SArt: Towards Innovation at the Intersection of Software Engineering and Art

Anna Trifonova, Salah U. Ahmed and Letizia Jaccheri

Department of Computer Science,
Norwegian University of Science and Technology (NTNU)
{trifonova, salah, letizia}@ idi.ntnu.no

Abstract. Computer science and art have been in contact since the 60´s. Our hypothesis is that software engineering can benefit from multidisciplinary research at the intersection with art for the purpose of increasing innovation and creativity. To do so, we have designed and planned a literature review in order to identify the existing knowledge base in this interdisciplinary field. A preliminary analysis of both results of our review and observations of software development projects with artist participation, reveals four main issues. These are software development issues, which include requirement management, tools, development and business models; educational issues, with focus on multidisciplinary education; aesthetics of both code and user interface, and social and cultural implications of software and art. The identified issues and associated literature should help researchers design research projects at the intersection of software engineering and art. Moreover, they should help artists to increase awareness about software engineering methods and tools when conceiving and implementing their software based artworks.

1 Introduction

Art finds expression in numerous products in society, where the development of products is complex, competitive, global and intercultural in scope. Art and computer science are in contact from the sixties (Sedelow 1970). The advent of multi media technology has changed art production processes and the way both music, video, and figurative is fruited by consumers. Our assumption is that the interaction between software technology and art is also beneficial for the software technologists.

In this context at the Norwegian University of Science and Technology (NTNU) we have initiated the SArt project that will explore the intersection between software engineering (SE) and art. Our first task is to answer the following questions: How do software engineering and art intersect? How do software engineering and art influence and can involve each other? What has been done until now in this area by other researchers in computer science?

According to our knowledge, this is the first attempt to answer these questions, although there has been a proposal to extend the IS research onto the art domain (Oates 2006a). There is no established knowledge base in this interdisciplinary field and finding what research has been done previously appears not to be a trivial task. We have initiated a literature review, the preliminary outcomes of which had confirmed that both software engineers and artists might profit by further research in this interdisciplinary field.

This paper has three main goals. Firstly, we aim at promoting interest and discussion in the interdisciplinary field between art and software engineering. Secondly, we present our understanding of how the two areas influence on each other and stress on the potential benefits for software engineering - innovation and creativity (Glass and DeMarco 2006). Last but not least, we discuss the process of making a survey of the state-of-the-art, aiming at completeness of the literature review. We show the preliminary outcomes of it which might be a starting point for other researchers and practitioners in the area.

The following section (section 2) gives some more details on our project – SArt - with its objectives and current involvements. In section 3 we discuss research methodology issues. A subsection is dedicated to the procedure of making the literature review in a systematic way. Section 4 shows the preliminary results of it. Section 5 is dedicated to a discussion. Section 6 – conclusions – is followed by references.

2 The SArt Project: Goals, Background, and Motivation

2.1 About the Project

The SArt project (http://prosjekt.idi.ntnu.no/sart/) is conducted inside the Software Engineering group at the Department of Computer Science at the NTNU. The focus of the project is the exploration of research issues in the intersection between software engineering and art. Our final objective is to propose, assess, and improve methods, models, and tools for innovation and creativity in software development. Our activities are mainly oriented at academic research. However, we also take part in different artistic projects.

This project confronts the interdisciplinary problem of integrating software designs originating from different academic genre into better products for society. The research methods will be based on empirical software engineering (Kitchenham 2004; Hevner, March, Park and Ram 2004; Wohlin, Runeson, Høst, Ohlsson, Regnell and Wesslen 2000) and will benefit from the interaction with experts from different disciplines, like artists.

2.2 Objectives

The principal objective of the SArt project is to propose, and assess, and improve methods, models, and tools for innovation and creativity in software development. Particular attention will be paid to art influenced software development. Sub-goals are:

G1. Develop knowledge on the interdisciplinary nature of software production in which the software engineering process interacts with the artistic process.

G2. Develop empirically based theories, models, and tools to support creativity and innovation in software technology development.

G3. Educate competent practitioners and researchers in the interdisciplinary field of software innovation and transfer of knowledge to the software industry and artist communities.

2.3 Projects

As part of SArt we are involved in three projects including both artists and software developers. Here we give a short description of those projects, showing the problems and the challenges the participants have encountered.

Flyndre (www.flyndresang.no) is a sculpture located in Inderøy, Norway. It has an interactive sound system that reflects the nature around the sculpture and changes depending on parameters like the local time, light level, temperature, water level, etc. The first version of the software was written by the sound composer 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 part of the "Experts in Team" course, see Jaccheri and Sindre 2007), was necessary. In this way (as OSS) the software attracts more developers and composers to utilize it and further improve and/or upgrade it.

Sonic Onyx (http://prosjekt.idi.ntnu.no/sart/school.php) is an installation project initiated by the Trondheim municipality in 2007. The goal is to decorate a junior high school with an interactive sculpture with which the people will be able to interact. The users will be allowed to send messages, images or sound files from mobile phones, laptops and hand held devices through Bluetooth technology. The received files will be converted into sound and played by the sculpture. For the development of the hardware and the software sound sub-systems a team of students from Hist (www.hist.no) is created and is working in collaboration with the artist (i.e. the sculpture designer). The members of the project use open source technology, like PureData (http://puredata.org/), Apache (http://www.apache.org/), Python (http://www.python.org/), due to availability of free software that allows development of low cost software and community support for maintenance and upgrade.

Open Digital Canvas (http://veggidi.opentheweb.org/) is a project which aims to embellish a white wall at NTNU with a number of main boards with LEDs on them, creating a big matrix of light pixels. We wish to push the concept of openness a step further by keeping the hardware, software, and behavior as open as possible. In the project the artist, the student and members of the computer science department work together to create a platform that will allow freedom of artistic expression and be a source of inspiration and reflection around interdisciplinary education and research.

Our experience in the described projects shows that software engineering theories and tools can play an important role for the improvement of the development process

and the design of the software architecture. We have observed functional requirements, software usage, development trends and challenges that are common or very similar for these projects. For example, the artists want to use some of the latest technologies in their artworks (e.g. different sensors in Flyndre or Bluetooth file interchange in Sonic Onyx sculpture), thus software developers are often not experienced and have to learn on the fly. The developing time and budget are limited. This might lead to neglecting the design of the software or the planning of the development process and might have many negative consequences (e.g. the software is created without proper architecture, thus is difficult for modification and upgrade; the documentation is poor or missing, thus reuse is hard). Software evaluation, testing and maintenance might also be problematic, as the funding is generally limited to the artwork creation. The software engineering perspective helps increasing the awareness in these issues. The artists seek for a cheap way to create their artwork and the software development is performed by students that are even less experienced. The software requirements are often not well defined and might change frequently. The artists would like to have a web site in addition to the main software that will present details about the artwork itself and provide a user-friendly interface to observe and/or control the system. Common interest of using Open Source software is also visible from all three projects. Interestingly, there is also a desire to provide the created software as open source. However, we observe difficulties in the choice of proper OSS licenses.

3 Methodological Issues

The following section explains our research approach.

We use the Information System Research framework developed by Hevner et al. (2004) to reflect about the SArt research framework (Fig. 1).

- *Knowledge base*: there is an established knowledge base of foundations and methodologies in software engineering (e.g. SWEBOK, Bourque and Dupuis 2004). For the field of software engineering and art there is no established knowledge base. The first step to establish such knowledge base is to perform a literature review (Oates 2006b).
- *Environment*: our experience in collaborating with local artists, observing artistic projects available on the web, participation to art festivals, tell us that artistic projects that involve both software engineers and artist exist and pose needs to our research. Moreover, we observe that the game industry (e.g., funcom.com in Norway) employ artists as part of their developing teams.
- *Research*: our task as researchers in this multidisciplinary field is that of contributing to establishing the knowledge base at the intersection of art and software engineering. In this phase of our research process we focus on literature review and projects observation. The next step will be that of developing new theories and artifacts that we will evaluate by means of empirical studies, having in mind our goals as introduced in section 2.

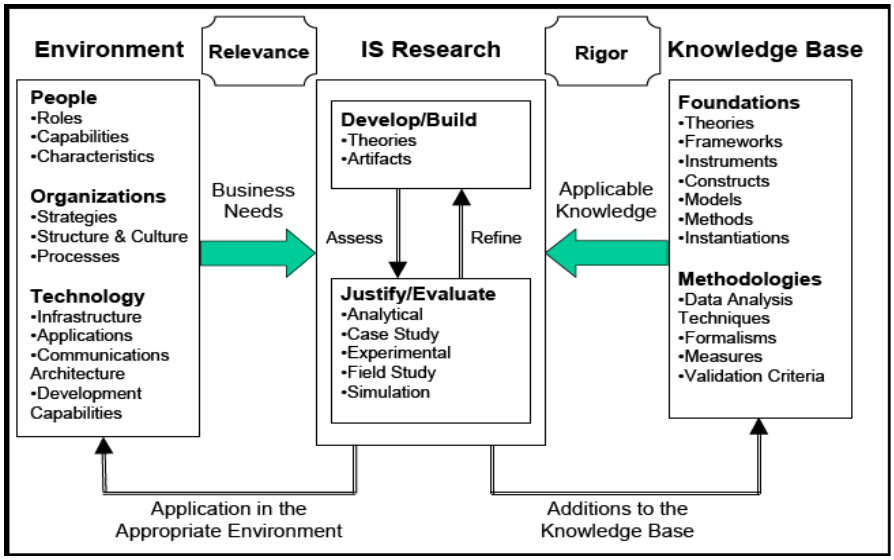

Fig. 1. Information Systems Research Framework (source Hevner et al. 2004)

3.1 Literature Review

A review of the available literature at the intersection between software engineering and art is necessary to answer the question posed above: *What has been done until now in this area by other researchers in computer science?* The goal of the review is to establish our knowledge base, as discussed above. It will, hopefully, reveal research gaps and opened questions in the field.

Performing an interdisciplinary review poses a set of challenges. One of them is to find the proper keywords to search for relevant articles. 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. This made the keyword search difficult. Another challenge is to define the criteria according to which to treat the article as relevant or irrelevant.

We have defined a process (Fig. 2) according to which to perform this literature review in a systematic way, following the advices, guidelines and suggestions of Hart (1998) and Kitchenham (2004). The procedure includes a three steps process - 1) planning the review; 2) conducting the review and 3) reporting the review. We have adopted this procedure in the following way.

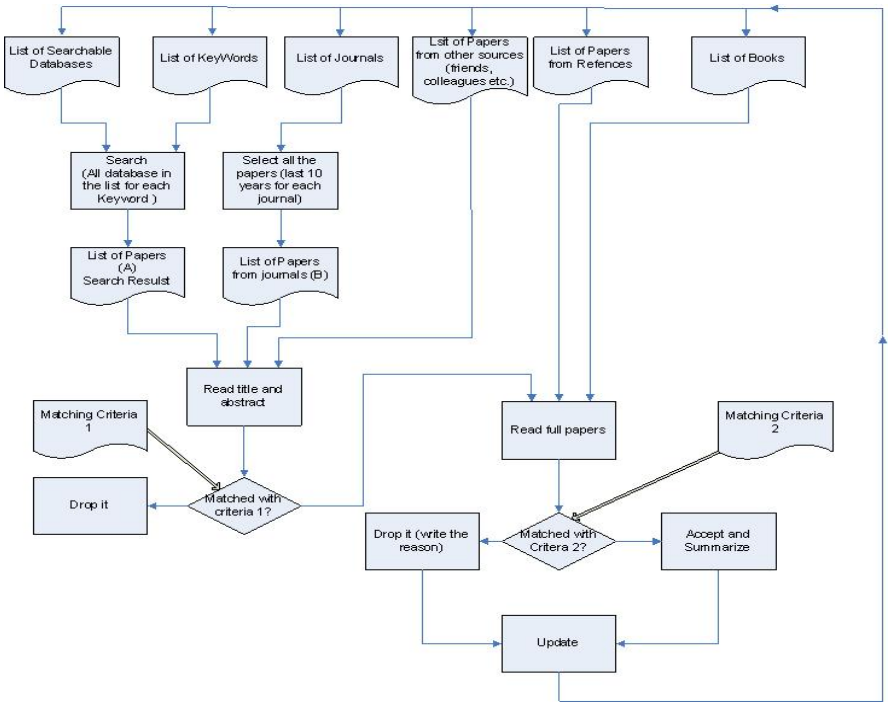

Fig. 2. SArt Process of Conducting the Literature Review

Planning the Review

During the planning phase we have set in advance and agreed on common relevance criteria for inclusion/exclusion of articles in the literature review. We have also identified the search strategies, including a list of searchable electronic databases of scientific publications and a starting list of keywords.

The selection criteria based on which to consider an article relevant or not were cleared and polished in several iterations at the beginning of the literature review. Relevant articles are those that address one or more of the following issues:

- artists' requirements for software and/or habits in utilizing it;
- software development processes and/or methodologies used for the software developed with artist's participation. This might include software created for artistic purposes (e.g. digital art, art installations, etc.), artists programming/producing software (e.g. software art) or participation in the development of software by other means;
- collaboration between artists and software developers or computer scientists;
- research questions at the intersection between art and software (pose and/or answer);

- the influence of software and digital culture on art and/or the influence of art on software development.

During the planning phase we have identified the following search strategies for finding relevant articles, to aim at completeness of our literature review:

- *keyword search in electronic databases* - We have identified a starting list of searchable electronic databases (e.g. IEEE Xplorer, ACM Digital Library, Google Scholar) and an initial list of keywords (see Table 1). Searches with each keyword (from the art related terms) and combination of keywords should be performed in each of the electronic databases. In most of the cases no limitation on the published year was applied. In those cases when the search resulted to over 100 entries the search was limited to the last 10 years of publication. Both initial lists (i.e. the search engines used and the keywords) are being continuously extended with new entries that appeared relevant during the literature review.

Software Engineering related terms	Art related terms
software engineering; software development; requirements; collaboration	art; software art; media art; digital art; blog art; net art; generative art; art installation; artist; artwork; aesthetics; creativity

Table 1. Initial List of Keywords

- *thoroughly search the papers published in selected relevant journals and conferences* - We have identified a list of relevant journals and conferences that, according to our previous experience or to their web-site description have a high probability of containing a large number of relevant articles. These journals and conferences are mainly focused to interdisciplinary research in software and art. We have planned to thoroughly examine/read the titles and abstracts of the papers from each of these journals and conferences for the past 10 years. New entries are continuously being appended to this list during the review process.

Journal/Conference
MIT Press Leonardo – Online Journal on Art, Science and Technology, 40 years of publications (http://www.leonardo.info/)
IEEE Computer Graphics and Applications – A journal on theory and practice of computer graphics (www.computer.org/cga)
Read_Me conference – The proceedings from years 2002-2004 are published in "READ_ME: Software Art & Cultures", by Olga Goriunova and Alexei Shulgin (eds.), Aarhus University Press (December 2004)
Convergence - The International Journal of Research into New Media Technologies, since 1995 (http://convergence.beds.ac.uk/)
ACM Siggraph - publications in Computer Graphics and Interactive Techniques (http://www.siggraph.org/publications)
Proceeding books of ARS Electronica – festival for art, technology and society (http://www.aec.at/en/global/publications.asp)

Table 2. Initial List of Journals and Conferences

- *finding articles from the references of papers that are reviewed* – On the initial step this list of papers from reference is empty and is filled on a later stage of the process. It is when we are reviewing, i.e. reading fully the relevant articles selected from search strategies described above, that we can find some referenced papers that appeared to be relevant (from the citation and title). In this case we include them directly into the list of related papers to be reviewed/read in full, in order to gain deeper understanding of the field.
- *other resources* – We have included in the review process the possibility of getting relevant articles from other sources. Some papers might be referred to us by colleagues and friends. Other articles we might find by looking at the publication lists of well-known researchers in the field and/or from institutions working in this area. The list of these papers is being continuously updated. Here we include also the articles and books that we knew from before and that have motivated our work, like Glass (1995), Green (2004), Harris (1999) and Sedelow (1970).

Conducting the Review

Articles Selection - The inclusion/exclusion of the articles is made in two steps. Initially, the selection criteria (i.e. the first list given above) are interpreted liberally, so that unless the articles identified by the electronic and manual searches could be clearly excluded based on titles and abstracts, full copies were obtained. The selected articles were independently reviewed (i.e. read fully) by two or more of the authors of this paper.

Data extraction – when fully reading the articles the authors analyze its content and summarize it in a table. The inclusion/exclusion in the review was discussed until agreement is reached. The authors also produced annotations about the content.

Statistics generation and synthesis – the produced table with articles' annotations and summaries allows easy generation of certain statistical data (years of publication; conferences in which relevant articles occur more regularly; authors actively working in the area, etc.).

The conducting of the review is an iterative process in which the initial lists of keywords, relevant journals and books is being updated with new entries found in other/previous articles.

Reporting the Review

We have conducted the first phase of the review. We have performed trial searches with most of the initial keywords defined and selected a number of papers (maximum ten from each search). In total about 50 articles have been reviewed so far. In this first iteration we have clarified the process steps and our relevance criteria which will allow us to decide the focus of our further study. In this paper we discuss the preliminary outcomes of the review. Additionally, relevant books and the observation of a number of web sites have contributed to our understanding of the field.

4 Research Issues Found in the Literature

In this section we present some of the research work/papers we have reviewed in our first phase of literature review and found relevant according to the selection criteria described in section 3. The issues presented here are neither complete nor exhaustive; rather it is aimed to make a further contribution to the review result by grouping the articles according to the issues that have discussed (Oates, 2006b). As certain papers discuss more than one of the issues they are referenced more than once. This is a preliminary trial to create some classification/taxonomy of the research in the field. We also shortly propose hints for further software engineering research in each category.

4.1 Software Development Issues

- *requirements/needs for software and software functionality within the artist community*. The artist Jen Grey (2002) discusses her experimentation with an alternative input mode (3D) – Surface Drawing © by Steven Schkolne – that she evaluates as very good for producing her artworks. The author have used the software in "a unique way to draw live models, a purpose for which it was not intended". Though it is not discussed by the author, this article points us the need for further research on the artists needs for software functionality of the general-purpose software. Such research might include studies on how the commercial and OSS available software match the artists needs. Meanwhile, a separate study might be needed on how to best discover the software requirements in projects that involve software development for art systems, like art installations reported by Marchese (2006) and the projects we described in section 2.
- *evaluation in art projects*. Marchese (2006) describes the software developer's perspective on the creation of an interactive art installation. The author reports that the understanding of the system requirements is "the most important part of the process". This is consistent with the software engineering body of knowledge. Software systems are developed according to customer requirements (in art as in other fields). The customer (the artist in this domain) will differ from the user of the product. The artist might want certain interactions to be triggered when a spectator approaches the artwork. For creating the software in the proper way, however, it is needed to understand how the artwork as a whole will be perceived by the spectator and a special attention should be paid on the fact that user's evaluation or acceptance testing might appear only after the product is released (i.e. the artwork is in the museum and the exhibition is opened).
- *software tools*. Edmonds, Turner and Candy (2004) have presented their approach to building interactive systems. Based on the collaborative work between artists and computer scientists (the authors themselves) they report that there is a need (for the artists) for "a bridge between the use of an environment that requires programming knowledge and the 'closed' application, which does not provide sufficient flexibility". They suggest that further research is needed in this direction. An example of bridging such a gap is shown by Machin (2002) where a

simulator and an application specific language has been developed for the artist's experimentation in the creation of an art installation. Machin reports that the offered solution was accepted very well by the artist and is a viable approach for augmenting the artists' freedom. Similarly, Biswas, Donaldson, Singh, Diamond, Gauthier and Longford (2006) also point out the need of an intermediate tool to support the designers (i.e. artists) when, together with software developers, are creating mobile context-aware applications. Such tool might also facilitate collaboration in multidisciplinary projects by minimizing the "semantic gap between the designer/artist's content design artifacts and technologist's system implementation" (Biswas et al. 2006). Many artists discuss their desire for more control over the software creation and experimentation (require to put their hands on the code), as currently available software and tools limit their creativity. However, they are not always very experienced in software development and programming. An additional software level between the artists and the low-level software development might be the way to solve the problem. Shoniregun, Logvynovskiy, Duan and Bose (2004) discuss issues like watermarking, streaming, encryption and conditional access for protecting the artworks. The authors propose a smaller granularity technique as appropriate to deal with this problem. A software engineering study on different possibilities, provision of guidelines and if necessary development of proper tools will be beneficial for the artists.

- *software development methods*. A case study discussed by Marchese (2006) shows that agile methods of software development, and more specifically the Adaptive Software Development, was suitable for the specific needs of a project in which a multimedia art installation was created. Meanwhile, Jaimes, Sebe and Gatica-Perez (2006) suggest that human-centered computing is appropriate in many cases when multimedia is used, including digital art and in museums to augment exhibitions. A further study is needed in order to understand and provide guidelines on what software development methods are appropriate when developing software within art and in what particular cases (e.g. one method might be appropriate for art installations, another for generative music; different methods might be appropriate according to the artists programming experience or programmers' specific knowledge).
- *collaboration issues* - Biswas et al. (2006), that we have mentioned above, point out that the tool they have provided (Mobile Experience Engine) is expected to also facilitate the design transfer from artists to technologies that use it. A necessity to understand the collaboration between artists and software developers in projects of common interest is discussed also in Harris (1999). For example, Harris mentions that some artists might be accustomed in working in studios instead of offices as the software developers. Different expression habits might be foreseen (e.g. artists sketching storyboards, while developers drawing software architectures); establishment of common language and understanding might be problematic. In the panel discussion of ACM Symposium on User Interface Software and Technology '98, panelists from both art and science background discussed the pitfalls, and issues of collaboration (Meyer and Glassner 1998). They pointed out that miscommunications occur frequently and due to the varying culture, col-

laboration is not always smooth. The strengthening of the collaboration between these two communities is also an aim in more recent events (CHI 2006). The organizers mention (Jennings, Giaccardi and Wesolkowska 2006) that one of the gaps between the HCI community and media artists is that the first one (HCI) is a formalized discipline, while artists prefer research-in-practice (experimentalism). Marchese (2006), also mentioned earlier, claims that the Adaptive Software Development method is useful in artist-scientist collaboration as it minimizes the management infrastructure and focuses on communication to foster collaborative problem solving. Candy and Edmonds (2002) have identified a number of factors underlying successful collaborations. These factors were evaluated against seven collaboration case studies which were conducted by the COSTART project (www.creativityandcognition.com/costart). They have also derived three models of co-creativity in the collaboration. A possible outcome of software engineering study over collaboration issues is to provide guidelines on how to ensure successful collaboration. Further outcome might include even the design of tools for fostering it.

- *business model* - In the recent years more artists are exploring the Internet as a medium to reach their audience/spectators. Nalder (2003) mentions that with the Internet as a way to transmit art the focus falls on the individual spectator, rather then museum type of art presentation. The OSS tools, developed in cooperative spirit and distributed via the internet, enable artists to experiment and create low-cost artworks. However, not all artworks are supposed to be free of charge – there are entities, like software{ART}space (www.softwareartspace.com) that are providing commercially digital artworks. The proposed by Shoniregun et al. (2004) small granularity technique for artwork protection is an example to deal with this problem. Other approaches might be proposed by the software engineering field to face this new business model of the digital/software art.

4.2 Educational Issues

- *artists' mastery of computer skills*. The need of interaction between artists and technology/technologists is discussed often in educational context. Garvey (1997) discusses the importance of the inclusion of computer graphics courses in the fine arts syllabus. However, keeping the balance between the traditional file arts skills (e.g. drawing, painting, sculpture) and digital skill mastery is of a great importance. According to the author, most of the largest companies in computer animation, film and game industry have growing requirement for personnel skilled in computer 2D and 3D modeling and animation, but give preference on strong traditional art education, stating that it is easier to teach an artist to use a computer, than a programmer to create art.

- *multidisciplinary collaboration between art and computer science students*. Multidisciplinary courses and common projects between art and science students are gaining importance. In fact, an essential part of the publications in between art and science report experiences of such initiatives. Ebert and Bailey (2000) discuss a 3D animation course with clear and well defined educational goals of such

course. The authors also report very positive outcomes: both types of students (artists and informatics) have interacted and produced not only interesting and nice looking animations and characters, but in some cases plug-ins for one of the commercial products for 3D animation (i.e. Maya). Morlan and Nerheim-Wolfe (1993) also discuss the educational goals of a course including collaboration between art and science students. These differ slightly for the two groups and apart from mastering certain skills in their own domain the course had to foster "the strengthening of interpersonal communication skills and the development of project management abilities in both art and computer science students". Parberry, Kazemzadeh and Roden (2006) describe their mainly positive experiences from a course and projects in game programming also offered to the two groups of students. The authors mention, however, that "there is a significant barrier to communication between the artists and the programmers". They also discuss practical issues that were found problematic, like the choice of game engine and incompatible file format (commercial vs. free tools). Different approaches are taken by the educators in different courses – combining lectures with hands-on work (Ebert and Bailey 2000), allowing the students to choose their project colleagues (Morlan and Nerheim-Wolfe 1993) or not. A further study might provide guidelines for optimizing the process of art and technology students working together in courses or educational projects.

- *art in computer science curricula*. Zimmerman and Eber (2001) describe a course on Virtual Reality and its "implications for computer science education". The authors state that "for the majority of the computer science students, the whole concept of artistic expression was not something they dealt with on a regular basis; simply raising their awareness and increasing their tolerance of this very different area was considered a successful outcome". Fishwick, however, shows bigger expectations about the influence of art on computer science. In his paper (Fishwick 2003) he describes the Digital Art and Science (DAS) curricula provided at the University of Florida. The author discusses the importance of combining computer science and art in the educational phase and the positive experiences obtained. The author suggests that such practices will stimulate creativity in computer science students – "A human-centered focus on experience, presence, interaction, and representation forms the core of the arts. Perhaps, then, the arts can lead computer science in new directions". The author discusses the need of interaction between art and science not only at College, but also in Master and PhD educational levels. Fishwick (2007) suggests that experiments and experience of the students in the aesthetic computing (see below) will "encourage students to design entirely new human-computer interfaces for formal structures."

4.3 Aesthetics Issues

- *aesthetic of the code* – Bond (2005) discusses several different aspects of the beauty of the software and the software as art. Firstly, he talks about Knuth's perfectionist's view of code and programming practices (Knuth 2001). With time it sometimes evolves into programs that do not have any utility value, like the one-

line programs, but are still showing the mastery of programming. Artists, however, take a different direction (the example given is a poem written in Perl) - the programming language is used to create a syntactically correct program to convey human sentiments to humans. More recently, the programs themselves are appreciated as artworks (at Transmediale - www.transmediale.de, Read_Me - http://readme.runme.org/ and other art festivals). Cramer and Gabriel (2001) state that the software "is inevitably at work in all art that is digitally produced and reproduced" and discuss its part in the aesthetics of the whole artwork. This need has led to the formation of the 'software art' movement in the artistic community.

- *aesthetic in software art* – Although in Software Art the software is considered artwork, it is not always about the beauty of the code, as in the previous section. Cramer (2002) writes "As a contemporary art, the aesthetics of software art includes ugliness and monstrosity just as much as beauty, not to mention plain dysfunctionality, pretension and political incorrectness". Manovich (2002), on the other hand, discusses the new aesthetics in the artworks created by software artists using Flash (see http://www.macromedia.com/software/flash/about/) and similar programs – "Flash generation invites us to undergo a visual cleansing". Such new aesthetics influences on the whole production of digital artifacts for the Internet.
- *aesthetics in user interfaces* – Aesthetic is addressed in the area of human-computer interaction considering interface design. Bertelsen and Pold (2004) propose aesthetics interface criticism as an alternative to the traditional assessment methods within human-computer interaction. The authors state that the interface criticism is to be done by someone with "at least some basic knowledge in aesthetics, and ideally some experience with art and literary criticism", and that the average systems engineer would not be able to perform the needed task without prior training. This shows the importance of incorporating art community in the user interface design, which is expected to lead to production of better interfaces to the developed software, but also to an expansion of the body of knowledge in user interface design.

4.4 Social and Cultural Implications of Software/Technology on Art

- software art, mentioned above, is concerned also with the cultural and social implications of the technology, and software as part of it, on the society and art. For example, Broegger (2003) discusses that one of the implications of the use of commercial software when creating an artwork is the de-facto co-authorship with the software developers (he exemplifies Macromedia). Manovich (2002) states that "Programming liberates art from being secondary to commercial media". Nalder (2003) provides a reflection on the way technology changes the society and how this affects on art. For example, with the wide use of Internet people have become afraid for their privacy and artists explore such implications and show them in their artworks. According to the author, NetArt treats the new information and communications technologies and systems "not merely as the tool of communication, but as the subject of their art".

5 Discussion

In this section we will try to summarize what we have learnt from the first iteration of the literature review in its early stage. As we have mentioned in the beginning we are looking at the intersection between art and software engineering and based on the above discussed articles we answer the following question:

How do software engineering and art intersect, i.e. how do software engineering and art influence and can involve each other?

The model we depict in Fig. 3 stems from our prior understanding of the field of software engineering and art, the knowledge that had motivated us and pushed us into this field. It has been refined by the knowledge we have developed during our literature review. In addition, it has been supported by our observations on artistic projects and web sites and participation in conferences and festivals.

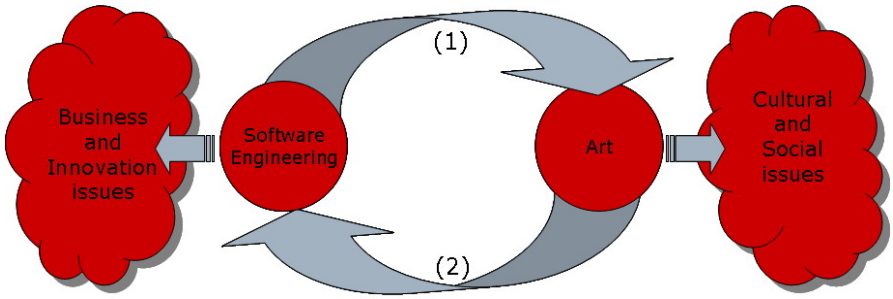

Fig. 3. Relations Between the Fields of Software Engineering and Art

The field between software engineering and art is interesting for theorists and practitioners from both areas. The review shows that both computer science researchers and contemporary artists discuss issues, experiences and problems encountered in this interdisciplinary field. The relationship is bidirectional, but the points of interest might differ according to from whose perspective we look at it.

On Fig. 3 the influences of software engineering on art is shown with the arrow (1). Contemporary artists often use digital technology in their artwork in a wide range of ways. Computer graphics, music and animation were already popular in 1980s (see PRIXARS - http://prixars.aec.at/). Artists follow closely the changes and upgrades of the technology, moving from simple digital image creation to the production of interactive installations. With the expansion of the Internet artists started exploring also the WWW technologies. There is a continuously growing need for the artists to be familiar with the computer tools provided by software engineers and developers to produce their artworks. This need is being incorporated in the art education (see the “Educational issues” in section 4), providing more and more courses for mastering skills in production of digital art (e.g. 2D and 3D modeling, computer animation, etc.). More recently interdisciplinary courses and projects are also offered, thus often putting informatics students and art students to work side-by-side. The depth in which the artists submerge into technology depends on the required functionality for the production of the artwork and on the availability of tools that

would satisfy artists' needs. In some cases when such tools do not exist or impose undesired limitations, the artists learn even to program and write their own programs (e.g. some cases in 'software art'). In other cases they turn to computer professionals for development of the specific hardware and/or software (most of the papers described in "software development issues" section above). This points to the need of particular attention of software engineering research towards the specific needs of artists.

Looking from the perspective of software engineer artists might be seen as customers that need especially designed software. In order to support and satisfy their requirements, the theories, the methods and the tools from software engineering might be used.

Interestingly, there is a trend in artistic community of returning to the roots of software, i.e. a return to programming. The 'software art' movement incorporates the idea that the artist digs into the software code for producing the artwork. Artist community acknowledges the importance of software in art and in the community in general. In fact, in 1999 one of the biggest Festivals on Digital Arts – Ars Electronica (www.aec.at) gave one of its prizes to Linux software. The returning of artists to programming, however, is not always due to appreciation of the software and with the goal to raise it as an artwork. It is sometimes due to the limitations it poses on artwork creation, as discussed in "Social and Cultural Implications of Software/Technology on Art" section. This gives us an indication that there is a serious gap in the software support of the art production. There is a need of research in order to understand the needs of the artist and provide them the appropriate tools for creating the artworks, without limiting their creative processes (see "software tools" subsection above). The research might also be directed on the processes of software development when artists and programmers work together for creating an artwork (like interactive installations). There is a need to understand what the most appropriate methods are and how to foster the collaboration between such multidisciplinary groups.

Arrow (2) on Fig. 3 shows the influences of art on software engineering. This relation is not as obvious as (1), but from the point of view of software engineers we are interested in it. The question is how can/is art influence on software development and what changes in the area of software engineering are brought by the intersection with art? This means to find out how art, artists, the collaboration with them and/or the study over existing artistic practices might help to enrich the theories, models and tools in software engineering.

One such possibility is suggested by Harris (1999) and was one of the main ideas behind the Xerox PARC Artists-in-Residence Program:

"PAIR is an opening into using some of the methodologies of art in scientific research, which is a creative activity itself and therefore is always on the lookout for new techniques to be borrowed from other professionals."

In other words, the creativity of computer scientists or software developers might be triggered by the close interaction with artists in such a manner that to propose innovative solution to specific problems, build novel tools, discover new approaches to processes, original methodologies, etc. Similar expectations might be seen in some

of the cases presenter in "art in computer science curricula" above (i.e. Fishwick 2007).

Another possibility is suggested by Briony Oates (2006a) – to explore the artists understanding of visual aesthetics, developed over hundreds and thousands of years and apply it on computer artifacts. What is meant here is that when digital visualization is involved artists might be the best judges. They might also be the best source of practical ideas for solving concrete visualization problems and/or propose innovative solutions. These issues are discussed in "aesthetics in user interfaces" subsection above. Meanwhile, there are many cases that computer scientists do not involve artists in such situations. For example, a paper presented at IEEE/WIC International Conference on Web Intelligence 2003 (Yazhong, Yueting and Yunhe 2003) proposes an approach for music retrieval based on mood detection, together with an innovative 3D visualization of the music in which the user should browse and select. The authors suggest that the visual representation of the songs should be related to some of the parameters that identify the mood of the song (e.g. the tempo). However, their study is not related with any study on how colors and moods are related. An inclusion of the artists' opinion and knowledge in such study would probably lead to much better results than the somehow confusing ones shown in the article.

6 Conclusions

Art and software engineering intersect in variety of ways. However, there is no established knowledge base for this interdisciplinary domain. This field is young and there are not established journals like in other interdisciplinary fields (e.g. medical informatics). Based on our projects (shortly presented in section 2) and our prior knowledge in this domain we have identified the need for deeper understanding of how software engineering and art intersect and how they influence and can help each other. To the best of our knowledge, this is the first attempt to make a systematic review of the state-of-the-art research in between software engineering and art.

Here we have presented the preliminary results of the literature review, showing in which directions research is being done. We have discussed our understanding of the intersection between art and software engineers and stressed on the possible benefits of such interaction for the software engineering. Creativity and innovation might be expected, together with improvements in the aesthetics direction.

We have discussed in details the process of making a systematic review of this interdisciplinary field. Our experience is that it is not a trivial task, but it is a necessary starting step for establishing the knowledge base. Some of the benefits of it are to reveal the gaps that need further research, create a collection of relevant papers and show previous positive or negative experiences and lessons learned.

Throughout the SArt project at NTNU we aim at gaining better understanding of how software engineering and art are connected by completing the systematic literature review described here. In our future work we will develop empirically based theories, models, and tools to support creativity and innovation in software technology development, fostered by the intersection with art.

References

Bertelsen, O. W. and Pold, S. (2004) Criticism as an Approach to Interface Aesthetics. Nordic conference on human-computer interaction (NordiCHI '04), Tampere, Finland, October 23-27, ACM International Conference Proceeding Series; Vol. 82.

Biswas, A., Donaldson, T., Singh, J., Diamond, S., Gauthier, D. and Longford M. (2006) Assessment of Mobile Experience Engine, the development toolkit for context aware mobile applications. ACM SIGCHI international conference on Advances in computer entertainment technology (ACE '06), Hollywood, USA, June 14-16.

Bond, G. W. (2005) Software as Art. Communications of the ACM. 48(8) August, 118-124.

Bourque, P. and Dupuis, R., eds. (2004) Guide to the Software Engineering Body of Knowledge. IEEE CS Press. ISBN 0-7695-2330-7.

Broegger, A. (2003) Software Art - an introduction. The online magazine Artificial, September 24, available online at http://www.artificial.dk/articles/software.htm, last visited on 21/03/07.

Candy, L. and Edmonds, E. (2002) Modeling Co-Creativity in Art and Technology. The fourth Conference on Creativity & Cognition (C&C'02), Loughborough, UK, October 14–16.

Cramer, F. (2002) Concepts, Notations, Software, Art. read_me 1.2 catalogue, available online at http://cramer.plaintext.cc:70/all/software_art_and_writing, last visited 13/04/2007.

Cramer, F. and Gabriel U. (2001) Software Art. American Book Review, issue "Codeworks" (Alan Sondheim, ed.), Sept. 2001, available online at http://cramer.plaintext.cc:70/all/software_art_and_writing, last visited 13/04/2007.

Ebert, D. S. and Bailey, D. (2000) A Collaborative and Interdisciplinary Computer Animation Course. ACM SIGGRAPH Computer Graphics. 34(3), 22 – 26.

Edmonds, E., Turner, G. and Candy, L. (2004) Approaches to Interactive Art Systems. In Yong Tsui Lee, Stephen N. Spencer, Alan Chalmers, Seah Hock Soon (eds), Proceedings of the 2nd International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia 2004, Singapore, June 15-18, 2004. ACM 2004, ISBN 1-58113-883-0.

Fishwick, P. (2003) Nurturing next-generation computer scientists. IEEE Computer Magazine. 36(12) December, 132 – 134.

Fishwick, P. (2007) Aesthetic Computing: A Brief Tutorial. Book Chapter to appear in Fernando Ferri (eds) Visual Languages for Interactive Computing: Definitions and Formalizations. Idea Group Inc.

Garvey, G.R. (1997) Retrofitting Fine Art and Design Education in the Age of Computer Technology. ACM SIGGRAPH Computer Graphics. 31(3), 29-32.

Glass, R. L. (1995) Software Creativity. Prentice Hall. ISBN 0131473646.

Glass, R. L. and DeMarco, T. (2006) Software Creativity 2.0. developer.* Books. ISBN 0977213315.

Greene, R. (2004) Internet Art. Thames & Hudson. ISBN 0500203768.

Grey, J. (2002) Human-Computer Interaction in Life Drawing, a Fine Artist's Perspective. Sixth International Conference on Information Visualisation (IV'02).

Harris, C., eds. (1999) Art and Innovation: The Xerox PARC Artists-in-Residence Program. The MIT Press. ISBN 0262082756.

Hart, C. (1998) Doing a Literature Review: Releasing the Social Science Research Imagination, SAGE Publications Ltd. ISBN 0761959742.

Hevner, A. R., March, S. T., Park, J. and Ram S. (2004) Design Science in Information Systems Research. MIS Quarterly. 28(1). 75-105.

Jaccheri, L. and Sindre, G. (2007) Software Engineering Students meet Interdisciplinary Project work and Art. To appear in proceedings of 11th International Conference on Information Visualisation, Zurich, Switzerland, 2-6 July.

Jaimes, A., Sebe, N. and Gatica-Perez D. (2006) Human-Centered Computing: A Multimedia Perspective. 14th international annual conference on Multimedia, Santa Barbara, CA, USA, October 23-27.

Jennings, P., Giaccardi, E. and Wesolkowska M. (2006) About Face Interface: Creative Engagement in the New Media Artis and HCI. CHI workshop 2006, Montreal, Canada April 22 - 23.

Kitchenham, B. (2004) Procedures for Performing Systematic Reviews. Keele University Technical Report TR/SE-0401. ISSN:1353-7776, available online at http://www.elsevier.com/framework_products/promis_misc/inf-systrev.pdf, last visited 13/04/2007.

Knuth, D.E. (2001) Things a computer scientist rarely talks about. In CSLI Lecture Notes. Number 136. Center for the Study of Language and Information, Stanford, CA, 2001.

Machin, C. H. C. (2002) Digital Artworks: Bridging the Technology Gap. The 20th Eurographics UK Conference, Leicester, UK, June 11-13.

Manovich, L. (2002) Generation Flash. International workshop Multimedia and the Interactive Spectator. University of Maastricht, May 16-18, available online at http://www.fdcw.unimaas.nl/is/generationflash.htm, last visited 13/04/2007.

Marchese, F. T. (2006) The Making of Trigger and the Agile Engineering of Artist-Scientist Collaboration. Tenth International Conference on Information Visualisation (IV'06).

Meyer, J. and Glassner, A. (1998) Artists and Technologists working together. The 11th annual ACM symposium on User interface software and technology (UIST '98), San Francisco, CA, USA, November 1-4.

Morlan, J. and Nerheim-Wolfe, R., (1993) Photographic Rendering of Environmental Graphics in Context: A Collaboration Between Art and Science Made Simple. ACM SIGGRAPH Computer Graphics Journal. 27(1), 10-12.

Nalder, G. (2003) Art in the Informational Mode. Seventh International Conference on Information Visualization (IV'03).

Oates, B. (2006a) New frontiers for information systems research: computer art as an information system. European Journal of Information Systems. 15. 617–626.

Oates B. (2006b) Researching Information Systems and Computing. SAGE Publications Ltd. ISBN 978141290224.

Parberry, I., Kazemzadeh, M. B. and Roden, T. (2006) The Art and Science of Game Programming. ACM SIGCSE Bulletin, Proceedings of the 37th SIGCSE technical symposium on Computer science education SIGCSE '06. 38(1).

Sedelow, S. Y. (1970) The Computer in the Humanities and Fine Arts. ACM Computing Surveys. 2(2): 89 - 110.

Shoniregun, C. A., Logvynovskiy, O., Duan, Z. and Bose, S. (2004) Streaming and Security of Art Works on the Web. Sixth IEEE International Symposium on Multimedia Software Engineering (ISMSE'04).

Wohlin, C., Runeson, P., Høst, M., Ohlsson, M. C., Regnell, B. and Wesslen, A. (2000) Experimentation in Software Engineering: An Introduction. Kleuwer Academic Publishers. ISBN 0792386825.

Yazhong, F., Yueting, Z. and Yunhe P. (2003) Music Information Retrieval by Detecting Mood via Computational Media Aesthetics, IEEE/WIC International Conference on Web Intelligence (WI), October 13-17, pp. 235 – 241.

Zimmerman, G. W. and Eber, D. E. (2001) When Worlds Collide! An Interdisciplinary Course In Virtual-Reality Art", ACM SIGCSE Bulletin, Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education (SIGCSE '01). 33(1).