Collaborative and social development environments: a literature review

Hani Bani-Salameh*
Department of Software Engineering, The Hashemite University, Zarqa 13115, Jordan
E-mail: hani@hu.edu.jo
*Corresponding author

Clinton Jeffery
Department of Computer Science, University of Idaho, Moscow, ID 83843, USA
E-mail: jeffery@uidaho.edu

Abstract: Software developers spend a majority of their time using a development environment coupled with a variety of collaborative tools. They use integrated development environments (IDEs) in different editing and debugging activities while solving particular technical problems or developing different pieces of software. These programming environments evolved from command-line tools, to text-based IDEs, graphical IDEs, and eventually to collaborative and social development environments (SDEs). This paper reviews the current state of the art in collaborative development environments (CDEs) and surveys the evolution in this area. The importance of collaboration within software engineering is addressed, followed by detailed information about collaborative systems for software development. Groupware architectures, taxonomies, and groupware applications relevant to software development are discussed, and various research systems that fall within these classifications are described. The paper concludes with recent research related to collaborative and social software engineering.

Keywords: awareness; CDE; collaborative development environment; collaboration; IDE; integrated development environment; SDE; software development environment; presence.


Biographical notes: Hani Bani-Salameh is an Assistant Professor of Software Engineering, in the Prince Al-Hussein Bin Abdallah II for Information Technology College at The Hashemite University. His research interests include Computer Supported Cooperative Work (CSCW), software development environments, collaborative software development in virtual environments, and social networking. He studies social interactions in social networks and online environments, including Facebook and SourceForge.

Clinton Jeffery is an Associate Professor in the Department of Computer Science, in the College of Engineering at University of Idaho. His research interests include collaborative virtual environments, programming languages, automatic debugging, and program visualisation.

1 Introduction

Collaboration is at the heart of software development, and it involves interactions among developers. Previous literature defines collaboration as any form of interaction between distributed software developers or teams (Cook, 2011; David and Borges, 2004). To achieve a common objective, software development requires collaboration among developers within and occasionally outside their project teams. Previous research studies have shown that almost 70% of software development time is spent on collaborative activities (Sarma, 2005).

In fact, research indicates that the way developers work together heavily influences the success of the project (Cook, 2011). Researchers and industry have produced a wide range of collaborative tools, and proposed many definitions of collaboration.

Collaboration is becoming more intertwined with daily aspects of software development processes, which require participation and interaction between team members.
Consequently, software development processes are becoming social activities where key decisions are made in the context of collaborative groups (Moran et al., 2004).

Collaborative development environments (CDEs) cover several related disciplines of computer science. Social development environments (SDEs) are social CDEs where social features are integrated within the environment. As illustrated in Figure 1, ‘Social development environment’ forms an intersection of these overlapping fields. For a successful study and implementation of the development environment tools, knowledge of the related disciplines is required.

Figure 1 Disciplines associated with social development environments (see online version for colours)

Most of these tools focus on a few aspects of collaboration. To understand the functionalities of existing collaborative tools and how they compare with one another, several classification frameworks have been proposed. Appendix A shows list of functional categories that represent a logical taxonomy for groupware tools.

There exist survey publications of existing tools in this area (Lanubile et al., 2010; Portillo-Rodriguez et al., 2010, 2012; Baghdadi, 2012; Serce et al., 2011). But there is no survey that distinguishes such tools, shows their evolution, and identifies what value each category of tools introduced to the development community.

There remains insufficient information regarding which tools assist developers with which collaboration challenges, and what value they can gain from using one over the other. The goal of this work, therefore, is to carry out a literature review of the collaborative software development tools, in order to obtain information about which tools are available for the distributed developers and what features they include.

The rest of this paper is organised as follows. Section 2 identifies the research approach and research questions. Related topics in the fields of groupware, computer supported cooperative work (CSCW) are presented in Section 3. Section 4 introduces CDEs, and gives a list of popular existing systems. An introduction and definition of SDEs are presented in Section 5. A comparison between CDEs and SDEs is presented in Section 6. Finally, this paper sums up current state of this work in Section 7.

2 Identification of research

The research and literature introduced here has been developed by following the guidelines presented in Petersen et al. (2008). In this section, we give an overview of this process.

2.1 Goal

The main objective of this study is to review and describe the functionalities of existing collaborative tools and how they compare with one another, and to review the classification of the frameworks.

2.2 Definition of research question

This study achieved a literature review through a search for existing research studies to answer the following research questions:

- What groupware and collaborative tools are available to support the software development community?
- What are the integrated features?

The review study aims to tell the following information about:

- tools that are commonly used by the software developers, and how they fall under the CSCW umbrella
- features that are integrated in these tools.

3 Research related to collaborative software engineering

This section presents an overview of previous work towards groupware in the field of software development. Before presenting details about groupware, it is appropriate to introduce the computer-supported cooperative work (CSCW) discipline. The term CSCW was coined in the mid-1980s and has many aliases including ‘computer-supported collaboration’ and ‘workgroup computing’.

CSCW is the study of how people use tools and technology to work together in shared time and space (Grudin, 1994; Rama and Bishop, 2006). It examines the design, implementation, and use of groupware (Schummer and Lukosch, 2006). CSCW is not just about ‘cooperation’ or ‘work’, it also examines competition, and socialisation. This field attracts those who are interested in software design and social behaviour, including business professionals, computer scientists, psychologists, and communications researchers, among other specialties.

3.1 Collaborative software systems (groupware)

“The reason I invent is to advance the evolution of society and its institutions. My crusade is to find much better ways for people to work together to make this world a better place.
-Douglas Engelbart (father of groupware)” (Bani-Salameh, 2011)
Groupware was first coined in 1984 by Peter and Trudy Johnson-Lenz at the New Jersey Institute of Technology. Trudy Johnson-Lenz defined groupware as “intentional group processes plus software to support them” (Ward, 2007). The term commonly refers to a specific class of tools such as e-mail, bulletin boards, asynchronous conferencing, group schedulers, group decision support systems, screen sharing software, whiteboards, video conferencing, news groups, and chat. Groupware is not just technology, it is also social. Groupware is collaborative activity that impacts the way people communicate with one another. In other words, groupware can be considered a set of activities people do, as much as it is a set of tools people use (Greenberg, 1991; Stahl, 2004).

Groupware supports job functions that require people to work together, even though they might not be together, in either time or space. Groupware systems enhance the sharing of information and ideas between distributed team members and make their effort more efficient (Mendes de Araujo et al., 2004). CSCW’s primary concern is with aspects of the behaviour of people and organisations. This is implicit in both of the ‘Groupware’ and ‘CSCW’ terms. CSCW and groupware research focuses either on utility and tools, or on usability.

Figure 2 shows the classical representation of the US research and development contexts for CSCW and groupware, where each ring defines a work level (organisation, project, group, or individual).

“...the outer ring represents entire systems designed to serve organisational goals, such as MIS systems. The inner ring represents applications designed for individual users, such as word processors, spreadsheets, and games. The middle two rings represent groupware, designed with groups in mind.” (Ward, 2007)

Figure 2 shows where software development environments fit as CSCW and groupware tools; they fit in rings B and C, somewhere between individual applications and information systems that support organisations.

Various groupware systems have been developed to support particular forms of CSCW. This section provides an overview of these systems. Table 1 compares the groupware systems against specific criteria defined by Rama and Bishop (2006). These criteria include functional, architectural, focus, time, and user involvement.

Researchers have divided groupware applications, based on the nature of the communication, into three groups (Dix et al., 1998).

- **Computer-mediated communication** aims to improve the communication between users. Such systems can be either asynchronous (such as e-mail, bulletin board systems or news feed), or synchronous systems that include types of computer assisted conferencing occurring in real-time (such as chat rooms, videoconferencing). They fall in the inner middle ring set of applications (A) shown in Figure 2.

- **Meeting and decision support systems** aim to help users establish common understanding about the task they support and generate ideas. Such systems can be asynchronous, for example, tools that record the discussions between users. Meeting rooms are usually synchronous and co-located; they support groups in face-to-face meetings. Shared drawing tools support synchronous meetings where users express their ideas in a shared drawing area. They fall in the outer middle ring set of applications (B) shown in Figure 2.

  - **Shared applications and artefacts** support interaction of users with shared tools/artefacts. There are many systems available, including shared editors that allow n-users to create documents in a controlled synchronous manner, co-authoring systems that allow for asynchronous collaboration on documents, and shared calendars that allow for the collaboration of several users on a combined schedule. They fall in the inner middle ring set of applications (A) shown in Figure 2.

Table 2 shows a range of systems and applications specified as groupware tools.

To summarise, within this paper, groupware is defined as: a research area that studies and supports the use of both synchronous and asynchronous software engineering tools for different tasks, programming languages, and software development processes, to help groups of developers work together toward a specific goal.

### Table 1 Common groupware criteria

<table>
<thead>
<tr>
<th>System</th>
<th>Functional</th>
<th>Architectural</th>
<th>Focus</th>
<th>Time</th>
<th>User involvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messaging</td>
<td>Central</td>
<td>User centred</td>
<td>Synchronised</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Conferencing</td>
<td>Replicated</td>
<td>Artefact centred</td>
<td>Mixed</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>Electronic meeting system (EMS)</td>
<td>Hybrid</td>
<td>Workspace centred</td>
<td>Synchronised</td>
<td>Unsynchronised</td>
<td></td>
</tr>
<tr>
<td>Group decision support</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document collaboration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compound document management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 3.1.1 Asynchronous systems

“Asynchronous groupware supports communication and problem solving among groups of individuals who contribute at different times, and typically also are geographically dispersed.” (Baeker, 1995, p.743)

Asynchronous groupware are group support systems that support distributed interaction occurring at different times. Such systems provide loosely-coupled interactions, so that
users can modify the shared artefacts or code pieces without having any awareness or direct knowledge of the changes made by the other users, either because they work at different times or because they do not have access to each other’s activities and actions (Graham, 1997).

3.1.2 Synchronous systems

Synchronous groupware (Wang, 2008; Bates, 1998), is a category of software application that facilitates real-time collaboration among co-located and geographically distributed group members. Synchronous groupware includes application sharing, voice and video conferencing, instant messaging, shared whiteboard, multi-player virtual games, shared editors, and group decision support systems.

Synchronous groupware is defined as “the class of applications in which two or more people collaborate in what they perceive to be real time” (Day, 1997).

According to Graham, “the defining property of synchronous groupware is that it helps people to work together at the same time, allowing participants to immediately see the effects of other participants’ actions. Synchronous groupware is meant to create group awareness, allowing people to work with the kind of direct communication they would have if they were all in the same room.” (Graham, 1997)

In recent years synchronous groupware has been widely used in at least the following three application areas (Graham et al., 2006):

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Systems specified as groupware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characteristics of system</td>
<td>Characteristics of environment</td>
</tr>
<tr>
<td>Applications</td>
<td>None</td>
</tr>
<tr>
<td>Applications</td>
<td>Networked computers, large databases</td>
</tr>
<tr>
<td>Any technology to support group productivity</td>
<td>None</td>
</tr>
<tr>
<td>Hardware, and software and Groups working together</td>
<td>None</td>
</tr>
<tr>
<td>Applications</td>
<td>Networked environment</td>
</tr>
<tr>
<td>Networked based software, web pages, electronic bulletin boards, discussion lists, file sharing, synchronous or asynchronous</td>
<td>Shared hypermedia environment</td>
</tr>
</tbody>
</table>
Communication tools: Tools such as MSN Messenger and Skype that allow people to communicate regardless of their distance (location), either by textual chat, or through voice over IP and video. They fall within both the middle ring set of applications (A and B) shown in Figure 2.

Multiplayer 3D games and virtual environments: Games allowing people to communicate, collaborate and compete have become enormously popular (Garcia et al., 2007). Examples of such massively multiplayer games are World of Warcraft (www.worldofwarcraft.com) and Second Life (http://secondlife.com) where people meet and socialise in a virtual world. They fall within both the central rings (A and B) shown in Figure 2, although these applications often have many social networking attributes as well.

Electronic meeting and conferencing tools: Tools that allow online meetings are increasingly used. Examples of such successful applications include GoToMeeting (www.gotomeeting.com), and WebArrow (www.namzak.com). They fall in the outer middle ring set of applications (B) shown in Figure 2.

Community and socialising tools: Tools that allow people to communicate, socialise, and build online communities. They allow and encourage the formation of social groups, and bring the group members together by helping them to stay connected with each other through online tools, blogs, discussion threads, event calendars and more. Examples of such tools are Facebook (www.facebook.com) and Myspace (www.myspace.com). They fall in the outer ring (E) shown in Figure 2.

Developers in distributed development environments use many tools and applications to help them communicate and collaborate within the development community. Software development environments integrate and merge such tools in a single environment in order to help reduce cognitive friction, by saving the developers from having to switch between those tools and manually dig for information. Also, the integration allows synergy where the interaction of those tools makes their combination greater than the sum of their separate effects, and reduces the steps required for finishing jobs than when tools are separated.

3.2 Groupware architecture analysis

Most groupware systems are inherently distributed (Zafer, 2001). A significant feature of groupware and CSCW is the distribution architecture that defines what part of the groupware application runs on a central server, and what parts run on decentralised sites, and how the decentralised sites are linked and cooperate with each other. The chosen architecture has an important influence on the way the groupware application is developed and used.

Choosing a distributed architecture affects the groupware application in several ways. For example, the application’s response time and fault tolerance can be improved using decentralised architectures. In general, decentralised architectures scale much better than centralised. On the other hand, some runtime services such as storing documents can be developed much easier when using a central server or centralised distribution architecture.

Researchers in computer science usually consider three distributed architectures: centralised, replicated, and hybrid. The diagrams in Figure 3(a) and (b) (Figures adapted from Begole et al. (2001)) illustrate the key components and communication paths between processes of a conceptual two-user collaborative system under fully centralised and replicated architectures (Begole et al., 2001).

Researchers in computer science usually consider three distributed architectures: centralised, replicated, and hybrid. The diagrams in Figure 3(a) and (b) (Figures adapted from Begole et al. (2001)) illustrate the key components and communication paths between processes of a conceptual two-user collaborative system under fully centralised and replicated architectures (Begole et al., 2001).

Graham et al. (2006) identified five distribution architectures for synchronous groupware by examining the implementation design of existing systems. These architectures are

- centralised core, thick client
- generic thin client
- centralised mixer with broadcaster
- replicated input broadcasting
- replicated state synchronisation (see Table 3).

4 Collaborative development environments

“While traditional IDEs focus on improving the efficiencies of individual developers, collaborative development environments focus on improving the efficiencies of the entire development team.” (Booch and Brown, 2003; Petropoulakis and Flood, 2007)
CDE is a term coined by Booch and Brown (2003, 2007) to describe applications that allow distributed developers to collaborate and work together. A CDE describes an online meeting space where the developers can work together, brainstorm, discuss, record, and go over ideas to solve design and coding problems, and create software products, regardless of their time-zone and region disparity. CDEs are IDEs that are designed with collaboration as a major focus (Lanubile et al., 2010).

Research into collaborative software development environments is increasing rapidly. Factors that led to the rise of this research include mainly the advent of open source IDEs, the high need for distributed development, and big advances in the computer and programming fields. Software development encompasses a wide range of tasks, and collaboration tools exist for almost every conceivable task in software engineering (SE).

Tools of this nature, such as FIELD (Reiss, 1995), Eclipse (Lewis, 2005), Visual Studio (Visual Studio Developer Center, 2007), and CASE tools such as Rational Rose (http://www.rational.com) are often designed for large teams of software engineers, where numerous procedures and methodologies that require tool support are in place. The rest of this section highlights some of the tools categories that support collaboration during the implementation of the SE project artefacts, including tools that focus mainly on the code-level development (Cook, 2011).

Integrated development environments are where most coding takes place and are the home of many different development tools. “IDEs were initially designed as a soloist tool, with features for the lone engineer to author, debug, refactor, and reuse code.” (Storey et al., 2010) Developers may interact and collaborate with the other range of developers to obtain advice about the code, fix each other’s bugs, and get a general understanding of what is happening in the project artefacts. They use a variety of collaborative tools in their everyday activities. These include tools such as: source control, bug tracking systems, e-mail, and instant messaging. Much collaboration in software development is accomplished using asynchronous technologies such as e-mail or revision control systems (Hupfer et al., 2004; Cubranic and Storey, 2005; Kamrani and Abouel Nasr, 2008).

Integrating such collaborative tools into the IDE, and enabling them with awareness of development processes and artefacts, may help reduce programmers’ cognitive context switches between tools inside and outside the IDE and make the connection between development and collaboration more seamless (Lewis, 2005).

Others point out that a rich CDE arises from the collection of many seemingly simple collaborative components to support coordination, collaboration, and community building. They also state that IDEs equipped with team-centric support and whose primary user experience focuses on the needs of the team, is a step up from those merely augmented with some collaborative support (Booch and Brown, 2007). The remainder of this section highlights some notable collaborative text editors (3.1) and more general CDE systems (3.2 and 3.3).

### 4.1 Collaborative editors

Collaborative editing is the practice of a group of individuals simultaneously editing a document. This section focuses on the synchronous collaborative editors, introducing two of those editing systems. A more detailed discussion of some of the existing systems is presented by Zafer (2001) such as ShrEdit (Dourish and Bellotti, 1992), GROVE (Ellis et al., 1993), SASSE (Baeecker et al., 1993), Calliope (Mitchell, 1996), Flexible JAMM (Begole, 1998), MUSE (Mullick, 2011), and others.

**NetEdit** (Zafer, 2001; Schanda, 2005) is a collaborative text editor implemented in Java that uses a replicated architecture. It provides the user with centralised file and session management, flexible editing ability among groups, and chat session management. A user in a collaborative session can modify any part of the document, and can communicate with others using the chat room.

**NetEdit** allows two or more users to remotely edit a document simultaneously. It provides collaborative awareness functionality that allows following other users through the document and tracking changes. NetEdit handles the concurrency with an $n$-way consistency algorithm wherein each client has a state-space graph maintained by the server. Each client keeps buffers for changes that have been made to its copy of the document, but have not yet been broadcast to the other clients in the group.

**GHT (Group Homework Tool)** (Langton et al., 2004) is an educational groupware tool built to support synchronous, collaborative coding among novice programmers. In addition to the real-time editor, GHT provides an assignment definition and resource page, a chat system, and a shared whiteboard. Also, GHT provides awareness information of users’ work in the shared place.
4.2 Eclipse-based CDE’s

Eclipse (Lewis, 2005) is an open source development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. Eclipse itself does not support code-level collaboration, but the Eclipse Communication Framework project (Lewis, 2005) aims to support the development of distributed Eclipse-based tools and applications and to allow the Eclipse code repository and project model to be shared and collaboratively edited. Eclipse provides an API to perform basic sharing, along with some prototype client applications. One such application is where a shared graph editing tool is hosted within the Eclipse IDE (Cook, 2011; Cheng et al., 2003).

A number of other Eclipse-based projects focus on the integration of collaborative features into IDEs. GILD is an example of a project that provides cognitive support for novice programmers and support for instructor activities (Storey et al., 2003). CodeBeamer is a commercial product that has plug-ins for integrating collaborative capabilities into IDEs such as chatting, messaging, project management, and shared data (CodeBeamer, 2013). Another example is Sangam, a plug-in for the Eclipse platform that features a shared editor and chat for pair programming (Ho et al., 2004).

Stellation (www.eclipse.org/stellation) is an open source effort (led by IBM Research) that introduces a fine-grained source control that supports the notion of activities and aims to simplify collaboration and provide awareness of changes to team members (Cheng et al., 2003). It enables developers to manage relevant work, notify the team of their current work, be informed of changes pertaining to their own activities, and provides a context for persistent conversations.

Palantir (Ripley et al., 2004; Hattori and Lanza, 2009) is another Eclipse plug-in that supports awareness features by showing which artefacts have been changed by which developers and by how much. Palantir provides workspace awareness such that developers can monitor other teams’ activities while working on their current task and without the need for switching contexts.

Borland’s JBuilder 2008 (JBuilder, 2011) is an enterprise-class Java IDE based on Eclipse. JBuilder is based on Eclipse 3.3 and Eclipse Web Tools Platform (WTP) 2.0. It supports real-time remote refactoring, distributed views of UML diagrams, and chat channels. It includes a CDE with integrated tracking, source code management, project planning, and continuous builds, allowing team members to easily monitor and manage project activity and progress.

JBuilder incorporates a shared pair-programming code editor and collaborative debugging capabilities (Lewis, 2005; Hupfer et al., 2004). JBuilder provides useful support for version management using a CVS repository. JBuilder shows the history of the checked projects in a history tab that allows developers to view different versions of an application, revert to an older version or examine differences between versions, show the revision information and work with merge conflicts (Product Review, JBuilder 8, 2013).

4.3 Other CDEs

GForge (Collaborative Development Environment (CDE), 2013) is an Open Source CDE. GForge provides developers and teams with tools to help them collaborate in files, artefacts, documents and tasks, and control access to source code management repositories such as CVS and Subversion.

Longhouse (longhouse, 2013) is an open-source CDE that is based on Google Code. It provides several different features for the software development communities, including: project hosting, issue tracking, and a space for collaborative work on software projects. It is designed to target small- to medium-sized communities who are interested in hosting their software development projects. “Longhouse aims to preserve the usability paradigms of Google applications while adding additional usability optimisations designed to maximise understandability, new-user-friendliness and productivity” (longhouse, 2013).

Collab.net (Bani-Salameh, 2011) is a commercial CDE that has both a public and a private face. Collab.net’s public face is SourceForge, an open source CDE that focuses on the development of open source software. SourceForge serves as a host to approximately 324,000 projects such as CVE (http://cve.sf.net). Collab.net’s private face is SourceCast (http://sourcecast.org/) a CDE that supports a number of important features not in SourceForge, such as greater security that is not a big issue for open source development.

A user may enter this web-based application using a specific project or from a personal web page. Within a project, Collab.net supports features for artefact storage, simple configuration management via CVS, bug tracking, task management, and discussion boards. SourceForge offers other features such as the ability to self-publish, change tracking, and project membership management.

Table 4 shows that CDE tools vary in the number of supported features; most CDE tools are designed for specific purposes and in general do not support all software development tasks.

Table 4 A variant of DeSanctis and Gallupe’s time/space taxonomy framework

<table>
<thead>
<tr>
<th>Time/place</th>
<th>Same</th>
<th>Different (predictable)</th>
<th>Different (unpredictable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same</td>
<td>Meeting facilitation</td>
<td>Work shift</td>
<td>Team rooms</td>
</tr>
<tr>
<td>Different (predictable)</td>
<td>Teleconferencing</td>
<td>Video conferencing</td>
<td>Desktop conferencing</td>
</tr>
<tr>
<td>Different (unpredictable)</td>
<td>Interactive multicast</td>
<td>Computer boards</td>
<td>Work flow</td>
</tr>
</tbody>
</table>

Source: Adapted from Graham (1997)

5 Social development environments

A SDE is a real-time collaborative programming tool with integrated social networking features. This emerging technology is important for distributed software developers,
e-learning and technical communities. Social development environments advance the state of the art for collaboration, coordination, and project management in software development.

5.1 Common properties

Social development environments provide a wide range of facilities for synchronous and asynchronous collaboration and information sharing between team members. This section provides an overview of SDEs and compares several state of the art examples.

An SDE is not a single application, it is a harmony between many different development tools. Several applications support the primary collaborative infrastructures for complete SDEs in a single environment. SDEs include a spectrum of collaborative tools that can be of a big benefit to the development communities, each of these tools adds its own taste and value to the integrated environment, including Web logs (blogs), Mailing lists, Walls, Chat rooms, Whiteboards, and Wikis. Section 4.2 presents a few related and well-known examples.

5.2 Examples

Jazz is the most widely-publicised example of a social development environment. IBM added enterprise social networking features to the rational team concert (RTC) development environment in their release of Mainsoft Document Collaboration for Rational Jazz mentioned on SPTechBlog (The SharePoint Technology Blog. Bridging the SharePoint-IBM divide, 2013). Developers are able to:

- access SharePoint My Sites, with links to blogs and wikis
- view SharePoint Personal Profiles
- use SharePoint People Search.

These features are available on top of the existing integration of SharePoint document libraries and workflows with the Jazz development process. Developers can view their other team members’ SharePoint My Sites from the RTC’s team artefacts view. Blogs and wikis linked from My Sites are also listed in the Team Artefacts view, and they can be opened directly from RTC (Frost, 2007).

Jazz extends Eclipse with collaborative capabilities to support coordination, communication, and awareness among a small close-knit team of developers. The Jazz environment includes: Jazz Band showing teams, members, and status icons, menu offering communication options, decorators and tooltips on resources, anchored chat marker, code modification indicator, and team member’s status message (Hupfer et al., 2004). It focuses on developing better team-building strategies, managerial processes, architectural designs, collaborative coding techniques, and software development practices. The Jazz goal is to provide a platform that allows integrating tasks across the software life cycle. However, Jazz also seeks to increase teams and teams’ productivity (Frost, 2007; Cheng et al., 2003).

The objective is to raise the feeling of the other developers’ activities as a social team, while capturing the team’s artefacts to provide a better communication and collaboration environment. Jazz provides a facility similar to an IM buddy list to monitor who is online and coding or not. The IM status message shows the other developers what file the developer is currently working on. Developers can initiate chats, which can be saved as code annotations or into a discussion forum, or use screen sharing and voice-over-IP (VoIP) telephony, without the need for any additional setup effort. Jazz also provides resource-centred awareness. Files and other resources in the file viewer are decorated with coloured icons to indicate what other developers are doing with their local copies of the files (e.g., indicating that a file is in focus and being edited at this very moment or that a file has been locally saved but not checked back into the code repository). Also it shows who is responsible for these changes. This awareness information reveals cues normally associated with files and users, such as file size, last modified date, and who is responsible for the changes that incorporate traceability (Hupfer et al., 2004; Cubranic and Storey, 2005).

MydeveloperWorks: IBM’s MydeveloperWorks social networking service has the motto: “social networking is the development process”. MydeveloperWorks is a new way for distributed developers to connect and interact with their fellow developers. MydeveloperWorks developers can create their own personal profile and customise their home page to get instant access to the people, feeds, tags, bookmarks, blogs, groups, forums, etc. that they care about, and search through user profiles for those with like-minded interests (IBM’s Social Network for Software Developers, 2013).

IBM’s goal with MydeveloperWorks is to connect the global community of software developers and make it easier for them to create new technologies based on open standards such as Java, Linux and XML (Communities of Practice, IBM’s Social Network for Software Developers).

CollabVS (Collaborative Development Environment using Visual Studio, 2013) is an SDE based on Microsoft Visual Studio that allows developers to work together whether planned or ad-hoc. For example, a pair of developers can agree to work together at a scheduled time; in addition, a developer can initiate collaboration with whomever happens to be available online at any given time. CollabVS extends the Visual Studio programming environment by adding collaboration tools, such as text and VoIP chat, watching the unchecked versions of files, and notification of presence in files. CollabVS targets collaboration among distributed developers (Hattori and Lanza, 2009; Hegde and Dewan, 2008).

It extends the Visual Studio programming environment by adding text and audio-video chat, browsing of remote unchecked versions of files, and notification of presence in elements inside a file. Although CollabVS targets collaboration among developers, it still relies on the classical ‘check in’/’check out’ model and treats files as the lowest level of granularity. CollabVS provides two kinds of presence. Real-time Presence information lets the user know what the other team members are currently doing. It shows what users are online and whether they are editing, debugging, engaged
in an instant messaging session, etc. Contextual Presence information facilitates finding relevant information and people quickly. It provides information based on the context of the project’s artefacts. It shows information such as: the last changes/updates made to the artefacts, who made the change, and who checked the artefacts out.

Zembly (Anderson et al., 2013) is a socially-networked development environment from Sun. Zembly supported the development of cloud applications on platforms including Facebook, OpenSocial, iPhone and other platforms. With Zembly users can do social programming, and develop applications with other people using social networking-type features. Not only Developers can reuse pieces and parts of other developers’ projects (a work that they previously implemented to construct new applications), but also inviting friends and colleagues for collaboration. Also, they can see what colleagues are working on via news feeds, and keep up with what others publish and even with what changes they make to their projects’ artefacts. It is a browser-based environment where all activities such as editing, testing, and documenting happen within the browser with the collaboration of other developers.

SCI (Bani-Salameh et al., 2010, 2011) is another social programming environment that supports developers’ collaboration, and interaction. It allows them to be aware of the actions of others in real time, aware of artefacts changes, avoiding coding errors, and potentially improves the collaboration during the software development process. An environment that combines: communication, collaboration, awareness, and social networking and online presence features from inside a single environment. SCI combines both synchronous and asynchronous features. It is composed of a presence and activity awareness information component, an IDE, and collaboration tools that all reside within a single environment, which serves as a virtual CDE. The merger of these tools increases their benefit to the development community.

SCI supports development in mainstream languages, such as C, C++, and Java. Depending on the obtained research results, augmenting related and specific awareness information, online presence, and social network features within a single environment makes SCI a usable and useful development environment for developers and teams working in a distributed environment. The combination of both the synchronous and asynchronous features makes the tools more usable. In general, the asynchronous features support the usability of the synchronous tools, and both categories complement each other.

Codebook is a social networking web service that helps software engineers track status and coordinate activities. They can use Codebook both to track their other team member’s activities, and changes to the status of work artefacts. Software engineers can look up a person or work artefact to find out their information and history. Developers can “be friends not only with other people but with the work artefacts they share with them” (Begel and DeLine, 2009).

StackOverflow is a community site where developers and programmers can ask and answer each others’ questions. Users can establish profiles and re-tag questions or voting to close a question.

GitHub is a web-based hosting environment for software projects that allows developers collaborate in their project artefacts. They share code, watch others activities, and search for others with common interests and expertise (Collaborative Development Environment using Visual Studio, 2013). GitHub includes social networking features such as: followers, feeds and blogs, and shows developers activities on the repository versions (GitHub Social Coding, http://github.com/; GitHub, http://www.crunchbase.com/company/github; Dabbish et al., 2012).

5.3 Discussion

The SDEs discussed in this section illustrate just the beginnings of this genre of software tools. To our knowledge, no existing tool takes the full ‘social networked IDE’ concept to its limit. Most of the cited and existing projects implement a subset of social networking, but do not fully integrate social networking features within an IDE. They may have some social network-like features, but not others such as user profiles and news feed.

The SDE tools presented previously in this section are categorised in Table 5. It shows that SDE tools vary in the number of awareness and social supported features.

Table 5 Possibilities regarding CSCW Characteristics

<table>
<thead>
<tr>
<th>CSCW Characteristics</th>
<th>Information</th>
<th>Sharing</th>
<th>Communication</th>
<th>Coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Source: Adapted from Zafer (2001)

Table 5 shows that among all of the discussed tools, only SCI and Jazz integrate most of the social networking and awareness features. Jazz supports many aspects of social networks. Like SCI, Jazz focuses on increasing the user’s awareness of people, resources, and activities, and on fostering communication among team members. Both Jazz and SCI support synchronous chat discussions. Also, Jazz provides team-centric discussion boards that compare to the asynchronous news feed supported by SCI. User profiles are not integrated in Jazz, but Jazz users can create their own profile by linking to SharePoint, where in SCI developers can view other developers’ profiles, friends, groups, projects, and activities directly from the SCI environment. Jazz supports awareness of the committed code changes with respect to the code repository. In contrast, SCI provides the developer awareness information of the committed and uncommitted
code changes, of the currently edited files, and indicates who is responsible for the changes.

The major difference between SCI and almost all the related work cited in this section is the full integration of social network features inside the software development environment, and without linking to third party products.

6 Comparison: CDEs vs. SDEs

Grady Booch (Booch and Brown, 2003; Lanubile, 2009) stated that the purpose of a CDE is to create a foundation for environments (tools) that reduces the frictions that have an impact on the daily work of the distributed software developers and reduces both individual and group efficiency, and to support the development process during the whole life cycle of the project. “CDEs are essentially team centric, meaning that their primary user experience focuses on the needs of the team (but with points of entry for different individuals)” (Booch, 2013).

In contrast with the support that CDEs provide for teams working closely together, social networks build online communities of people loosely connected by their common interests or activities. Community is a vital aspect of software development, but software developers tend to focus their attention primarily on their programming environment tools. Software development projects usually include communities of users. Community members play vital roles that reflect the success of such projects, such as reporting bugs, helping other users, and analysing problems. These observations lead to a new category of tool, the SDE.

In this paper the term social software development refers to software development in collaborative online communities with social relationships. Software development requires interaction between the people involved in the development process. For this reason, social activities form a big part of their daily work. In software development, developer networks are an instance of object-centred sociality (Bouillon et al., 2005), where developers do not usually interact merely to socialise, as in conventional social networks. In contrast, they interact and collaborate primarily through shared project artefacts.

Communication is an important factor for software projects to succeed. With the geographical, cultural, time-zone, or language barriers that face the project members involved in the development, making communication and collaboration more successful is a not an easy task. Communication leads to collaboration between the project’s team to accomplish certain tasks. As teams become more and more geographically distributed, collaboration becomes more critical and difficult to achieve. Team members usually use different tools to communicate and that leads to difficulties to collaborate and finish their tasks.

As mentioned earlier, there are hundreds of communication applications that are used by software development communities and these can be combined to support distributed development teams (Bouillon et al., 2005; Bani-Salameh et al., 2008). CDEs integrate collaboration and programming tools in one environment. SDEs integrate the development environment and the social network in a single environment, and support the interaction between the social networking features and the communication and collaboration tools. SDEs add value because they touch on the social and presence elements of software development. An SDE aims to help users maximise their productivity, and to help distributed team members maintain a level of social awareness regarding other team members’ roles, activity patterns and contributions to the project, as well as the resources in the community relevant to a given project. In general, the purpose of the SDE is to provide a frictionless environment for software development by eliminating the need to switch between different tools in order to perform their various solitary and collaborative tasks.

To summarise, CDEs and SDEs share substantially overlapping purpose and objectives. SDEs add value by integrating features that help establish and support interaction among distributed developers, strengthen social bonds, communication and interactions, and make it possible for them to work hand in hand, build trust, and have the ability to network. SDEs are a category of CDEs where the system provides a fully featured social network for distributed developer communities.

In and SDE, developers can browse profiles, create development projects, and form professional teams. In addition, SDEs provide views for listing people and groups, project data, activities, etc. all from inside the environment. By using these views, users will be able to request help and assist others in their projects, and form strong social bonds among the community members. In general, an SDE is an environment that allows distributed developers to work hand in hand and also have the ability to network. It is a combination of social networking and collaborative software development.

7 Summary

The relevant literature for this paper comes from the groupware and CDE fields. Many tools discussed here help to identify the research issues. The review showed that many tools lack support for the awareness and online presence. The review also demonstrated an absence of the main social networking features support.

Social support for software development is an important emerging field of research. Conventional single-user tools are not able to provide the needed environment for smooth collaboration between distributed developers due to the size and complexity of today’s development projects. CDE tools that support and provide project artefacts’ updates in real time have the potential to raise the level of communication, and coordination between distributed developers.

Most current CDEs have inherent limitations, including little support for awareness and online presence, missing social networking features, and weak support for source code repositories’ features. The multitude of tools increases the friction that results from switching among different tools.

Supporting full software development environment functionality has been challenging to achieve since most conventional development environment tools are inherently single user tools, and cannot easily be augmented by
groupware to solve all development environments problems. Even though specific purpose development environment tools provide limited facilities to support collaboration among the distributed developers’ community, a more general purpose solutions may be needed for a cohesive development teams.

There is great promise in exploring tool support for the social side of software development. Collaboration plays a crucial role in software development. For this reason, continuing to improve the collaborative tools available inside IDEs is of great potential benefit. Collaborative tools can be used alongside a non-collaborative IDE, but integration adds qualitative and quantitative awareness information and reduces the cost of collaboration during the development process, particularly for distributed teams.

While CDEs support distributed communities’ work effectively, we believe SDEs will add better benefit development communities and help them increase their productivity and produce better software products.

References


Collaborative and social development environments: a literature review

101


Zafer, A. (2001) NetEdit: A Collaborative Editor, Master of Science, University of Virginia, USA, p.82.

Notes

1 Stack Overflow: www.stackoverflow.com
2 GitHub: http://github.com/

Websites

GitHub Social Coding Available at http://github.com/
GitHub, Available at http://www.crunchbase.com/company/github

Appendix A: Groupware taxonomies

Researchers have developed several framework classifications to help them understand the functionalities of these tools. Each classification has a different focus: some provide a detailed taxonomy to compare tools in a particular area, some classify tools based on:

- their functionality
- predictability of the actions that they support
- the collaboration approach that they take
- the user effort required to collaborate effectively.

This section summarises some of those frameworks with a list of functional categories that represent logical taxonomies for groupware tools.

Time/space matrix

Groupware tools have been classified in many ways, but most of them are based on an original matrix created by Johansen in 1988 as illustrated in Table 6 (Ellis et al., 1993; DeSanctis and Galleu, 1987; Johansen, 1988). Groupware technologies are typically categorised along two primary dimensions (Ward, 2007):

Whether users of the groupware are working together at the same time (‘real-time’ or ‘synchronous’ groupware) or different times (‘asynchronous’ groupware), and whether users are working together in the same place (‘co-located’ or ‘face-to-face’) or in different places (distributed or ‘distance’). In 1989, Ellis et al. (1993) refined DeSanctis and Galleu’s (DeSanctis and Galleu, 1987) space and time classification framework to create a $3 \times 3$ matrix (see Table 7) (Grudin, 1994; Bafoutsou and Mentzsa, 2002; Pumareja and Sikkel, 2006). He improved DeSanctis and Galleu’s framework by distinguishing the tools based on the predictability of the
Table 6 Features table of existing CDE systems

<table>
<thead>
<tr>
<th></th>
<th>GILD</th>
<th>CodeBeamer</th>
<th>Stellation</th>
<th>Palantir</th>
<th>Jazz</th>
<th>JBuilder 2008</th>
<th>CollabVS</th>
<th>SCI</th>
<th>GForge</th>
<th>Longhouse</th>
<th>Collab.net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous features</td>
<td>X</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Asynchronous features</td>
<td>X</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Artefact management</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Purpose</td>
<td>E</td>
<td>C</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>C</td>
</tr>
<tr>
<td>Multiple language support</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3D virtual environment</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Awareness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Social presence</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Historical</td>
<td>X</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Group-structural</td>
<td>X</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Workspace (Artefacts)</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SN features</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Groups</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Friends</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Help/Support</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

[C: Commercial, D: Development, E: Educational, G: General, P: Partial].

actions that they support. Johansen’s framework classifies tools based on the temporality of activities, location of the teams, and the predictability of the actions (Sarma, 2005).

Workflow

Nutt classifies workflow systems based on the characteristics of the underlying workflow model. He created a three dimensional domain space that models a work procedure as illustrated in Figure 4 (adapted from Sarma (2005)). The three dimensions in the model space are:

- **X-axis**: the amount of conformance that is required by the organisation for which the process is a model
- **Y-axis**: the level of detail of the description of the process
- **Z-axis**: the operational nature of the model.


**Pyramid taxonomy from the user’s effort perspective**

All of the above frameworks look at coordination tools from a functionality point of view. Sarma introduces a classification framework that revolves around different classes of user effort required to collaborate effectively (Sarma, 2005). This framework is based on two principal characteristics of collaborative tools:

- the level of coordination support provided to users
- the focus of a tool on one of the three essential elements of collaboration: communication, artefact management, and task management.

![Figure 5 The model domain space](Source: David and Borges (2004))

It combines these two characteristics to form a pyramid, as illustrated in Figure 5 where five levels of coordination support
Table 7  Awareness features of existing SDE systems

<table>
<thead>
<tr>
<th>Awareness features</th>
<th>Jazz</th>
<th>My developerWorks</th>
<th>CollabVS</th>
<th>SCI</th>
<th>Zembly</th>
<th>CodeBook</th>
<th>StackOverflow</th>
<th>GitHub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Informal</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Group-Structural</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Social/Presence</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Historical</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Workspace/Artefacts</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

SN features

<table>
<thead>
<tr>
<th>SN features</th>
<th>Groups</th>
<th>Friends</th>
<th>Help/Support</th>
<th>Others</th>
<th>IDE Integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Groups</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Friends</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Help/Support</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Others</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

[Source: Adapted from Bates (1998)]

are organised vertically and called ‘layers’ and three different foci of tools are organised horizontally, and called ‘strands’.

Other researchers such as Penichet et al. (2007), have presented a more flexible taxonomy for the CSCW systems. Their proposed solution is to classify the tools and systems by showing the relation between the system with the time-space features and with the typical CSCW characteristics: information sharing, communication and coordination. A specific system can have several characteristics. Table 8 shows the possibilities regarding CSCW characteristics.

This paper, along with Sarma et al. (2003), agrees that most of the previous frameworks look at coordination tools from a functionality point of view, and none classify tools based on the user’s effort required to collaborate effectively.

Table 8  Time/space taxonomy

<table>
<thead>
<tr>
<th>Time/place</th>
<th>Same</th>
<th>Different</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same</td>
<td>Face to face (F2F) interaction</td>
<td>Asynchronous interaction</td>
</tr>
<tr>
<td>Different</td>
<td>Synchronous distributed interaction</td>
<td>Asynchronous distributed interaction</td>
</tr>
</tbody>
</table>

[Source: Adapted from Bates (1998)]