Целенаправленное обучение на примере модели и классификатора инструментов автоматизации тестирования по

А.В.Сухоруков
M.sc.comp., докторант кафедры технологий разработки программного обеспечения,
Рижский технический университет,
ул. Межа, 1/3, г. Рига, Латвия, LV-1048, (371) 67089571
Aleksandrs.Suhorukovs@rtu.lv

Аннотация

В условиях необходимости решения конкретной задачи, специалист, не имеющий достаточных знаний в специфической предметной области, вынужден затрачивать много времени на ее изучение и освоение. Предлагается метод, комбинирующий построение модели решения и классификацию возможных решений, призванный решить проблему быстрого выявления узкой части предметной области, изучения которой достаточно для нахождения решения поставленной задачи. Такая целенаправленность дает возможность существенно сократить время, затрачиваемое на обучение. Метод излагается на примере построения модели и классификатора области знаний об инструментах автоматизации тестирования ПО.

In order to solve some specific problem a specialist who doesn't have enough knowledge in particular field has to spend much time to learn and master it. Proposed method addressing this issue combines development of solution model and classification of possible solutions. The method allows for narrowing the field of knowledge to small part which is sufficient to learn in order to solve the topical problem. Such goal-orientation makes it possible to significantly shorten time necessary for learning. The method is described by example in which solution model and classifier are getting built for knowledge field about software test automation tools.

Ключевые слова

целенаправленное обучение, модель решения, классификация, автоматизация тестирования ПО

goal-oriented learning, solution model, classification, software test automation

Введение

После появления новых областей знаний, как правило, проходит значительное время, прежде чем новый материал будет включен в образовательные программы подготовки специалистов. Причем чем специфичнее знания, то есть чем более узкому кругу специалистов они необходимы, тем больше это время. Тем самым затрудняется применение новых знаний на практике, и любой специалист должен непрерывно повышать квалификацию, чтобы не отставать от новых тенденций [2]. Однако во многих быстроразвивающихся сферах деятельности объем новых знаний растет очень быстро [1], и специалисты постоянно сталкиваются с задачами, для решения которых имеющихся у них знаний оказывается недостаточно. Даже для решения вполне конкретных небольших задач, специалист бывает вынужден проводить масштабное исследование с целью найти нужный ему инструмент или метод среди множества имеющихся.

С одной стороны такое исследование является вариантом самоорганизованного проблемно-ориентированного обучения [6] и расширяет кругозор специалиста, повышая его компетентность, с другой – существенно замедляет решение той конкретной задачи, которая перед ним поставлена, причем потраченное на исследование время может во много раз превышать время, необходимое для самого решения. Для таких случаев необходим метод, который бы позволил существенно сократить время поиска подходящего решения.

Предлагаемый в статье метод основывается на классификации возможных решений по параметрам решаемой задачи. Специалист, который знает эти параметры – условия поставленной задачи, и которому доступен соответствующий классификатор, собирающий воедино знания в нужной предметной области, может, существенно ограничить круг поиска. Тем самым он целенаправленно сужает интересующую его предметную область до той ее части, которую ему действительно необходимо изучить в данный момент времени.

Подобные классификаторы уже существуют и развиваются для многих новых предметных областей, например классификация шаблонов проектирования ПО по параметрам назначения и контекста применения [5]. Однако далеко не всегда бывает легко в условии задачи выявить необходимые значения параметров. Предлагаемый в статье метод решения этой проблемы состоит в построении абстрактной модели задачи, легко сопоставимой с любой задачей данной предметной области. Сопоставление поставленной задачи с моделью позволит проявить специфику конкретного случая в виде определенных значений критериев классификатора.

Целью данной работы является демонстрация предлагаемого метода на примере выбора подходящего инструмента автоматизации тестирования ПО в зависимости от определенных нужд тестировщиков.

Задачи классификации инструментов автоматизации тестирования ПО

В наши дни для поддержки тестирования ПО вообще и в особенности для автоматизации тестирования имеется множество инструментов, число которых измеряется сотнями. Чтобы ориентироваться в этом разнообразии, необходимо организовать информацию об этих инструментах в единую и понятную систему. В отсутствие такой системы тестировщик будет вынужден проводить подобное исследование самостоятельно, что существенно замедлит его работу.

В соответствии с предлагаемым методом, для решения данной проблемы необходимо решить следующие задачи:

1) построить модель динамического автоматизированного теста, достаточно общую для того, чтобы тестировщик мог спроецировать на нее конкретный тест, который ему необходимо автоматизировать;
2) выделить в модели ключевые свойства теста, влияющие на выбор подходящего инструмента автоматизации, которые и станут критериями классификации;
3) проанализировать возможные значения этих свойств (критериев классификации) и оформить в виде иерархических списков;
4) рассмотреть достаточно большое количество различных инструментов автоматизации тестирования и оценить их по выделенным критериям, выбирая их конкретные значения.

Выбор критериев классификации инструментов автоматизации тестирования – непростая задача, поскольку они имеют множество свойств, а у каждого отдельно взятого инструмента есть свои уникальные, присущие только ему свойства. Поэтому

классифицировать инструменты тестирования можно по-разному в зависимости от задачи, которую эта классификация призвана решить.

В рамках данной работы были выбраны следующие принципы классификации:

– удобство – необходимо группировать инструменты таким образом, чтобы принадлежность каждого отдельно взятого инструмента было бы легко оценить;

– однозначность – по возможности нужно выбрать такой метод классификации, чтобы инструмент можно было бы отнести к одному и только одному классу;

– многосторонность – классифицируя, необходимо принимать во внимание различные свойства инструментов, которые могли бы характеризовать их с различных существенных точек зрения;

– полезность – классификация должна служить практическим целям, среди которых можно особо выделить поддержку выбора инструмента и поддержку проектирования тестируемости. Поддержка выбора инструмента означает возможность либо выбрать наиболее подходящий инструмент, либо сузить круг кандидатов, чтобы окончательный выбор делать из ограниченного числа наиболее подходящих инструментов. Поддержка проектирования тестируемости означает помощь в принятии проектировочных решений, которые в дальнейшем упростят автоматизацию тестов.

Для обоснованного выбора критериев классификации, соответствующих указанным принципам, была построена модель автоматизированного теста, позволяющая выделить основные аспекты, влияющие на пригодность инструмента для решения определенных задач тестирования.

Модель динамического автоматизированного теста

В отличие от статического тестирования, которое не предусматривает запуск программы, в динамических методах программа выполняется с целью оценки ее свойств [3]. В случае классического динамического тестирования, выполнением программы занимается тестировщик, возможно используя вспомогательные инструменты (например, для тестирования отдельно взятого модуля, можно использовать интерфейс, специально разработанный с целью тестирования этого модуля). При таком подходе человек является исполнителем теста. Отличие автоматизированного теста в том, что тест выполняется автоматически, а участие человека ограничивается запуском теста (или множества тестов) и анализом результатов. Чтобы это было возможным, тест сначала нужно автоматизировать, т.е. разработать тестовое ПО [4].

Тестовое ПО можно разделить на тестовый сценарий и тестовые данные. Сценарий по сути является программой, включающей взаимодействие с тестируемым объектом, проверка корректности реакции и другие действия, необходимые для того чтобы сценарий во время выполнения смог бы оценить определенные свойства тестируемого объекта. Тестовые данные – это данные, используемые в сценарии для выполнения конкретных тестовых примеров. Тестовые данные можно разделить на входные данные, данные об ожидаемых результатах и вспомогательные данные.

Принимая во внимание указанные аспекты, можно построить модель автоматизированного теста, которая показана на рис. 1. Назначение представленной модели состоит в том, чтобы любую задачу автоматизированного тестирования тестировщик мог легко рассмотреть в ее свете. Поэтому в ней намеренно отсутствует четкий формализм и детализация – простота и интуитивность в этом случае важнее точности.

Тестируемый объект может быть отдельным фрагментом кода, модулем или целой системой. Сценарий взаимодействует с объектом посредством интерфейса

объекта. Сценарию не известно внутреннее устройство объекта (этим инструменты автоматизации тестов отличаются от инструментов анализа кода), а только интерфейс, через который и происходит взаимодействие – вызов операций и проверка корректности. Таким образом, обмен данными между сценарием и тестируемым объектом происходит в обоих направлениях, что и отображено на рисунке. Данные сценарий получает из некоторого источника, но не меняет их.

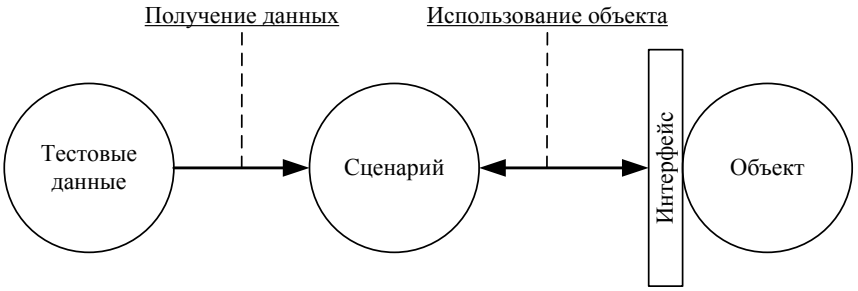

Рис. 1. Схема модели автоматизированного теста.

В данной модели не учтены многие аспекты инструментов автоматизации тестов, которые, однако, не являются существенными для оценки пригодности. Например, взаимодействие сценария с тестировщиком, который запускает тест, отображение результатов и др. Эти аспекты связаны скорее с удобством использования, которое зависит от опыта пользователей, вкусов и других субъективных факторов.

В построенной модели можно выделить четыре аспекта, которые и будут служить критериями классификации:

1) метод получения сценарием данных (обозначим буквой D);
2) тип строения сценария (S);
3) метод взаимодействия с объектом (M);
4) тип интерфейса объекта (I).

Другие критерии, связанные с этими, например способ хранения данных или тип объекта, не существенны для анализа пригодности инструментов автоматизации. Данные могут храниться в любом виде, для инструмента важно только то, как сценарий получит эти данные. Тип объекта и его внутреннее устройство не существенны для инструмента, поскольку взаимодействие происходит только через интерфейс.

Критерии классификации

В результате анализа более 30 различных инструментов тестирования, для каждого критерия классификации были выбраны множества возможных значений, которые обобщенно показаны на рис. 2.

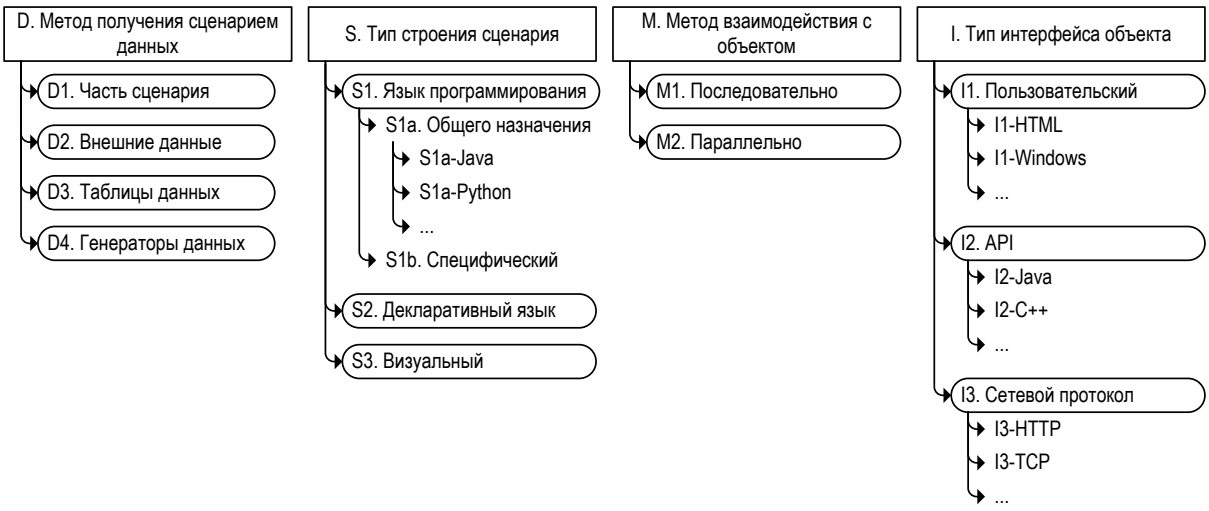

Рис. 2. Классификатор инструментов автоматизированного тестирования.

Для метода получения сценарием данных варианты оказались следующими:

– данные, как неотъемлемая часть сценария (D1) – это означает, что сценарий содержит константные значения данных и выполняется всегда с одним и тем же комплектом данных;

– внешние данные (D2) – данные получаются сценарием из некоторого внешнего источника, таким образом возможно изменять данные, не изменяя самого сценария;

– таблицы данных (D3) – данные также получаются из внешнего источника, при этом существует возможность выполнить один и тот же сценарий с разными комплектами тестовых данных;

– генераторы данных (D4) – в этом случае инструмент избавляет тестировщика вручную вводить тестовые данные, вместо этого данные генерируются автоматически по определенному шаблону.

Для типа строения сценария варианты оказались следующими:

– сценарий пишется на языке программирования (S1) – языки могут быть либо общего назначения либо специализированные;

– сценарий пишется на декларативном языке (S2) – в отличие от первого класса используется какой-либо декларативный язык, что упрощает написание простых сценариев, но делает невозможным создание алгоритмически сложных сценариев;

– сценарий создается с помощью визуальных средств (S3) – используется визуальный интерфейс инструмента, в текстовом виде сценарий недоступен.

Поскольку первая группа инструментов (S1) оказалась достаточно большой, она была разделена на две подгруппы: языки общего назначения (S1a) и языки, специфичные для конкретного инструмента (S1b). Подгруппу S1a можно далее классифицировать по конкретным языкам (например, Java, Python, и т.д. или Multiple в случае если возможно использовать разные языки).

В зависимости от метода взаимодействия с объектом инструменты разделились на две группы:

– последовательное выполнение команд (M1) – сценарий выполняется инструментом пошагово, в одном экземпляре;

– параллельное выполнение команд (M2) – инструмент способен запускать несколько экземпляров сценария параллельно, имитируя таким образом нескольких клиентов объекта.

Таблица 1

Классификация инструментов по четырем критериям

Название	**D**	**S**	**M**	**I**	**Сайт инструмента или разработчика**
Abbot	D1	S1a-Java	M1	I1-Swing	http://abbot.sourceforge.net/
Business Process Testing	D2	S3	M1	I1-Multiple	http://www.hp.com/
Canoo WebTest	D1	S2	M1	I1-HTML	http://webtest.canoo.com/
cPAMIE	D1	S1a-Python	M1	I1-HTML	http://pamie.sourceforge.net/
CppUnit	D1	S1a-C++	M1	I2-C++	http://cppunit.sourceforge.net/
CUnit	D1	S1a-C	M1	I2-C	http://cunit.sourceforge.net/
HttpUnit	D1	S1a-Java	M1	I3-HTTP	http://httpunit.sourceforge.net/
Jemmy	D1	S1a-Java	M1	I1-Swing	https://jemmy.dev.java.net/
JMeter	D3	S3	M2	I3-Multiple	http://jakarta.apache.org/jmeter/
JUnit	D1	S1a-Java	M1	I2-Java	http://www.junit.org/
LoadRunner	D3	S1a-C	M2	I3-Multiple	https://www.hp.com/
MessageMagic	D3	S1a-TTCN	M2	I3-Mutiple	http://www.elvior.com/messagemagic/
NUnit	D1	S1a-Multiple	M1	I2-.NET	http://www.nunit.org/
OpenSTA	D3	S1b	M2	I3-HTTP	http://www.opensta.org/
OpenTTCN	D3	S1a-TTCN	M2	I3-Multiple	http://www.openttcn.com/
QALoad	D3	S1a-C++	M2	I3-Multiple	http://www.microfocus.com/
QF-Test	D1	S3	M1	I1-Swing	http://www.qfs.de/en/qftest/index.html
Oracle Functional Testing	D3	S3	M1	I1-HTML	http://www.oracle.com/
Oracle Load Testing	D3	S3	M2	I3-HTTP	http://www.oracle.com/
QuickTest Professional	D3	S1a-VBS	M1	I1-Multiple	http://www.hp.com/
Rational Functional Tester	D3	S1a-Multiple	M1	I1-Multiple	http://www.ibm.com/
Rational Performance Tester	D3	S3	M2	I3-Multiple	http://www.ibm.com/
Rational Robot	D1	S1b	M1	I1-Multiple	http://www.ibm.com/
Selenium	D1	S3	M1	I1-HTML	http://seleniumhq.org/
SilkPerformer	D3	S3	M2	I3-Multiple	http://www.borland.com/
SilkTest	D3	S3	M1	I1-Multiple	http://www.borland.com/
TestComplete	D3	S1a-Multiple	M1	I1-Multiple	http://automatedqa.com/
TestNG	D1	S1a-Java	M1	I2-Java	http://testng.org/
TestPartner	D3	S1a-VBA	M1	I1-Multiple	http://www.microfocus.com/
The Grinder	D1	S1a-Jython	M2	I3-HTTP	http://grinder.sourceforge.net/
TOSCA	D2	S3	M1	I1-Multiple	http://www.tricentis.com/
WAPT	D3	S3	M2	I3-HTTP	http://www.loadtestingtool.com/

По сути этот критерий разделяет все инструменты на два больших класса – инструменты тестирования функциональности и инструменты нагрузочного тестирования.

По типу интерфейса объекта инструменты разделились следующим образом:

– уровень пользовательского интерфейса (I1) – инструмент имитирует реального пользователя, взаимодействуя с видимыми на экране объектами (окнами, кнопками, полями), и проверяя корректность состояния видимых объектов;

– уровень API (I2) – инструмент имитирует некоторый модуль системы, который использует тестируемый объект на уровне вызовов его операций – по сути этот вариант формирует класс инструментов модульного тестирования;
– уровень протокола коммуникации (I3) – в этом случае инструмент имитирует клиентскую часть некоторой системы, взаимодействуя с тестируемым объектом с помощью сетевых протоколов.

Каждый из этих трех вариантов можно детализировать соответственно по типу пользовательского интерфейса (например, HTML, Swing, Multiple), языку программирования определенного API (например, C++, Java) и протоколу коммуникации (например, HTTP, TCP).

Таким образом, инструменты классифицируются по четырем различным критериям. По каждому критерию инструмент попадает в свой класс и, возможно, подклассы более низкого уровня, в зависимости глубины классификации. В таблице 1 приведена классификация проанализированных инструментов по четырем рассмотренным критериям – D, S, M и I.

Рассмотрим конкретный пример. Предположим, что перед тестировщиком стоит задача провести тест производительности веб-системы, в котором поведение имитируемых пользователей будет зависеть от реакции системы, и при этом в каждой транзакции должны использоваться уникальные данные. Уникальные данные можно получать из таблицы (D3), нужно использовать язык программирования для описания логики поведения пользователя (S1, при этом под условие задачи подходит как S1a, так и S1b), выполняться сценарии должны параллельно для имитации нескольких пользователей (M2), сама имитация должна происходить на уровне сетевого протокола (I3-HTTP или I3-Multiple). В таблице 1 всего 5 инструментов, отвечающих данным параметрам. Не обязательно, что каждый из них подойдет для решения, но круг поиска значительно сузился.

Заключение

В ходе работы был разработан классификатор инструментов автоматизации тестирования ПО, но сформулированный метод разработки и принцип применения таких классификаторов могут быть использованы для множества предметных областей. Например, на сегодняшний день существует множество систем дистанционного обучения, и выбор наиболее подходящей системы может оказаться сложной задачей, несмотря на то, что существуют известные способы их классифицировать, например [7]. Создание простой модели таких систем и основанного на ней общего классификатора, оптимизированного для поддержки целенаправленного обучения, может существенно облегчить задачу изучения и выбора наиболее подходящих систем дистанционного обучения.

Разработанный классификатор, как и подобные ему классификаторы для других предметных областей могут решать две существенные проблемы. С одной стороны они решают проблему ориентации в широкой предметной области и тем самым сокращают время, затрачиваемое специалистами на поиск нужного им решения. С другой стороны они могут служить компактным экскурсом в предметную область для желающих освоить новую область знаний в целом. Кроме того, результаты подобной классификации могут служить отправной точкой для формирования обучающих курсов, таким образом решая проблему отставания образовательных программ от новшеств, появляющихся в быстроразвивающихся сферах деятельности.

Открытым остается вопрос о том, кто будет создавать подобные классификаторы. Одним из возможных решений является создание онлайн-сообществ, заинтересованных в освоении, создании и развитии новых знаний в своей

предметной области. Создание модели решения и выбор критериев классификации изначально может осуществляться узким кругом лиц, добавление же новых знаний, расширение и углубление классификаторов должно быть доступно каждому члену сообщества.

Эта работа выполнена при содействии Европейского социального фонда в рамках проекта „Поддержка развития докторантуры РТУ”.

Литература

1. Еляков А.Д. Современная информационная революция // Социологические исследования. – 2003. – № 10. – С. 29 – 38.
2. Жуковская З.Д., Квасова Л.В., Фролов В.Н. О концепции непрерывного образования // Высшее образование сегодня. – 2007. – № 8. – С. 12 – 17.
3. Канер С., Фолк Д., Нгуен Е.К. Тестирование программного обеспечения. – М.: ДиаСофт, 2000. – 544 с.
4. Fewster M., Graham D. Software Test Automation: Effective use of test execution tools. – New York: ACM Press, 1999. – 574 p.
5. Gamma E., Helm R., Johnson R., Vlissides J.M. Design Patterns: Elements of Reusable Object-Oriented Software. – Boston, MA: Addison Wesley, 1995. – 416 p.
6. Loyens S.M.M., Magda J., Rikers R.M.J.P. Self-directed Learning in Problem-Based Learning and its Relationships with Self-Regulated Learning // Educational Psychology Review. – 2008. – Vol. 20. – Nr. 4. – pp. 411 – 427.
7. Zaitseva L., Bule J., Kuplis U. Advanced e-Learning System Development // Proceedings of the International Conference “Advanced Learning Technologies and Applications” (ALTA’03). Kaunas, Lithuania, September 11–12. – 2003. – pp. 14 – 18.