stand-alone application is probably preferable.	3	The suitability of the system for a particular application will depend on the complexity of devices as well as the scale of the project. The system performs well at core tasks, but as applications become more specialised or conditional specifications more complex, suitability will diminish. This has proved difficult to measure reliably, however.
		Able to accommodate a wide range of performers or devices at a time	The system can accommodate between 3 and 50 performers or devices within a performance without degradation of response.	1	The system has run reliably with up to eight connected devices; however, full stress testing to validate operation with higher numbers has not taken place.
		Accommodation of intensive processing requirements	A requirement for intensive processing should not affect other parts of the system's operation.	1	The system's architecture is distributed; each component runs on a separate machine[187] and only has access to the processing resources of that machine.
		Manage and support multiple performances	The system should be able to accommodate multiple performance projects over the same time period.	3	The system maintains and runs each project separately in software, although they all share the same central database store.
Extendibility	System development	Defined architecture, API and documentation for creating extensions	There should be a template for adding functionality to the system for adding new sensing or other capabilities to meet the specific needs of creators that are not already catered for.	2	A template was established and used during the development process, but it is not documented to the standard required for wider use.

[187] It is possible to run both the Central Controller and a client instance on the same machine without problems; while much more complex to set up, this configuration allows operation outside Internet-connected locations.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
	Interoperability	Protocol and format for interconnection between system components	System components should communicate using a bandwidth efficient, flexible, well-documented and portable format.	1	The XML messaging format is simple and efficient and is supported by a wide range of devices and programming systems. It is reasonably well documented in a small set of XML schema.
		Interface with digital tools that artists are already using	The system should work with the digital tools and devices already used by artists in their work.	2	The system can interface easily with the large number of devices and systems that use MIDI for control. Specialist interfaces (software and hardware) are available that increase the range of devices which can be used.
		Capability to interface with bespoke software and hardware	It should be comparatively easy for specialists to create interfaces in new or reworked bespoke systems that work with the system.	3	The system can connect to bespoke systems either using XML messages on standard networks or via a client using MIDI. Both of these are well-understood technologies.
	Source code	Use of version control system	Development should use a version control system for effective management of the development process and for enabling collaboration in future stages.	1	The development process used a standard public-facing Subversion version control system which provides the full range of functionality needed for efficient management of software assets[188].
		Open Source licence	As much as possible of the code should be made available free, preferably under an Open Source licence.	3	It is planned that, subject to commercial licensing of specific components, all code will continue to be freely available under an Open Source licence.

[188] The Subversion project identifies its aim as " Enterprise-class centralized version control for the masses" see http://subversion.apache.org/

7.2.4 Functional Requirements

Category	Function	Description	Purpose	Priority	LIMPT system implementation
Connecting	Routing	Specific routing	The system should facilitate connections between a specified performer or device and any, or all, other performers or devices.	1	The system allows any single performer or device to send a specified item of information to any, or all, other performers or devices involved in a performance.
		Multiple routings	The system should allow many specified routings simultaneously.	1	So far, no limit on the number of rules (and hence conditional connections) that can be active simultaneously has been observed. The upper limit would depend on the processing power of the computer hosting the Central Controller.
		Dynamic routing	Routings should be capable of being changed at any point during preparation or in performance.	2	Rules (which specify the conditional routings) can be activated and deactivated dynamically at any point in preparation or performance.
	Contextuality	Conditional routing	Routings should be conditional, where connections are made, or not, depending on specific activity.	1	Rules specify connections that are made when one or more specified events occur.
	Specification	Storage and validation	The system should accept, validate and store routing specifications.	2	Rules are parsed when they are sent to the Central Controller. If they are valid, they are added to the current rule-state and stored in the database. If they cannot be parsed successfully, an error is generated specifying what is wrong.
		Formulation of routing specifications	The system should accept routing specifications couched in language and structures that are based as far as possible on those used in performance rather than computing.	2	The eMerge language uses the vocabulary and structures of performance-practice wherever possible. Workshop participants reported finding it easy to understand.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
		Variable level of detail for specifications	Routing specifications should involve only the level of detail required.	3	Event specifications are scalable; full specifications are not necessarily required, see appendix 3, p 7.
Performance	Tracking	Track the activity of performers and devices	The system should maintain a data structure representing the activity of all performers and devices involved in a performance.	1	The Central Controller maintains a separate state-space for each project that represents the activity of all elements involved in that project.
		Track time	The system should enable timing to be part of a performance's properties.	2	The system supports an unlimited number of time-based objects which model formal units (e.g. sections, acts, phases), see appendix 3, p 12.
		Track physical spaces or locations	The system should enable spaces or locations to be part of a performance's properties.	2	The system supports an unlimited number of objects (zones) which model positions or places, see appendix 3, p 11.
	Journaling	Storage of tracking data	The system should store its tracking data.	4	The series of state-spaces are not stored, although the Central Controller's reactions to changes in state space are logged.
		Support reviewing performances	The system should allow users to review a performance's progress through its stored tracking data.	5	This is not supported for general users; specialists can review events by looking at the log file.
System and configuration	Components	Auto-connection and configuration	System components should be automatically connected and configured.	1	This is supported and is a client configuration option after the first run.
	Users	User settings should be saved automatically	Users' settings for any aspect of a performance should be saved automatically.	2	Users settings for all client and plug-in configuration options are saved locally automatically. However, they do not persist if a user changes the machine they use the system with.
		User access management	Users access to performances is managed.	4	There is no access management in the current implementation; all users have access to all projects.

Category	Function	Description	Purpose	Priority	LIMPT system implementation
Input	Processing	Quantisation	The system should quantise performance activity with some user control over the scale.	1	The client quantises user activity, although not all quantising is currently configurable by the user. The level of MIDI capture can be at event or note level; the sound level sensitivity is configurable, but the timing for click and key events is not.
		Mapping from one activity event-type to another	The system should be capable of appropriate mapping of events between performers and devices.	1	Any supported activity type can be mapped to any supported notification type.
		Passing activity data	The system can pass data from one device to another.	2	This is not currently supported dynamically[189], although it is planned for in the language specification (see section 8.3).
	Extensibility	Support for addition of extra sensor-types or new devices	The system should allow additional information types or formats or new devices to be used as inputs.	4	This is possible (and has been rehearsed during the development process), although as changes to both parser and Central Controller would be required, careful planning is required.
Output	Triggering	Conditional cues or triggers	The system should provide cues or triggers for performers or devices that are appropriate in format and medium and are conditional on specific activity.	1	The system can generate a wide range of triggers or cues.
	Extensibility	Support for addition of extra output-types	The system should allow additional information types or formats to be used as outputs.	4	This is possible (and has been rehearsed during the development process), although as changes to both parser and Central Controller would be required, careful planning is required.

[189] It can be simulated for a restricted set of values, but at present, each value to be passed would require a separate rule.

7.2.5 Requirements Evaluation Summary

The evaluations detailed above suggest that the prototype LIMPT system succeeds in meeting all of the highest priority requirements identified in chapter 4. It also meets almost all of those requirements with priorities of 2 or 3 although in some cases, this is partial. The system is able to work reliably[190] and flexibly and demonstrates the underlying model of integration with live performance in a way that is both easy to understand and quick for new users to engage with at an introductory level. It is capable of working with a wide range of information types and, because of this, can be used to support work in a range of performance situations and genres. It is available to practitioners over extended time-periods in any location where an Internet connection is possible and will run on standard combinations of hardware and operating system. Although the proposed upper limits on active connected devices have not been tested, the scales of operation that have been used have shown no signs of limitations caused by processing, bandwidth or storage.

The *eMerge*language models the domain of live performance in ways that practitioners find comfortable and which they are able to use quickly. Because of the emphasis on meeting the needs of the target users and limiting any specialised learning required, the language clearly shows the contradictions inherent in a DSL designed for non-technical specialists; some aspects of the bounding conditions for‘correct’ (i.e.meaningful) statements are represented in

[190] There is a persistent problem affecting the client when views are switched. At irregular intervals this operation crashes the client (although never the server). The problem is intermittent and only occurs when actually changing views – so use that concentrates on a specific task (preparation, performance) is not affected and other problems are very rare.

the grammar and checked by the parser, whereas others are dealt with in the control and logic of the Central Controller. In designing the language a decision was made that familiarity and coherence with users' existing patterns of language use about their practice would be prioritised over simplicity of structure and computational expediency. While from a computer science viewpoint the compromises involved may seem excessive, my interviews with practitioners suggest there is a widespread view that digital technology is often too inflexible in the way it presents itself to the non-technical user and that this does inhibit work (particularly non-goal-directed, exploratory work) involving incorporating technology withinlive performance practice[191].

Specific areas which the evaluation against requirements suggests are not yet adequately supported include: the need for a user-interface view enabling the direct use of commands within performances; the ability to use a wider range of MIDI events for input and output; language enhancements to permit passing information between performance elements; an asset management system; and an access control system which links users to specific projects and grants specific permissions based on their role(s). To some extent, these gaps in functionality represent the necessary priorities of a development process where having a system which could demonstrate some functionality was a priority and was judged as more important than implementing a complete feature-set. A provisional route-map for further development work is presented in section 8.3.

[191] I also found that practitioners reported that this perception of inflexibility tended to make their use of technology more static; once they had developed a facility with a device or system, they tended to keep on using it even if newer and more appropriate or capable systems became available.

7.3 Comparative Study

In forming an evaluation of the LIMPT system, one important metric is that set of functionalities and affordances provided by currently used resources in digital performance (see section 2.2.2). If there are already tools that provide any of the functionality offered by the LIMPT system, how does the prototype measure up to them? The approach used for this comparative evaluation is a two-stage process; firstly overlapping areas of functionality are identified and then the criteria for evaluation developed in section 2.2.1 are used to form judgements about the relative strengths of the two pieces of software.

There is no single piece of software that has a feature set that exactly matches that of the LIMPT system. This is to be expected, of course, since the current study began as a response to the repeated production of bespoke software. Less of that initially-provoking programming effort would have been required had there been tools available that met those needs that have been established as common across much performance practice (see section 4.1.1). Many of the applications currently used in digital performance (see section 2.2.2) have little or no feature overlap with the LIMPT system. Additionally, some resources that do have a degree of common functionality (Atlantic Waves, Auracle, Peersynth, Qunitet.net, EyesWeb, vvvv) are tied to a particular set of media types or interaction possibilities such that, although they facilitate interactive digital performance, they are very far from implementing the generic aspirations of the LIMPT system.

However, there are a small number of applications which do require a systematic comparative evaluation, these fall into two groups: systems designed for the production of live performance digital tools (Max, Isadora, Fieldand,to some extent,Module8) and those which are designed to provide flexible, customisable, controller mapping and data routing (JunXion, Mrmr, OSCulator).

Max, as has been discussed above (section 2.2.3), is an enormously powerful system for producing instruments and other resources for use in live performance. Indeed, it has been used to produce some of those specialised networked performance systems mentioned above (Atlantic Waves, Peersynth). Based on this and on the complex projects and diverse applications that Max has been used for, it is clear that the system *could* be used to produce a software system that had the same functionality as either component of the LIMPT system. However, it would also represent a significantly difficult task[192] and not one that could be attempted by anyone who wasn't already an expert Max developer. Further, while the feature-set of the LIMPT system could be produced using Max, the usability requirements would be significantly more difficult; while Max can certainly produce highly efficient specialist interfaces,

[192] This applies particularly to the Central Controller. The client, or rather an application offering a subset of its functionality (for example, a specialist sensor requiring limited user interaction), certainly could be implemented in Max – indeed this is one of the requirements for the LIMPT system, the ability to easily integrate with bespoke software.

this area of application is not regarded as a particular strength of the package[193].

Field is perhaps an even more generic and powerful system than Max in the sense that it is basically a set of libraries and an editing system. However, it also requires an extremely advanced knowledge of programming to use which suggests, based on experience and the results of my survey, a quite limited number of potential users. Similarly Module8, while ostensibly a VJ system (albeit a very flexible one), allows an expert Python programmer to create plug-ins and extensions which can radically alter both the look and functionality of the package.

Isadora in contrast, has a very similar approach to the identification of its target users and of how best to meet their needs to that of the LIMPT system. Potential users are creative artists working with technology in performance and they do not want to have to acquire extensive programming or computing skills to facilitate this. However, while the visual data flow metaphor (particularly in the Isadora implementation) is an excellent fit with the intuitive, exploratory approach of choreographers, building applications that support specific lower-level functionality that is not provided by an 'actor'[194] (i.e. custom networking) is

[193] Many users work with control surfaces or other systems to collect user input, which is then passed to Max using MIDI or OSC. The output capabilities of Max, particularly in the latest version are, in contrast, very powerful.
[194] '*Actor'* is the name given by Isadora to a self-contained functional block together with its visual representation. They are close to Classes in conventional OOP programming.

difficult[195]. Based on these limitations, one can conclude that while Isadora offers a very large area of functionality that the LIMPT system does not (especially the real-time manipulation and processing of video and audio signals), it is limited in the complexity and scale of performance it could support. Even the powerful extensions supporting specialist control of lighting (e.g. using the LanBox LCX with the specialist actors provided) imply a single agent directing that control rather than a flexible structure where control is exercised according to the creative demands of the project and artist[196]. For applications where more than two or three elements need to be interconnected, Isadora becomes an increasingly inappropriate choice for supporting live performance.

JunXion, the most sophisticated of the input mapping systems, is in some ways, very close to the functionality of the LIMPT system. It takes input events and then conditionally generates output (actions). The range of supported input sensors is extremely wide[197], indeed much wider than either Mrmr or OSCulato*r*.Junxion accepts input data from Arduino, Wii, iPhone, standard MIDI, sound (pitch and level), simple video tracking as well as from the standard computer keyboard and mouse. This activity is then mapped to OSC or MIDI events which can then be sent to the intended device for processing. Its range of supported input activity is thus wider than that of the current LIMPT system

[195] It can be done in a limited way using OSC messages, TroikaTronics publishes an example of master-slave operation of Isadora compiled applications, see: http://www.troikatronix.com/izzy-download.html

[196] It is also worth noting that Isadora's consistency and easy of use are not entirely maintained by the LanBox actors; they require some knowledge of networking to set up and use different conventions of image dimension specification from other actors.

[197] JunXion is produced by Steim in Amsterdam, an organisation with a long history of innovation in digital performance, see http://www.steim.org/steim/junxion_v4.html.

although this is not the case for output;JunXion is only an input mapping system. While it would be possible to use JunXion with more than one performer, this is not part of the usual deployment which is focussed on collecting events from a given set of sensors or data streams. Large-scale interconnection of many performance elements is not supported, although JunXion is not tied to a particular interface device like the other systems mentioned, it is still envisaged as a single-point device, collecting and merging a set of data streams in a more or less static performance architecture[198]. Although JunXion is an excellent choice for interfacing with input sensors for a single performer or small-scale performance, it does not provide the structurally all-embracing functionality of the LIMPT system or enable providing output cues to performers.

[198] Mapping specifications are stored in *patches* and these can be saved and loaded. Even so, this does not facilitate dynamic reconfigurations involving several elements.

7.4 Chapter Summary

This chapter has described three different approaches to evaluating the prototype LIMPT system. It was felt necessary to use three approaches partly because of the small sample size in the limited user-trial, but also because of the need for triangulation in establishing the reliability of the overall evaluation. The survey of workshop participants showed that the system could be deployed effectively in a 'real-world' context, and that users could get started and feel that they understood both the control language and underlying model of performance quickly and easily. The evaluation against the statement of requirements, established earlier in the study, demonstrated that the prototype system implemented almost all the functional specifications although there were a few areas of medium and low priority functionality which are outstanding, some of which have been identified as targets in future development (see section 8.3). The comparative evaluation suggested that, while there were no currently available packages with an identical feature set to the prototype LIMPT system, a few resources had functionality which overlapped that provided. However, once requirements for specialised knowledge and for scalability were taken into account, none of the resource-production systems[199] provided the mix of simplicity, scalability and flexibility of the prototype. Of the input-mapping and routing applications, JunXion, the most capable, has some underlying design goals that are very similar to those of the LIMPT system.

[199] I.e.those packages whose intended purpose is the development of applications for use in live performance.

However, it is not able to provide the scalability or output facilities of the prototype LIMPT system.

8Conclusions

The research project documented in this thesis began, as described in the Rationale (section 1.2), with the realisation that every performance project involving interactive multimedia seemed to begin with the creation, by a programmer, of the tools required for that project, and that this might not be the most effective way for the digital performance field as a whole to develop and prosper. This chapter draws together all the results established in earlier chapters and describes how they contribute to a coherent and systematic attempt to answer those original misgivings and, further, shows how the goals identified in section 1.4.1 have all been met. The chapter then looks at and evaluates the original contributions to knowledge made by this project and discusses possible further research activity and ways that the current research outcomes will be disseminated. It concludes with a brief discussion of the contributions the project might make to the future of live performance involving interactive multimedia. Some material in this chapter is drawn from the paper*Words of Power*, published in the International Journal of Humanities and Arts Computing (Willcock 2010).

8.1 Evaluation of Research Activity

Stimulated by the personal misgivings described at the start of this chapter, a systematic enquiry was planned and executed which sought to test the hypothesis that there are generic elements in practice involving interactive multimedia within live performance and, that if this was so, one might identify common features of artists' work and hence specify the requirements for a system which could facilitate these aspects of their practice. A survey instrument was developed and employed with expert practitioners from a range of performance disciplines, which produced six substantial interviews. The resulting data was then analysed using established qualitative methodology and an authoritative account of current practice across a range of performance disciplines and genres was produced. Significant patterns in this data were identified; the first observation was that there do indeed appear to be generic[200] features of work within interactive multimedia and live performance.

Following on from this result, these generic features were identified (section 4.1.1) and, together with evidence drawn from personal creative activity (section 4.2), were used to establish a set of formal requirements for a system designed to support generic features of work in digital performance. Working from the interview data and from the critical survey of previous academic activity presented in section 2.1, a novel model of live performance was developed (section 5.1) which suggested possibilities for the incorporation of digital

[200] 'Generic' was defined (section 1.2) as activity or aspects of activity which is common across a majority of practice, but which is not part of the production of identity for practice. So video projection was very widely used, but it was not considered as generic since the content and manner of its use was an important factor in differentiating one artist's work from another.

technology within performance practice in ways that were essentially generic, an integration that did not enforce or depend upon particular practices of creation or production but that could support many different kinds of work and artists. The combination of the statement of requirements and the conceptual model of performance was then used to define a formal computational model (section 5.2) which detailed the operations required by any system implementing the requirements.

Alongside the review of previous academic work, an appropriate set of criteria for evaluating performance-specific software resources (section 2.2.1) and a comprehensive and detailed survey of software currently used in live performance (section 2.2.2) were produced. The resource-survey stands alongside the account of practice described earlier and, taken together, these two outcomes represent a more detailed and comprehensive account of the diversity of activity and technology across multimedia and live performance than has previously been available.

Having established that it was possible to identify generic features of work in live performance and having identified the features of a system which could support creative activity across a range of practices together with conceptual and computational models which provided an integrative context and a formal specification for a software system, the project then began realising a prototype system. A Domain-Specific Language, eMerge was developed (section 6.1), which encapsulated the conceptual model of performance within the functional

framework of the computational model to provide users with a means of controlling the system. A lightweight XML-based messaging format was created (section 6.2.1) for system components to communicate with each other efficiently and robustly without requiring high data-rates. A software architecture was devised which both met the functional requirements for a system and which represented and reflected current high quality software design practices. This involved a Central Controller which keeps a record of the ongoing activity within a performance and generates cues or triggers as required. The Central Controller application (section 6.2.2) was realised in Java[201] with a mySQL database for persistent storage and incorporates the highly efficient *Jess*[202]inference engine for scrutiny of its state-space in determining if an output should be generated. A reference implementation for the client application (section 6.2.3) was realised as a Flash/Director hybrid incorporating a flexible plug-in architecture allowing additional input or output information types to be added if required.

The prototype Live Interactive Multimedia Performance Toolkit (LIMPT) system was then evaluated using three separate and different approaches to provide an element of triangulation to support the validity of conclusions. A survey of participants in a conference workshop (see section 7.1) provided evidence that the system could be deployed successfully and easily in real-life contexts and that practitioners found the system and its underlying metaphor clear and

[201] As has been described above (section 6.2.2), the Central Controller is based on code originally produced by Nick Rothwell for the eMerge project in 2003, although this has been extensively revised and extended.
[202] Jess is produced by Sandia National Laboratories, see http://www.jessrules.com/jess/.

relevant[203]. A detailed assessment against the statement of requirements (section 7.2) showed that the prototype LIMPT system met all the high-priority and most of the middle-priorityspecifications and was flexible and highly available. A comparative evaluation (section 7.3) against currently available tools using in digital performance showed that, while the LIMPT system had very little overlap of functionality with most, there were a few types of tools (performance resource development systems and data routing and mapping applications) that did provide at least some of the functionality offered by the prototype LIMPT system. However, none of them provided the same breadth of functionality, particularly when scalability and extensibility were taken into account.

Taken together, the three evaluations suggest that while further testing (particularly longer-term user testing) would be desirable, the LIMPT system does seem capable of facilitating the generic aspects of practitioners' work with interactive multimedia in live performance.

[203] The small number of participants meant that data derived from the survey could not be considered reliable on its own and needs to be considered in conjunction with findings from the other evaluations.

8.2 Contributions to Knowledge

The first of the original contributions to knowledge provided by this study is an account of practice and practitioners across the field of digital performance characterised not by differences, but by underlying similarities in work that might appear highly heterogeneous if the perceived, surface phenomenology is considered alone. This approach is entirely novel and provides access to a range of critical and evaluative strategies which have not previously been available. Further, while several authors (e.g. Dixon 2007, Chatzichristodoulou 2009) have sought to establish the identity and boundaries of the digital performance domain, this current study suggests an additional approach to topic definition which might avoid at least some of the pitfalls of categorisation by genre or content and could lead to a more secure basis for establishing the extent of the field.

The analysis of practitioners' accounts of their technology-use has provided new understandings of the basis for the establishment of requirements for software designed for use by creative artists and of appropriate ways to plan and manage software development aimed at serving the needs of this particular group. In particular, it is clear that where software is designed to enable a range of work rather than meeting a specific, limited need, asking potential users 'what they would like' is not a valid approach in a highly differentiated field of

activity[204] and that approaches based on analysis of artists' working-practices and attitudes towards their work and technology are required for success.

The study has also produced a highly innovative model of live performance that is able to account for an extremely wide range of practices and materials through focussing on the formal operation of performer activity. While the model has been used in the current study as a basis for the incorporation of interactive multimedia with live performance, it offers a potentially new and powerful way for critics and scholars to (re)consider performance in general and to examine specific performances. By positioning conditional connectivity as an underlying structural device, one can propose new approaches to analysis and meaning based on taxonomies of connection and qualitative evaluation of performer-choice.

By presenting creative practitioners with the opportunity of considering all sensed activity as potentially significant and capable of establishing conditional networks of connection between all elements involved in a performance, the *eMerge* language and LIMPT implementation provide access to a range of possibilities for devising and controlling performance which have not been facilitated to the same extent by previously available resources. Existing limitations may have been those of availability, requirements for specialist knowledge, scalability or flexibility as well as those of functionality; before even using the system, the constraints (explicit and implicit) imposed on creative

[204] See deLahunta 2002a. I'd suggest that one of the reasons for the relative lack of success of the *Software for Dancers* project was the methodology of defining requirements.

thought and experimentation by currently-used resources can be questioned if not entirely thrown off. A process within which novel possibilities for individual practice can be imagined will have begun.

8.3 Suggestions for Future Work

The enquiry process that formed the basis of this thesis has suggested a number of interesting possibilities for future work. The set of interviews which produced much of the data on which subsequent work has been based took practitioners from the three main performing arts traditions. However, some authorities (e.g. Dixon 2007, Digital Stages Festival 2011)place those interactive installations in which the audience take on some of the roles associated with conventional performancewithin the domain of digital performance. Further interviews with subjects drawn both from the disciplines already represented and from others would serve to validate the reliability of the original results and extend the coverage of the study with the potential for new insights into interdisciplinary working.

The evaluation of the prototype LIMPT system against the statement of requirements identified a number of medium priority functional targets which could inform a route map for further system development (see Figure 59).

Target	**Rationale**	**Priority**
Client 'Command' view	A client user interface view enabling the direct issuing of commands to means they could be issued in performances for real-time control. The feature is already supported by the Central Controller.	3
Asset management	The system needs to provide support performance-makers in managing any assets (images, etc.) that are used in projects.	2
User management	The system needs to control access to projects and rule-sets using a simple, but robust permissions scheme.	3

OSC event in/out	OSC is almost as widely used as MIDI in some sectors of digital performance and the ability to use it for input and output would greatly increase the connectivity of the LIMPT system. OSC connection has been successfully demonstrated in installation work (see section 4.2.3).	1
Enhanced MIDI support	Extending the range of MIDI events which are supported for input and output would increase connectivity options; the language already supports an increased range.	3
Improved client stability	The client sometimes crashes when users switch views. This needs to be investigated and solved.	1

Figure 59 – Further system development goals

In the longer term, future development of the language and system would seek to implement ways of passing dynamic values to performance elements.In working with users, a desire to be able to pass values belonging to position and time objects has been identified, so that the command, ‘show performance time to clockDisplay’ would check the current performance time and send that value when it was executed (either in response to a rule being triggered or as a direct command).

Another language feature that has been asked for is a way of using the value of an event which triggers a rule in the command(s) of that rule (see Figure 60). The major difficulty is implementing this in software rather than the language-design (the keyword ‘it’ has been reserved).This would enable passing values from one performer or device to another, allowing complex and subtle integrations and enhancements of performance possibilities while still retaining the close relationship to the ways performers and producers talk and think about performance activity in their current practice.

if
nick gives a noteOn midi signal
then
give**it** to isadoraDisplay

Figure 60 –Passing data from event to command

Beyond the developmental goals discussed above, the most important feature of future work with the LIMPT system needs to be getting the system adopted by other practitioners and to begin feeding back their impressions, gained over extended use, into the development process. While the adoption of the system would not have been an appropriate metric to make judgements about the success of the original research project, it is very clear that widespread and enthusiastic adoption by artists is crucially important in building an effective and supportive user-community for a system. Future work would need to make establishing and supporting such a community, through continuing reliable provision of the Central Controller service and the widespread dissemination of code and results, its highest priority.

8.4 Dissemination of Results

While the responsibilities of a researcher to the wider research community make dissemination of results an important part of any enquiry process, ‘spreading the word’ is particularly important for a project that requires the participation of others if it is to thrive. While papers have already been published discussing some aspects of this research, it is envisaged that others, exploring aspects of the data that have not been featured in this thesis, such as the influence of technology on working patterns and practitioners’ conceptualisation of technology, will form the basis for future academic work.The interviews conducted for this project were, because of the nature of the field, covered by confidentiality agreements. In order that artists felt able to be candid about their practice (and so produced more reliable data), it was agreed the complete interviews would not appear in this thesis. However, each subject was supplied with a transcript and at least one subject now plans to publish an edited version of the interview.

The early stages of this research included the programming of a relational database system linking practitioners, their projects and venues with the technical resources used for each digital performance. While the database did not produce results that were used in the main project itself, it has grown to become a valuable resource in its own right, particularly for practitioners who want to find out what resources are available for specific purposes. A project involving postgraduate students from a number of UK universities is planned to

develop this resource further and increase its profile among practitioners at the start of their careers.

The code produced for the project will, with the exception of those parts covered by commercial licences, be released under an Open Source licence and the repository URL advertised[205]. Part of the advantage of establishing a user-group would be the sharing of responsibility for code integrity and documentation as these are areas that, moving forwards, would need greater emphasis as the number of participating contributors grew.

[205] This might involve moving the code to an Open Source group development repository such as Source Forge (http://sourceforge.net).

8.5 Chapter Summary,Concluding Remarks

This chapter began by revisiting the initial impetus for the enquiry and then showed how, through carefully structured process, successive goals and outcomes were successfully realised. This thesis sets out the justifications and evidence for asserting that there are generic features of work across the wide range of current practice in live performance involving interactive technology, and that it is possible to identify these and establish the features of a system that would support these common features of practice. From meeting these interim goals, the enquiry was then able to develop a prototype system which, it has been shown, is liked by users and which provides all the most important required features. Further, the specific mix of functionality, usability and scalability is not provided by any other currently available resource.

It was suggested in the first chapter of this thesis (section 1.4) that, while the achievement of the research goals is the yardstick for making judgements about the enquiry so far, it will be the development of understanding, the stimulation of interest and the production of exciting, innovative performances that incorporate interactive multimedia in ways that have not even been considered yet that are the real long-term benchmark of success. The research described in this thesis has the capacity to stimulate interest in, and renew critical and scholarly attention on, the diverse range of practice in digital performance. If, in addition to this, the use by creative artists of the LIMPT system (or other resources that encompass its feature-set) facilitates those areas of practice that are common, it seems reasonable to suggest that this could perhaps leave practitioners able

to devote more of their creative energies to those radical elements of digital performance that embody the unique and characterful features of their individual artistic practice, to participate more strongly in that movement of transformative cultural activity identified by Steve Dixon:

> “The calls for artistic revolution made by the futurists, the surrealists, and the constructivists of the past palpably rematerialized. They echoed, phantomlike, around the virtual walls of the new cybertheater – and not only were the voices heard, they were also acted upon.” (Dixon 2007:662)