ВЕСТНИК НАЦИОНАЛЬНОГО ИССЛЕДОВАТЕЛЬСКОГО ЯДЕРНОГО УНИВЕРСИТЕТА “МИФИ”, 2021, том 10, № 6, с. 540–549

ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

УДК 004.5

ПРИНЦИПЫ ДЕКОМПОЗИЦИИ В РАЗРАБОТКЕ ПРОГРАММНОЙ АРХИТЕКТУРЫ ДЛЯ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ BCI-ВЗАИМОДЕЙСТВИЯ

© 2021 г. Т. И. Возненко[1,*], А. И. Петрова[1], Е. В. Чепин[1]

[1] Институт интеллектуальных кибернетических систем, Национальный исследовательский ядерный университет “МИФИ”, Москва, 115409, Россия

**e-mail: TIVoznenko@mephi.ru*

Поступила в редакцию 03.02.2022 г.
После доработки 04.02.2022 г.
Принята к публикации 08.02.2022 г.

Нейрокомпьютерные интерфейсы (Brain-Computer Interfaces, BCI) являются современным и развивающимся принципом взаимодействия с различными вычислительными устройствами, значительно расширяющим возможности человека. Актуальной задачей является быстрая и гибкая разработка адаптивных BCI-приложений под различные задачи и для широкого круга пользователей – в особенности, для людей с ограниченными возможностями. В данной статье рассмотрен такой аспект разработки, как декомпозиция программной архитектуры системы на составные модули, с учетом специфики BCI-интерфейсов. Был проанализирован популярный в сфере разработки программ, в том числе и включающим BCI-взаимодействие, подход функциональной декомпозиции и предложено использование декомпозиции на основе нестабильности. Сравнение двух данных подходов было проведено на примере разработки системы, реализующей управление роботизированным устройством с помощью BCI – обозначены ее исходные требования, описаны варианты ее программной декомпозиции по признакам функциональности и нестабильности, а также проведен анализ необходимости выполнения доработок полученных компонентов при различных модернизациях системы. В результате был сделан вывод о применимости обоих подходов декомпозиции для решения задачи проектирования систем с BCI-взаимодействием, однако декомпозиция на основе нестабильности показала большую устойчивость архитектуры приложения к изменениям в связи с появлением новых требований по функционалу. Следовательно, при таком методе декомпозиции упрощается и ускоряется адаптация BCI-приложения к изменяющимся условиям эксплуатации, что является выгодным с экономической точки зрения и положительно влияет на пользовательский опыт.

Ключевые слова: нейрокомпьютерный интерфейс, функциональная декомпозиция, декомпозиция на основе нестабильности, архитектура программного обеспечения

DOI: 10.56304/S2304487X21060122

ВВЕДЕНИЕ

В настоящее время нейрокомпьютерные интерфейсы (Brain-Computer Interfaces, BCI) являются довольно популярной темой для многих исследований в сфере человеко-машинного взаимодействия. Данная технология уже довольно давно известна сама по себе, однако с развитием данной отрасли в результате многочисленных исследований и проектов появляются новые типы BCI и парадигмы его применения, новые методы обработки сигнала с BCI и оптимизация существующих, различные аспекты применения BCI в медицине и многое другое. Данные достижения направлены в первую очередь на удобство, надежность и качество использования BCI человеком и расширение его возможностей взаимодействия с внешней средой. Однако помимо таких основополагающих и, безусловно, важнейших направлений исследований BCI, существует также ряд вторичных, но в перспективе играющих важную роль задач, заключающихся в разработке BCI как программного продукта, предоставляемого конечному пользователю.

Рассматривая программные системы с самыми привычными широкому кругу лиц интерфейсами человеко-машинного взаимодействия – графическими интерфейсами приложений – можно отметить, что в большинстве случаев проектирование и разработка таких систем осуществляются в соответствии с определенными лучшими практиками. Эти практики работают на различных этапах жизненного цикла программного обеспече-

ния (ПО) – от сбора и систематизации требований (пользовательских и функциональных) до применения в архитектуре программных компонентов приложения паттернов проектирования. Благодаря применению подобных практик процесс разработки ПО становится определеннее, согласованнее и быстрее, а выпускаемый программный продукт будет сопровождаемым, надежным и отвечающим всем необходимым требованиям конечных пользователей.

В случае с BCI мы наблюдаем иную ситуацию. На данный момент серийно поставляемые BCI-решения, позволяющие конечному пользователю делать какие-либо ежедневные задачи без сложной предварительной настройки, отсутствуют. Если не рассматривать медицинские системы, то BCI-приложения можно разделить на 2 группы: экспериментальные образцы – которые были заявлены отдельными группами исследователей в рамках своих докладов и научных статей, но так и не вошли в массовое производство и использование; и коммерческие решения для исследователей (такие, как BCI-устройства от Emotiv, g.tec, Muse и др.), предоставляющие широкий спектр возможностей по настройке, программированию и исследованию BCI-систем ученым и инженерам, но никак не конечным пользователям. И при этом именно в первом варианте многие экспериментальные образцы делаются с расчетом на ряд пользовательских задач. Однако в процессе исследований основное внимание уделяется преимущественно не архитектуре ПО, а основному объекту исследования – что является естественным и правильным. Но не перенеся подобный проект в рамки более строгих архитектурных решений при дальнейшей активной разработке и эксплуатации потенциально можно столкнуться с рядом сложностей, которые могут сделать разработку более медленной и дорогой.

С целью повысить шансы активного развития экспериментальных BCI-продуктов до полноценных сервисов и приложений с широкой пользовательской аудиторией необходимо в рамках их реализации придерживаться определенных методик проектирования и разработки. Не последнюю роль среди методик играет декомпозиция программной системы на отдельные компоненты. В данной статье мы рассмотрим различные подходы, применяемые для декомпозиции систем на модули, и их применимость в сфере проектирования приложений BCI-взаимодействия с учетом специфики сбора и обработки сигналов ЭЭГ (электроэнцефалограммы) и возможных парадигм взаимодействия с пользователем.

СОВРЕМЕННЫЕ МЕТОДЫ ДЕКОМПОЗИЦИИ

Для оценки текущей ситуации по обозначенной проблеме проектирования программных средств в BCI-системе рассмотрим основные тенденции и подходы к разработке таких систем. На сегодняшний день в большинстве проектов применяется архитектура, состоящая из модулей, разделенных по функциональному признаку: сбор сигнала, обработка сигнала и взаимодействие – с роботом и с пользователем [1, 2]. Модули инкапсулируют в себе соответствующие данные и алгоритмы, и реализуют необходимые программные интерфейсы, что позволяет разрабатывать данные модули независимо и упрощает их интеграцию. Подобный подход к декомпозиции является легким для понимания и интуитивным, поэтому довольно широко распространен.

Повышение универсальности и возможности повторного использования и гибкой интеграции BCI-решений в различные системы является актуальной проблемой, но большинство предложенных подходов к ее решению сводились в основном к решению данной задачи для внутренних компонентов системы – как, например, в ряде фреймфорков, предлагаемых Ventur [2, 3], OpenVibe [4], BCILAB [5], где четко определено разделение между компонентами записи сигнала и его обработкой. Несомненным преимуществом данного подхода является совместимость с различными устройствами для записи данных. И хотя это по-прежнему накладывает на разработчика такой системы обязанность обеспечить взаимодействие с тем или иным устройством, но с точки зрения программной логики данные устройства являются взаимозаменяемыми и используются в качестве неких “плагинов”, а подобное ПО становится расширяемым.

Функциональное разделение может быть также применено и внутри перечисленных выше основных модулей. Например, в рамках общего функционала по обработке сигнала OpenVibe [4] позволяет конструировать конвейеры обработки ЭЭГ-данных, включающие в себя различные подзадачи. Отдельного упоминания заслуживает то, что конструирование подобных сценариев производится с помощью графических элементов и может осуществляться далекими от программирования людьми, что значительно повышает степень эффективного использования такого ПО. BCILAB [5] предоставляет возможность программной интеграции между собой ряда независимых модулей, реализующих ту или иную обработку сигнала и выделения признаков из него.

Разбиение системы на составляющие компоненты также способствует определенной гибкости реализации, позволяя каждый изолированный модуль реализовать максимально оптималь-

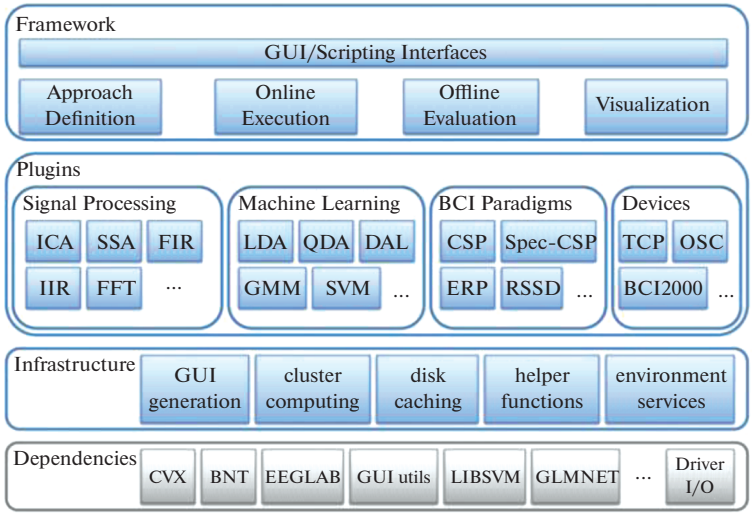

Рис. 1. Архитектура BCILAB [5].

но с самыми подходящими для этого технологическими решениями. Так, например, блок обработки сигналов с помощью математических методов или машинного обучения может быть реализован в виде параллельно выполняющихся блоков в рамках единого вычислительного узла [6], либо размещен на выделенном сервере [7], либо реализован в различных облачных решениях [8], что ускорит в целом обработку сигналов и упростит хранение данных. Также большим преимуществом является возможность одновременного развертывания и использования различных блоков сбора и обработки сигналов – как для источников сигналов разных типов [9] (в рамках гибридных технологий), так и для одновременного сбора сигнала у нескольких пользователей.

Разделение по функциональному признаку не является исчерпывающим и единственно верным способом – зачастую в рамках проектирования различных программных систем такой подход в принципе полностью не отражает особенностей системы, в частности подверженных изменениям в будущем, что в ряде случаев спровоцирует проявление ряда проблем, которые скажутся на дальнейшей разработке, а также сопровождении продукта. Альтернативным способом декомпозиции, учитывающей изменчивость компонентов системы является декомпозиция на основе нестабильности [10, 11], которая довольно распространена в качестве лучшей практики в проектировании программных систем – особенно, для сервис-ориентированных решений. Однако в случае систем с использованием BCI применение такого подхода представлено не было. Тем не менее, принимая во внимание все преимущества данного метода, следует рассмотреть возможность и целесообразность его применения для проектирования BCI-систем, учитывая все их особенности.

Отдельного внимания заслуживает внутренняя архитектура и возможности BCILAB [5] (представлена на рис. 1). Фреймворк заявляется как подходящий для исследовательских проектов и макетирования приложений, связанных с BCI. Архитектура разделена на несколько уровней, каждый из которых включает в себя ряд тех или иных компонентов. Сам фреймворк позволяет использовать отдельные компоненты, комбинировать между собой, а также разрабатывать собственные плагины – таким образом, является довольно гибким инструментом для разработки различных приложений исследовательского уровня. Из описания архитектуры, данного разработчиками, можно утверждать, что она наиболее приближена к вышеупомянутому принципу декомпозиции на основе нестабильности, что будет только повышать ее повторное использование и расширение.

Таким образом, как мы видим, современное состояние BCI-проектов исследовательского уровня включает в себя много удачных архитектурных решений и позволяет оптимально осуществлять подготовку и проведение экспериментов, а также реализацию прототипов устройств. Тем не менее, жизненный цикл и требования исследовательских проектов будут существенно отличаться от коммерческих. В свою очередь, различные коммерческие продукты, данные о которых удалось найти в открытом виде, предоставляют решения, ориентированные в основном на пользователя-разработчика BCI-систем, а не на того конечного пользователя, для улучшения качества жизни которого данная разработка и осуществляется. Ввиду того, что мы видим по-насто-

ящему мало примеров использования методик проектирования систем, готовых для массового выпуска и длительной эксплуатации и поддержки, следует рассмотреть их применимо к данному сегменту ПО и выявить возможные ограничения и проблемы их использования, которые накладывает предметная область.

АНАЛИЗ ТРЕБОВАНИЙ

Среди аспектов, которые влияют на архитектуру BCI-системы, присутствует способ взаимодействия с пользователем и внешними устройствами, а также ее назначение, что выражается в требованиях к системе. Проектирование любой системы должно осуществляться с учетом ее требований. Данный этап также описан как отдельная стадия жизненного цикла системы по стандарту ISO/IEC/IEEE 12207:2017 [12]. В данной работе мы конечно не сможем охватить весь спектр требований, связанных с BCI и подходящих большинству приложений, однако проанализируем самые основные.

Требования могут в первую очередь касаться выбранной парадигмы взаимодействия с пользователем (например, с помощью вызванных потенциалов или мысленных команд) и детализации по ней (сколько команд может выдать пользователь, по какому алгоритму и с каким временным интервалом). В рамках конкретной парадигмы также следует обратить внимание на пользовательский опыт и повышение удобства использования интерфейса – например, если система ориентирована на взаимодействие посредством мысленных движений [13] или других типов мыслеобразов [14–16], а также если система включает в себя элементы обратной связи с пользователем [17]. Помимо этого следует учитывать возможную интеграцию BCI с другими устройствами – как в рамках управления данными устройствами [18, 19], так и в рамках реализации гибридного интерфейса [20, 21]. Наконец, необходимо рассмотреть среду, в которой будет использоваться система – и учесть как влияющие непосредственно на оператора факторы [16], так и различные условия эксплуатации технических средств, особенно вне лабораторных условий.

Помимо BCI-специфичных требований должны присутствовать и чисто функциональные требования к системе – описания задач, которые она должна выполнять, результаты ее работы. В качестве примера рассмотрим в контексте составления требований и проектирования 3 типичных BCI-приложения с базовыми требованиями к таким приложениям:

1) Проведение экспериментов. Данное приложение должно осуществлять параллельную демонстрацию стимулов испытуемому и запись данных с того или иного BCI-устройства в логфайл в оговоренном формате и с заданным именем. Демонстрация стимулов должна осуществляться в соответствии с той или иной схемой эксперимента.

2) Управление роботизированными устройствами. Данная система должна предоставлять пользователю возможность с помощью того или иного BCI-устройства передавать команды на то или иное роботизированное устройство.

3) Непрерывный сбор данных с целью мониторинга. Данная система должна обеспечивать запись данных с того или иного BCI-устройства, их сохранение и отображение на панели оператора в режиме реального времени.

Среди специфичных для каждого приложения требований мы можем выделить некоторые общие признаки, благодаря которым можно наглядно представить то, на сколько и в чем разнятся данные приложения. Данные признаки представлены в виде критериев в таблице 1, где для каждого приложения определено то, насколько оно должно удовлетворять данному критерию. Представленная в таблице 1 информация сформирована на основе анализа функциональности существующих разработок для каждого типа приложения [17–19].

Конечно, абсолютно все BCI-приложения не будут делиться на эти 3 категории, и к тому же в одном и том же приложении может быть реализовано несколько упомянутых функций. Более того, сами эти функции могут иметь более конкретизированные требования в большем количестве, чем представлено тут. Но в целом, данные упрощенные примеры могут, во-первых, служить хорошей отправной точкой для анализа. Во-вторых, все богатство требований и его влияние на систему в зависимости от выбранного способа ее декомпозиции будет также рассмотрено далее. Из данного раздела можно сделать вывод о том, что само по себе наличие BCI-интерфейса в системе является источником дополнительных требований, которые могут сильно варьироваться в зависимости от целей и задач системы.

ДЕКОМПОЗИЦИЯ НА ОСНОВНЫЕ КОМПОНЕНТЫ

Декомпозиция системы на программные компоненты является давно устоявшимся этапом проектирования и применяется для удобства независимой разработки различных компонентов. Рассмотрим уже упомянутые методы декомпозиции на основе функциональности и нестабильности. В качестве критериев того, насколько данные методы декомпозиции уместны и приемлемы для проектирования BCI-приложений, будут рассмотрены а) возможность масштабирования ком-

Таблица 1. Сравнение базового набора требований для BCI-приложений

	Проведение экспериментов	Управление роботизированными устройствами	Мониторинг
Работа в режиме реального времени	Да	Да	Да
Пользователю нужно активно взаимодействовать с системой	Да	Да	Нет
Присутствует обратная связь	Да	Опционально	Нет
Режим “администратора” для настройки/сервисного обслуживания приложения	Да	Нет	Да
Хранение данных сигнала	Локально	Нет	Локально/Удаленно
Интерпретация сигнала в команды	Нет	Да	Нет

Таблица 2. Изменение программных компонентов функционально спроектированной системы

Новая возможность	Изменяемые текущие компоненты	Добавляемые компоненты
Изменение количества управляемых устройств	Логика формирования команд с учетом наличия новых устройств (*F: 2.3, F: 2.4*)	Контроллер управления несколькими устройствами (+*F: 3.3*), элементы интерфейса для работы с несколькими устройствами (+*F: 1.3*), компоненты-драйверы для взаимодействия с новыми устройствами (+*F: 3.2.[1...N]*)
Расширение возможных парадигм взаимодействия	Реализация дополнительных методик обработки сигнала в соответствии с выбранной парадигмой (*F: 2.2*), внесение изменений в логику формирования команд (*F: 2.3*), реализация/доработка интерфейса для выбранной парадигмы взаимодействия (*F: 1.1, F: 1.2),* модификация количества выдаваемых команд (*F: 2.4*)	Модуль настройки системы для работы (интерпретации команд) в рамках той или иной парадигмы взаимодействия (+*F: 2.5*)

понентов в зависимости от количества используемых устройств; б) простота адаптации функционала системы к изменяющимся требованиям на примере внедрения новых парадигм взаимодействия.

Функциональная декомпозиция (functional-based) подразумевает разбиение системы на элементы в соответствии с функциями, которые они выполняют. Как уже было упомянуто выше, в настоящее время широко используется разбиение на модули, отвечающие за запись и обработку сигнала, его интерпретацию, взаимодействию с роботизированным устройством и пользователем [1–3]. Исходя из этого, разбиение системы по функциональному признаку будет представлять собой набор перечисленных компонентов. Данное разбиение можно проиллюстрировать рисунком 2, на котором представлен пример разбиения по функциональному признаку для описанной выше системы управления роботизированным устройством. Диаграмма отображает разбиение системы на ряд основных программных компонентов, логически сгруппированных в рамках 3 уровней. Каждый из компонентов представляет из себя программный элемент функциональности, который может быть реализован как набором классов в рамках единой системы, так и самостоятельным сервисом. Каждый уровень имеет порядковый номер, и также все компоненты пронумерованы в пределах каждого уровня.

Система может в будущем потребовать изменений на всех трех обозначенных уровнях, которые могут произойти от расширения круга пользователей системы, либо из-за изменений в контексте и задачах текущих пользователей. В качестве примера рассмотрим ряд возможных изменений системы, связанных с добавлением новых возможностей и доработки существующих. В таблице 2 приведено, какие из существующих компонентов, представленных на рисунке 2, будут подвергнуты доработке в рамках реализации этих изменений.

Таким образом, мы видим, что в рамках каждого такого расширения функционала система

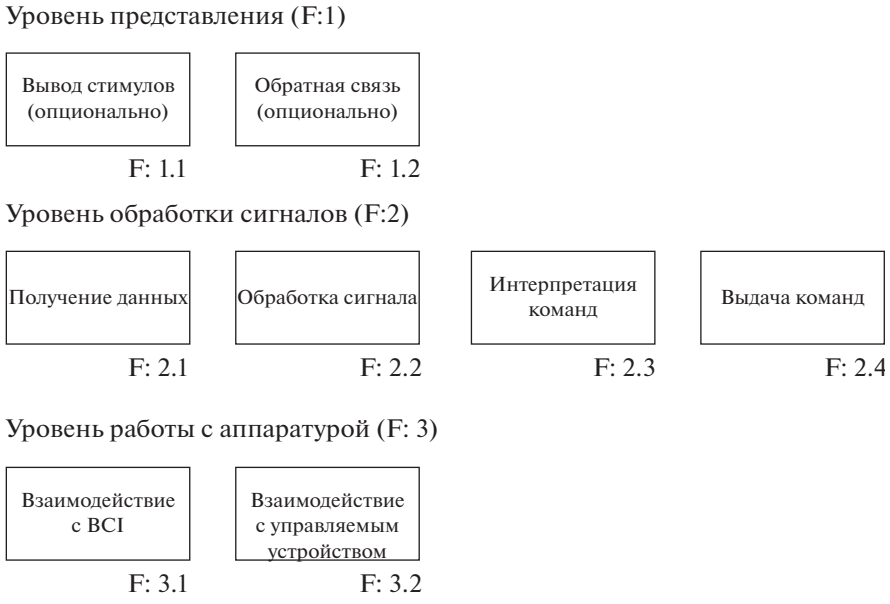

Рис. 2. Функциональная декомпозиция системы управления роботизированным устройством с помощью BCI.

претерпевает значительные изменения – множество ее компонентов изменяются сами либо расширяются новыми, за этим также следуют изменения интерфейсов этих компонентов. Некоторые из компонентов являются излишне хрупкими и модифицируются практически по любому поводу – к таким компонентам относятся *Обработка сигнала (F: 2.2)*, *Интерпретация команд (F: 2.3)*, *Выдача команд (F: 2.4)*.

Декомпозиция на основе нестабильности (volatility-based) подразумевает разбиение системы таким образом, чтобы каждый отдельный компонент инкапсулировал тот или иной потенциально меняющийся функционал и “принимал на себя удар” ввиду изменений данной нестабильной области [10]. Рассмотренная выше BCI-система содержит в себе разделенные функционально, но при этом все равно сильно связанные логически между собой компоненты – и обработка сигнала, и интерпретация команд, и пользовательский интерфейс сильно зависят от конкретной парадигмы взаимодействия с пользователем, и их совместная модификация при изменении парадигмы является хоть и сложным, но логичным решением. Проанализируем, может ли данная проблема быть решена с помощью декомпозиции на основе нестабильности – при этом, в качестве отправной точки возьмем все те же первоначальные требования, представленные в разделе Анализ требований. Результат декомпозиции представлен на рисунке 3. Диаграмма была составлена на основе общих правил применения используемого принципа декомпозиции, а также с использованием обобщенного набора программных компонентов BCI-приложения и их компоновки [5].

Аналогично функциональной декомпозиции, необходимо рассмотреть данное решение на наличие изменений в компонентах при расширении системы. Составим таблицу, аналогичную таблице 2, и рассмотрим в ней такие же новые возможности системы. Результат анализа представлен в таблице 3.

В данном случае мы видим минимальные изменения системы в рамках каждого из приведенных изменений. Например, в системе уже предусмотрен *контроллер устройств (V: 4.1)*, который

Таблица 3. Изменение программных компонентов системы, спроектированной с учетом нестабильностей

Новая возможность	Изменяемые текущие компоненты	Добавляемые компоненты
Возможность работы с несколькими разными устройствами	Логика передачи команд на новые устройства (*V: 4.1*)	Модуль взаимодействия с добавляемым устройством (*+V: 5.1.[1...N]*)
Расширение возможных парадигм взаимодействия		Новая схема взаимодействия (*+V: 2.1.[1...N]*), пользовательский интерфейс управления (*+V: 1.1.[1...N]*), конвейер обработки сигнала (*+V: 3.1.[1...N]*).

Рис. 3. Декомпозиция системы на основе нестабильности.

отвечает за конфигурацию всех управляемых устройств. Наличие контроллера учитывает нестабильность количества и типа устройств. Изменчивость неустойчивых компонентов функциональной декомпозиции – обработка сигнала, интерпретация и выдача команд связана с нестабильностью парадигмы взаимодействия, что тоже учтено в рамках данного разбиения. В итоге система при изменении требований расширяется, а не переделывается, и, таким образом, предлагаемый способ разбиения делает систему более гибкой и готовой к сопровождению в ходе эксплуатации и расширению новыми возможностями.

ОБСУЖДЕНИЕ

Рассмотренные методы декомпозиции в обоих случаях решают задачу проектирования архитектуры системы по обозначенным требованиям. Оба способа позволяют разрабатывать компоненты отдельно и независимо с их последующей интеграцией. Таким образом, обе представленные диаграммы в целом корректны и могут служить отправной точкой в дальнейшей разработке той или иной системы.

Тем не менее, в рамках декомпозиции рассмотренной системы управления роботизированными устройствами и анализа ее возможных изменений было выявлено, что способ декомпозиции на основе нестабильности проявляет большую устойчивость к внесению изменений, чем декомпозиция на основе функциональности ввиду того, что при функциональной декомпозиции гораздо большее количество программных компонентов требует доработки, а значит, влечет за собой дополнительные временные и экономические затраты. Нами не были подробно рассмотрены системы другого типа – например, предназначенные для проведения экспериментов, мониторинга и т.д., однако данный вывод будет справедлив и для этих систем.

Специфичность BCI-систем выражается в применении специализированных пользовательских интерфейсов для взаимодействия с элементами предметной области, что в свою очередь подразумевает введение в систему дополнительных компонентов формирования пользовательских воздействий с помощью BCI (и других человеко-машинных интерфейсов, например, на основе электромиограммы) на различных уровнях: от логики формирования команд из исходного сырого сигнала и его математической обработки до непосредственного подключения и сбора сигнала с устройства-интерфейса. На основе составленных схем на рисунках 2 и 3 можно выявить наиболее критичные и хрупкие компоненты – это компоненты, отвечающие за *обработку сигнала* и *логику интерпретации/выдачи команд*. Например, компонент *обработки сигнала* зависит от множества дополнительных элементов системы, таких как выбранная парадигма и количество команд. Ввиду этого нецелесообразно выделять данный компонент из системы по функциональному признаку, т.к. будет существовать множе-

ство причин для его изменения, что противоречит принципу единой ответственности. Вместо этого рациональным решением будет сделать данный функционал настраиваемым под конкретные требования контекста, реализовав для каждого варианта использования свой конвейер обработки сигнала с использованием некоторых общих блоков математических методов, изолировав подобным образом изменчивость способа обработки сигнала в отдельном компоненте системы.

Таким образом, оба способа декомпозиции позволяют учесть аспекты BCI-систем в ходе проектирования, однако функциональная декомпозиция не отражает их изменчивость, и спроектированная подобным образом система в будущем может неконтролируемо модифицироваться и разрастаться в ходе сопровождения. Итеративный процесс пересмотра ее архитектуры внесет в ее состав необходимые компоненты, однако система, изначально не учитывающая подобные модификации, будет значительно (и при этом неоправданно) усложняться. Декомпозиция на основе нестабильности изначально предполагает продумывание архитектуры на основе анализа требований всех возможных вариантов использования системы в текущем обозначенном контексте и в перспективе, что делает программную архитектуру такой системы более устойчивой к изменениям.

ВЫВОДЫ

В данной статье была рассмотрена одна из методологий, обеспечивающих выполнение задачи реализации системы с BCI-взаимодействием – декомпозиция системы на программные компоненты – что позволяет реализовать более гибкую и устойчивую архитектуру системы. Распространенным способом декомпозиции во многих проектах является функциональная декомпозиция, которая естественным образом сопоставляет требования к системе с ее компонентами. Несмотря на легкость использования, данный способ накладывает ограничения в случае расширения системы новыми возможностями. В противоположность этому, декомпозиция на основе нестабильности предполагает более сложный и тщательный анализ предметной области и требований к системе, однако в перспективе данный способ декомпозиции позволит осуществлять доработку системы быстрее и проще, что будет являться безусловной ценностью данной системы для адаптации и расширения ее функционала под нужды пользователей.

Повышение доступности BCI-технологий является приоритетной и актуальной задачей в настоящее время, и грамотное проектирование архитектуры ПО является лишь одним, но немаловажным ее аспектом. В данной статье предложено применение одного из приемов декомпозиции применительно к специфике BCI-технологий и показана его целесообразность. В будущих работах мы аналогичным образом планируем рассмотреть и другие различные лучшие практики разработки программных систем и их соотношение с разработкой систем с пользовательским взаимодействием на основе BCI с целью выявить еще больше потенциальных возможностей повышения гибкости и качества эксплуатации данных систем широким кругом пользователей.

СПИСОК ЛИТЕРАТУРЫ

1. *Martinez-Ledezma J.A., Barron-Zambrano J.H., Diaz-Manriquez A., Elizondo-Leal J.C., Saldivar-Alonso V.P., Rostro-Gonzalez H.* Versatile implementation of a hardware–software architecture for development and testing of brain–computer interfaces // International Journal of Advanced Robotic Systems, 2020. V. 17. № 6. https://doi.org/10.1177/1729881420980256
2. *Venthur B., Blankertz B.* A Free And Open Source BCI System In Python. Conference Abstract: XII International Conference on Cognitive Neuroscience (ICON-XII), 2015. https://doi.org/10.3389/conf.fnhum.2015.217.00406
3. *Venthur B., Blankertz B.* Mushu, a free- and open source BCI signal acquisition, written in Python. Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference, 2012. P. 1786–1788. https://doi.org/10.1109/EMBC.2012.6346296
4. *Renard Y., Lotte F., Gibert G., Congedo M., Maby E., Delannoy V., Bertrand O., Lécuyer A.* OpenViBE: An Open-Source Software Platform to Design, Test and Use Brain-Computer Interfaces in Real and Virtual Environments // Presence : teleoperators and virtual environments, 2010. V. 19. № 1.
5. *Kothe C.A., Makeig S.* BCILAB: a platform for brain-computer interface development // Journal of neural engineering, 2013. V. 10. № 5. P. 056014. https://doi.org/10.1088/1741-2560/10/5/056014
6. *Šťastný J., Doležal J., Černý V., Kubový J.* Design of a modular brain-computer interface. 2010 International Conference on Applied Electronics, 2010. P. 1–4.
7. *Zhang X., Yao L., Zhang S., Kanhere S., Sheng Q., Liu Y.* Internet of Things Meets Brain-Computer Interface: A Unified Deep Learning Framework for Enabling Human-Thing Cognitive Interactivity. IEEE Internet of Things Journal, 2018. P. 1–1. https://doi.org/10.1109/JIOT.2018.2877786
8. *Hosseini M.-P., Soltanian-Zadeh H., Elisevich K., Pompili D.* Cloud-based deep learning of big EEG data for epileptic seizure prediction. 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2016. P. 1151–1155. https://doi.org/10.1109/GlobalSIP.2016.7906022.
9. *Gonzalez-Sanchez J., Chavez-Echeagaray M.E., Atkinson R., Burleson W.* ABE: An Agent-Based Software Architecture for a Multimodal Emotion Recognition

Framework. 2011 Ninth Working IEEE/IFIP Conference on Software Architecture, 2011. P. 187–193. https://doi.org/10.1109/WICSA.2011.32.

10. *Левис Дж.* Совершенный софт. СПб.: Питер, 2021. 480 с.: ил. (Серия "Для профессионалов"). ISBN 978-5-4461-1621-8
11. *Banker R., Slaughter S.* The Moderating Effects of Structure on Volatility and Complexity in Software Enhancement. Information Systems Research, 2000. № 11. P. 219–240. https://doi.org/10.1287/isre.11.3.219.12209
12. ISO/IEC/IEEE 12207: 2017. Systems and software engineering – Software life cycle processes https://www.iso.org/ru/standard/63712.html
13. *Alonso-Valerdi L.M., Mercado-García V.R.* Towards designing Brain-Computer Interfaces in terms of User-Profiles, Neurophysiological Factors and User Experience // Mexican Journal of Biomedical Engineering, 2019. V. 40. № 2. P. 1–12. https://doi.org/10.17488/RMIB.40.2.3
14. *Dhindsa K., Carcone D., Becker S.* Toward an Open-Ended BCI: A User-Centered Coadaptive Design // Neural Comput, 2017. V. 29. № 10. P. 2742–2768. https://doi.org/10.1162/neco_a_01001
15. *Petrova A., Voznenko T., Dyumin A., Chepin E., Cherepanova A.* The Influence of Using Different Mental Images as BCI Commands on the Quality of Control. Procedia Computer Science, 2020. V. 169. P. 235–239. https://doi.org/10.1016/j.procs.2020.02.141
16. *Cherepanova A., Petrova A., Voznenko T., Dyumin A., Gridnev A., Chepin* E. The Research of Distracting Factors Influence on Quality of Brain-Computer Interface Usage: Proceedings of the Ninth Annual Meeting of the BICA Society, 2019. https://doi.org/10.1007/978-3-319-99316-4_6.
17. *Kosmyna N., Lécuyer, A.* Designing Guiding Systems for Brain-Computer Interfaces // Frontiers in human neuroscience, 2017. V. 11. P. 396. https://doi.org/10.3389/fnhum.2017.00396
18. *Corralejo R., Hornero R., Alvarez D.* A Domotic Control System Using Brain-Computer Interface (BCI). In: *Cabestany J., Rojas I., Joya G.* (eds) Advances in Computational Intelligence. IWANN 2011. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2011. V. 6691. P. 345–352. https://doi.org/10.1007/978-3-642-21501-8_43
19. *Vega-Barbas M., Pau I., Martín-Ruiz M.L., Seoane F.* Adaptive Software Architecture Based on Confident HCI for the Deployment of Sensitive Services in Smart Homes // Sensors, 2015. V. 15. № 4. P. 7294–7322. https://doi.org/10.3390/s150407294
20. *Niknamian S.* The Introduction of Designing a Hybrid Brain Computer Interface System // Biomedical Journal of Scientific & Technical Research, 2019. V. 16. https://doi.org/10.26717/BJSTR.2019.16.002828
21. *Petrova A., Chepin E., Voznenko T.* Using Environmental Objects as Visual Stimuli in BCI-based Interaction System: Theoretical Approach // Procedia Computer Science. Elsevier B.V., 2021. V. 190. P. 670–677. https://doi.org/10.1016/j.procs.2021.06.109

Vestnik Nacional'nogo Issledovatel'skogo Yadernogo Universiteta "MIFI", 2021, vol. 10, no. 6, pp. 540–549

Decomposition Principles in Software Architecture Design for Systems with the BCI-Based Interaction

T. I. Voznenko[a,#], A. I. Petrova[a], and E. V. Chepin[a]

[a] *Institute of Cyber Intelligence Systems, National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, 115409 Russia*

[#] *e-mail: TIVoznenko@mephi.ru*

Received February 3, 2022; revised February 4, 2022; accepted February 8, 2022

Abstract—Brain-computer interfaces (BCIs) are a modern and promising interaction technology that significantly expands human capabilities. An actual objective is the fast and flexible development of adaptive BCI applications for various tasks and for a wide range of users, especially for people with disabilities. The decomposition of the system software architecture into components, which is one of the aspects of the development process, has been considered including the features of the BCI-based applications. The approach of functional decomposition, which is popular in the field of software development, including BCI interaction, has been analyzed. The application of volatility-based decomposition principle has also been proposed. These two principles have been compared on the example of designing a system that implements the control of a robotic device using BCI, for which common initial requirements have been determined and two functionality-based and volatility-based options for its software decomposition have been considered. Finally, the necessity to perform any modifications of the components due to various upgrades of the system has been analyzed. It has been shown that both decomposition approaches are applicable to solve the problem of designing systems with BCI interaction, but volatility-based decomposition demonstrates a higher resistance of the application architecture to changes due to the forthcoming of new functional requirements. Therefore, this decomposi-

tion method simplifies and optimizes the adaptation of the BCI application to changing operational conditions, which is beneficial from an economic point of view and positively affects the user experience.

Keywords: brain-computer interface, functional decomposition, volatility-based decomposition, software architecture

DOI: 10.1134/S2304487X21060122

REFERENCES

1. Martinez-Ledezma J.A., Barron-Zambrano J.H., Diaz-Manriquez A., Elizondo-Leal J.C., Saldivar-Alonso V.P., Rostro-Gonzalez H. Versatile implementation of a hardware–software architecture for development and testing of brain–computer interfaces. *International Journal of Advanced Robotic Systems*, 2020, vol. 17, no. 6. doi: 10.1177/1729881420980256
2. Venthur B., Blankertz B. A Free And Open Source BCI System In Python. *Conference Abstract: XII International Conference on Cognitive Neuroscience (ICON-XII)*, 2015. doi: 10.3389/conf.fnhum.2015.217.00406
3. Venthur B., Blankertz B. Mushu, a free- and open source BCI signal acquisition, written in Python. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference,* 2012, pp. 1786–1788. doi: 10.1109/EMBC.2012.6346296
4. Renard Y., Lotte F., Gibert G., Congedo M., Maby E., Delannoy V., Bertrand O., Lécuyer A. OpenViBE: An Open-Source Software Platform to Design, Test and Use Brain-Computer Interfaces in Real and Virtual Environments. *Presence: teleoperators and virtual environments*, 2010, vol. 19, no. 1.
5. Kothe C. A., Makeig S. BCILAB: a platform for brain-computer interface development. *Journal of neural engineering*, 2013, vol. 10, no. 5, p. 056014. doi: 10.1088/1741-2560/10/5/056014
6. Šťastný J., Doležal J., Černý V., Kubový J. Design of a modular brain-computer interface. *2010 International Conference on Applied Electronics*, 2010, pp. 1–4.
7. Zhang X., Yao L., Zhang S., Kanhere S., Sheng Q., Liu Y. Internet of Things Meets Brain-Computer Interface: A Unified Deep Learning Framework for Enabling Human-Thing Cognitive Interactivity. *IEEE Internet of Things Journal*, 2018, pp. 1–1. doi: 10.1109/JIOT.2018.2877786.
8. Hosseini M.-P., Soltanian-Zadeh H., Elisevich K., Pompili D. Cloud-based deep learning of big EEG data for epileptic seizure prediction. *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2016, pp. 1151–1155. doi: 10.1109/GlobalSIP.2016.7906022.
9. Gonzalez-Sanchez J., Chavez-Echeagaray M.E., Atkinson R., Burleson W. ABE: An Agent-Based Software Architecture for a Multimodal Emotion Recognition Framework. *2011 Ninth Working IEEE/IFIP Conference on Software Architecture*, 2011, pp. 187–193. doi: 10.1109/WICSA.2011.32.
10. Löwy, J. Sovershenyy soft [Righting software]. SPb.: Piter, 2021, 480 p.: il. (Seriya "Dlya professionalov"). ISBN 978-5-4461-1621-8 (in Russian)
11. Banker R., Slaughter S. The Moderating Effects of Structure on Volatility and Complexity in Software Enhancement. *Information Systems Research*, 2000, vol. 11, pp. 219–240. doi: 10.1287/isre.11.3.219.12209.
12. ISO/IEC/IEEE 12207:2017. Systems and software engineering – Software life cycle processes. Available at: https://www.iso.org/ru/standard/63712.html. (accessed 31.10.2021)
13. Alonso-Valerdi L.M., Mercado-García V.R. Towards designing Brain-Computer Interfaces in terms of User-Profiles, Neurophysiological Factors and User Experience. Mexican Journal of Biomedical Engineering, 2019, vol. 40, no. 2, pp. 1–12. doi: 10.17488/RMIB.40.2.3
14. Dhindsa K., Carcone D., Becker S. Toward an Open-Ended BCI: A User-Centered Coadaptive Design. *Neural Comput*, 2017, vol. 29, no. 10. pp. 2742–2768. doi: 10.1162/neco_a_01001
15. Petrova A., Voznenko T., Dyumin A., Chepin E., Cherepanova A. The Influence of Using Different Mental Images as BCI Commands on the Quality of Control. *Procedia Computer Science*, 2020, vol. 169, pp. 235–239. doi: 10.1016/j.procs.2020.02.141.
16. Cherepanova A., Petrova A., Voznenko T., Dyumin A., Gridnev A., Chepin E. The Research of Distracting Factors Influence on Quality of Brain-Computer Interface Usage. *Proceedings of the Ninth Annual Meeting of the BICA Society*, 2019. doi: 10.1007/978-3-319-99316-4_6.
17. Kosmyna N., Lécuyer, A. Designing Guiding Systems for Brain-Computer Interfaces. *Frontiers in human neuroscience*, 2017, vol. 11, p. 396. doi: 10.3389/fnhum.2017.00396
18. Corralejo R., Hornero R., Alvarez D. A Domotic Control System Using Brain-Computer Interface (BCI). *In: Cabestany J., Rojas I., Joya G. (eds) Advances in Computational Intelligence. IWANN 2011. Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2011, vol. 6691, pp. 345–352. doi: 10.1007/978-3-642-21501-8_43.
19. Vega-Barbas M., Pau I., Martín-Ruiz M.L., Seoane F. Adaptive Software Architecture Based on Confident HCI for the Deployment of Sensitive Services in Smart Homes. *Sensors*, 2015, vol. 15, no. 4, pp. 7294–7322. doi:10.3390/s150407294
20. Niknamian S. The Introduction of Designing a Hybrid Brain Computer Interface System. *Biomedical Journal of Scientific & Technical Research*, 2019, vol. 16. doi: 10.26717/BJSTR.2019.16.002828.
21. Petrova A., Chepin E., Voznenko T. Using Environmental Objects as Visual Stimuli in BCI-based Interaction System: Theoretical Approach. *Procedia Computer Science*. Elsevier B.V., 2021, vol. 190, pp. 670–677. doi: 10.1016/j.procs.2021.06.109.