DOI: 10.18721/JCSTCS.10406
УДК 004.81

ОПИСАНИЕ ПРОЦЕССОВ РАЗРАБОТКИ ПРОГРАММ С ПОМОЩЬЮ СКРЫТЫХ МАРКОВСКИХ МОДЕЛЕЙ

Д.А. Тимофеев, А.В. Самочадин

Санкт-Петербургский политехнический университет Петра Великого,
Санкт-Петербург, Российская Федерация

Рассмотрены процессы работы программиста над индивидуальными задачами в рамках программного проекта и предложен способ моделирования таких процессов на основе скрытых марковских моделей. Модель процесса может использоваться для решения трех задач: анализа и сравнения процессов, повышения эффективности работы и снижения утомляемости работника, обучения студентов технологическим процессам опытных разработчиков. Две последние задачи предложено решать с помощью введения обратной связи в виде подсказок, формируемых на основе анализа текущего вида деятельности программиста и предыдущих состояний процесса разработки. Идентификация вида деятельности программиста осуществляется путем восстановления наиболее правдоподобной последовательности состояний модели на основе анализа наблюдаемых действий программиста в среде разработки и других приложениях.

Ключевые слова: разработка программного обеспечения; моделирование процессов; скрытые марковские модели; обратная связь; производительность труда.

Ссылка при цитировании: Тимофеев Д.А., Самочадин А.В. Описание процессов разработки программ с помощью скрытых марковских моделей // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. 2017. Т. 10. № 4. С. 70–77. DOI: 10.18721/JCSTCS.10406

MODELING OF SOFTWARE DEVELOPMENT PROCESSES WITH HIDDEN MARKOV MODELS

D.A. Timofeev, A.V. Samochadin

Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russian Federation

This paper describes the individual process of programming as a specific part of a more general software development process. We discuss the task of programming process modeling and propose a new approach based on hidden Markov models. The model may be used to solve three kinds of problems: the analysis and comparison of processes, making the process more efficient and less exhausting and transferring the process knowledge from experienced developers to students. The latter two problems are solved by detecting the kind of programmer's activity. We do it by inferring the most likely sequence of process model states based on observed actions the developer performs using the integrated development environment and other relevant applications.

Keywords: software development; process modeling; hidden Markov model; feedback; productivity.

Citation: Timofeev D.A., Samochadin A.V. Modeling of software development processes with hidden Markov models. St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunications and Control Systems, 2017, Vol. 10, No. 4, Pp. 70–77. DOI: 10.18721/JCSTCS.10406

Введение

Процесс разработки программного обеспечения включает в себя планирование работ и выполнение их конкретными исполнителями. Вопрос планирования работ изучен достаточно хорошо. Все распространенные методологии задают критерии разбиения задач на подзадачи, способы оценки их трудоемкости и порядок контроля выполнения работ. В это же время вопрос о том, как именно разработчик работает над конкретной задачей, исследован значительно меньше.

Одной из причин этого является природа самого процесса. Работа программиста рассматривается (и прежде всего самими разработчиками) как творческий акт.

В качестве иллюстрации можно привести заключение тьюринговской лекции Д. Кнута: «Итак, мы видели, что программирование – это искусство, поскольку оно является приложением накопленных знаний для практических целей, поскольку оно требует умения и мастерства, и в особенности потому, что продукты программирования могут представлять эстетическую ценность. Программист, который бессознательно ощущает себя художником, получает удовольствие от своей работы и справляется с ней лучше» [1].

Одновременно процесс программирования предполагает использование определенной технологии[1]. Она определяет правила декомпозиции задачи (структурное, объектно-ориентированное, компонентное и другие виды программирования), использование стандартных приемов организации программы (например, шаблонов проектирования [2]), применение тех или иных инструментов (языков программирования, сред разработки, систем контроля версий). Принятая в рамках проекта технология в существенной степени определяет способ организации программы, подход к ее проектированию и реализации. Тем не менее даже в рамках одной технологии производительность труда программистов может существенно отличаться друг от друга из-за использования ими различных процессов работы[2].

Данная статья посвящена одному подходу к моделированию процесса работы программиста на уровне отдельных задач. Целью создания такой модели является решение следующих задач:

- экспериментальное исследование и сравнение процессов, характерных для программистов с различным опытом;
- выработка рекомендаций по организации процесса работы и формирование обратной связи в процессе работы, позволяющих увеличить производительность труда и снизить утомляемость работника;
- обучение начинающих программистов приемам и процессам работы опытных программистов путем формирования подсказок в ходе выполнения заданий.

Моделирование процесса программирования

Процесс программирования относительно редко рассматривается с точки зрения технологии. В работе [3] исследован вопрос различий между способами решения задачи, характерными для опытных и для начинающих программистов. Для выявления этих различий использовался метод, при котором участник эксперимента рассуждает вслух и комментирует свои действия. Так как предметом исследования в данной работе являлся сам способ рассуждения о задаче, формальное описание процесса работы над задачей не выполнялось. Аналогичный подход использован в работе [4], где видеозапись процесса разработки в сочетании с проговариванием и объяснением всех выполняемых действий применялась для обучения начинающих студентов-программистов.

Построение формальной модели про-

[1] **Гагарина Л.Г., Кокорева Е.В., Виснадул Б.Д.** Технология разработки программного обеспечения: учебное пособие. М.: ИД «ФОРУМ»; ИНФРА-М, 2008. 400 с.

[2] **Kamma D.** Study of Task Processes for Improving Programmer Productivity. PhD thesis. Indraprastha Institute of Information Technology, Delhi, 2007 // URL: https://repository.iiitd.edu.in/jspui/bitstream/handle/123456789/513/PhD1004.pdf.

цесса разработки было предпринято в диссертации[2]. В ходе исследования шесть разработчиков были разделены на две группы на основании своего опыта. Каждый из них решал типовую задачу разработки модульных тестов, комментируя выполняемые действия, а процесс разработки записывался на видео. Модели процессов строились на основе полученных видеозаписей и комментариев с использованием аппарата марковских цепей. Полученные модели затем использовались для определения степени сходства между процессами различных разработчиков и анализа их эффективности.

Существенно большее количество работ посвящено моделированию других выполняемых людьми процессов. Основным формальным аппаратом при этом являются скрытые марковские модели (СММ) [5] и их модификации. Одно из направлений исследований связано с задачами анализа, индексирования и поиска видеозаписей. Например, в [6] анализируется структура теннисного матча. Трансляция разбивается на базовые фрагменты (ошибка при первой подаче, розыгрыш, повтор момента и перерыв в трансляции), из которых затем составляется иерархическая структура матча (сеты, геймы, перерывы) и трансляции. На основе вручную размеченных записей строится СММ, которая затем служит для классификации фрагментов новых трансляций. Аналогичный подход использовался для анализа структуры трансляции матчей по футболу и бейсболу [7, 8].

Другой важной областью применения формальных моделей поведения человека является здравоохранение. В ряде работ анализ повседневной деятельности людей, особенно пожилых, используется для принятия решений об оказании помощи [9] или для ранней диагностики деменции и других состояний, приводящих к изменению поведения пациента [10]. В работе [9] модель строится на основании размеченных и неразмеченных данных, получаемых с датчиков. Для классификации неразмеченных элементов пространства признаков использовалась их близость к известным точкам этого пространства с учетом локальной структуры многообразия. Результат классификации наблюдений используется для построения СММ. Авторы работы [10] применили аппарат двухуровневых иерархических СММ [11], в которых каждому состоянию соответствует своя СММ, а в качестве наблюдаемых величин использовали признаки, извлекаемые из видеоизображения. Если целью работы является не анализ процесса, а только идентификация отдельных состояний, то могут применяться и более простые методы на основе стандартных алгоритмов классификации [12, 13].

В данной статье рассматриваются новые возможности использования формальной модели процесса программирования, основанные на сопоставлении модели с фактическим процессом работы над программой и формировании на этой основе обратной связи. Анализ текущего состояния и истории процесса разработки позволит идентифицировать состояния, в которых процесс целесообразно прервать или изменить, либо, напротив, следует максимально ограничить воздействие отвлекающих факторов для поддержания максимальной эффективности и удовлетворения от работы (состояние «потока» [14]).

Процесс работы программиста над задачей включает несколько видов деятельности, среди которых можно выделить следующие:

- обсуждение задач, требований и ограничений, путей решения;
- получение или предоставление информации о работе компонентов системы, справочной информации;
- изучение документации (знакомство с предметной областью, изучение документации по используемым инструментам, по внешним компонентам, поиск справочных материалов);
- изучение программного кода (чтение и анализ кода внешних компонентов, чтение и анализ существующего кода, инспекция кода других разработчиков);
- планирование и проектирование (планирование работы, архитектурное проектирование, разработка структур данных и алгоритмов, разработка тестов, планирование рефакторинга);

• кодирование (написание кода, тестов, редактирование ранее написанного кода);

• отладка (проверка работоспособности написанного или отредактированного кода, поиск и диагностика ошибок или проблем производительности);

• тестирование (модульное, интеграционное, нагрузочное);

• сборка и развертывание (подготовка установочных образов, развертывание системы на сервере, настройка взаимодействия компонентов распределенной системы);

• документирование (комментирование кода, разработка внутренней и внешней документации, разработка учебных и демонстрационных материалов);

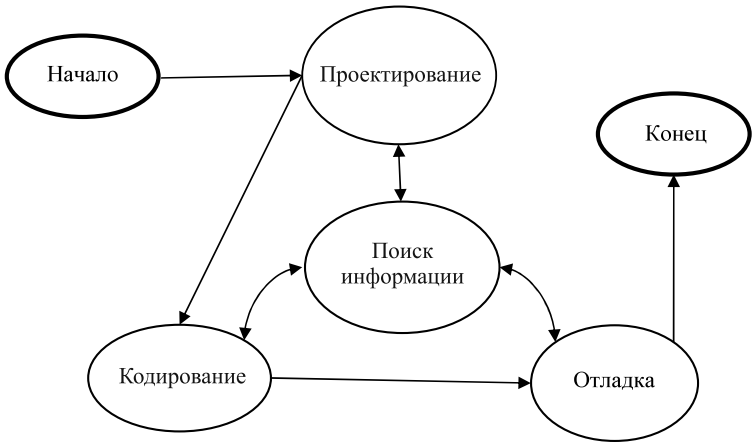
• обеспечение процесса разработки (настройка сред разработки, редакторов, сред выполнения программы, управление версиями кода и данных, обновление и анализ информации в системе управления задачами).

Перечень задач определяется ролью конкретного разработчика в проекте. В крупной компании задачи тестирования, сборки и развертывания, документирования, а также в определенной степени настройки среды разработки могут быть делегированы выделенным сотрудникам, в то время как в малых коллективах к набору функций разработчика могут быть также добавлены такие задачи, как анализ требований или поддержка и обучение пользователей.

Перечисленные виды деятельности с точки зрения управления процессом могут как входить в одну задачу, так и составлять отдельные задачи. Обычно вместе объединяются задачи планирования и проектирования (особенно на уровне структур данных и алгоритмов), кодирования и отладки, к которым по мере необходимости добавляется поиск справочной информации. Такой процесс в идеализированном виде представлен на рисунке.

На практике такой почти линейный процесс реализуется не всегда. Одной из причин нарушения линейности является обнаружение разработчиком ошибок, допущенных на одном из предшествующих этапов. В этом случае ошибка исправляется, затрагивая все последующие этапы. Почти всегда происходит возврат из состояния «отладка» к состоянию «кодирование», но возможен и возврат к этапу «проектирование». Вторая причина, по которой процесс может выполняться нелинейно – декомпозиция задачи (в частности, программирование «сверху вниз», при котором вместо отсутствующих функций используются «заглушки», и «снизу вверх», когда сначала разрабатываются и отлаживаются отдельные базовые функции, объединяемые затем в более сложные алгоритмы). В этом случае процесс становится циклическим, а в каждом состоянии работа ведется только над частью задачи.

Исходя из этих наблюдений, можно,



Базовый процесс работы над задачей

The basic process of work on the task

следуя описанному в диссертации[2] подходу, моделировать процесс работы над задачей с помощью марковских цепей. Марковская цепь задается множеством состояний $S = \{s_1, s_2, ..., s_n\}$ и матрицей переходов A, каждый элемент которой a_{ij} соответствует вероятности перехода цепи из состояния s_i в состояние s_j. Вероятность перехода в каждое из состояний зависит только от текущего состояния вне зависимости от того, какими были предшествующие состояния.

В реальном процессе разработки переходы между состояниями не случайны, но они зависят от большого количества факторов, в том числе недоступных или только частично доступных для наблюдения (от опыта и знаний разработчика, его психофизиологического состояния, плана работы, особенностей решаемой задачи, присутствия других людей и т. д.), что позволяет считать использование статистического моделирования допустимым упрощением.

Чтобы построить модель конкретного процесса, необходимо зафиксировать множество состояний S и найти матрицу A. Для поиска матрицы переходов необходимо идентифицировать последовательность состояний процесса. При наличии явного комментария, включающего переходы между состояниями, для построения элемента a_{ij} матрицы A достаточно вычислить количество переходов из состояния s_i в состояние s_j к общему количеству переходов из состояния s_i:

$$a_{ij} = \frac{\left|s_i \to s_j\right|}{\left|s_i \to s_k, 1 \le k \le n\right|}. \qquad (1)$$

Моделирование процесса программирования с помощью марковских цепей в принципе позволяет решить первую из трех сформулированных во введении задач: анализ и сравнение процессов, характерных для программистов с различным опытом работы. Тем не менее для решения двух других задач этой модели недостаточно. Формирование обратной связи в процессе работы требует идентификации текущего состояния. Эта задача тривиальна при наличии эталонной разметки – комментария, описывающего все изменения состояния. Однако в повседневной деятельности подобные комментарии оказываются недоступными по двум причинам. Во-первых, комментирование всех действий неудобно для пользователя. Во-вторых, автоматический анализ комментариев в свободной форме на естественном языке представляет собой трудную задачу, которая в настоящее время не может быть решена с достаточной точностью без существенного ограничения словаря. Таким образом, идентификация текущего состояния процесса должна выполняться неявно на основе косвенных признаков.

Моделирование процесса с помощью скрытых марковских моделей

Скрытые марковские модели [5, 15] представляют собой модификацию марковских цепей. Скрытая марковская модель также имеет n состояний $S = \{s_1, s_2, ..., s_n\}$, переходы между которыми определяются матрицей вероятностей перехода A, но теперь наблюдатель не может непосредственно определить, в каком состоянии находится модель. Вместо этого модель определяет m наблюдаемых величин $V = \{v_1, v_2, ..., v_m\}$. На каждом шаге t модель находится в состоянии $q_t \in S$ и формирует случайное значение наблюдаемой величины $o_t \in V$. Распределение наблюдаемой величины в состоянии s_i задается вектором b_i: вероятность в состоянии s_i наблюдать величину v_k равна $b_i(k)$. Состояние s_i выбирается в качестве начального состояния q_0 с вероятностью $\pi(i)$. Так как процесс работы над задачей конечен, среди состояний модели выбирается одно состояние q_F, являющееся заключительным.

При моделировании процесса работы программиста скрытые состояния S соответствуют состояниям процесса разработки (видам деятельности программиста). Наблюдаемые величины соответствуют действиям, которые разработчик выполняет в каждом из состояний.

В разрабатываемой в настоящее время версии системы сбора данных и анализа процессов разработки используются следующие виды наблюдаемых величин:

• низкоуровневые события, связанные с взаимодействием пользователя с операционной системой: нажатия клавиш, перемещение и нажатие кнопок мыши, переключение между окнами приложений, запуск и завершение процессов;

• высокоуровневые события, связанные с взаимодействием пользователя с прикладными программами: действия в среде разработки (начало и завершение сеанса работы, открытие и закрытие проектов и файлов, запуск и завершение отладки) и браузере (открытие и закрытие вкладок, переходы по ссылкам).

Дополнительно предусмотрено получение физиологических характеристик состояния разработчика. В настоящее время поддерживается измерение частоты сердечных сокращений с помощью фитнес-браслета MioLink. Тем не менее эти данные пока не используются для построения моделей.

Анализ событий, возникающих в ходе программирования, производится на компьютере с установленными приложениями для перехвата событий, включая модули расширения для среды разработки Microsoft Visual Studio 2017 и браузера Google Chrome. Данные о каждом событии сохраняются в центральной базе данных с указанием типа события, его параметров и отметки времени. Использование центральной базы данных позволяет формировать и анализировать набор данных о действиях многих пользователей. Такой способ работы с данными ориентирован в первую очередь на использование в лабораторном окружении. Одновременно с этим в системе предусмотрена возможность ограничения передачи данных за пределы локального компьютера, что позволяет пользователям проводить локальный анализ данных о своей повседневной деятельности без риска утечки приватной информации. Управление сбором и передачей данных о каждом из видов событий осуществляет сам пользователь.

Построение скрытой марковской модели может быть выполнено двумя методами. Первый способ заключается в итеративном восстановлении матрицы вероятностей переходов A, набора распределений b_i (который можно задать одной матрицей B) и распределения π путем последовательных приближений на основе анализа последовательностей наблюдаемых величин. Для скрытых марковских моделей эта задача может быть решена с помощью алгоритма Баума-Велша [15]. Достоинством данного подхода является отсутствие необходимости знать истинную последовательность скрытых состояний процесса. Вместо этого алгоритм восстанавливает параметры модели в соответствии с принципом максимального правдоподобия. Использование данного метода целесообразно для выявления закономерностей в действиях многих разработчиков, когда идентификация «истинных» состояний (например, в ходе явного проговаривания переходов между видами деятельности) невозможна или слишком дорога. В этом случае достаточно организовать сбор событий о наблюдаемых действиях пользователей, не требуя дополнительных аннотаций или участия экспериментатора.

Тем не менее данный подход требует большого количества наблюдений, а для быстрой сходимости нужно хорошее начальное приближение для матриц A и B. Предварительные эксперименты также показали, что для использования описанного подхода целесообразно проводить предварительную обработку данных о событиях. В частности, вместо использования элементарных событий типа «нажатие клавиши» следует анализировать последовательности нажатых клавиш и формировать события более высокого уровня, позволяющие идентифицировать вид вводимого текста. Такой анализ выполняется на основе языковых моделей для естественных языков и языков программирования. Использование событий более высокого уровня позволяет существенно снизить размерность задачи.

Второй подход возможен при наличии явного знания о последовательности состояний. Хотя в общем случае последовательность состояний скрытого марковского процесса недоступна для наблюдения, при построении модели процесса разработки эти данные могут быть получены для некоторого ограниченного количества экспериментов, в которых действия разработчика протоколируются. В таком случае исполь-

зуется следующий алгоритм оценки параметров модели.

1. С помощью формулы (1) на основе собранных данных вычисляются элементы матрицы A.

2. Для каждого состояния s_i, для каждого наблюдаемого действия v_i вычисляется $b_i(j) = \frac{|o = v_j, s = s_i|}{|s = s_i|}$ (отношение количества наблюдений действия v_i, выполняемого в состоянии s_i, к общему числу действий в этом состоянии).

Несмотря на то что при использовании этого подхода модель строится для относительно небольшого числа экспериментов, ее можно использовать либо непосредственно в качестве эталонной модели (для простых моделей), либо в качестве начального приближения для алгоритма Баума-Велша. При этом также целесообразно снижать размерность пространства наблюдаемых величин путем построения более высокоуровневых признаков вида «ввод текста программы на языке Java» или «ввод текста на русском языке».

Заключение

В данной статье рассмотрен подход к построению формальной модели процессов, выполняемых разработчиками программного обеспечения в ходе работы над отдельными задачами. Такая модель может применяться для анализа и повышения эффективности используемых процессов, более точной оценки производительности труда разработчика при выполнении определенного класса задач, а также для обучения студентов-программистов и начинающих разработчиков эффективным процессам работы над задачей.

Если в качестве наблюдаемых величин использовать также показатели психофизиологического состояния (частота сердечных сокращений, пульсовая волна, электрическая активность кожи и др.), измеряемые с помощью фитнес-браслетов или других доступных приборов, аналогичные модели можно также применять для оценки и прогнозирования психофизиологического состояния разработчика в течение рабочего дня или на больших интервалах времени.

В настоящее время реализована первая версия системы, осуществляющей сбор и хранение данных о выполняемых разработчиками действиях и о значениях физиологических показателей. С ее помощью ведется формирование корпуса данных о действиях разработчиков программного обеспечения.

Работа подготовлена в ходе реализации проекта в рамках Постановления Правительства РФ от 09.04.2010 г. № 218 при финансовой поддержке Министерства образования и науки РФ. Договор № 03.G25.31.0247 от 28.04.2017 г.

СПИСОК ЛИТЕРАТУРЫ

1. **Эшенхерст Р.** (ред). Лекции лауреатов премии Тьюринга. Пер. с англ. М.: Мир, 1993. 560 с.

2. **Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д.** Приемы объектно-ориентированного проектирования. Паттерны проектирования. Пер. с англ. СПб.: Питер, 2014. 366 с.

3. **Jeffries R., et al.** The processes involved in designing software // Cognitive Skills and their Acquisition. 1981. Pp. 255–283.

4. **Bennedsen J., Caspersen M.E.** Revealing the programming process // Proc. of SIGCSE'05. St. Louis, Missouri, 2005.

5. **Baum L.E., Petrie T.** Statistical inference for probabilistic functions of finite state Markov chains // The Annals of Mathematical Statistics. 1966. Vol. 37 (6). Pp. 1554–1563.

6. **Kijak E., et al.** Audiovisual integration for tennis broadcast structuring // Multimedia Tools and Applications. 2006. Vol. 30. No. 3. Pp. 289–311.

7. **Xie L., et al.** Structure analysis of soccer video with Hidden Markov Models // Proc. of the IEEE Internat. Conf. on Acoustics, Speech, and Signal Processing. IEEE, 2002. Vol. 4. Pp. IV-4096–IV-4099.

8. **Chang P., Han M., Gong Y.** Extract highlights from baseball game video with Hidden Markov Models // Proc. of the 2002 Internat. Conf. on. IEEE, 2002. Vol. 1. Pp. I-I.

9. **Ghazvininejad M., et al.** HMM based semi-supervised learning for activity recognition // Proc. of the 2011 Internat. Workshop on Situation Activity & Goal Awareness. ACM, 2011. Pp. 95–100.

10. **Karaman S., et al.** Hierarchical Hidden Markov Model in detecting activities of daily

living in wearable videos for studies of dementia // Multimedia Tools and Applications. 2014. Vol. 69. No. 3. Pp. 743–771.

11. **Fine S., Singer Y., Tishby N.** The hierarchical Hidden Markov Model: Analysis and applications // Machine Learning. 1998. Vol. 32. No. 1. Pp. 41–62.

12. **Stikic M., Schiele B.** Activity recognition from sparsely labeled data using multi-instance learning // Location and Context Awareness. 2009. Pp. 156–173.

13. **Zhou W., Vellaikal A., Kuo C.C.** Rule-based video classification system for basketball video indexing // Proc. of the 2000 ACM Workshops on Multimedia. ACM, 2000. Pp. 213–216.

14. **Чиксентмихайи М.** Поток: Психология оптимального переживания. Пер. с англ. 7-е изд. М.: Смысл; Альпина нон-фикшн, 2017.

15. **Rabiner L.R.** A tutorial on Hidden Markov Models and selected applications in speech recognition // Proc. of the IEEE. 1989. Vol. 77. Issue 2. Pp. 257–286.

Статья поступила в редакцию 11.11.2017.

REFERENCES

1. **Eshenherst R. (ed).** *ACM turing award lectures*. Moscow: Mir Publ., 1993, 560 p. (rus)

2. **Gamma E., Helm R., Johnson R., Vlissides D.** *Design patterns: elements of reusable object-oriented software*. St. Petersburg: Piter Publ., 2014, 366 p. (rus)

3. **Jeffries R., et al.** The processes involved in designing software. *Cognitive Skills and their Acquisition*, 1981, Pp. 255–283.

4. **Bennedsen J., Caspersen M.E.** Revealing the programming process. *Proceedings of SIGCSE'05*, St. Louis, Missouri, Febr. 23–27, 2005.

5. **Baum L.E., Petrie T.** Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 1966, Vol. 37 (6), Pp. 1554–1563.

6. **Kijak E., et al.** Audiovisual integration for tennis broadcast structuring. *Multimedia Tools and Applications*, 2006, Vol. 30, No. 3, Pp. 289–311.

7. **Xie L., et al.** Structure analysis of soccer video with Hidden Markov Models. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, IEEE, 2002, Vol. 4, Pp. IV-4096–IV-4099.

8. **Chang P., Han M., Gong Y.** Extract highlights from baseball game video with Hidden Markov Models. *Proceedings of the 2002 International Conference* on, IEEE, 2002, Vol. 1, Pp. I-I.

9. **Ghazvininejad M., et al.** HMM based semi-supervised learning for activity recognition. *Proceedings of the 2011 International Workshop on Situation Activity & Goal Awareness*, ACM, 2011, Pp. 95–100.

10. **Karaman S., et al.** Hierarchical Hidden Markov Model in detecting activities of daily living in wearable videos for studies of dementia. *Multimedia Tools and Applications*, 2014, Vol. 69, No. 3, Pp. 743–771.

11. **Fine S., Singer Y., Tishby N.** The hierarchical Hidden Markov Model: Analysis and applications. *Machine Learning*, 1998, Vol. 32, No. 1, Pp. 41–62.

12. **Stikic M., Schiele B.** Activity recognition from sparsely labeled data using multi-instance learning. *Location and Context Awareness*, 2009, Pp. 156–173.

13. **Zhou W., Vellaikal A., Kuo C.C.** Rule-based video classification system for basketball video indexing. *Proceedings of the 2000 ACM Workshops on Multimedia*, ACM, 2000, Pp. 213–216.

14. **Csikszentmihalyi M.** *Flow: The psychology of optimal experience*. Moscow: Smysl; Alpina non-fiction publ., 2017. (rus)

15. **Rabiner L.R.** A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 1989, Vol. 77, Issue 2, Pp. 257–286.

Received 11.11.2017.

СВЕДЕНИЯ ОБ АВТОРАХ / THE AUTHORS

ТИМОФЕЕВ Дмитрий Андреевич
TIMOFEEV Dmitrii A.
E-mail: dtim@dcn.icc.spbstu.ru

САМОЧАДИН Александр Викторович
SAMOCHADIN Alexander V.
E-mail: samochadin@gmail.com