Software Engineering Issues in Interactive Installation Art

Anna Trifonova
Department of Computer and Information Science
Norwegian University of Science and Technology
Sem Saelands vei 7-9
7491 Trondheim, Norway
trifonova@idi.ntnu.no

Letizia Jaccheri
Department of Computer and Information Science
Norwegian University of Science and Technology
Sem Saelands vei 7-9
7491 Trondheim, Norway
letizia@idi.ntnu.no

ABSTRACT
Software engineering has been in contact with new media art for years, although the connections between the two fields have rarely been explicit. In this article we discuss the important software engineering issues that appear in one of the new media art subfields, namely interactive installation art. Our deductions and suggestions are based mainly on reports available in the literature (i.e. published papers). Interactive installation art is often heavily dependent on software and thus software engineering issues are important to consider. Software requirements, which are vague and frequently changeable, appear to be one of the major and most difficult issues to be considered in the development of interactive installation. Timely evaluation, validation and testing with potential users are helpful for successful completion of the artwork Special attention should be paid to the choice of process model and software architecture to allow flexibility. The final goal is to provide a road map for artists who need software engineering skills to communicate with software engineers and/or to act themselves as programmers or software engineers of their artworks. Additionally, software engineers who start working with interactive installation art will profit from this summary of relevant reports.

Categories and Subject Descriptors
D.2.0 [**Software Engineering**]: General; J.5. [**Arts and Humanities**]: Performing Arts

General Terms
Management, Documentation, Design, Human Factors.

Keywords
Software Engineering; New Media Art; Interactive Installation Art; Development of Interactive Installations.

Submitted for DIS 2008, February 25–27, 2008, Cape Town, South Africa.

1. INTRODUCTION
Computer art dates back to the 60s. The first computer art exhibition took place at Technische Hochschule in Stuttgart in 1965. The same year at the Howard Wise Gallery in New York City the earliest computer art exhibition took place in the United States [1].

The first software engineering conference was held in Garmish in 1968 [2]. Software engineering and computer art, blooming at the same time, have met several times, even if these relationships may not have been rendered explicit.

The use of digital technology in contemporary art is often referred to as new media art. Since the early 90s within the New Media Art realm there is a growing production of interactive art installations[1]. These artworks are generally complex and they are heavily dependent on software for controlling the whole system. The production of the software often needs the involvement of programmers and software engineers.

As in other countries, interactive installations are often made in Norway (several examples might be seen in [3]). Within the Software Engineering group of the Norwegian University of Science and Technology (NTNU) we have started a project, called SArt (http://prosjekt.idi.ntnu.no/sart/). The members of SArt participate in multidisciplinary projects for the development of interactive installations that involve collaboration between software engineers and artists. These projects motivate us for a profound investigation in this domain and our goal is to find, expose and bridge possible gaps between the two fields.

This paper contains several contributions. On one hand, artists will find useful the list of software engineering issues with their descriptions. This knowledge will help their work in projects with software intensive systems. On the other hand, software engineers in interactive installation art projects will profit from practitioners' experience reported in the literature and summarized here. This includes a list of utilized tools.

[1] ARS Electronica - Festival for Art, Technology and Society – is one of the most popular annual events on art and technology. As part of it, since 1987 prices are given to the best artworks in different categories (Prix Ars). The evolvement of these categories shows also the changes in the trends in the domain (see www.aec.at/en/prix/). Interactive art, including installations, is one of the categories. It first appeared in 1990.

Further this article is organized as follows: in section 2 we shortly discuss interactive installation art field and some of its specifics, namely multidisciplinarity and interactivity. In section 3 we synthesize the major software engineering concepts and show in section 4 how they apply in the production of interactive installations. Section 5 presents a summary of the software engineering issues in installation art projects and shows different approaches found in the literature. A discussion (section 6) is followed by conclusions (7) and references.

2. NEW MEDIA ART AND INTERACTIVE INSTALLATION ART

New media art is a subclass of contemporary art that involves the use of new media technology. In this work we do not try to give an exact definition of new media art, we do not limit what it includes, and do not intend to define how it relates to other art domain. Instead, we point to the work of new media art theoreticians like Manovich [4] and Tribe et al. [5] and reference the practitioners. For example, Biswas and Singh [6] in their article describing the software engineering issues in two case studies define new media art as "type of new media application where an artistic idea is expressed using technology or new media artifacts".

As software engineers, we have to be critical to the notion of new media as what used to be new in the early 90´s, for example the HTML language and web browsers, is now main stream technology. Web 2.0 which can be regarded as new at the time of writing will not be new in a couple of years from now. Manovich gives an explanation of what artists intend as new in the New Media: "... new media today can be understood as the mix between older cultural conventions for data representation, access and manipulation and newer conventions of data representation, access and manipulation. The "old" data are representations of visual reality and human experience, i.e., images, text-based and audio-visual narratives – what we normally understand by "culture." The "new" data is numerical data." [7].

Installation art is a phenomenon which starts in the late 30s, for example with artists like Duchamp. Interactive installations are the evolution of installation art and are a part of New Media Art, because of "their origins in, and reliance upon, computer-based technology" [8]. An interactive installation includes a physical construction which is generally placed in a public space. Usually certain parts of the installation are changing in time (e.g. video, audio, mechanical parts movement, etc.). Often these changes are due to spectator(s) presence and/or actions. The creation of an interactive installation commonly requires specialists with different areas of competence to collaborate with the artist[2]. A multidisciplinary team can involve artists (painters, composers, etc.), constructors, hardware designers, electrical engineers, software engineers, programmers, art curators, etc.

[2] We often talk about the artist and/or developer (in singular) for simplicity, although there might be cases where a group of artists/developers (two or more) are working together on the same artwork, either simultaneously on the whole work or on different parts of it (e.g. one artist on the music components, another on the visualisation; one software engineer on the software architecture and another on implementation, etc.)

Interactivity is a major issue in interactive installations. Different interactivity types might be defined. In fact, interaction is extensively discussed in [almost] all articles describing interactive installations.

Hannington and Reed [9] discuss three distinguished types of interaction in multimedia applications: *passive* interaction is where the content has a linear presentation and users interact by only starting and stopping the presentation; *interactive* is when users are allowed to choose a personal path through the content; *adaptive* is the interaction in which users are able to "enter their own content and control how it is used".

In [10] Sommerer and Mignonneau discuss two types of interaction that they have observed in existing interactive artworks: *pre-designed* or *pre-programmed* paths of interaction, as in interactive CDs where the viewer can choose his/her path, but the possibilities are limited; and *evolutionary*[3] interaction in which the artwork's processes are linked to interaction and is evolving continuously.

Edmonds et al. [11] discuss four categories of "relationship between the artwork, artist, viewer and environment: *static*, *dynamic-passive*, *dynamic-interactive* and *dynamic interactive (varying)*. While the first is a lack of interaction, the latter three describe situations in which the artwork responds to its context. In *dynamic-passive* the artwork response is triggered by environmental factors as temperature, humidity, etc. In *dynamic-interactive* in addition to the environmental factor the human presence and/or actions (purposeful or not) are captured and are used as parameters for changing the artwork. The rules about how the parameters are treated are static in this case. When an agent (either human or program) is modifying it's original specifications the artwork is *dynamic interactive (varying).*

We find these three categorizations of interaction important, but also each one is incomplete. In our viewpoint there are three perspectives to be taken into consideration:

- Interaction Rules – the rules that control the interaction might be static or dynamic. In [11] this difference is shown by introducing the *dynamic interactive (varying)* category. The *evolutionary* interaction in [10] is also based on dynamic interaction rules, but limits the rules to evolutionary algorithms.
- Triggering parameters – The interaction rules generally depend on environment parameters that are changed by the audience. Most often the audience is directly participating in the interaction intentionally, but it is possible that no intention is required and only the spectators' presence is enough to trigger the interaction rules. However, in some cases the changes in the artwork might not depend on the audience at all, but only on the environment – such option is foreseen only in [11] with the *dynamic-passive* interaction category.

[3] The term 'evolutionary' is used by the authors of the cited article as reference to evolutionary image processes, as their works are bio-inspired.

- Content origin – weather the artwork presents visual or audio content to the spectators this content might be dynamically generated or predefined by the artist. The predefined content might also be dynamically manipulated. In particular cases the audience might also input content to the artwork, for example by sending pictures/music from their phones. Such option is only seen in [9] - category *adaptive*.

3. SOFTWARE ENGINEERING

Here we make an attempt to synthesize software engineering concepts [12, 13] for the purpose of defining the intersection between software engineering and interactive installation art. In other words, this list points to software engineering theories that one can use to reflect about new media art. We have developed this list of concepts by looking at the topics of the latest International Conference of Software Engineering (ICSE 2007[4]) and we have modified it supported by our experience and discussions with colleagues. These concepts are:

1. Requirements – software requirements are the real-world goals, needed functionality and constraints for the software to be developed. The process of software requirements engineering includes identifying the stakeholders and their needs and documenting these for analysis and implementation. For a software development project to be successful, the software engineers and the client have to agree on the requirements to be implemented.
2. Software Architecture and Design – software architecture is a description of the high-level design of a system (i.e. the product), its main parts and their relations and interactions. Software design is the process of making and analyzing such architectures.
3. Evaluation, Validation and Testing - Validation (of both process and product) means showing that a delivered product/system satisfies the user's real or future needs. Testing is the controlled execution of program code. There are different levels of testing: unit, module, subsystem, system, acceptance etc. to check that actual execution with given inputs produces the expected results.
4. Process Models and Project Management – process models describe the activities with ordering and compositional relations, artifacts being produced or consumed by such activities, human work roles, what tools/techniques to use, and possibly what measurements to apply (i.e. a formalization of the software development process - e.g. agile). Project management deals with planning and control of execution process model, including schedulers, budget, etc.
5. Development environments and tools (e.g. Eclipse) are programs used by software engineers and programmers as aid for the software design and implementation, synonyms are CASE – computer aided software engineering, or IPSE – integrated process support environment.
6. Maintenance (software maintenance) - Further development of a software product after its first release, also usually organized as a project. 2/3 of total software costs may fall on software maintenance. We distinguish between perfective (new or revised requirements), adaptive (new technologies/platform), corrective (fixing faults) and preventive maintenance (internal reorganization) – e.g. with relative distribution 50%, 25%, 20%, and 5%. Reuse is a way of software development that includes systematic activities for creation and later incorporation ("reuse") of common, domain-specific artifacts. Reuse may have profound technological, practical, economic, and legal obstacles – but the benefits may be huge. It mostly concerns program artifacts in the form of components, see below. Standard use of platform components – i.e. commodities like OS, DBMS, Internet netware, GUI etc. – are normally not called reuse.
7. Open source software – Software for which the access to the source code is open, the distribution and re-distribution is global, free, the licenses are non-restrictive, etc. Traditionally, open source software has been developed by interested communities and software users and developers have been volunteers. In the recent years, large and small enterprises (among the large ones we mention IBM and Sun Microsystem) are exploiting business model centered around production, customization, and service of OSS products.
8. Quality Attributes (performance, reliability, security, safety, etc.) Performance is the measuring of the speed or volume offered by a service, e.g. delay/transmission time for data communication, storage capacity in a database, image resolution on a screen, or sound quality over a telephone line. Reliability is the probability of failure-free behaviour (vs. stated requirements), in a specific context (executing environment and usage profile) and time period. Security is the level of protection against unauthorized access (e.g. read / write / search) of data / information. Safety is the degree of protection against dangerous events, i.e. events with possible serious consequences for humans, environment, business, and/or society.

4. INTERACTIVE INSTALLATIONS AS SOFTWARE ENGINEERING PRODUCTS

In 2005 Briony Oates [14] proposes to extend IS research in the domain of computer art. The first suggestion is that “computer art might be seen as a kind of information system”. In this section we extend this idea and discuss more concretely the mapping between an interactive installation and a software engineering product. Schematically, a software engineering product might be presented as a black box that receives a digital input, processes it and the result is outputted to the user. Interactive installations might be mapped to this schema, as they receive certain input that is digitally processed and the output is given back to the audience.

[4] ICSE 2007 took place in Minneapolis, 19-27 May 2007 http://web4.cs.ucl.ac.uk/icse07/

Figure 1: Interactive Installation as Information System.

Machine [15] states that the technology for controlling the interactive installation artwork in general does not differ from the technology used for controlling industrial machines; what differs is that the artists require the technology to be more accessible, so that they can experiment while creating the artwork.

We would like to extend this idea by adding the details around - by describing the processes involved, the stakeholders and their roles and the tools used.

Figure 2: Software Development.

4.1 Product

The final product in the creation of an interactive art installation is the artwork as a whole. This includes its hardware and its software and is physically placed as desired by the artist in its context (e.g. public spaces, galleries, etc.).

In software engineering terms, however, the product is the software that controls the interactive installation. The software system takes care of the input (e.g. data from motion detectors, light sensors, images from video cameras, content sent by the audience, etc.), applies certain digital processing and gives the output to the audience. Additional products, like documentation, user manuals, web-site, supplementary software tools, etc. might also be expected.

4.2 Roles (Stakeholders)

A project that aims to develop and interactive art installation is often multidisciplinary, as it involves production of physical and software components. The stakeholders of such installation include the artist, the software designer and developer, the hardware designer and developer, etc. but also the final audience/spectators.

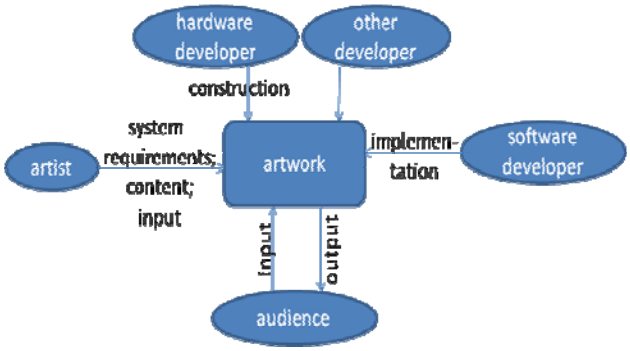

Figure 3: Interactive Installation Art Stakeholders.

The artist has the key role in the project. He/she comes with the idea of the whole system. The artist might have a global view of what message the artwork should send to the audience or what reaction it should trigger. It is possible, however, that the artist goal is to experiment with certain technology without defined in advance goal or message for the audience. The artist might be seen as a client to the software engineering team. He/she has to agree on the software requirements, the final product quality attributes, projects scheduler, etc.

Software and hardware engineers have the mission to convert the artist's desires and visions about the artwork into formal requirements that are later implemented into the final product within the time and the budget available. Depending on the requirements and the technology involved software engineers propose appropriate process model, software architecture and tools for implementation.

The audience/spectators participate at the final stage, when the whole system is ready, integrated and put in place. The audience influences of the artwork by its presence or by its actions and in this way the spectators are becoming part of art.

4.3 Tools

As mentioned in section 3(5), software engineering tools are referencing to development environments – software that aids in the design and the implementation of the final product or in the management of the software project. Many different CASE tools are available commercially or free of charge and practitioners choose the most suitable ones for their task, thus different tools might be used by people with different roles in a project.

In some cases the final product of a project might be a tool. For example, the software development team might implement a software system that the artist will use for experimenting with the artwork design or for implementing and controlling the interaction.

5. SOFTWARE ENGINEERING ISSUES IN INTERACTIVE INSTALLATION ART

In this section we show a synthesis of 12 papers describing 23 interactive installations (see Table 2). These articles were found within a larger study – a systematic literature review on the intersection between software engineering and art. With some exceptions (i.e. some articles were pointed to us by colleagues) the articles were found by searching IEEE Xplorer, ACM Digital Library, Google and the NTNU library meta-search engine with a combination of the keywords "art installation", "software engineering", "artist and software". More information about the complete survey process and initial results is available in [16].

5.1 Requirements

During the development of a software product, requirements definition is a task of major importance. It starts at an early stage of the project and might continue to evolve, update, increase or change throughout the near end of the project. Highly appreciated by software engineers are projects where the agreement on software requirements is reached with the client at the initial stage and they are locked against changes. Changes in requirements, especially in later stages, might lead to drastic changes in the software architecture design and implementation and thus might cause increase in project cost and delays in the scheduler.

Table 1: Interactive Installations Found in our Literature Survey.

Installation	Exhibited (where and when)	Supported by	Ref	Artist	Scientists
Locative sound-scape	Park Emile Gamelin, Montreal	Concordia University; Hexagram: Institute for Research/Creation in Media Arts and Technology; etc.	[6]		Amitava Biswas, Jagmit Singh
15 seconds of fame	Exhibited several times, first at the 8th International Festival of Computer Arts, 28 May–1 June 2002, Maribor, Slovenia		[17]	Franc Solina	Franc Solina and four Master Students in CS, Slovenia
A-Volve	Multiple exhibitions 1994-2007, first at Ars Electronica '94	ICC-NTT Japan, and NCSA, Urbana IL, U.S.A.	[10]	Christa Sommerer, Laurent Mignonneau	Christa Sommerer, Laurent Mignonneau, Tom Ray (A-Life scientist)
Books of sand	Several exhibitions, including Daum Museum of Contemporary Art. Missouri. USA and Museum of Modern Art of Buenos Aires. MAMbA (2002-2003)	UCLA Theater, Film and TV Department (residency program), Univeristy of Tres de Febrero and Linda Lighton Foundation of Kansas City	[18]	Mariano Sardón	Laurence Bender
Gender Specific	Santa Monica Museum of Art, and Bliss House, Pasadena, California. November 11 - 18, 1989. (Simultaneous one-person exhibitions) and Part of the *LA Freewaves* video festival.	Sponsored by the Foundation for Art Resources.	[8]	Jennifer Steinkamp	
GENMA (Genetic Manipulator)	Ars Electronica Center (AEC) in Linz, Austria, as part of a permanent exhibition, 1996; extended several times	ATR Media Integration and Communications Research Lab, Kyoto Japan	[10]	Christa Sommerer Laurent Mignonneau	Laurent Mignonneau
Iamascope	Several times since 1998, including Play Zone, Millenium Dome 2000, Video Construct, Kyoto Two		[11]	Sidney Fells	Sidney Fells
Interactive plant growing	Permanent collection of the Media Museum at the ZKM Karlsruhe, Germany, 1997		[10]	Christa Sommerer, Laurent Mignonneau	Christa Sommerer, Laurent Mignonneau
Intro Act	Biennale de Lyon at the Musée d'Art Contemporain in Lyon, France, as part of the museum's collection, 1996/97		[10]	Christa Sommerer, Laurent Mignonneau	Laurent Mignonneau
Memichi	Banff national park, Alberta, Canada		[6]		A. Biswas
MIC Exploration Space	ATR Media Integration and Communication Systems Laboratories in Kyoto, Japan, 1996.		[10]	Christa Sommerer, Laurent Mignonneau	Christa Sommerer, Laurent Mignonneau
Nautilus		VTT Information Technology, Cube Oy, Nokia Research Center, Särkänniemi Adventure	[19]		Hanna Strömberg Antti Väätänen Veli-Pekka Räty

		Park, Tekes, the National Technology Agency and the University of Lapland, Finland			
Phototropy	Shiroishi, Japan 1998	Artifices, Saint-Denis, France.	[10]	Christa Sommerer, Laurent Mignonneau	Laurent Mignonneau
Priva-Lite Panel Construction "Digital Garden"	Outdoors, 1998	COSTART project and Gallery of the Future, Loughborough, UK	[15]	Esther Rolinson	Colin Machin
Remote furniture	Exhibitions in Yokohama, Queens mall, Aug 1999; Tokyo, Ginza subway station, Aug 1999		[20]	Fujimura, Noriyuki	-
Stiffs	Art Center College of Design's Williamson Gallery, 2000	ACME Gallery, Los Angeles	[8]	Jennifer Steinkamp in collaboration with Jimmy Johnson (soundtrack)	
SwarmArt	Two installations for a Calgary gallery, 2002-2005		[21]	Gerald Hushlak	Jeffrey E. Boyd, Christian J. Jacob
SWELL	Several exhibitions until 2005, first ACME, Santa Monica, California. (One-person exhibition) November 10 - December 8, 1995	The Museum of Contemporary Art, Los Angeles, California with funds provided by the Ruth and Jake Bloom Young Artist Fund	[8]	Jennifer Steinkamp in collaboration with Bryan Brown (soundtrack)	
Swimming across the Pacific			[22]	Alzek Misheff, Fels S.	Fels S., Kinoshita Y., etc.
The TV Room	Santa Monica Museum of Art, 1998	ACME., Los Angeles	[8]	Jennifer Steinkamp in collaboration with Andrew Bucksbarg (soundtrack)	
Trans Plant	Permanent collection of the Tokyo Metropolitan Museum of Photography, Tokyo, Japan, 1995-1998	Advanced Telecommunications Research (ATR) Laboratories, Japan.	[10]	Christa Sommerer, Laurent Mignonneau	Christa Sommerer Laurent Mignonneau
Trigger	Pace University Digital Gallery, New York, NY, October 18 - November 8, 2005		[23]	Jody Zellen	scientists from Pace University's Center for Advanced Media (CAM)
Untitled	Several exhibitions between 1993 and 2006, first at FOOD HOUSE, Santa Monica, California, 1993		[8]	Jennifer Steinkamp	

As in many other fields, requirements definition is one of the most difficult tasks for the software developers in the interactive installation projects - "It is the most important part of the process because without a precise understanding of the system requirements it is possible to build a well functioning system that does not perform the tasks requested by the user" [23]. However, requirements are often found difficult to capture. Machine [15] underlines that requirements definition is the hardest part and states that "we find the greatest challenges in even identifying what the artist requires". The author emphasizes that the requirements by the artist might change repeatedly until he/she is satisfied. Similarly, Marchese reports changes in requirements during the implementation of Trigger - "the system design was updated to reflect experiments with different types of sensors" [23]. Biswas and Singh [6] share their experience from two installation projects, stating that often "the emergent system specifications cannot be defined in sufficiently tangible terms till the very end of the project", especially because they might be very vague at the beginning. The reason for that might be due to the different working style of the artist – more exploratory rather than rationally planned and with explicit goals, as it generally is in business domain.

As earlier stated, the installations that we examine are most often highly interactive. What differs from the common interaction in software systems is the final goal. In information systems the goal is to increase the effectiveness and the efficiency in the correct completion of a concrete task (or set of tasks). On the other hand, "humor and play are important aspects of the art" [8]. Additionally, often “the system usage context is entirely absent or it is not well understood” [6]. Hannington and Reed [9] state that the difficulties in “capturing human activities in a manner that is sufficiently informal for non-programmers to understand, yet sufficiently precise for developers to use as a specification” are stronger in multimedia domain than in other domains. Interactive installations are often used by a large number and variety of spectators – adults and kids, people with different education and knowledge, men and women form different nationalities. Thus, “requirement elicitation should encompass sufficiently large variety of usage situations.” [6]

It is important that both software developers and artists are aware of these properties of the requirements. Requirements might be difficult to capture, vague at the beginning and frequently changeable. Having this in mind will allow choosing the most appropriate software development methods, designing the most suitable architecture of the product, good risk assessment and proper planning of budget and schedule.

The literature review shows also that the software developers have to be an active side in the requirements definition when working with artists. In many cases artists have clear ideas of what they want the final effect of the artwork on the audience to be. They might have also decided on what technology they want to explore. However, they might not be aware of the full potential of this technology and how it might influence on what the system will do. They expect suggestions and proposals from the technologists on what the technology allows. These ideas would not be directly applied, but would provoke/inspire the artist’s creativity and will be put together with his/her ideas and goals for the final artifact – the artwork.

5.2 Software Architecture and Design

The software architecture depends on the functionalities which should be provided by the system, on the technology chosen, on designers’ preferred styles, etc. Thus, the software architecture will most probably differ from one project to another.

For several of the interactive installations the software architecture is reported. Marchese [23] describes a simple architecture with 3 components - microcontroller-sensor system, application software, and an interface software between sensors and the application with "high level interrelationships among components without specifying the processing details". Boyd et al. [21] report a pipeline architecture where “the output of a module can provide input to one or more other modules in a pipeline”. Several standard software (i.e. previously available software not developed by the authors) were combined, including the software for simulating a swarm and a video interaction server widely used for surveillance tasks. The pipeline is made dynamically configurable through a graphical interface.

The software behind the interactive installation “Swimming across the Pacific” [22] has also a modular architecture. The use of object-oriented software engineering methods is reported in [19]. Machin in [15] describes an especially designed simulator and a specific language that allows the artist to easily experiment with the installation design and several supplementary tasks (e.g. calculation of the overall cost which depends on the changes of the materials used and their quantity). Similarly, Biswas and Singh [6] have developed a Mobile Experience Engine (MME) which helps in the simulation of the final artwork. Their system contains two parts – a visualizer that generates low-fidelity prototype and a code generator that generates high-fidelity prototype with optimized implementation for several interaction devices. The authors find that most suitable is to “wisely splitting the application architecture into two parts, one dealing with interactivity, the other tackling core functionality”. Finally, Edmonds et al. state that “In art and technology environments, we need environments for building environments” [11].

The observation on the published work on interactive installation art shows that whenever possible the development teams tend to us software that is already available (reuse). This decreases the overall effort for implementation and the final price of the software. However, in most of the cases the standard components have to be integrated into the full system and custom parts have to be implemented.

Although artists often have much more profound technological knowledge then expected from clients in software engineering projects, they often would like to have the freedom of experimenting with all technological possibilities by themselves even in cases when their experience is not enough. Adding an extra layer between the artist and the underlying programming fosters the artwork creation without limiting the artist’s creativity.

5.3 Evaluation, Validation and Testing

Software products are usually checked for their correctness during execution (i.e. testing), for satisfying the requirements specifications (i.e. validation) and on how well the end-product satisfies the user expectations (i.e. evaluation).

The testing might be done automatically by using other software that executes [pieces of] the system with various parameters and controls the correctness of the outputs. It might be also done manually by the developers and/or users, which is commonly done in small projects. For example, [23] reports such manual approach for the integration testing of the interactive installation Trigger - “The developers systematically walked through the space triggering all video and sound sequences”, thus simultaneously testing the hardware (e.g. sensors) and the software of the artwork. This, together with the validation was done on-site when the installation was mounted in the gallery several days before opening - “Multiple walkthroughs of the installation by the artist before the opening constituted the final acceptance test of the system”.

Strömberg et al. in [19] report the use of heuristic expert evaluation for the technical aspects of the system. Multiple (iterative) evaluations were performed in different phases of the design and implementation with participants that were considered potential final users. The evaluation was done by observation, interviews and open-ended questionnaires and in earlier stages the feedback was used for design improvements. Fells et al. [22] evaluated their artwork with the visitors of Siggraph 2004 exhibition, collecting opinions, positive and negative experience, comments and suggestions.

The evaluation against the final user utilization might be essential for creating the user-system interaction as it is planned and

expected by the artist. For example, Steinkamp [8] reports that "children immediately understand that they are expected to play in the projection". On the other hand, adults were examining and analyzing the system instead of actively interacting with it. This was not exactly the desired by the artist behavior/effect, but it was noticed only when observing the audience during the exhibition.

While Hannington and Reed [9] affirm that "most multimedia process models advocate use of strict evaluation and revision within the iterative cycles of development" this might not be possible in all cases due to budget or other limitations. Furthermore, clear distinction should be done between testing and evaluation, as "testing ensures correct technical operation of the system, but it does not ensure its appropriateness or its effectiveness in delivering the expectations" [6]. Nevertheless, in some cases the testing of certain system parameters might be done in real-world situation – for example the robustness of the innovative face-detection software behind the "15 seconds of fame" [17] was tested during the exhibition and based on participants evaluation the authors judged the algorithm as reliable and effective.

Summing up, the development team in interactive installation art project should consider evaluating the artwork and especially the interaction as early as possible with final users, as the effect might not be as expected by the artist. Different users should be considered - culture, gender, age, etc. Some of the quality attributes, like reliability, robustness, etc. might be tested in real environment during exhibition.

5.4 Process Models and Project Management

The software development model is a formalization of the activities and the modes in which the software development is organized. Generally this is part of the policy which the software development company has incorporated and is valid for all the projects within the company. Many interactive installations are developed during artists-in-residence programs and the software development process might be influenced by the hosting institution practices.

Agile method of software development (i.e. Adaptive Software Development) was chosen at the beginning of the project described in [23] and was evaluated as a good choice by the software engineers. The developers predicted the possibility of vague requirements which would change frequently. The chosen method was suitable also because it dealt well with the strict schedulers of both artist and technologists and with budget limitations.

Biswas and Singh [6] discuss several possible software development methods and their advantages and disadvantages for interactive art projects. For example they state that "Creative artist's work processes do not necessarily follow "analyze-model-design-build" trajectories like engineers. They [artists] iteratively and intuitively generate creative ideas and evolve their design based on their perception and experience". More suitable from the traditional software engineering approaches is found to be the "evolutionary prototyping" in which "artists will generate creative ideas, technologists will receive briefing from artists, build prototypes, elicit modifications/corrections and further requirements from the artists and this cycle will be repeated till the artist is progressively satisfied". However, this was far from the perfect model, as "system developed by evolutionary prototyping may suffer from lack of coherency in its architecture due to inadequate planning". The authors build upon this model and enhance it with low-fidelity and high-fidelity prototype generation. Biswas and Singh also suggest the need of further investigation of other methods, like user assisted prototyping, participatory prototyping and prototyping combined with usability studies.

While in the previously discussed cases the development of the artwork was rather isolated from the final users, in other projects different approach has been chosen. Human-centered design has been used in [19]. The authors report that the users were not only participating in the final evaluation, but were "essential part of the design process from the early stages". This process is iterative and the necessary changes were made according the feedback from the users. The application functionalities were designed by the artists in the form of scenarios and storyboards that were used by the software designers for deriving an object-oriented architecture.

Considering the whole project management several issues should be mentioned. In interactive installation art often the creation of the artwork is sponsored by a public entity which does not influence on the artist's creativity. However, the budgets are tight and often relatively small. In many cases the work is created during artists-in-residence programs and it is possible that some members of the team have additional work duties. Commonly, there is an opening day for the artwork, thus the final deadline for the full system might not be flexible. Apart from budgeting and scheduling probably most important issue in the project management is the risk assessment. According to [23] "Any component of the system or member of the project could pose a risk". Artists often explore and incorporate in their interactive art installations one or more new/emergent technologies (e.g. sensors, location awareness, mobile and wireless, etc.). Together with mostly limited budget, this leads to the high probability that the technologists will not be well acquainted with the necessary skills and need time to learn. The newest technology might also be problematic in terms of instability, lack of documentation and support materials and communities.

5.5 Development Environments and Tools

The development tools used by software developers are not discusses in the reviewed articles. Our guess is that standard CASE tools were utilized (the ones which the technologists are most used to) with no particular advantages or disadvantages for the software development in interactive installation art. Although artists sometimes prefer "access to deeper levels of the computer's programming system" [11] the tools that are suitable for software engineers does not seem to be proper for them. Interestingly, they use tools like Macromedia Flash as CASE tools – for implementing their programs (e.g. [23]), but also for supportive tools, such as for creating the storyboard in [19]. In several cases (e.g. [6, 11, 15]) technologists provide additional software layer for the artists – tools that will give them the freedom to experiment without limiting their creativity. Such tools are found to be well accepted and positively evaluated by artists. In Table 2 we provide a list of the software tools mentioned in the discussed articles, together with additional information and links.

Table 2: Software tools

Software [5]	Description	Ref	URL
3D Studio Max	3D modeling and animation package	[19]	www.autodesk.com/3dsmax/
Breve swarm simulation	A package for building 3D simulations of multi-agent systems and artificial life	[21]	www.spiderland.org/breve/
GigaStudio 160	Software for music and sound effects	[19]	www.tascamgiga.com/
Macromedia Director	Multimedia authoring tool	[18]	www.adobe.com/products/director/
Macromedia Flash	Professional software for creating rich, interactive content for digital, web, and mobile platforms.	[23] [19]	www.macromedia.com/software/flash/
MAX/MSP	A graphical environment for music, audio, and multimedia	[23] [11]	www.cycling74.com/
Maya	3D animation package	[8]	usa.autodesk.com/adsk/servlet/index?siteID=123112&id=7635018
Mobile Bristol toolkit	A tool to create and share mobile, location–based media	[6]	www.mobilebristol.com/
Mobile Experience Engine	A software development platform for creating advanced context-aware applications and media-rich experiences for mobile devices	[6]	www.open-mee.org/
OpenGL	Industry standard and API for high performance graphics	[22]	www.opengl.org/
Particle dynamics	A software tool set for simulating natural phenomena.	[8]	
Pfinder	Advanced camera/gesture tracking software, developed by MIT	[10]	vismod.media.mit.edu/vismod/demos/pfinder/
Sculpture simulator	A sculpture simulator with its own programming language	[15]	
SoftVNS video toolkit	A real time video processing and tracking software for MAX/MSP	[11]	homepage.mac.com/davidrokeby/softVNS.html

[5] The Commercial products are marked in *Italic*. The products in **Bold** are free or open source products. For the rest this information is not available.

6. DISCUSSION AND CONCLUSIONS

Interactive installation art is an active field with many artworks created and exhibited, both in Norway and worldwide. Interactive installations are frequently heavily dependent on software and often software engineers are part of the team in projects for creating interactive installation. During our involvement in several such projects we have asked the question: How much does the Artist need to know about Software Engineering in order to interact with Software Engineers and programmers?

In this article we outline the key software engineering concepts in relation to the development of interactive installation art. This will give an easy start for artists and will facilitate the creation of common language and understanding within the team. We also summarize the software engineering issues and solutions reported in published articles describing interactive installations. This collection of experiences from interactive installation art projects might guide the software engineers and programmers in their future practices and direct their attention to difficult issues in this specific domain. We also provide a list of utilized tools.

It is extremely important that both software developers and artists are aware that in interactive installation art requirements are difficult to capture, vague at the beginning and frequently changeable. Software engineers have to be an active side in the requirements definition. The choice of appropriate software development method and the design of suitable software architecture might capture this specific of the domain and might help for the proper planning of the budget and schedules. Careful risk assessment should be performed, so that to guarantee a successful ending of the project.

Interaction is a key issue in interactive installation art. Sometimes the environmental parameters trigger the interactivity, sometimes only the audience presence is enough to cause changes in the artwork, but in most of the cases the interaction is directly linked to the audience actions. But how to understand the audience and how to predict their behavior to the extent to reach the Artist's desired effect? In section 5 we have shown that different process models might be used to face this issue, like human-centered design. Prototyping is found suitable in order to ensure correct understanding between artist and software developers (i.e. correctness in requirements specification). Prototypes might also foster the evolvement of the artist's creative ideas. Additionally the evaluation of the artwork interactivity might be done in different stages of the project by the expected audience. However, a large variety of spectators (i.e. gender, age, nationality, education, physical capabilities, etc.) might be anticipated.

It should be kept in mind that different stakeholders in interactive installation projects might need and prefer the use different set of tools for their tasks. Software developers are acquainted with CASE tools appropriate for the software design and implementation. These tools, however, might not be appropriate for artists. Artists often use tools like Flash or Max/MSP as CASE tools. They might prefer to create storyboards instead of UML diagrams. In their work artists often adopt more experimental style and might require an additional software layer (e.g. a tool) that will allow them to experiment with the chosen technology without limiting their creativity.

Additional important software engineering issues, like maintenance and open source software, are not mentioned in any

of the reviewed articles. We ask if it depends from the fact that they are not important in interactive installation art or there is some other reason why they are omitted. Our experience implies that they should be important. Artists often continue work on their interactive installations; they evolve and are exhibited in consecutive occasions. Then why software maintenance is not discussed? Our assumption is that this might be related to limited budgets of projects in interactive art installations. As we mention in section 3 up to 2/3 of the software cost might fall on software maintenance.

Furthermore, the reuse of software components is a common target-domain specific issue in software engineering, which might be worth exploring in art. In addition, the open source phenomenon seems a promising area. Open source software is not only a free of charge basis for software projects. It also provides a community support and further development which artists might explore. Further investigation on these questions is needed.

Interdisciplinary research should provide benefits to all disciplines involved. In our case the primary goal is that of providing guidelines for artists that need software engineering knowledge. On the other direction there are at least the following beneficial issues:

1.Education and recruitment of students – students are motivated when working on their art-related projects [24]. They can touch with their hands their work and show it to family and friends.

2.Culture and social - computer scientists and software engineers need to reflect about the nature of our discipline and on our profession The dialogue with artists and the concrete projects and problems are a stimulus to such reflection.

3.Innovation – Artists work in a different way and generate requirements that may result in innovative products and methods (e.g. [25]).

7. REFERENCES

[1] Digital Art Museum, "History", http://www.dam.org/history/index.htm, last visited 26/06/07.

[2] Naur, P. and B. Randell, "Software Engineering: Report of a conference sponsored by the NATO Science Committee", in *Software Engineering*, Garmisch, Germany, 1968, p. 231.

[3] Lysaa, P. A., Y. Sandboe, and B. Kvinnsland, "Intravision art presentation", Intravision Presentation, March, 2006, available online at www.intravision.no/downloads/iv_art_2006.pdf (13/06/07).

[4] Manovich, L., *The Language of New Media (Leonardo Books)*: The MIT Press, 2002.

[5] Tribe, M., R. Jana, and U. Grosenick, *New Media Art (Taschen Basic Art)*: Taschen America, LLC, 2006.

[6] Biswas, A. and J. Singh, "Software Engineering Challenges in New Media Applications", in *Software Engineering Applications (~SEA 2006~)*, Dallas, TX, USA, 2006.

[7] Manovich, L., "New Media from Borges to HTML", in *The New Media Reader*, N. Wardrip-Fruin and N. Montfort, Eds.: The MIT Press, 2002, pp. 13-28.

[8] Steinkamp, J., "My Only Sunshine: Installation Art Experiments with Light, Space, Sound and Motion", *Leonardo*, vol. 34 (2), pp. 109-112, 2001.

[9] Hannington, A. and K. Reed, "Towards a taxonomy for guiding multimedia application development", in *Ninth Asia-Pacific Software Engineering Conference (APSEC'02)*, Gold Coast, Queensland, AUSTRALIA, 2002, pp. 97-106.

[10] Sommerer, C. and L. Mignonneau, "Art as a Living System: Interactive Computer Artworks", *Leonardo*, vol. 32 (3), pp. 165-173, 1999.

[11] Edmonds, E., G. Turner, and L. Candy, "Approaches to interactive art systems", in *Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* Singapore: ACM Press, 2004.

[12] Bourque, P., R. Dupuis, A. Abran, and J. W. Moore, "Guide to the Software Engineering Body of Knowledge", 2004 ed, P. Bourque, R. Dupuis, A. Abran, and J. W. Moore, Eds.: IEEE Press, 2004, p. 204.

[13] Conradi, R., "Software engineering mini glossary", http://www.idi.ntnu.no/grupper/su/publ/ese/se-defs.html, last visited 19/06/07.

[14] Oates, B. J., "New frontiers for information systems research: computer art as an information system", *European Journal of Information Systems*, vol. 15 (6), pp. 617-626, Dec 2006.

[15] Machin, C. H. C., "Digital artworks: bridging the technology gap", in *Proceedings of The 20th Eurographics UK Conference, 2002* 2002, pp. 16-23.

[16] Trifonova, A., S. U. Ahmed, and L. Jaccheri, "SArt: Towards Innovation at the intersection of Software engineering and art", in *Proceedings of The 16th International Conference on Information Systems Development* Galway, Ireland: Springer, 2007.

[17] Solina, F., "15 seconds of fame", *Leonardo*, vol. 37 (2), pp. 105-110, 2004.

[18] Sardon, M., "Books of sand", in *Proceedings of the 14th annual ACM international conference on Multimedia* Santa Barbara, CA, USA: ACM Press, 2006.

[19] Strömberg, H., A. Väätänen, and V.-P. Räty, "A group game played in interactive virtual space: design and evaluation", in *Proceedings of the conference on Designing interactive systems: processes, practices, methods, and techniques* London, England: ACM Press, 2002.

[20] Fujimura, N., "Remote furniture: interactive art installation for public space", in *ACM SIGGRAPH 2004 Emerging technologies* Los Angeles, California: ACM Press, 2004.

[21] Boyd, J. E., G. Hushlak, and C. J. Jacob, "SwarmArt: interactive art from swarm intelligence", in *Proceedings of the 12th annual ACM international conference on Multimedia* New York, NY, USA: ACM Press, 2004.

[22] Fels, S., Y. Kinoshita, C. Tzu-pei Grace, Y. Takama, S. Yohanan, A. Gadd, S. Takahashi, and K. Funahashi, "Swimming across the Pacific: a VR swimming interface", *Computer Graphics and Applications, IEEE*, vol. 25 (1), pp. 24-31, 2005.

[23] Marchese, F. T., "The Making of Trigger and the Agile Engineering of Artist-Scientist Collaboration", in *Proceedings of the conference on Information Visualization (IV)*, 2006.

[24] Jaccheri, M. L. and G. Sindre, "Software Engineering Students meet Interdisciplinary Project work and Art", in *11th International Conference on Information Visualisation (IV)* Zurich, Switzerland, 2007.

[25] Harris, C., "Art and innovation: the Xerox PARC Artist-in-Residence program", C. Harris, Ed.: MIT Press, 1999, p. 293.