17 BCI Software

Peter Brunner and Gerwin Schalk

CONTENTS

17.1 Introduction .......................................................................................................................... 323
  17.1.1 Technical Demands .................................................................................................. 324
  17.1.2 Scope of Investigation ............................................................................................... 325
17.2 Implementation ..................................................................................................................... 325
  17.2.1 BCI Software Platforms Using Commercial High-Level Platforms ................. 327
    17.2.1.1 MATLAB................................................................................................... 327
    17.2.1.2 SIMULINK ................................................................................................ 327
    17.2.1.3 LabVIEW ................................................................................................ 329
  17.2.2 Self-Contained BCI Software Platforms ......................................................331
    17.2.2.1 OpenViBE .................................................................................................. 331
    17.2.2.2 BCI2000 ..................................................................................................... 331
17.3 Impact of BCI Software ........................................................................................................ 335
17.4 Summary and Conclusions ................................................................................................... 335
References ...................................................................................................................................... 336

Abstract

In this chapter, we highlight relevant characteristics of BCI software, discuss the importance of these characteristics in different contexts of BCI research, and document the important impact of BCI software on productivity. The central purpose of BCI software is to facilitate implementation, verification, and dissemination of a wide range of BCI approaches. To do this, BCI software needs to satisfy complex technical demands across a wide array of scientific, clinical, and commercial investigations. BCI software can be implemented either from scratch, using low- or high-level programming environments and toolboxes, or on top of pre-existing BCI software platforms. We assign contemporary BCI software into these categories and investigate their impact on the field of BCI research. Our results demonstrate that only BCI2000 and OpenViBE have enjoyed sustained development and widespread dissemination and have had a strong impact on productivity of the whole field of BCI research.

17.1 INTRODUCTION

It is becoming increasingly clear that BCI systems will eventually realize important new technologies that replace, restore, supplement, or improve function in people affected by neurological disorders. The extent to which these possibilities are realized, that is, the extent to which BCI technologies achieve substantial practical impact, is critically dependent on the ability to effectively and efficiently implement and test different BCI approaches. The purpose of BCI software is to facilitate such implementation and testing across the whole range of research and development, that is, from basic research to clinical translation and, in some cases, even to commercialization. For BCI software to fulfill this purpose, it needs to satisfy two important requirements.

The first requirement, satisfying the technical demands of a particular BCI system, is daunting. An effective BCI must perform three major functions: (1) acquire signals from the brain and/or other
physiological or behavioral sources; (2) analyze these signals to produce output commands; and (3) produce the output and associated feedback. It most often must interact with several different devices (e.g., a biosignal amplifier and an eye tracker) simultaneously, and it must do all this in real time with minimal delays and stable timing. Implementation and integration of these three functions are difficult and time consuming, and require substantial competence in different areas of science and engineering. Thus, unlike in other fields of research in which limited expertise in software engineering can suffice (e.g., implementation of a mathematical algorithm), the design of BCI systems requires individuals who are highly qualified not only in one subject area (e.g., signal processing) but also in other areas.

The second requirement for BCI software is to readily facilitate the implementation, verification, and dissemination of any and all BCI experiments planned in a particular laboratory. Implementation of any specific BCI design, and any subsequent modification, should be relatively easy and should not require extensive reprogramming. This principle is particularly important for human BCI studies since they usually test many different BCI approaches. The ease at which any BCI software can be modified depends mainly on the BCI framework’s architecture, that is, the flexibility, capacity, and practicality of the BCI system’s components and their interactions. This architecture must be general enough to accommodate changes to particular BCI system configurations. As an example, BCI software should not be limited to a certain number of signal channels, a specific sampling rate, or a specific signal analysis method. To facilitate the dissemination of the BCI system among a group of researchers or to other scientists in the field, it should also be possible and practical that system or experimental development, data collection, and data analysis can be performed by different personnel in different locations and with varying hardware. In addition, data stored during online operation should include all meta information (e.g., recording or stimulation parameters) that is necessary for interpretation by personnel other than those individuals who collected the data.

In summary, implementation of BCI software that functions properly, that can be easily adapted to different experimental situations, and that can facilitate operation of whole research programs rather than execution of an individual study, is complex, difficult, and costly. The premise of existing open-source or commercial BCI software is to reduce this complexity, difficulty, and cost.

17.1.1 **Technical Demands**

BCI systems depend on hardware and software to acquire, synchronize, and process electrophysiological signals (electroencephalography/magnetoencephalography [EEG/MEG], electrocorticography [ECoG], or single-neuron activity) and behavioral data (e.g., finger movements, eye-gaze) in real time. The acquisition of these signals is accomplished using specialized hardware devices and requires the implementation of proprietary software interfaces to configure the device and acquire data in real time. Both configuration and type of interface can vary substantially across different devices. Some devices are directly supported by the operating system and require no configuration (keyboard, mouse, etc.), but others require the use of application programming interfaces (APIs) provided by the manufacturer and extensive configuration to define complex parameters of signal acquisition. These APIs can vary substantially in their suitability to provide the acquired data in real time. For example, while some APIs provide the acquired signal data in precise intervals, for example, through call-back notifications, others may not support such notifications or may have imprecise timing owing to limitations in the underlying data connection to the hardware (as is the case with most wireless connections). BCI software may account for these shortcomings, but doing so typically requires the use of substantial processing resources and additional buffers that entail additional acquisition delays.

As an additional complication, to properly relate physiological and behavioral signals, they need to be synchronized with each other. This can be technically difficult, because it may require adjustments for differences in sampling rates and acquisition delays across these signals.

BCI systems that provide continuous feedback have further requirements. To enable users to naturally regulate their neural activity, a BCI system must compute and output feedback within a short period of time (~100 ms or less). In addition, this feedback must be perceived as continuous, which
BCI Software

requires BCI software to update the output signal rapidly (>20 times per second). At this rate, the BCI software has less than 50 ms to perform signal acquisition, processing, and feedback calculations.

Demanding BCI configurations (e.g., including high channel counts/sampling rates, complex signal processing algorithms) may require more than that brief time period, thus necessitating highly optimized coding.

17.1.2 Scope of Investigation

For BCI systems, the scope of investigation is determined by its functional, scientific, and experimental complexity. Understanding these three factors is critical to selecting proper BCI software.

Functional complexity depends on the nature of the investigated BCI problem and is influenced by signal and stimulus modalities, online processing, and feedback and timing requirements. For example, a study in which a single type of brain signal is acquired while visual stimuli are presented at a predetermined timing and sequence could be considered to have relatively low functional complexity. In contrast, a study in which auditory or visual stimuli are presented, and in which their timing or other properties depend on signals acquired from multiple physiological or behavioral signal modalities, could be considered to have relatively high functional complexity.

Scientific complexity increases with the number of different experiments, personnel, and institutions. For example, a study performed by one investigator and one student at one institution could be considered to have low scientific complexity. In contrast, a program project that is performed by multiple investigators and students located at different institutions (with associated variations in experiments and equipment) could be considered to have high scientific complexity. In the first example, the BCI software needs to support a particular experiment executed by one person using one particular kind of equipment. In the latter example, the BCI software needs to support a wide range of experiments executed by many people using diverse equipment in different environments. The difference in technical requirements for BCI software implied by this difference in scientific complexity is enormous but not widely recognized.

Experimental complexity increases with varying experimental situations. For example, a study may conduct a reaction-time experiment to investigate the relationship between sensory stimulus duration and the delay of the neural and behavioral responses for different sensory stimulus modalities (e.g., auditory, visual, or tactile). Investigating this relationship requires post hoc analysis of neural signals (e.g., EEG) and behavioral signals (e.g., button press) for the different experimental conditions under consideration. Properly facilitating not only the conduct of the experiment but also the requisite corresponding offline analyses requires that the BCI software store all signals and experimental parameters in a cohesive and general fashion, and that they can be readily made available to third-party applications (such as MATLAB®) for offline analysis.

Functional, scientific, and experimental complexity typically decreases as a BCI system moves from basic scientific endeavors to a commercial clinical application, concomitant with an increase in technological, scientific, and regulatory maturity and a sharp decrease in variations in experimental or other parameters. Ideally, a BCI software system can accommodate the differing requirements of these different stages of development.

17.2 Implementation

Once investigators have identified the technical demands and scope of their investigation, they are in a good position to select the proper BCI software to implement a specific BCI system (Figure 17.1). The initial and important choice is whether to implement their BCI software from scratch using low- or high-level programming environments, or whether to proceed with a preexisting BCI software platform (see Table 17.1).

Given the limited human and financial resources in most academic laboratories, implementing BCI software from scratch is often only practical if the technical demands and the scope of investigation are limited, such as for early-stage technical demonstrations.
In this case, BCI software may be implemented using open-source or commercial programming platforms such as Python, MATLAB, SIMULINK, or LabVIEW. These platforms provide the ability to rapidly prototype BCI systems, but, without complex additions, with only limited technical complexity.

Some commercial or academic BCI software platforms are implemented on top of these programming languages. For example, BCILAB and g.BCIsys expand MATLAB/SIMULINK with a wide range of BCI-specific functionality and with the convenience of implementing the BCI system.

### FIGURE 17.1
Technical complexity and scope of investigation. This figure highlights the capabilities of different BCI software to implement BCI systems of varying technical complexity and scope of investigation.

### TABLE 17.1
BCI Software Overview

<table>
<thead>
<tr>
<th>Language</th>
<th>License</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Self-Contained BCI Software Platforms</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI2000</td>
<td>C++</td>
<td>GPL GNU</td>
</tr>
<tr>
<td>OpenViBE</td>
<td>C++</td>
<td>GPL Affero</td>
</tr>
<tr>
<td>MATLAB</td>
<td>MATLAB</td>
<td>Commercial</td>
</tr>
<tr>
<td>SIMULINK</td>
<td>SIMULINK</td>
<td>Commercial</td>
</tr>
<tr>
<td>LabVIEW</td>
<td>G</td>
<td>Commercial</td>
</tr>
<tr>
<td><strong>Commercial High-Level Platforms</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EEGLab</td>
<td>MATLAB</td>
<td>GPL GNU</td>
</tr>
<tr>
<td>BCILAB</td>
<td>MATLAB</td>
<td>GPL GNU</td>
</tr>
<tr>
<td>PsychoPy</td>
<td>Python</td>
<td>GPL GNU</td>
</tr>
<tr>
<td>BioSig</td>
<td>MATLAB</td>
<td>GPL GNU</td>
</tr>
<tr>
<td>rtsBCI</td>
<td>MATLAB</td>
<td>GPL GNU</td>
</tr>
<tr>
<td>FieldTrip</td>
<td>MATLAB</td>
<td>GPL GNU</td>
</tr>
<tr>
<td>MNE</td>
<td>MATLAB</td>
<td>BSD</td>
</tr>
<tr>
<td>Psychophysics</td>
<td>MATLAB</td>
<td>MIT</td>
</tr>
<tr>
<td>g.HIsys</td>
<td>LabVIEW</td>
<td>Commercial</td>
</tr>
<tr>
<td>g.BCIsys</td>
<td>SIMULINK</td>
<td>Commercial</td>
</tr>
</tbody>
</table>

**Toolboxes**:
- EEGLab: http://sccn.ucsd.edu/eeglab
- BCILAB: http://sccn.ucsd.edu/wiki/BCILAB
- PsychoPy: http://www.psychopy.org
- BioSig: http://biosig.sourceforge.net
- rtsBCI: http://biosig.sourceforge.net
- FieldTrip: http://www.fieldtriptoolbox.org
- MNE: http://martinos.org/mne
- Psychophysics: http://psychtoolbox.org
- g.HIsys: http://www.gtec.at/Products/Software
- g.BCIsys: http://www.gtec.at/Products/g.BCIsys
using visual data flow programming. This convenience allows for rapid prototyping and verification
of a specific BCI approach. At the same time, the underlying platform (i.e., MATLAB, SIMULINK,
or LabVIEW) usually imposes important limitations in system performance or technical complexity
(e.g., processing rate and latency, limited ability to simultaneously acquire signals from different
devices). It also impedes commercialization because of prohibitive licensing costs.

In contrast, self-contained BCI software platforms do not depend on underlying commercial
software and do not have their limitations. For example, BCI2000 and OpenViBE are based on stan-
dard C++ and provide general-purpose BCI functionality at high performance and without depend-
ing on other software. The modular and general-purpose nature of these software packages lends
itself well to a wide scope of systematic BCI investigations. For example, the support for a wide
range of data acquisition systems in BCI2000 and OpenViBE facilitates the use of the BCI system
in other environments. These systems also facilitate translating the BCI system into a clinical (and
eventually commercial) application. OpenViBE is focused on providing visual data flow program-
ing to facilitate the rapid prototyping of BCI systems. In contrast, BCI2000 is focused on pro-
viding a highly stable and performant general-purpose BCI software infrastructure that facilitates
systematic and large-scale investigations.

17.2.1  BCI Software Platforms Using Commercial High-Level Platforms

Commercial high-level platforms (MATLAB, SIMULINK, or LabVIEW) have been used in the
development of early real-time BCI demonstrations (Lauer et al. 1999; Guger et al. 1999). The sec-
tions below give an overview of their impact on BCI research.

17.2.1.1  MATLAB

MATLAB (MATrix LABoratory) is a commercial programming language for numerical com-
puting that supports Linux, Windows, and Mac OS X and that has been continuously developed
by Mathworks Inc. since 1984. MATLAB is principally focused on supporting the development
of mathematical algorithms but can also plot data and create user interfaces. Because of its ease
of use and expansive functionality, MATLAB has become popular with scientists and engineers.
MATLAB is relatively affordable for students and faculty (e.g., $100 for the basic functionality and
$29 for each additional toolbox) but carries a steep price tag for commercial users (e.g., $2150 for the
basic functionality and $1000 for each additional toolbox). Thus, MATLAB is mostly used within
the academic setting and in a limited fashion (for rapid prototyping) in the industrial setting. The
high cost of MATLAB has motivated the development of the open-source GNU Octave software,
which replicates large portions of the basic MATLAB functionality.

MATLAB’s popularity among scientists and engineers has led to the development of specialized
toolboxes for the presentation of behavioral paradigms (e.g., Psychophysics Toolbox Brainard 1997)
and the post hoc analysis and visualization of biosignals (e.g., EEGLab Delorme and Makeig 2004,

EEGLab, BioSig, and FieldTrip have recently been extended into MATLAB toolboxes for the
rapid prototyping and evaluation of online BCIs, called BCILAB, rtsBCI, and FieldTrip buffer,
respectively. These extended toolboxes make use of MATLAB functionality to process, classify,
and visualize signals in real time (see Figures 17.2 and 17.3). The capabilities of BCILAB have
been illustrated in several real-time demonstrations that range from EEG signal visualization
(Mullen et al. 2013) to EEG state classification (Mullen et al. 2015) and the decomposition of
EEG signals into independent components (Hsu et al. 2016).

17.2.1.2  SIMULINK

SIMULINK is a commercial graphical user interface that expands MATLAB with a graphical block pro-
gramming interface. Each block in this interface can contain either custom or preexisting code. The ability
to seamlessly combine preexisting code into new applications makes SIMULINK well suited for users
FIGURE 17.2 Basic architecture of BCILAB. It is based on four layers: (1) The dependency layer provides external functionality, for example, MATLAB and required toolboxes, machine learning toolboxes, and user interfaces. (2) The infrastructure layer provides functionality for GUIs, parallel computing, disk caching, and helper functions for testing and deployment. (3) The plugin layer provides BCI-specific signal processing and machine learning functionality. (4) The framework layer provides the scripting interface to interact with the first layers of the BCILAB toolbox.

FIGURE 17.3 User interface of BCILAB. This screenshot shows the main menu (top center), a model visualization (bottom), a model configuration dialog (left), an evaluation setup dialog (top right), and the script editor (bottom right).
BCI Software

with limited programming skills. The g.BCIsys software, a commercial SIMULINK-based software package, makes use of this functionality to provide a front-end interface for rapid BCI research and development to users of g.tec hardware (see Figure 17.4). The capacities of g.BCIsys have been demonstrated in several real-time demonstrations (Guger et al. 2012a,b; Ortner et al. 2013; Kapeller et al. 2013).

17.2.1.3 LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a commercial graphical block programming interface software for data acquisition, signal processing, visualization, and controlling of instrumentation. The most recent version of LabVIEW (2016) runs on Windows 7, 8.1, and 10 and, with limited functionality, on Mac OS X (10.10, 10.11) and Linux (Red Hat, Scientific, openSUSE). National Instruments has been developing LabVIEW since 1983, mainly to serve as a real-time processing frontend for their NI-DAQ (Data Acquisition) series hardware. This type of hardware provides affordable analog and digital input/output at relatively high sampling rates (e.g., 100 kHz) and resolution (e.g., ±500 mV dynamic range, 3 μV resolution at 16 bits). In combination with appropriate pre-amplifiers, NI-DAQ hardware is suitable for a wide range of biosignal applications. While these boards lack regulatory approval for clinical use in humans, their affordable price (e.g., 1/10th of a comparable biosignal acquisition system approved for human use) made them popular with animal experiments. The ability to interface with external instrumentation through analog and digital NI-DAQ board inputs and outputs, together with an increased availability of data acquisition modules for third-party biosignal acquisition systems (e.g., g.HIsys, g.tec, Austria†; OpenEEG‡; OpenBCI§), has led to some adoption of LabVIEW within the BCI community. In this context, LabVIEW has been the basis for several BCI demonstrations that range from EMG-based assistive technology (Huang et al. 2006), to NIRS-based communication (Matthew et al. 2008),

* g.BCIsys—g.tec’s brain–computer interface research environment. http://www.gtec.at/Products/Complete-Solutions/g BCIsys-Specs-Features.
† g.HIsys—g.tec’s high-speed online processing under LabVIEW. http://www.gtec.at/Products/Software/Software -options/ (show)/1711/#p1711.
‡ OpenEEG for LabVIEW. https://sourceforge.net/projects/ openeeglabview.
to steady-state visual evoked potential (SSVEP)–based wheelchair control (Singla et al. 2014). In addition to these demonstrations, LabVIEW has been used to implement the BCI system Craniux (Degenhart et al. 2011), which is illustrated in Figures 17.5 and 17.6. Craniux has been used for BCI studies in monkeys and for humans with tetraplegia (Wang et al. 2013; Collinger et al. 2013, 2014).

17.2.2 Self-Contained BCI Software Platforms

Self-contained BCI software platforms are implemented in low-level programming languages (e.g., C++), compiled into native machine code, and executed without dependencies on commercial libraries. This makes these platforms ideal for supporting systematic scientific and clinical studies.
that entail complex experiments with high technical demands (e.g., high channel count or sampling rate, low latencies) as well as a wide scope of investigation (e.g., the economical deployment across many laboratories and users). While many self-contained BCI software platforms have been proposed (e.g., BCI++ Perego et al. 2009, xBCI Susila et al. 2010, and TOBI Breitwieser et al. 2012), only BCI2000 and OpenViBE have enjoyed significant adoption beyond the authors’ own laboratory. Sections 17.2.2.1 and 17.2.2.2 describe and summarize these two platforms.

17.2.2.1 OpenViBE

The OpenViBE software platform has been developed since 2007 (Reynard et al. 2010). The most recent version of OpenViBE (1.3.0) is licensed under the GNU Affero General Public License, a copyleft license that provides the right to freely distribute copies and modified versions of a work with the stipulation that the same rights be preserved in derivative works. OpenViBE is implemented in C++ and compiled into binaries that run on Windows 7 and 10, as well as on several Linux distributions (Ubuntu 12.04, 14.04, and 16.04; Fedora 20 and 21). The recently founded startup Mensia Technologies is translating OpenViBE’s core into CertiViBE, a hardened and medically certifiable BCI software platform for clinical applications.

OpenViBE is based on an architecture that facilitates the integration, expansion, and configuration of modular functionality (Figure 17.7), while the graphical interface makes OpenViBE easy to use for a wide range of investigators, including engineers, scientists, and clinicians (Figure 17.8). These two factors make OpenViBE well suited to supporting the implementation of different BCI approaches. The capabilities of OpenViBE have been demonstrated in several publications, including EEG-based workload estimation (Mühl et al. 2014; Frey et al. 2016), SSVEP-based BCIs (Zhao et al. 2015; Martisius and Damasevičius 2016), and motor imagery BCIs (Jeunet et al. 2015, 2016a,b).

17.2.2.2 BCI2000

The BCI2000 software platform has been developed since 1998 (Schalk et al. 2004; Schalk and Mellinger 2010). Its development is a major component of the recently established NIH-funded National Center for Adaptive Neurotechnologies.* The most recent version of BCI2000 (3.6) is
licensed under version 3 of the GNU General Public License, a copyleft license that is compatible with other open-source licenses (e.g., LGPLv3, Apache 2.0, XFree 1.1). GPL v3 facilitates the integration of functionality from other open-source projects. BCI2000 is implemented in C++ and compiled into binaries that run on Windows 7, 8.1, and 10. Its implementation is based on a model that can describe any BCI system (Schalk et al. 2004; Mason and Birch 2003). In accord with this model, and as shown in Figure 17.9, BCI2000 has four modules that communicate with each other: Source (data acquisition and storage), Signal Processing, User Application, and Operator Interface. The modules communicate through a documented network-capable protocol based on TCP/IP.

The implementation of BCI2000 is highly optimized, so that it can support even very demanding BCI configurations with good timing characteristics (Wilson et al. 2010).

The BCI2000 data storage format accommodates variations in the digitized signals (e.g., in number of channels, sampling rate), defines the operating protocol, and includes a record of all events (e.g., feedback to user, device control, artifact detection) that occur during operation. BCI2000 has a roster of existing implementations with primarily technical documentation. These implementations can realize different BCI designs and methods that are readily usable and employ readily available and relatively inexpensive hardware components (Figure 17.10). BCI2000 is described in detail in a book [38], as well as in multiple book chapters (Mellinger and Schalk 2007, 2010; Wilson and Schalk 2010; Wilson et al. 2012; Allison et al. 2012; Brunner et al. 2013) and peer-reviewed articles (Schalk et al. 2004; Wilson et al. 2010, 2009; Schalk 2009).
FIGURE 17.9  Basic structure of BCI2000. It has three interchangeable modules (Source, Signal Processing, and User Application) and an Operator module that facilitates the configuration and communication across all modules.

FIGURE 17.10  User interface of BCI2000. This screenshot shows the use of BCI2000 in a P300 speller study that acquires EEG signals along with eye gaze. The user is presented with a P300 speller window (top left), while the operator monitors the user’s EEG (top right) and eye gaze as presented as an overlay over a copy of the P300 speller window (green dot, top left) and as a time trace (center right). The configuration of this experiment can be changed through a parameter window (bottom).
The software architecture and functionality make BCI2000 well suited for studies with high technical complexity and a wide scope of investigation, for example, laboratory research conducted across multiple laboratories. The utility of BCI2000 is documented by the more than 1000 published studies that have used it for BCI-related experimentation. These studies include cursor control using EEG (Wolpaw and McFarland 2004; McFarland et al. 2008, 2010), ECoG (Leuthardt et al. 2004, 2006; Wilcon and Lacoboni 2006; Felton et al. 2007; Blakely et al. 2009; Miller et al. 2010; Bundy et al. 2016), and MEG signals (Mellinger et al. 2007). BCI2000 was used to work toward a speech prosthesis that uses ECoG signals (Leuthardt et al. 2011; Pei et al. 2011; Mugler et al. 2014; Herf et al. 2015; Martin et al. 2016). It was also used for control of a wheelchair (Kaufmann et al. 2014), a humanoid robot (Bell et al. 2008), and a quad-copter (Doud et al. 2011; LaFleur et al. 2013) by a noninvasive BCI and for exploring the value of P300 evoked potentials (Sellars et al. 2006, 2010; Vaughan et al. 2006; Nijboer et al. 2008; Furdea et al. 2009; Kübler et al. 2009; Townsend et al. 2010, 2016; Cecotti and Gräser 2011; Kaufmann et al. 2011, 2013; Brunner et al. 2011; Aloise et al. 2012; Käthner et al. 2013), steady-state auditory evoked potentials (SSAEPs) (Hill and Schölkopf 2012), SSVEP (Allison et al. 2008; Yin et al. 2013), and the combination of P300 with SSVEP signals (Yin et al. 2014) for BCI purposes. BCI2000 has been used to implement real-time BCIs for high-resolution EEG techniques (Mattia et al. 2008) and for BCI control of assistive technologies (Cincotti et al. 2008). BCI2000 has also provided the basis for the first large-scale clinical evaluations of BCI technology for the needs of people with severe motor disabilities (Vaughan et al. 2006; Nijboer et al. 2008; Kübler et al. 2005; Spüler et al. 2012; Kaufmann et al. 2013; Zickler et al. 2012, Holz et al. 2015; Vansteensel et al. 2016) and the first applications of BCI technology to functional restoration in patients with chronic stroke (Buch et al. 2008; Daly et al. 2009; Wisneski et al. 2008; Cincotti et al. 2012). Finally, several studies have used BCI2000 for purposes other than online BCI control, for example, the optimization of BCI signal processing routines (Fuentes Cabrera and Dremstrup 2008; Royer and He 2009; Yamawaki et al. 2006) and the mapping of cortical function using ECoG (Leuthardt et al. 2007; Miller et al. 2007a, b; Schalk et al. 2007, 2008; Kubanek et al. 2009; Brunner et al. 2009; Roland et al. 2010; Gupta et al. 2014; Korostenskaja et al. 2014). Finally, BCI2000 has provided the basis for a commercial functional mapping system called cortiQ (Prueckl et al. 2013).

FIGURE 17.11 Impact of BCI software platforms as measured by scientific publications.
The ultimate purpose of BCI software is to facilitate implementation and testing of different BCI approaches. Thus, the success of BCI software in achieving this purpose can be measured by the number of scientific studies that they have supported. The illustration in Figure 17.11 indicates the number of publications that have been supported by different BCI software packages. Among these software packages, OpenViBE and BCI2000 together account for 93% of all publications. Additional analyses demonstrate evidence (Figure 17.12b) that the user base of OpenViBE and BCI2000 is substantially different: users of OpenViBE are likely to be active in the computer science and engineering domains, whereas BCI2000 appears to primarily cater to scientists in the neuroscience and medicine domains.

The benefit of the use of BCI software is a reduction in the time, risk, and cost associated with BCI system development. The large number of BCI studies enabled by existing BCI software implies that many of these studies have likely only been possible because of existing BCI software and that the accumulated economic benefits of using existing BCI software are likely in the range of tens of millions of US dollars.

### 17.3 IMPACT OF BCI SOFTWARE

The ultimate purpose of BCI software is to facilitate implementation and testing of different BCI approaches. Thus, the success of BCI software in achieving this purpose can be measured by the number of scientific studies that they have supported. The illustration in Figure 17.11 indicates the number of publications that have been supported by different BCI software packages. Among these software packages, OpenViBE and BCI2000 together account for 93% of all publications. Additional analyses demonstrate evidence (Figure 17.12b) that the user base of OpenViBE and BCI2000 is substantially different: users of OpenViBE are likely to be active in the computer science and engineering domains, whereas BCI2000 appears to primarily cater to scientists in the neuroscience and medicine domains.

The benefit of the use of BCI software is a reduction in the time, risk, and cost associated with BCI system development. The large number of BCI studies enabled by existing BCI software implies that many of these studies have likely only been possible because of existing BCI software and that the accumulated economic benefits of using existing BCI software are likely in the range of tens of millions of US dollars.

### 17.4 SUMMARY AND CONCLUSIONS

Systematic research and development of BCIs is critically facilitated by existing BCI software platforms that support the ability to effectively and efficiently implement and verify BCI systems. To fulfill this promise, BCI software platforms need to satisfy complex technical demands and support the implementation, verification, and dissemination of many different BCI approaches. Two fundamental types of BCI software platforms exist, those implemented using existing high-level platforms, and those implemented using low-level programming languages. Over the past two decades, many BCI software platforms have been developed, but only two of them have enjoyed sustained development and widespread dissemination and have succeeded in making a strong impact on productivity of the whole field of BCI research. The increasing complexity of BCI approaches (e.g., multimodal signal acquisition and processing) will likely further increase the value of BCI software and will place new demands on their capacities.
REFERENCES


