Conceptual framework for the intersection of software and art

Salah Uddin Ahmed, Letizia Jaccheri, Guttorm Sindre, Anna Trifonova

Norwegian University of Science and Technology (NTNU), Trondheim, Norway

{salah, letizia, guttors, trifonova @idi.ntnu.no}

Abstract

The interaction between art and technology, especially computing technology, is an increasing trend in the recent days. The context of this intersection is growing in numbers, size and aspects each year. The number of artists participating in multimedia software or games development projects is continuously increasing and so is the number of software engineers participating in art projects like interactive art installations. As this intersection of art and technology grows, it involves people from different disciplines with varying interests creating a milieu of interdisciplinary collaborations. In this context at the Norwegian University of Science and Technology, we explore the intersection of software and art to understand different entities that are involved in the intersection. This is done by literature review and inspired by our previous experiences from participation in art projects. The objective is to conceptualize the framework of the intersection between software and art and develop a knowledge base at this interdisciplinary domain.

1 Introduction

Art finds expression in numerous products in society, where the development of products is complex, competitive, global and intercultural in scope. The literature is full with examples of artists applying mathematics, technology, and computing e.g. genetic art, algorithmic art, artificial intelligence to the creation of art. With the rapid development of technology, software is being used in almost every sector of life. The interconnection between art and computer science has a long history that dates back to the early sixties (1970) and it has interested many artists, researchers, art critics and theorists in the recent years. As the intersection is drawing attention of people from a diverse background and growing in size and scope, it is beneficiary for people interested in software and art to know each other's background and interests well. In a multidisciplinary collaboration, the success depends on how well the different actors in the project collaborate and understand each other. Both researchers and artists report problems regarding collaboration in multidisciplinary projects involving technologists and artists (Meyer, Staples, Minneman, Naimark, & Glassner, 1998). Thus understanding each other's interests and background knowledge is important for having a smooth collaboration and successful cooperation with all actors during an interdisciplinary project. The objective of this chapter is to provide a basic understanding of the interdisciplinary domain of software and art through a literature review. We describe the intersection through a conceptual framework which is represented by the different entities that we have encountered in our literature review as part of *SArt* project at the intersection of software and art. The framework is described by several entities such as *who, why, where, what* which stand, respectively, short for *who* are the people involved, *why* the people are interested to the intersection, *where* the intersection takes place, *what* tools and software are used in this intersection of software and art.

2 Background and Research Method

2.1 Background

The SArt project is conducted inside the Software Engineering group at the Department of Computer Science in the Norwegian University of Science and Technology. 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 software development in art context while facilitating collaboration with artists. Oates (2006) looks at computer art as an information system and proposes to extend IS research

agenda to include computer art. Similarly we regard the software developed in the context of art as to be considered for software engineering research and thus extend the scope of software engineering research to include research issues found in the intersection of software and art.

Since 2006, members of SArt have taken part in three interdisciplinary projects involving both artists and software engineers: *Flyndre* (http://www.flyndresang.no), *Sonic Onyx* (http://www.soniconyx.org) and *Open Digital Canvas* (http://mediawiki.idi.ntnu.no/wiki/sart/index.php/Main_Page). In the first two projects the artworks are sculptures with interactive sound systems. *Flyndre* takes as input parameters from the environment such as the local time, light level, temperature, water level and depending on these parameters creates music by exploiting algorithmic composition techniques. *Sonic Onyx* takes as input audio files and text files which are sent by users from their handheld devices such as mobiles or PDAs through Bluetooth technology. It converts those files into sound files which are later played by the sculpture. The third project, *Open Digital Canvas*, aims to embellish a white wall with a number of main boards with LEDs (Light Emitting Diodes) on them, creating a big matrix of light pixels. The project creates a platform that allows freedom of artistic expression and holds the concept of openness by keeping the hardware, software and behavior as open as possible.

Our experience shows that software engineering can play important role in such interdisciplinary projects. For example, often the developing time and budget are limited in an art project which might lead to neglecting the design of the software and lead to a quick trial and error based solution. As a consequence, the software is often created without proper architecture, thus becomes difficult for later modification and upgrade; the documentation is poor or missing which makes software reuse hard and complex. We believe that software engineering perspective can help increasing awareness in these issues. We have also observed some common interests of the artists, for example they tend to use latest technologies in their art work. For example, Bluetooth technology is used in Sonic Onyx and a variety of sensors are used in Flyndre. Artists also want publicity of their works, in all of the mentioned projects, artists wanted websites to publish their artworks. Common interest of using open source software is also noticeable from the projects.

The experience gained from the projects give us insights about some of the issues in the development of software based artworks and provides us interesting information about the artists, their ideas and interests. Furthermore, a preliminary literature investigation showed that art's relationship with software involves people from diverse background and interest, for example art critics, software developers, educators and each has his/her own interests and attitudes. In order to get a wider picture of the intersection and capture issues related to software engineering beyond software dependent art projects, we extended our focus and conducted the review

on the intersection of software and art. The objective of our literature review here in the context of SArt is to answer to the questions: where and how software and art intersect each other? Who are the important actors / entities in this intersection and how can we conceptualize the intersection with respect to these entities?

2.2 Research Method

The conceptual framework presented in this chapter is the result of our literature review which has been conducted following the systematic review process suggested by Kitchenham (Kitchenham, 2004). The details of the process have been published in our article (Trifonova, Ahmed, & Jaccheri, 2007). In this section we present the criteria that we have used to select the articles in order that the readers get an overview of the scope and the focus of our literature review. 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. Criteria 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:

- artists attitude towards software
- software engineers attitude towards art
- influence and usage of software in art
- the influence of arts in computing
- artists' and software developers' joint works such as art projects, multimedia installations, multidisciplinary courses.
- working process related issues such as collaboration problems, communication problems between artists and software developers.
- software development related issues such as maintenance, requirements, development process and CASE (Computer Aided Software Engineering) tools in context of software development projects involving artists.
- features of artistic software, their development history, usage and evolvement including both the communities of artists and software developers.

Articles that are not published in any scientific journal or conference are excluded from the review, for example, online articles, articles accompanying art festivals and workshops were excluded on account of this reason. We want to mention here that the conceptual framework presented later in this chapter includes not only reviewed articles from our literature review, but also includes relevant books and knowledge gained from our participation into several art festivals, art projects and conferences.

3 The Intersection of Software and Art

The intersection between software and art includes people with varying backgrounds such as artists, developers, critics who involve themselves into the intersection with purposes that vary for each of them. This leads to the people at the intersection having different viewpoints on different issues in this interdisciplinary domain. For example, a software engineer has a different purpose/interest than that of an art critic when he/she involves him/her self in the intersection. In the following section we describe the intersection of software and art in the view of the entities that exist in the intersection, such as actors (who are involved?), interests/viewpoints (Why are the people interested?), places (where this involvement takes place?), tools and technologies (what tools and technologies has bound this relationship?).

3.1 Who (are the people involved in this domain)

The people involved in the intersection of software and art can be identified as artists, designers, software developers, software engineers, theorists, critics and researchers. It should be mentioned here that these categorizations are based on the roles of the individuals involved and they are not mutually exclusive. This list is neither complete nor exhaustive; rather it covers the roles that we have found in our literature review. One person can have many roles at the same time, for example, when the interactive filmmaker Florian Thalhofer creates interactive documentary software Korsakov; he is both an artist and a software developer (Blassnigg, 2005).

3.1.1 Artists

Artists here refer to those artists who are using software for realizing some part of their artwork. This includes new media artists, photographers, filmmakers, curators, animators and all others who utilize software to accomplish their work. Many artists use software technology not directly for their work, but for publishing their work or communicating with others through web tools and technologies. It is interesting to note that some of the

artists coin the term ‘info-architect’ to refer to the artists working with new media technologies (Blassnigg, 2005).

Artists, together with theorists and art critics (see below) are more involved in discussions about the intersection of software and art compare to the software practitioners. For example, according to Bond (2005), it is the new media artists, not the software practitioners that take part in the movement of software art. This movement of software art refers to the viewpoints of the artists who believe that software itself, that is, the raw code of software that works behind every software application should be treated as a medium or material of art. Bond (2005)states, “Casting software as an artistic medium might strike many readers as odd, or even objectionable, but there is a growing body of evidence to show that it is perceived and utilized in just this way” (p. 118). Theorists Florian Cramer and Ulrike Gabriel extend the view of software art by focusing on the underlying code (Cramer & Gabriel, 2001).

3.1.2 Theorists and Art Critics

Development of digital, information and communication technologies has built a complex corpus called cyber-culture. “Media and multimedia information and communication technologies generate new promises, problems and threats; and artists undertake efforts to examine this emerging area that has been repeatedly considered as a post biological syndrome. In other words artists do not only use media technologies but also scrutinize and challenge them” (Kluszczynski, 2005, p. 124). Thus, in the intersection of software and art we find a number of theorists and critics as well. As an example, Erkki Huhtamo, Mathew Fuller, Florian Cramer, Jeffery Cox, Lev Manovich are a few of the names to list in this category. Many of the people mentioned here have several roles, varying from artist, teacher, theorist and programmer. For example Erkki Huhatamo is a lecturer, researcher, writer and curator all by the same time. Manovich is a lecturer and writer of many articles and books. His book, “The Language of New Media” is considered by many reviewers to be the first rigorous and far reaching theorization of the subject. Even though there might not be a person who can be termed as only theorist, we mention them as a separate category here as we find a significant portion of research articles that we have reviewed are contributed by these theorists and art critics.

3.1.3 Software Engineers

In context of the intersection between software and art, by software engineers we mean any person who is involved in the development of what we call 'artwork support tools', i.e. software developed for the artists and intended to be used for some art purposes. These software engineers or developers might be students who follow some interdisciplinary courses and/or work together with artists in art projects. In many cases, software developers take part in art projects and work with artists for a short period. Sometimes artists working with digital media recruit their personal developers, for example, Paris based artist Christophe Bruno employs his personal software developer to realize his concepts with the help of software (http://www.christophebruno.com). However, there are new media application developing groups or companies run by artists that recruit software developers for long time collaboration, example of which might be the Mediamatic Lab in the Netherlands (www.mediamatic.nl) and Soundscape Studios in Norway (www.soundscape-studios.no).

In addition, there are some software engineers who do not have direct involvement with artists, but are involved with the debate of the role of art versus science in software engineering. The debate focuses on the creativity and innovativeness in software engineering discipline. For example, referring to the process of solving complex problems where there is no perfect solution achievable through rigorous methods of science, Bollinger (1997) states "the artistic part of this process lets science move unexpectedly into new currents and previously unmapped understanding" (p. 125). It is interesting to see that the panel discussion of OOPSLA 2004 was titled "Software Development: Arts & Crafts or Math & Science?" But there are also others who think different, for example, McConnell says that this debate was appropriate at the beginning, at least 30 years back when it started, as at that time software engineering didn't have an established body of knowledge. But now, the author recommends that it should be called engineering as there is significant body of knowledge (McConnell, 1998). A viewpoint combining the both parties could be "Building software is a complex and exciting task that is a unique confluence of engineering, mathematics and artistic insight, and it is important we resist the tendency to view it in a single dimension"(Wei-Lung, 2002, p. 29).

There are other software engineers who refer some software as "art for art's sake". Knuth includes in this categorization programs which are appreciated in light of the challenge the programmers face in creating it, for example one line programs, programs that output themselves, etc. (Bond, 2005). Knuth also compares the

beauty of a program to the beauty in literature or music – programs whose tasks are stated elegantly and whose parts come together symphonically are beautiful programs.

3.1.4 Researchers

Many people are involved with software and art for the purpose of research in areas such as user interface, collaboration, creativity and innovation. These researchers may come both from art and computing disciplines. Human Computer Interaction (HCI) community is interested in aesthetics of user interface. Bertelsen and Pold assert that interacting with the interface is a cultural activity. They propose an approach to interface design in which the interface should be 'criticized' by someone with knowledge in aesthetics and ideally some experience with art and literature. This approach might be used for increasing the aesthetic value in the user interface (Bertelsen & Pold, 2004).

Creativity and Cognition Studios research group is an active group interested and involved in the intersection of software and art. Researchers like Linda Candy, Ernest Edmonds and many others working in the Creativity and Cognition Studios have contributed in numerous collaborative research works involving art and technology. In fact, creativity and cognition conference which was instigated by Ernest Edmonds and Linda Candy has turned into a regular event in Association of Computing Machinery's SIGCHI calendar.

There are also many art researchers who have addressed significantly the collaboration between artists and technologists. For example, UIST 98 panel discussion was about artists and technologists working together where artists like Michael Naimark, Loretta Staples were in the panel with the technologist-researchers Jon Meyer, Andrew Glassner and Scott Minneman (Meyer et al., 1998). Individual researchers and researchers whose main stream research is not focused on 'software and art' are also found to be interested in the intersection. An example is Briony J. Oates who proposes to extend the scope of information systems' research by including computer art (Oates, 2006).

3.2 Why (art and software need interaction)

Software engineers and artists work together for many reasons, varying from personal interests to the more general interests of their respective disciplines. In this section we highlight some of the reasons that we have found in the literature review that brings artists together with the developers.

3.2.1 Artists need tools support

One of the main reasons artists seek help from the technologists is to get support with the tools that they need for the realizations of their artwork. "… they [artists] are often ill equipped to work with complex technologies. It is this factor that may incline any artist wishing to work within the domain of contemporary Art & Technology and digital art towards collaboration as a necessary means for realising their intentions" (Jones, 2005, p. 76).

Supporting artists with necessary tools invoke many questions, such as: What kind of tools artists want? What desired features might be included in these tools? For what purposes the available tools are used? and so on. Many researchers have addressed issues related to the creation of tools to support artists, designers or creative people in general. Here are some examples: (Machin, 2002) aims at reducing the gap between developers and artists and enhancing creativity; Warr and O'Neill (2007) focus on visualization at early stage of collaborative work, supporting individual creativity; Mamykina et al. (2002) discuss the importance of the ability to track progress and revisit design decision. Referring to Macromedia Director Machin (2002) states "this kind of software provides the artists with a powerful tool which assist in visualising the piece even at a very early stage of its design" (p.3). Biswas et al. (2006) described that a development toolkit (for new media content design) should support separate design (by the artists) and implementation (by software developers) of the final application. Thus, it should be able to decrease the semantic gap between artists and technologists and assist their dialogue. Gross (2005) believes that when it comes to build software tools for artists, pragmatics analysis (context and behaviour) is crucial because art is heavily immersed in practice and action, and because art is valued on its ability to communicate.

3.2.2 Art projects need software engineering

Software developers who have the experience of working together with artists in artistic projects have realized the need of software engineering knowledge in those projects. This is what we have also realized from our participation into the art projects. Machin (2002) states "What is required now is to carry out further work in order to establish methods that can be utilized in future requirements capture software. By working with artists in the installation art field, we can seek out ways to enable the capture of the artist's ideas without inhibiting the artistic process." (p. 8). Many researchers/software engineers are also interested in comparing the software development method and analyze which ones suit better an art project in a certain context. Furthermore, Candy and Edmonds (2002) investigate what are the most appropriate evaluation methods in software intensive art and

if the evaluation should be done only by the artists or should include the software engineers as well. Where the artworks are implemented in limited time and budget and where artists leads the project, the maintenance and upgrading issues are often overlooked. Thus the maintenance and upgrade of these kinds of software supported artworks becomes one of the prime sectors where art projects need software engineering help.

3.2.3 Computing needs aesthetics

Fishwick (2005) reports the result of a survey on the usefulness of aesthetic methods on several areas of computer science. The result shows that data structure, algorithms, digital logic, computer architecture was chosen by the respondents as some of the fields where aesthetic computing can be used. Information visualization and software visualization are other fields that can contribute to bringing art/aesthetics inside of computing (P. A. Fishwick, 2007).

Adams (1995) addresses the importance of teaching aesthetics in engineering education and the role of aesthetics in engineering. Paul Fishwick has coined a term "Aesthetic Computing" to refer to a new area of study, which is concerned with the impact and effects of aesthetics on the field of computing. As an example, the discrete models found in computing can be transformed into visual and interactive models which might increase the understanding of the students. Fishwick et al. (2005) represents a method for customizing discrete structures found in mathematics, programming and computer simulation. Based on the method, they transform some discrete models (for example finite state machine) to geometric models and assess students' perception on how customized models affected their understanding and preferences regarding visual and interactive model representations.

3.2.4 Technology changes art's medium, audience and business

In the recent years more artists are exploring the Internet as a medium to reach their audience/spectators. When Internet is used as a way to transmit art the focus falls on the individual spectator, in contrast from the museum type of art presentation (Nalder, 2003). The Open Source Software tools developed in cooperative spirit and distributed via the internet, enable artists to experiment and create low-cost artworks. Besides the new way of reaching the target audience, Internet and Web has created new business models as well. For example, softwareARTspace (http://www.softwareartspace.com) is a website which provides digital artworks commercially. Usage of information technologies implies further improvements of tools and technologies to

address the issues related to business and publicity, such as protection of artworks and copyrights. Thus, it brings artists and technologists together to solve these issues. As an example, small granularity technique has been proposed by Shoniregun et al. (2004) for the protection of artwork. The potential reciprocal interaction between artists and technologists in the form of demands of the user (artists) which stimulate an engineer to extend the technology in some way thus extends the possibility of its use (Jones, 2005). This helps both technology and art to co-evolve in a configuration of mutual interdependence.

3.3 Where (the intersection happens)

The influence of technology on art has also created a number of new genres of arts, such as internet art, generative art, new media art, net art, software art and many more (Nalder, 2003). Many of these fields might still not be well defined or still at an early stage of evolvement. Professor Peter Weibel coins all of these fields by a single term "Computer art". For example, he states in the jury statement at the Ars Electronica '92, "Computer art is concerned with artworks that 'were impossible to produce before the invention of the computer … even unimaginable" (Oates, 2006). Some sectors of art where interaction between software and art happens extensively are computer generated sound and electronic music, visual arts, virtual reality, performance arts etc. Interdisciplinary events and activities are often supported by institutions or organizations. Thus, even though software is accessible to individual artists, it is not unusual to find most of the cases of intersection described in the reviewed articles have occurred in the context of some institutional support. Here we present some major places/sectors that we have identified from the literature where art and software meet each other.

3.3.1 In Educational Institutes

In art schools and computing schools interdisciplinary courses are conducted which include students from both art and computing discipline. Besides these interdisciplinary courses, there are also cases where the need of computing education is realized in the art discipline, and the need of art education in the computing discipline.

In Art Schools

Donna (1991) mentions "It is a fundamental mistake for a university not to provide such training [computer training] to non computer science majors because of the rapid growth of computing in arts and humanities" (p.2). The importance of the inclusion of computer graphics courses in the fine arts syllabus is recognized by art

institutes (Garvey, 1997; Sardon, 2006). However, keeping the balance between the traditional fine arts skills (e.g. drawing, painting, sculpture) and digital skill mastery is of a great importance. Designing and conducting multidisciplinary courses need consideration of special issues such as ensuring good team work, having common language and understanding each other's skills and abilities as well as having respect for each other's domain. While presenting their experience of organizing and conducting an interdisciplinary computer animation course in Ebert and Bailey (2000) mention the importance of effective teamwork and learning from each other's disciplines.

In Computing Schools

Parberry et al. (2006) describe their mainly positive experiences from a course and projects in game programming offered to the two groups (art and computer science) of students. Argent (2006) describes two game development courses offered at the University of Denver which were built upon a four way partnership between computer science, digital media studies, electronic media arts design and studio art. The role of teaching aesthetics to engineering students was addressed by Adams almost 12 years earlier (Adams, 1995). Zimmerman et al. (2001) describe a course on Virtual Reality and its "implications for computer science education". Jaccheri et al. (2007) reports their experience of running an interdisciplinary course for three years where they have reported that such an interdisciplinary course gives software students learning outcomes that are quite different from what they get from more traditional software engineering team projects, in particular concerning interdisciplinary skills and self insight. Fishwick (2003) describes the Digital Art and Science (DAS) curricula provided at the University of Florida where he discusses the importance of combining computer science and art in the educational phase and the positive experiences obtained. He suggests that such practices will stimulate creativity in Computer Science students.

3.3.2 In Software Industry

Art meets with software in the software industry as well, for example in building entertainment and leisure related applications, games, and demo-scenes. Computer based entertainment and leisure related applications range over a wide spectrum evolving from console based ones to augmented/mixed reality, pervasive, ubiquitous, immersive, context aware and social computing experiences. The involvement of artists and developers takes place in the industry in the following way: technologists develop more intelligent interactive context-aware systems while designers, artists and design-art researchers use the existing technologies as a "new

media" to design with, and deliver "values" and "experiences" which go beyond the notional functionalities of technology (Biswas et al., 2006). The major areas where the collaboration takes place can be identified as Game Development, Human Computer Interaction, User Interface Design and Web Design. Game development is an interdisciplinary field which requires both technical and creative appreciation. Games are considered as a rich field for art not only because of the design, but also because of the cultural issues and perspectives that might be identified within the games (Blais & Ippolito, 2006).

3.3.3 In Research Institutes

Apart from the industry and the art or computing schools, there are also many research institutes where a research setting is intentionally created for artists and technologists to work closely together. Even though these research institutes may be a part of a university or an industry, the objective of research setting is different from general purpose art projects. Often, this kind of collaboration is done through “Artist-in-residence” programs, for example, the Xerox PARC artist-in-residence program (Harris, 1999), COSTART project (Candy, 1999). The objectives of these programs are different, in a way that they are aimed for innovation, creativity or elegant solutions of a complex problem, whereas in a general art project, the objective is to realize an art work without any explicit intention of innovation. Many industries also arrange artists in residence program from time to time. This situation differs from the one where artists work in the same enterprise with other technologists as part of regular production work.

3.3.4 In Art Projects

In public art especially new media art projects, art meets software for the realization of the projects. Sometimes artists learn to use or code the software by themselves; sometimes they work together with the technologists to get the software developed. The art projects can be creation of simple piece artwork such as internet art, software art, and digital art and so on where a single artist works with the software. On the other hand it can be an interactive art installation, or a multimedia presentation, film or electronic music where one or more artist work with software and collaborates with other technologists.

3.3.5 In Art Centres and Festivals

Art is promoted by many festivals where artists get chance of exhibiting their works to the audience, communicating ideas and concepts, and evaluating and/or criticising artworks. Many art festivals along with art institutes promote new media art which lies at the intersection of art and software. Examples include Ars

Electronica (http://www.aec.at/de/index.asp), PixelACHE (http://www.pixelache.ac/), Read_me (http://readme.runme.org/), Transmediale (http://www.transmediale.de), Piksel (http://www.piksel.no/), Make Art (http://makeart.goto10.org/2007/), Trondheim Matchmaking (http://matchmaking.teks.no/). Competitions like PrixArs (International competitions for Cyber Arts organized by Ars Electronica) offers prizes on different categories such as computer animation, films, interactive art, digital music, hybrid art, digital communities and media art research. ARCO BEEP new media art awards targets the goal of advancing the production and exhibition of new media art and art linked to new technologies (www.arco.beep.es). PixelACHE presents projects experimenting with new media and technology with a goal to act as a bridge between the traditional creative discipline and the rapidly developing electronic subcultures. Other institutes and festivals mentioned here also stress importance on media art and use of technology.

3.4 What (software tools are used in this domain)

After we have identified people (who), reasons behind their interest at the intersection (why) and the places/sectors (where) art intersects with software, here we present some practical examples of what (tools and technologies) binds the relationship between software and art. Artists tend to use software for different purposes. Quite often they use commercial software; often they are interested in open source software as a cheap alternative. In few cases, artists develop their own software. Most of the time they use the software as it was intended to be used by the creator of the software but sometimes they can be creative and use it in a different way which was not intended. For example the artist Jen Grey used the proprietary software *Surface Drawing* in a unique way to draw live models, a purpose which was not intended (Grey, 2002). Some software is used as a tool to develop artwork; some as a media to support artists' activities indirectly (for example collaboration) while others are general purpose programming languages used to build applications. Besides these, there is also customized software i.e., software that is built for a specific artistic purpose. Several articles mention this kind of software (example, Datareader, Korsakov mentioned in the table below) which was developed by either artists alone, or with the help of programmers as part of an art project. In this section we list the software/tools that were mentioned in the literature. These tools provide the reader an overview of what type of software and tools are used or required by the artists.

Artwork support tools, i.e. tools used to develop artworks are mainly special purpose artistic software which specializes on some tasks such as visualization, sound manipulation or animation. The following table gives a list of these kinds of software that were mentioned in different articles found in our literature review along with

their purposes/ functionalities. The list gives the readers an idea about the tools that artists are using/ interested in using for different art purposes. The list also contains customized tools that were developed for some specific artwork or art purpose, for example Datareader was developed for taking as input meteorological data.

Table1. Software tools mentioned in different articles

Category	Software Name	Description	Referenced in
1. Graphics Manipulation	Illustrator	Vector based drawing program	(Garvey, 1997)
	Freehand	Tool to create layouts and illustrations for print /Web designs	(Garvey, 1997)
	CorelDraw	A vector graphics editing software	(Garvey, 1997)
	Photoshop	A graphics editor software	(Grey, 2002)
	Painter 6.0	Painting tool for graphic designers and fine artists.	(Grey, 2002)
2. Multimedia Authoring	Flash	Animation and interactivity	(Sung-dae, Jin-wan, & Won-Hyung, 2006) (Sardon, 2006) (Marchese, 2006)
	Director	Creating interactive content for fixed media and internet	(Garvey, 1997) (Sardon, 2006) (Machin, 2002)
3. 3D graphics Manipulation	SoftImage 3D	3D animation software for games, film and television	(Jennings, Giaccardi, & Wesolkowska, 2006)
	3DStudio Max	Professional 3D modeling, rendering and animation software	(Garvey, 1997) (Strömberg, Väätänen, & Räty, 2002)
	Mini CAD, Auto CAD	A suite of CAD (Computer Aided Design) software products for 2- and 3-dimensional design and drafting	(Garvey, 1997)
	Alias /Wavefront	3D graphics software	(Garvey, 1997)
	WorldUp	Software development environment for building 3D/VR applications	(Zimmerman & Eber, 2001)

	Maya	High end 3D computer graphics software	(Zimmerman & Eber, 2001) (Steinkamp, 2001)
	Breve swarm simulation	A package for building 3D simulations of multi-agent systems and artificial life	(Boyd, Hushlak, & Jacob, 2004)
4. Sound Manipulation	Pure Data	Tool for creating interactive computer music and multimedia works	(Jennings et al., 2006)
	Max/MSP	A graphical programming environment for music and multimedia	(Jennings et al., 2006) (Marchese, 2006) (Edmonds, Turner, & Candy, 2004)
	GigaStudio 160	Software for music and sound effects	(Strömberg et al., 2002)
5. Video Manipulation	SoftVNS video toolkit	A real time video processing and tracking software for MAX/MSP	(Edmonds et al., 2004)
	Jitter	Extends MAX/MSP to support real time manipulation of video data	(Polli, 2004)
	Korsakov	Interactive Documentary software	(Blassnigg, 2005)
6. Other Applications	ELE (Expressive Lighting Engine)	Control Lighting for a virtual environment	(Gross, 2005)
	Mobile Bristol toolkit	A tool to create and share mobile, location–based media	(Biswas & Singh, 2006)
	Mobile Experience Engine	A software development platform for creating advanced context-aware applications and media-rich experiences for mobile devices	(Biswas & Singh, 2006)
	Sculpture simulator	A sculpture simulator with its own programming language	(Machin, 2002)
	Particle dynamics	A software tool set for simulating natural phenomena.	(Steinkamp, 2001)
	Datareader	A Max/MSP object to read text data	(Polli, 2004)

		and convert them into sound	
7. Programming languages /API	VRML	Virtual Reality Modeling language	(Nalder, 2003)
	General purpose programming languages such as python, C++, Java etc.	To create wide range of applications	(Nalder, 2003)
	OpenGL	Industry standard and API for high performance graphics	(Fels et al., 2005)

Apart from the artwork support tools there are other tools and software that artists use for supporting other activities such as communication, publicity, sharing works, ideas etc. Internet and Web tools has become not only a medium for the artist to publish and present their work and activities but also a medium for communicating and collaborating with other artists. “The digital arts site Rhizome is recognized for the crucial role it plays enabling exchange and collaboration among artists through the network” states Walden in his review on the book Net_Condition: Art and Global Media (Walden, 2002). The tendency of publishing artists work through websites to reach audience was also observed during our participation in the three projects described in section 2. The other purposes of website includes, publishing artworks, selling art products, virtual tour of museums and creating online communities, discussion groups or forums, and blogging.

Domain specific programming language is preferred by the artists compared to the general purpose programming languages unless the artist does not aspire to be a professional programmer. This is because general languages can be daunting due to the steep learning curve associated with learning programming. Besides, artists often prefer to work with intermediate tools where the need for programming is reduced. But that does not make any limitations for artists to learn the general purpose programming languages. In some of the articles that we have reviewed mentions a number of general purpose languages which was used to realize artworks or some artistic software, for example, C++, ActionScript, UML, 2D OpenGL.

Last but not the least, the role of open source software has to be mentioned as an important factor for making artists more interested to software. Artists tend to move towards using open source technology not only because they are cheap, even free of charge, but also because many artists believe in the open source ideology. Halonen (2007) mentions that new media art is based on cooperation to a greater degree than many art forms that can be

created alone. He identified four groups with diverse motives: i) using open source network as an important reference for professional image, ii) using open source projects as a platform for learning, iii) an opportunity to seek jobs and iv) enrich professional networks. From our project experience, we identified that some artists want to have open source projects so that they can build an interested community around the project which might assist in the further development, upgrade and maintenance of the project at a low cost (Flyndre and SonicOnyx). Open source and free software usage in artists community is also encouraged by different art festivals such as piksel (http://www.piksel.org), makeart (http://makeart.goto10.org/). The interest is also visible by the activities of different art organizations/institutes such as APO33 (http://apo33.org) ap/xxxxx (http://1010.co.uk/) Piet Zwart Institute (http://pzwart.wdka.hro.nl/).

3.5 The Framework

In the earlier sections we have presented the framework of the intersection in terms of different entities in the intersection, i.e. people and views, place and tools. Here the framework is presented through a table (table2). In the table, rows represent where (software and art meet) column represent who (are the actors), entry in each cell represent the reason why the actors participate in a given place.

Table2: Where, who and why dimension of the intersection of software and art.

Who / Where	Artists	Software Engineers	Researchers	Theorists & Art Critics
Educational Institutes (Computing and Art schools)	4, 5, 10	5	1, 2, 3	X
Research Institutes	3, 17	3,6, 17	3, 18, 17	X
Software Industry	8,9	7, 8, 9	7	X
Public Art, Art Projects	10	11, 12	18	16
Festivals	13, 14, 15	12, 15	17	16

Here we present the list of reasons (Why):

1. Design and conduct interdisciplinary courses
2. Conduct research and disseminate knowledge of conducting interdisciplinary courses
3. Foster innovation and creativity
4. Learn technology
5. Learn to work in multidisciplinary projects
6. Apply aesthetics in computing
7. Do research and development of products
8. Design user interface and enhance/improve human computer interaction
9. Enhance user experience
10. Use technology in artworks
11. Realize the artwork through software
12. Provide tools and technology support
13. Share and exchange knowledge
14. Exhibit art work in public
15. Extend collaboration between artists and software engineers
16. Follow the changes of art and their social and cultural effects
17. Present research works
18. Conduct research on artists-technologists collaboration and reduce the gap between them

Different people have different reasons to involve themselves with the interdisciplinary domain. The fact that different actors have different viewpoints is largely due to their varying roles and background knowledge. That is why, while we see artists picking up different software tools and technologies and even the naked codes as a material for artworks, software engineers are seen debating over the role of art in software engineering.

4 Future Trends

The interaction between software and art is an increasing trend followed by many artists, software developers and technologists. As the technology advances, more and more sectors of life are influenced by the use of technology. Art finds its way to reflect on every artefact of life, so the intersection of art with software takes place naturally as it happens with other technologies. The application of different computing tools and

technologies has already been established in arts. The trend to include the aesthetics in computing, especially in aspects of design is a recent but growing trend. In future when the computing speed and network speed will increase further, the value of aesthetics will be even more recognized in computing (Hoffmann & Krauss, 2004). Until now aesthetics is recognized mostly in human computing interaction, but in future other fields of computing such as data structure, algorithms, digital logic, computer architecture might be also using more aesthetics as anticipated by Paul Fishwick (2005).

As time moves on and the interdisciplinary domain matures, this collaboration between art and computing attracts not only more people but also more disciplines and it gives birth to new issues and new perspectives. For example, the need of software technology for artists leaded to the inclusion of different computer courses in art and computing institutes. This in turn has raised the pedagogic perspectives of conducting interdisciplinary courses involving art and computing students. In future this might also raise the importance of learning technology through art and fun especially for children. Software dependent arts such as installation arts that are placed in public space attract the attention of many viewers including children and old people. Thus it involves social and cultural aspects which include perspectives of learning, consciousness in the society through computational arts. In future when there will be more software dependent arts placed in public space, it will be interesting to consider these findings, in relation to the different people and different perspectives of life surrounding computational arts as well as building a knowledge base in this growing interdisciplinary domain.

5 Conclusion

In this article we present an outline of different entities (who, why, where, what) at the interdisciplinary research domain of software and art. The interaction between art and software happens naturally as technology finds its way to influence every sphere of our life and artists reflect, scrutinize and challenge technology at the same time as they use it for extending their expressions. Artists working with technology are faced with multiple tasks that demand them to perform different roles such as researcher, engineer, programmer etc. In many cases artists have background knowledge of technology from their previous career or education, in other cases they learn a bit of the technology while working with it. But in general case, artists require support from the technologists to realize their visions. This brings artists together with technologists and the whole phenomenon brings together other professionals who are interested or get involved with this intersection of technology and art. The conceptual framework that we present in this chapter provides a detailed insight of how the intersection between software and art is. This intersection is mostly based on the articles from our literature survey and our

experience from art projects. So it might not include all the actors and entities in this interdisciplinary domain. We have limited our literature review to scientific publications, but many recent trend and relevant information in the intersection of art and software is available in sources such as websites, online articles, and artists' biographies. The framework might look different if we include these sources.

Even though art has connection with software since a long while, but the intersection between art and software is a very recent trend and it is continuously changing and growing. Thus, in future this framework will have more entities to include. The conceptual framework adds value to the knowledge base of the interdisciplinary domain by structuring the information about the intersection. A knowledge base is a necessary element for an interdisciplinary domain based on which the domain can prosper and grow. The conceptual framework will thus act as basis for getting into detailed understanding of this domain. Future work regarding this conceptual framework might include extension and improvement which will further enhance the knowledge base of this interdisciplinary domain.

6 References

Adams, C. C. (1995). *Technological allusivity: appreciating and teaching the role of aesthetics in engineering design.* In Proceedings of the Frontiers in Education Conference, 1995. (pp. 3a5.1-3a5.8). Washington, DC, USA: IEEE Computer Society.

Argent, L., Depper, B., Fajardo, R., Gjertson, S., Leutenegger, S. T., Lopez, M. A., et al. (2006). Building a game development program. *Computer, 39*(6), 52-60.

Bertelsen, O. W., & Pold, S. (2004). *Criticism as an approach to interface aesthetics.* In Proceedings of the third Nordic conference on Human-computer interaction (pp. 23-32). New York, NY, USA: ACM Press.

Biswas, A., Donaldson, T., Singh, J., Diamond, S., Gauthier, D., & Longford, M. (2006). *Assessment of mobile experience engine, the development toolkit for context aware mobile applications.* In Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology (pp. 8). New York, NY, USA: ACM Press.

Biswas, A., & Singh, J. (2006). *Software Engineering Challenges in New Media Applications.* In the Proceedings of Software Engineering Applications (~SEA 2006~) (pp. 7). Dallas, TX, USA: ACTA Press.

Blais, J., & Ippolito, J. (2006). *At the Edge of Art*. London: Thames & Hudson Ltd.

Blassnigg, M. (2005). Documentary Film at the Junction between Art and Digital Media Technologies. *Convergence-The International journal of New Media Technologies, 11*(3), 104-110.

Bollinger, T. (1997). The interplay of art and science in software. *Computer, 30*(10), 128, 125-127.

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

Boyd, J. E., Hushlak, G., & Jacob, C. J. (2004). *SwarmArt: interactive art from swarm intelligence.* In Proceedings of the 12th annual ACM international conference on Multimedia (pp. 628-635). New York, NY, USA: ACM Press.

Candy, L. (1999). *COSTART Project Artists Survey Report: Preliminary Results.*: Loughborough University.

Candy, L., & Edmonds, E. (2002). *Modeling co-creativity in art and technology.* In Proceedings of the 4th conference on Creativity & cognition (pp. 134-141). New York, NY, USA: ACM Press.

Cramer, F., & Gabriel, U. (2001). Software Art and Writing. *American Book Review, 22*(6).

Donna, J. C. (1991). Interdisciplinary collaboration case study in computer graphics education: “Venus & Milo”. *SIGGRAPH Computer Graphics, 25*(3), 185-190.

Ebert, D. S., & Bailey, D. (2000). A collaborative and interdisciplinary computer animation course. *ACM SIGGRAPH Computer Graphics, 34*(3), 22-26.

Edmonds, E., Turner, G., & Candy, L. (2004). *Approaches to interactive art systems.* In Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia (pp. 113-117). New York, NY, USA: ACM Press.

Fels, S., Kinoshita, Y., Tzu-pei Grace, C., Takama, Y., Yohanan, S., Gadd, A., et al. (2005). Swimming across the Pacific: a VR swimming interface. *Computer Graphics and Applications, IEEE, 25*(1), 24-31.

Fishwick, P. (2003). Nurturing next-generation computer scientists. *Computer, 36*(12), 132-134.

Fishwick, P. (2005). Enhancing experiential and subjective qualities of discrete structure representations with aesthetic computing. *Journal of Visual Languages & Computing, 16*(5), 406-427.

Fishwick, P., Davis, T., & Douglas, J. (2005). Model representation with aesthetic computing: Method and empirical study. *ACM Trans. Model. Comput. Simul., 15*(3), 254-279.

Fishwick, P. A. (2007). Aesthetic Computing: A Brief Tutorial. In F. Ferri (Ed.), *Visual Languages for Interactive Computing: Definitions and Formalizations*: Idea Group Inc.

Garvey, G. P. (1997). Retrofitting fine art and design education in the age of computer technology. *ACM SIGGRAPH Computer Graphics, 31*(3), 29-32.

Grey, J. (2002). *"Human-computer interaction in life drawing, a fine artist's perspective".* In Sixth International Conference on Information Visualisation (IV'02) (pp. 761-770). Los Alamitos, CA, USA: IEEE Computer Society.

Gross, J. B. (2005). *Programming for artists: a visual language for expressive lighting design.* In IEEE Symposium on Visual Languages and Human-Centric Computing, 2005 (pp. 331-332). Los Alamitos, CA, USA: IEEE Computer Society.

Halonen, K. (2007). Open Source and New Media Artists. *Human Technology - An interdisciplinary journal on humans in ICT environments, 3*(1), 98-114.

Harris, C. (Ed.). (1999). *Art and innovation: the Xerox PARC Artist-in-Residence program.* Cambridge, Massachusetts: MIT Press.

Hoffmann, R., & Krauss, K. (2004). *A critical evaluation of literature on visual aesthetics for the web.* In Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries (pp. 205-209). South Africa: South African Institute for Computer Scientists and Information Technologists.

Jaccheri, M. L., & Sindre, G. (2007). *Software Engineering Students meet Interdisciplinary Project work and Art.* In Proceedings of the 11th International Conference on Information Visualisation (pp. 925--934). Washington, DC, USA: IEEE Computer Society.

Jennings, P., Giaccardi, E., & Wesolkowska, M. (2006). *About face interface: creative engagement in the new media arts and HCI.* In CHI '06 extended abstracts on Human factors in computing systems (pp. 1663-1666). New York, NY, USA: ACM Press.

Jones, S. (2005). *A cultural systems approach to collaboration in art & technology.* In Proceedings of the 5th conference on Creativity & cognition (pp. 76--85). New York, NY, USA: ACM Press.

Kitchenham, B. (2004). *Procedures for Performing Systematic Reviews*: Keele University Technical Report TR/SE-0401 and NICTA Technical Report 0400011T.1.

Kluszczynski, R. W. (2005). Arts, Media, Cultures: Histories of Hybridisation. *Convergence-The International Journal of New Media Technologies, 11*(4), 124-132.

Machin, C. H. C. (2002). *Digital artworks: bridging the technology gap.* In Proceedings of the 20th Eurographics UK Conference, 2002. (pp. 16-23). Washington, DC, USA: IEEE Computer Society.

Mamykina, L., Candy, L., & Edmonds, E. (2002). Collaborative creativity. *Communications of the ACM, 45*(10), 96-99.

Marchese, F. T. (2006). *The Making of Trigger and the Agile Engineering of Artist-Scientist Collaboration.* In Proceedings of the conference on Information Visualization (pp. 839-844). Washington, DC, USA: IEEE Computer Society.

McConnell, S. (1998). The art, science, and engineering of software development. *IEEE Software, 15*(1), 120, 118-119.

Meyer, J., Staples, L., Minneman, S., Naimark, M., & Glassner, A. (1998). *Artists and technologists working together (panel).* In Proceedings of the 11th annual ACM symposium on User interface software and technology (pp. 67-69). New York, NY, USA: ACM Press.

Nalder, G. (2003). *Art in the Informational Mode.* In Proceedings of the Seventh International Conference on Information Visualization (pp. 110). Washington, D.C, USA: IEEE Computer Society.

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

Parberry, I., Kazemzadeh, M. B., & Roden, T. (2006). *The art and science of game programming.* In Proceedings of the 37th SIGCSE technical symposium on Computer science education (pp. 510-514). New York, NY, USA: ACM Press.

Polli, A. (2004). *DATAREADER: a tool for art and science collaborations.* In Proceedings of the 12th annual ACM international conference on Multimedia (pp. 520-523). New York, NY, USA: ACM Press.

Sardon, M. (2006). *Books of sand.* In Proceedings of the 14th annual ACM international conference on Multimedia (pp. 1041-1042). New York, NY, USA: ACM Press.

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

Shoniregun, C. A., Logvynovskiy, O., Duan, Z., & Bose, S. (2004). *Streaming and security of art works on the Web.* In IEEE Sixth International Symposium on Multimedia Software Engineering (ISMSE'04) (pp. 344-351). Washington, DC, USA: IEEE Computer Society.

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

Strömberg, H., Väätänen, A., & Räty, V.-P. (2002). *A group game played in interactive virtual space: design and evaluation.* In Proceedings of the conference on Designing interactive systems: processes, practices, methods, and techniques (pp. 56-63). New York, NY, USA: ACM Press.

Sung-dae, H., Jin-wan, P., & Won-Hyung, L. (2006). *Designing Audio Visual Software for Digital Interactive Art.* In 16th International Conference on Artificial Reality and Telexistence--Workshops, 2006. ICAT '06. (pp. 651-655). Washington, DC, USA: IEEE Computer Society.

Trifonova, A., Ahmed, S. U., & Jaccheri, L. (2007). *SArt: Towards Innovation at the intersection of Software engineering and art*. Paper presented at 16th International Conference on Information Systems Development, Galway, Ireland.

Walden, K. L. (2002). Reviews : Peter Weibel and Timothy Druekrey (eds), Net_Condition: Art and Global Media. *Convergence-The International journal of New Media Technologies, 8*(1), 114-116.

Warr, A., & O'Neill, E. (2007). *Tools to Support Collaborative Creativity*. Paper presented at Tools to Support Collaborative Creativity workshop held as part of Creativity and Cognition conference 2007, Washington D.C., USA.

Wei-Lung, W. (2002). Beware the engineering metaphor. *Communications of the ACM, 45*(5), 27-29.

Zimmerman, G. W., & Eber, D. E. (2001). *When worlds collide!: an interdisciplinary course in virtual-reality art.* In Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education (pp. 75-79). New York, NY, USA: ACM Press.

Key Terms and Their Definitions

Installation Art: Merriam Webster defines Installation as "a work of art that usually consists of multiple components often in mixed media and that is exhibited in a usually large space in an arrangement specified by the artist". In wikipedia we find following definition of Installation Art: "Installation art uses sculptural materials and other media to modify the way we experience a particular space. Installation art is not necessarily confined to gallery spaces and can be any material intervention in everyday public or private spaces. Installation art incorporates almost any media to create an experience in a particular environment. Materials used in contemporary installation art range from everyday and natural materials to new media such as video, sound, performance, computers and the internet. Some installations are site-specific in that they are designed to only exist in the space for which they were created."

Art Projects: In the context of this chapter, with the word art projects we mean Art projects that use technology for the development of the final product which is usually an artwork. These projects include at least one artist and a number of technologists. In context of this chapter, technologists are software engineers. But in general, besides the software engineers there can be many other technologists such as sound engineers, electrical engineers and so on. Often art projects are multidisciplinary projects which involve many people including the sponsors and public administration, researchers and so on.

Artwork Support Tools: Software used to realize a certain artwork by implementing the core functionalities of the artwork. This software can be commercial off the shelf (COTS) software or open source software or even customized software that is developed only to realize a particular art project. By artwork support tools we mean the whole application that works behind the artwork.

Conceptual Framework: Conceptual framework can be described in many ways: i) a set of coherent ideas or concepts organized in a manner that makes them easy to communicate to others. Or ii) an organized way of thinking about how and why a project takes place, and about how we understand its activities or iii) An overview of ideas and practices that shape the way work is done in a project. Wikipedia defines it as "A conceptual framework is used in research to outline possible courses of action or to present a preferred approach to a system analysis project. The framework is built from a set of concepts linked to a planned or existing system of methods, behaviours, functions, relationships, and objects".

Software Engineering: According to IEEE Standard Glossary of Software Engineering Terminology, Software engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. According to the Software Engineering Body of Knowledge, the discipline of software engineering encompasses knowledge, tools, and methods for defining software requirements, and performing software design, computer programming, user interface design, software testing, and software maintenance tasks. It also draws on knowledge from fields such as computer science, computer engineering, management, mathematics, project management, quality management, software ergonomics, and systems engineering. In industry software engineers can have several specialized roles such as analysts, architects, developers, testers (according to Wikipedia). In context of this chapter, we call all people who are involved in the development of software for art projects as software engineers irrespective of their specialization.

Computer Art: Wikipedia defines Computer art as any art in which computers played a role in production or display of the artwork. Such art can be an image, sound, animation, video, CD-ROM, videogame, web site, algorithm, performance or gallery installation. Often computer art is used in general to refer to artworks that were impossible to create before the invention of computer.