Towards Good Software Engineering Practices 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 products and processes are important in the production of new media art as new media artworks heavily depends on software. We have run a literature review and analyzed papers published in computer science journals and conferences to establish a knowledge base of the multidisciplinary field of software engineering and art with focus on interactive installation art. In addition we have observed several projects of new media art production.

This paper presents the findings of our review of interactive art installation. It organizes the theme according to five concepts which are: requirements, architecture, validation, process, and tools. Furthermore, we compare our findings with the general practices in contemporary software development. We discuss which issues are peculiar for software intensive art projects and need further software engineering research: 1) *Experimentation* as a working style of artists; 2) *Entertainment or reflection* as a final goal; 3) *Ideological* influence on the choice of technology.

This information should help researchers who work at the intersection between software engineering and the new media art as it provides a map of the published literature and outlines which research issues are peculiar of this multidisciplinary field.

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.

1. INTRODUCTION

As software becomes more and more important in all aspects of business and life, software practitioners cooperate with experts from disparate fields, like banking, automobile industry, health sector, and cultural institutions. As a consequence, the software engineering research community has the challenge to provide good practices that are specific for the multidisciplinary domains. Computer art is an important example of a multidisciplinary field at the intersection between computing and art.

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 [5]. The first software engineering (SE) conference was held in Garmish in 1968 [16]. 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. These artworks are generally complex and they are heavily dependent on software for controlling the whole system. The production of the software often requires the involvement of programmers and software engineers.

In 2005 Briony Oates [17] proposes to extend Information Systems (IS) research agenda in the domain of computer art. The first suggestion is that “computer art might be seen as a kind of information system”. We build up on this idea by giving a concrete example of an artistic discipline (i.e. interactive installation art) which intersects with software engineering. 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. We, further, study from SE perspective the production of interactive installations.

The field of software engineering has a well established knowledge base (e.g. SWEBOK [2]) of foundations and methodologies, with many books available on the market, many conferences organized yearly and active research bodies. For the multidisciplinary field of software engineering and art there is little established knowledge base [1, 15, 17, 25]. This field is young and there are not specialized journals like in other multidisciplinary fields, like automotive software and information and communication technology for health. The first step to establish such knowledge base is to perform a literature review [18] and determine what has been published until now in this area by researchers and practitioners.

Within the Software Engineering group of the Norwegian University of Science and Technology (NTNU) we have started a project, called SArt, in which software engineering research will be conducted on art as target domain (http://prosjekt.idi.ntnu.no/sart/).

We have started a systematic literature review with a broad focus following the guidelines in [11]. The full review, which covers many artistic disciplines in which software is used, is not completed yet, but detailed description of the process we follow and the preliminary results are available in [25]. Part of the review about interactive installations is considered complete. In this paper we present some of the outcomes of the review - the reported in the literature software engineering practices in interactive installation projects. Furthermore, we compare these practices with the practices in contemporary information systems development (ISD). This allows us to identify which characteristics are peculiar for art projects and deserve further research attention.

As part of SArt we are involved in three projects where the teams include both artists and software developers. The main objective of each of these projects is the production of an interactive art installation. These projects serve us as researchers in SE as case studies to gain deeper understanding on the emerging issues and problems. We act as observers, often participating at projects' meetings, following the communication between the members and analyzing produced documentation. We give a description of one of the projects further in the paper (see *Flyndre* in section 2) and show in which parts of the interactive installation software is required and developed. We also show the roles of the software developers and the artist in such projects.

Further the article is organized as follows: section 2 is dedicated to an introduction of the targeted application domain - new media art field and its subclass interactive installation art. We give an example of an interactive installation, pointing out the needs of software development in different levels. Section 3 summarizes the outcomes of a literature review of software engineering practices in interactive installation art projects which are later (in section 4) compared with the persistent SE problems. In section 5 we list the peculiarities in art projects. Section 6 is dedicated to a short discussion, conclusions and future work. Finally, acknowledgements (7) are followed by references (8).

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. We point to the work of new media art theoreticians like Manovich [13] and Tribe et al. [24]. 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." [14]. 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.

Installation art is a phenomenon which starts in the late 30s 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" [22]. 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 action(s). The creation of an interactive installation commonly requires specialists with different areas of competence to collaborate with the artist[1]. A multidisciplinary team can involve artists (painters, composers, sculptors, etc.), constructors, hardware designers, electrical engineers, software engineers, programmers, art curators, 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 (e.g. [6, 8, 21]).

In our viewpoint there are three perspectives of the interaction that have to be taken into consideration:

- Interaction Rules – the rules that control the interaction might be static or dynamic. Static rules are defined in advance and the dynamicity of the interaction depends on the triggering parameters. In other words, when the interaction rules are static if the same triggering parameters are inputted twice the response/reaction will be the same in both cases. On the other hand, dynamic rules are dynamically modified, thus that even with repeating input parameters the interactivity changes. The *evolutionary* interaction described in [21] might be considered dynamic interaction rules, but limits the rules to evolutionary algorithms. In [6] this difference between static and dynamic rules is shown by introducing the *dynamic interactive (varying)* category of interactivity.
- Triggering parameters – The interaction rules generally depend on environment parameters that are changed during the artwork exhibition. 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

[1] 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.)

foreseen only in [6] with the *dynamic-passive* interaction category.

- 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 [8] - category *adaptive*.

Flyndre is a real example of an interactive art installation. It is a sculpture located in Inderøy, Norway. It has an interactive sound system that has the goal to reflect the nature around the sculpture. To implement this goal the produced sound changes depending on parameters like the local time, light level, temperature, water level, etc.

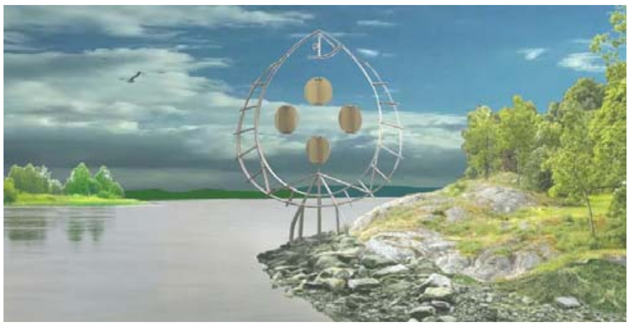

Figure 1. Interactive Installation - Flyndre, Norway

On Figure 1 one can see the physical construction of the artwork which is placed in the natural site[2]. People can walk around the sculpture or sit nearby to watch it and listen to its music.

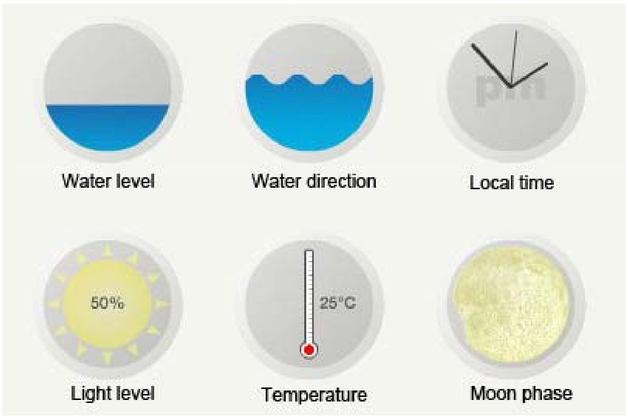

Figure 2. Environmental parameters in Flyndre

Figure 2 represents the triggering parameters from the environment that are captured by sensors and which influence the generation of the music. These parameters are integrated by interaction rules into the music which is played by the sculpture all the time. Some of the rules are static (i.e. fixed results are produces by certain parameters), while other rules are dynamic (e.g. "algorithmic composition, generating new sounds in a semi-autonomous fashion" [19]). The 'content' in this interactive installation is the music. It includes samples pre-defined by the composer, which are combined with and overlaid by the dynamically generated sounds.

[2] This is a screenshot of *Flyndre* web site www.flyndresang.no

Flyndre relies on software in several of its main parts – 1) for controlling the numerous sensors that are capturing environmental parameters, 2) for incorporating the captured parameters into music that is played by the sculpture, 3) for maintaining an archive of the music generated in the past, 4) for online presentation of the artwork, the artist, the music archive and the software. The web site is an important part of the whole project and includes on-the-fly animated Flash application that displays the current parameters of the environment and the current music played by the sculpture. The archive of the previously played by the sculpture music is also accessible through the web site.

The sound installation of Flyndre is created by the composer, musician and programmer Øyvind Brandtsegg over an existing sculpture by the sculptor Nils Aas. At the controlling core of the sound installation there is a custom version of the software ImproSculpt[3]. The first version of the software was written by the composer in 2001 and was a single CSound (http://csounds.com/) script file that was hard to modify, maintain and upgrade. However, the author discovered that it created a lot of interest among other composers and music technology enthusiasts. In order to improve the architecture of the software and publish it as open source software (OSS) with appropriate licenses, a multidisciplinary team, including software developers and NTNU students (as Master thesis work, see [19] and as part of the "Experts in Team" course, see [9]), was necessary. When published as OSS and available for download the software ImproSculpt attracts more developers and composers to utilize it and further improve and/or upgrade it.

The artist has started the software project that implements the functionalities of Flyndra. He has been the sole responsible for development of the modules which implement the composition logic (e.g. the sampling algorithms or the integration of particular sounds which are generated based on the environmental parameters). The generated music reflects the composer's aesthetical values and/or the desired [by the artist] effect/impression on the audience.

During the year the cooperation between the composer and software experts has given room to software development by other persons. The software engineers developed the technical framework for the networking and the sensors systems (i.e. for capturing parameters by the sensors and for transferring them via the Internet to the sound processing station). Another group of software engineers have re-factored the software modular architecture. During the following years the artist has continued his solo work on ImproSculpt and the software has evolved and changed several times. The artist, however, has consulted and reused whenever possible the modular architecture proposed by the software engineers and he found it very useful. In addition, the artist found helpful the publishing of the software as open source in SourceForge[4] and the utilization of the Wiki[3] and the Concurrent Versioning System (CVS) introduced by the software engineers. The composer has always acted as a project owner retaining the control of the whole software.

[3] http://improsculpt.sourceforge.net/pmwiki/pmwiki.php

[4] http://sourceforge.net/projects/improsculpt/

3. SOFTWARE ENGINEERING PRACTICES IN ART PROJECTS

In this section we present some of the findings of our literature review. In advance we have synthesized a list of software engineering concepts [2, 4] for the purpose of defining the intersection between software engineering and interactive installation art which allows us to reflect about new media art. We have developed this list of important software engineering concepts by looking at the latest International Conference of Software Engineering (ICSE 2007) and we have modified it supported by our experience and discussions with colleagues. The list includes: Requirements; Software Architecture and Design; Testing; Process Models and Project Management; Development Environments and Tools; Maintenance; Open Source Software; Quality Attributes; etc.

The following subsections describe those five software engineering concepts addressed in the reviewed articles considering new media art.

3.1 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.

Requirements definition is one of the most difficult tasks reported by 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" [15]. However, requirements are often found difficult to capture. Machine [12] 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 the artwork ‘Trigger’ - "the system design was updated to reflect experiments with different types of sensors” [15]. Biswas and Singh [1] 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 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" [22]. Additionally, often “the system usage context is entirely absent or it is not well understood” [1]. Hannington and Reed [8] 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.” [1].

It is important that both software developers and artists in interactive installation art 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.

Our literature review shows 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.

3.2 Software Architecture and Design

The 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. The software architecture depends on the functionalities which should be provided by the system, on the technology chosen, on software engineers’ 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 [15] 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. [3] 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, which is 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’ [7] has a modular architecture. The use of object-oriented software engineering methods is reported in [23]. Machin in [12] 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 in the artwork construction and their quantity). Similarly, Biswas and Singh [1] 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" [6].

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 programming experience is not enough of developing the whole software. Adding an extra layer between the artist and the underlying programming fosters the artwork creation without limiting artists' creativity.

3.3 Testing

Software products are usually checked for their correctness during execution, for satisfying the requirements specifications and on how well the end-product satisfies the user expectations. 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.

Marchese [15] 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 [23] 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. [7] evaluated their artwork with the visitors of Siggraph[5] 2004 exhibition, collecting opinions, positive and negative experience, comments and suggestions.

[5] Siggraph is a large annual conference and exhibition on computer graphics and interactive techniques www.siggraph.org

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 [22] 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 [8] 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" [1]. 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 artwork '15 seconds of fame' [20] 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.

3.4 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 [15] 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 [1] 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 [23]. 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 engineers for deriving an object-oriented architecture.

Considering the whole project management several issues should be mentioned. The creation of the artwork in interactive installation art is often sponsored by a public entity which does not influence on 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 [15] "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. variety of sensors, location awareness, mobile and wireless devices, etc.). Together with mostly limited budget, this leads to the high probability that the involved 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.

3.5 Development Environments and Tools

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.

The development tools used by software engineers in the design and implementation of the software of interactive installations are not discusses in the reviewed articles. Our guess is that standard CASE tools were utilized (the ones which the technologists are most familiar with) without 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" [6] 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 tool – for implementing their programs (e.g. [15]), but also for supportive tools, such as for creating the storyboard in [23]. In several cases (e.g. [1, 6, 12]) 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.

4. PERSISTENT SE PROBLEMS VERSUS SE IN ART PROJECTS

In the previous section we have reported the software engineering practices found in the literature from projects involving artists and software developers. The goal of this section is to distinguish which practices and problems are specific for art related projects and which are common in SE. Understanding this is important for properly selecting which SE theories to apply in future software intensive art projects. Additionally, it will give indication of which features are unique in art and should be further studied from SE point of view. We compare our literature findings with a study [10] that synthesizes the characteristics of contemporary IS development[6].

In [10] three inherent and interrelated problems are identified in the information systems development – 1) *diversity*, in terms of application domains, underlying technology, people involved and variety of end users, 2) *knowledge* needed to cope with the diversity and 3) *structure* in terms of process methodologies. The authors discovered that the practices for coping with these challenges are a) organization and specialization (in three different levels – business, company and personal), b) constant verbal communication and negotiation and c) pragmatic application of certain development methods and methodical concepts.

4.1 Diversity

Technology - The continuously decrease of hardware prices and increase of speed and storage capabilities of computers allows the use of computation in many domains with many companies strongly relaying on their back-end applications. In addition, new technologies (i.e. the Internet and the Web) allow flexible access to data globally, at high speed and low cost. Companies utilize the opportunity to present themselves and their products and services via web sites or access their back-end applications online. The creation of these web sites often involves artists'/designers' participation.

In the art domain, however, we can see the use of computational technology (i.e. software) for larger variety of purposes often within a single project. Similar to companies, active contemporary artists often create their presentation on the web. Meanwhile, in interactive installations the software is developed for controlling the installation and the interactions of the users with the physical construction. Furthermore, software modules might be used for generating artistic expression (like the music in

[6] Further in this section when giving generalized statements about ISD, although the reference might be sometimes omitted, we reference the article [10] Kautz, K., Madsen, S. and Norbjerg, J. Persistent problems and practices in information systems development. *Information Systems Journal*, *17* (3). 217-239.

Flyndre, described in section 2). Such modules are usually created by the artist and through artists' experimentation. The desired outcome often cannot be translated into formal requirements and is continuously evolving.

In addition to the larger functionalities needed in art we should mention that artists often influence on the choice of the technological solution due to their ideological point of view. Frequently, artists experiment with emerging technologies in order to provoke audience's attention and invoke reflection. Furthermore, in many cases we observe that artists favor Open Source ideology and may impose the use of OSS tools and require licensing of the developed software as OSS.

Multidisciplinarity is mentioned in [10] as an emerging characteristic of IS. It is due to the increased complexity of the technology which leads to the need of specialization of the team members. Sometimes members of the multidisciplinary teams might lack basic IT knowledge and skills which is a major challenge. Multidisciplinarity is a big issue in new media art too. In different art-related projects experts from many domains have to work together to produce the final artifact – the artwork. In these projects artists might be either members of the team or 'clients' to the SE team or both. In the cases when artists are part of the development team the general ISD issues apply.

End users - The Internet has brought an increase in the variety of users of IS which often means that designers and developers have to create systems for a large and diverse audience without knowing very well the potential users. This characteristic of IS completely maps to our findings about artistic projects. Although this diversity and unfamiliarity with the users creates challenges in terms of identifying, describing and managing requirements it is not new in IS. When dealing with such issues in art we should consult previous studies and possible guidelines from the literature (i.e. SE theories and best practices).

4.2 Knowledge

Domain knowledge - Another reported ISD problem is the "thin spread of application domain knowledge". In other words, often in ISD projects only small number of members in the development team understand well the targeted domain. This lack of domain knowledge often leads to inability to map between the targeted domain information and the computer representation. The problems related to vague and conflicting requirements often appear as a result of lack of knowledge of the domain area. To overcome this problem "truly exceptional people, i.e. gurus" that understand/know well the targeted domain bridge this gap by face-to-face interactions to negotiate and communicate shared understanding. Such people might become bottleneck in the communication, thus in ISD projects it is important to plan and budget time and resources for learning about the application domain. This is one of the things that we think differs drastically in art-related projects. Artists rely on being different from one another and artworks are rarely very similar. Thus, even if one knows/learns a lot about art theories and is familiar with contemporary art practices it still might be impossible to overcome problems with vague requirements. In any case, the importance of open and intensive communication to reach shared understanding between software developers and artists is underlined by practitioners in interactive installation art. Partially, misunderstandings might be overcome with the help of simulation and prototyping. Nevertheless, we should mention again that artists' work-style is often experimentation.

On the other hand, a trend we observe is that artists (when they act as clients) most often have significant interest in being familiar with (i.e. gaining knowledge of) the utilized technology. The reasons for such desire might differ from artist to artist. In some cases this is due to personal interest, which might have ideological background. In other cases the reason might be practical – when the artist uses certain technology in an artwork often he/she wants to continue with it in consecutive projects. Consecutive artistic projects, however, might be developed with different teams, often due to different means of funding. This leads to the artist being the connecting branch between versions, thus the desire of understanding the details.

4.3 Structure

Development methods - Traditional ISD relies on methods and management techniques that are not widely used in web-based ISD. In the development of web-based systems ad hoc methods are most often observed. Several studies have reported that features like high time pressure, dynamic environment, short development cycle, unstable requirements, etc. which are typical for web-based ISD make many of the traditional ISD methodologies unfit. In fact, evolutionary or incremental software process models are recommended as more appropriate to cope with these specific features. We have observed these features in the interactive installation projects reported in the previous section of this article (see section 3.4). Note that incremental prototyping was also mentioned as a good choice in one of the interactive installation projects.

Projects characteristics - Similar to web-based ISD projects art projects are often small, with teams of a couple of people. It is shown that in smaller organizations and teams have less need of structure (i.e. formalization and application of methodologies), but disciplined development approach is often related to the quality of the IS. The authors of [10] mentions that the overall low use of methodologies might be indicator of low maturity of the web-based ISD field. Although this might be also the case with interactive installation art, further study might be necessary on how to incorporate the artists' experimentation needs and work styles. In any case, project planning and management issues are commonly difficult but extremely important in ISD, which we observe also in artistic projects.

5. PECULIARITY OF ART PROJECTS

In previous sections we present the software development practices in the domain of interactive installation art and we compare this practices with the persistent problems in IS development. We see that many of the characteristics and problems of modern software development are similar to the ones we observe in art projects. Based on the comparison and on our experience we identify several characteristics which are peculiar for artistic projects:

1) *Experimentation* as a working style of artists: This is a major challenge and further research is needed to fully understand how to support such experimentation without limiting artists' creativity. Constant verbal communication and negotiation is crucial and prototyping might be very helpful. The acquisition

of domain knowledge and learning about the art domain by software engineers might not resolve difficulties of vague and changing requirements in art projects, due to the experimenting nature of artists working styles. On the other hand, we observe that artists are interested in gaining software engineering knowledge, which, in our opinion, is not typical for typical clients in ISD.

2) *Entertainment or reflection* as a final goal: One of the main differences we observe in art related projects is the application aim/goal. While in other domains such as business, commonly the goal is to increase task efficiency, in art the goal is to entertain and to raise awareness in the audience about contemporary phenomenon and its implications for the future of the society. This characteristic leads often to the production of non-critical applications. Due to this, multidisciplinary projects might be very good for educational purposes (i.e. Flyndre or Experts in team course in NTNU, described in section 2, involving software engineering students and artists/art students). Unexpected (even incorrect) functionalities of the developed programs might be considered by the artists as non-negative, interesting and inspiring effects.

3) *Ideological* influence on the choice of technology. Artists' preference for utilization of [emerging] technologies, like mobile devices, surveillance cameras, sensors, etc. is often due to ideological motivation - they are driven by their desire to provoke reflection in the audience. The technological choice influences on the use of software technology too. This differs from other domains, where the choice of software technology is based on the workflow processes that the new software has to support and the needs to integrate new systems with previous available systems. Additionally, we observe ideological choice even of the software technology: for example artists use open source software for ideological reasons. What typically happens is that when they try it the user interface is unfamiliar and difficult and some features they are used to and they need are missing. Thus, they switch back to proprietary applications, like Final Cut Pro/Acid/Photoshop. A challenge for SE is to keep the artists' interest in participating in open software development, as their artistic and innovative input of ideas for new functionalities and solutions to problems might prove to be very valuable.

In addition, special attention should be paid on artists' tendency to utilize wide variety, mostly emerging technologies. In interactive installation art projects we often observe the use of software to support large variety of functionalities (e.g. control of sensors, interaction, content generation, web presentation, etc.). The software is often developed through a single project by a small team with generally tight budget and schedules. This implies to the need of cautious choice of software development methodology and project management, including careful risk assessment.

6. CONCLUSIONS, DISCUSSION AND FUTURE WORK

In this paper we present the software engineering practices in projects for creating software intensive art. We compare the reported in the literature issues with the common characteristics and problems in information systems development, pointing to the similarities. We also discuss the particularities of art projects which might need further study from software engineering perspective. This work will give a roadmap to researchers and practitioners interested in the multidisciplinary domain where software intensive art is developed.

In principle, this work should be relevant to artists who want to communicate with software engineers and to software engineers who want to work with artists.

The difficulty of systematic literature review of software engineering issues in art domain: Performing an multidisciplinary systematic literature review poses a set of challenges. One of them is to find the relevant publication sources (journal and conferences). There are multidisciplinary domains (e.g. e-learning or bioinformatics) with well established traditions and journal/conferences. We have found out that in the art domain there are few specialized publications on art and technology. The bigger problem, however, is that the existing ones (e.g. MIT Leonardo[7] or Convergence[8]) most often have a very broad focus and address mainly the artistic perspective, so software engineering issues are either not discussed or addressed superficially. Another option is to search SE publications for art-related articles. The defining of useful keywords to search for relevant articles proved to be a difficult task. The keyword 'art' is problematic to use as part of a search in the computer science world as the combination "state of the art" is used in almost all papers.

The importance of software engineering concepts which are not reported in relation to interactive installation art: In the beginning of section 3 we have listed a number of software engineering concepts that are important in SE. While performing our literature review we tried to cover all of them. Nevertheless some concepts, like maintenance or quality attributes of the 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 in the published articles. Our experience with industrial SE makes us hypothesize 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.

Our current and future work is to extend the systematic literature review. It has a broad focus and our goal is to cover different artistic disciplines unveiling the importance of the software engineering perspective. In addition, we intend to proceed with case studies on the projects we are involved in.

Finally, we think it is essential to inform members of both communities (software engineering and art) of the important concepts of the other field. This will raise awareness, interest and multidisciplinary discussions.

[7] Leonardo: Journal of the International Society for the Arts, Science and Technology www.leonardo.info

[8] Convergence: The International Journal of Research into New Media Technologies www.luton.ac.uk/convergence

7. ACKNOWLEDGMENTS

We thank Øyvind Brandtsegg for our good cooperation and for the permission of publishing information about the artistic project Flyndre.

Part of this work was carried out by Anna Trifonova during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

8. REFERENCES

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

[2] Bourque, P., Dupuis, R., Abran, A. and Moore, J.W. (eds.). *Guide to the Software Engineering Body of Knowledge*. IEEE Press, 2004.

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

[4] Conradi, R. Software engineering mini glossary, http://www.idi.ntnu.no/grupper/su/publ/ese/se-defs.html.

[5] Digital Art Museum. History, Digital Art Museum, 2007, http://www.dam.org/history/index.htm.

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

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

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

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

[10] Kautz, K., Madsen, S. and Norbjerg, J. Persistent problems and practices in information systems development. *Information Systems Journal*, *17* (3). 217-239.

[11] Kitchenham, B. Procedures for Performing Systematic Reviews, Keele University Technical Report TR/SE-0401 and NICTA Technical Report 0400011T.1, 2004.

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

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

[14] Manovich, L. New Media from Borges to HTML. in Wardrip-Fruin, N. and Montfort, N. eds. *The New Media Reader*, The MIT Press, 2002, 13-28.

[15] 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), IEEE Computer Society.

[16] Naur, P. and Randell, B., Software Engineering: Report of a conference sponsored by the NATO Science Committee. in *Software Engineering*, (Garmisch, Germany, 1968), Brussels, Scientific Affairs Division, NATO (1969), 231.

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

[18] Oates, B.J. *Researching Information Systems and Computing*. SAGE Publications Ltd, 2006.

[19] Semb, T.A.G. and Småge, A. Software Architecture of the Algorithmic Music System ImproSculpt *Software Engineering Master Thesis at Department of Computer and Information Science*, NTNU, Trondheim, 2006, 70.

[20] Solina, F. 15 seconds of fame. *Leonardo*, *37* (2). 105-110.

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

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

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

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

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