ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
Л.Н. ГУМИЛЕВ АТЫНДАҒЫ ЕУРАЗИЯ ҰЛТТЫҚ УНИВЕРСИТЕТІ

СОВЕТ МОЛОДЫХ УЧЕНЫХ

Студенттер мен жас ғалымдардың «ҒЫЛЫМ ЖӘНЕ БІЛІМ - 2014» атты IX халықаралық ғылыми конференциясы

IX Международная научная конференция студентов и молодых ученых «НАУКА И ОБРАЗОВАНИЕ - 2014»

The IX International Scientific Conference for students and young scholars «SCIENCE AND EDUCATION-2014»

2014 жыл 11 сәуір
11 апреля 2014 года
April 11, 2014

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
Л.Н. ГУМИЛЕВ АТЫНДАҒЫ ЕУРАЗИЯ ҰЛТТЫҚ УНИВЕРСИТЕТІ

Студенттер мен жас ғалымдардың «Ғылым және білім - 2014» атты IX Халықаралық ғылыми конференциясының БАЯНДАМАЛАР ЖИНАҒЫ

СБОРНИК МАТЕРИАЛОВ IX Международной научной конференции студентов и молодых ученых «Наука и образование - 2014»

PROCEEDINGS of the IX International Scientific Conference for students and young scholars «Science and education - 2014»

2014 жыл 11 сәуір

Астана

УДК 001(063)
ББК 72
Ғ 96

Ғ 96

«Ғылым және білім – 2014» атты студенттер мен жас ғалымдардың IX Халықаралық ғылыми конференциясы = IX Международная научная конференция студентов и молодых ученых «Наука и образование - 2014» = The IX International Scientific Conference for students and young scholars «Science and education - 2014». – Астана: http://www.enu.kz/ru/nauka/nauka-i-obrazovanie/, 2014. – 5831 стр. (қазақша, орысша, ағылшынша).

ISBN 978-9965-31-610-4

Жинаққа студенттердің, магистранттардың, докоранттардың және жас ғалымдардың жаратылыстану-техникалық және гуманитарлық ғылымдардың өзекті мәселелері бойынша баяндамалары енгізілген.

The proceedings are the papers of students, undergraduates, doctoral students and young researchers on topical issues of natural and technical sciences and humanities.

В сборник вошли доклады студентов, магистрантов, докторантов и молодых ученых по актуальным вопросам естественно-технических и гуманитарных наук.

УДК 001(063)
ББК 72

ISBN 978-9965-31-610-4

внедрения разрабатываемая автоматизированная система должна включать максимальный спектр возможностей, необходимых для организации учета в сфере услуг на предприятиях квартирного бюро.

Список использованных источников

1. 1С: Бухгалтерия 8 для Казахстана. – М.: Фирма «1С», 2009.
2. 1С: Предприятие 8.1. Практическое пособие разработчика. Примеры и типовые приемы. – Радченко М.Г. – СПб: Питер, 2009.
3. http://www.v8.1c.ru (актуально на 25.03.14)

ИСПОЛЬЗОВАНИЕ ФОРМАЛЬНЫХ МЕТОДОВ ДЛЯ СПЕЦИФИКАЦИИ, ВАЛИДАЦИИ И ВЕРИФИКАЦИИ ПРОГРАММ

Сатекбаева Айжан Жанабековна
aika_satekbayeva@mail.ru
PhD докторант 3 курса специальности Информационные системы
ЕНУ им. Л.Н. Гумилева, Астана, Казахстан
Бейсеева Жанара Адилбековна
beiseeva@mail.ru
Магистрант 2 курса факультета информационных систем
ЕНУ им. Л.Н. Гумилева, Астана, Казахстан

I Введение

В течение почти трех десятилетий было предложено большое количество формальных методов, их цель состоит в том, чтобы снизить стоимость строительства компьютерных систем и улучшить их качество. Неофициально, формальный метод - это математическая техника или инструмент, который полезен в разработке аппаратного или программного обеспечения. В последнее время формальные методы сыграли значительно активную роль в проектировании аппаратных средств. Все больше и больше компаний, которые продают микропроцессоры и аппаратные чипы, такие как Intel, IBM, и Motorola, используют инструменты на основе формальных методов, например, модели контроллеров [1] и программы автоматического доказательства теорем [2], для обнаружения дефектов в проектах.

Формальные методы всё реже применяются в разработке программного обеспечения, однако в недавних случаях, были обнаружены ранее неизвестные дефекты в программах. Один из ярких примеров является результат исследований в SLAM проекта от Microsoft, в котором Болл и Раджамани разработали несколько формальных методов для автоматического обнаружения дефектов в драйверах устройств [3]. В 2006 году Microsoft выпустила Статическую проверку драйверов (SDV-Static Driver Verifier) [4] в рамках Windows Vista, SDV использует SLAM программную модель для проверки двигателя, выявляя драйверы устройств, нарушающие правила из набора правил интерфейса. Таким образом SDV помогает раскрыть дефекты в драйверах устройств.

Одним важным шагом в разработке высококачественного программного обеспечения является понимание и документирование требовании к программному обеспечению. Исследования показали, что многие программные ошибки можно проследить в неоднозначных, неполных и противоречивых технических заданиях, установление этих дефектов может быть очень дорогостоящим, особенно когда дефекты обнаруживаются на поздних стадиях разработки.

Эта статья описывает способ требований под названием "Сокращение затрат" (SCR - Software Cost Reduction) [5], [6], [7], первоначально разработанная Парнасом, Хенингером в Морской исследовательской лаборатории (NRL), начиная с конца 1970-х годов. Одна из

основных целей исследования NRL заключалась в оценке полезности и масштабируемости инженерных принципов программного обеспечения, используя принципы реконструирования в практической системе. Метод SCR был сформулирован и продемонстрирован, на основе технического задания [5] и нескольких проектных документов [8] для программы полета самолета A-7 американского военно-морского флота.

SCR использует табличное обозначение для представления поведения программного обеспечения системы, чтобы сделать требования понятным для практиков. После того, как требования спецификации SCR было сформулировано, набор инструментов формально называется SCR Toolset [9], [10] может быть использована для проверки логичности, полноты и правильности спецификации.

II. SCR формальной модели и табличные обозначения.

Цель установления технического задания SCR является захват необходимого внешнего поведения программной системы. В спецификации SCR [6], [10], переменные контролируют и управляют количеством систем в окружающей среде. Необходимое поведение системы определяется как отношение, которые система должна поддерживать между контролируемыми и управляемыми переменными. Чтобы указать эти отношения, SCR язык предоставляет два типа вспомогательных классов: переменные – режим класса и терм, а также условия и события.

Условием является предикат, определенный на состоянии системы. Основное событие представлено в виде @T(c), показывает, что состояние меняется с ложного на истинное. Событие @F(c) определяется как @T(¬c). Если значение *c* в текущем состоянии обозначается как *c* и его значение в следующем состоянии, как *c′*, то семантика @T(c) определяется: ¬c∧c' из @F(c) по c∧¬c'.

Два отношения, NAT и REQ [11], определяющие связь между текущими и следующими значениями всех контролируемых и зависимых переменных. NAT определяет естественные ограничения на управление и контроль переменных, это ограничения связанные с физическими законами окружающей среды. REQ использует метод SCR, чтобы указать необходимое поведение системы на зависимых переменных. Учитывая набор зависимых переменных, REQ определяется с функцией определенных таблиц.

SCR Требования к спецификациям: Пример.

Для иллюстрации обозначения SCR, в этом разделе представлена спецификация SCR простой системы управления под названием Safety Injection System (SIS), который контролирует уровень давления воды в системе охлаждения атомной электростанции. Спецификация SIS показывает, как SIS реагирует на изменения в своих контролируемых переменных, изменив одну управляемую переменную, которая контролирует аварийную ситуацию, и находится в режиме "включен" или "выключен". В SIS, система начинает введение безопасности, когда давление воды падает ниже определенного постоянного значения Low. SCR спецификация для SIS, управляемые переменные - mBlock, mReset и mWaterPres - обозначает состояния двух переключателей, блока, сброс переключателей и определение давления воды; режим класса mcPressure указывает один из трех режимов системы, TooLow, Permitted и High; логический терм tOverridden указывает переопределение аварийного поведения и регулируемая величина cSafetyInjection указывает состояние аварийного поведения, "включен" или "выключен".

Рисунок 1 показывает отношения между SIS и управляемыми переменными. Когда, например, система находится в состоянии TooLow и гидравлическое давление изменяется от низкого до высокого, состояние SIS изменяется на Permitted; точно так же, когда SIS находится в Permitted состоянии, гидравлическое давление изменяется от высокого до низкого, если ниже, то состояние изменяется на событие SIS TooLow.

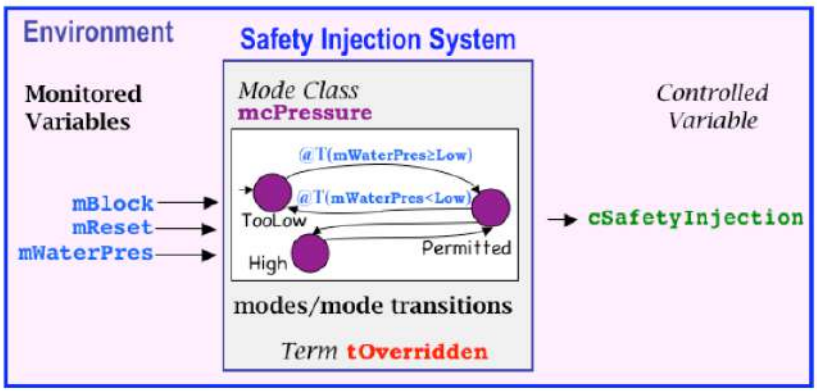

Рисунок 1 SCR техническое задание из строя системы безопасности.

Таблицы 1-3 содержат режимы перехода, событие и таблицу условий, определяющих переход отношении для SIS. Таблица1 содержит таблицу режимов перехода, которая определяет переходы режима для класса mcPressure. Первая строка таблицы 1 означает, что если система находится в режиме TooLow при изменении давления воды от менее низкого до величины, превышающей или равной Low, режим меняется в Permitted. Таблица 2 содержит таблицу событий, определяющую переменную tOverridden. Запись "Никогда" в таблице событий означает, что если система находится в режиме High, ни одно событие не может вызвать tOverridden и изменить на истину. Средняя запись во втором ряду таблицы 2 означает, что система находится в режиме TooLow или Permitted и пользователь поворачивает переключатель Block, если переключатель Reset выключен, то значение tOverridden в следующем состоянии истина. Таблица 3 является таблицей условий, определяющая значение контролируемой переменной cSafetyInjection в качестве инварианта. Этот инвариант определяет требуемое отношение между значениями cSafetyInjection, tOverridden и mcPressure.

Old Mode	Event	New Mode
TooLow	@T(mWaterPres ≥ Low)	Permitted
Permitted	@T(mWaterPres ≥ Permit)	High
Permitted	@T(mWaterPres < Low)	TooLow
High	@T(mWaterPres < Low)	Permitted

Таблица 1 Режимы переходов таблицы для mcPressure.

Mode Class mcPressure	Events	
High	Never	@T(mcPressure=High)
TooLow, Permitted	@T(mBlock=On) WHEN mReset=Off	@T(mcPressure=High) OR @T(mReset=On)
tOverridden	True	False

Таблица 2 Событие таблицы для Overridden

Mode mcPressure	Condition	
High, Permitted	True	False
TooLow	tOverridden	NOT tOverridden
cSafetyInjection	Off	On

Таблица 3 Событие таблицы для cSafetyInjection

III. Инструменты

SCR Toolset представляет собой интегрированный пакет инструментов поддержки для метода SCR требования [9], [10]. Рисунок 2 иллюстрирует инструменты, которые включает в себя редактор спецификации для создания и изменения технического задания, для проверки спецификации, корректности, симулятор для того, чтобы символически выполнить систему, основанную на спецификации, модель контроллера для анализа спецификации прикладных свойств и график зависимостей браузера для отображения переменных зависимостей. Кроме того, набор инструментов включает в себя инвариантный генератор, тестовый случайный генератор и генератор исходного кода.

SCR набор инструментов была также оценен в многочисленных пилотных проектах. В одном проекте, исследователи IV и V фонда NASA использовали инструменты для обнаружения неоднозначности и недостающие предположения в спецификации требований к программному обеспечению для NASA Международной космической станции [12].

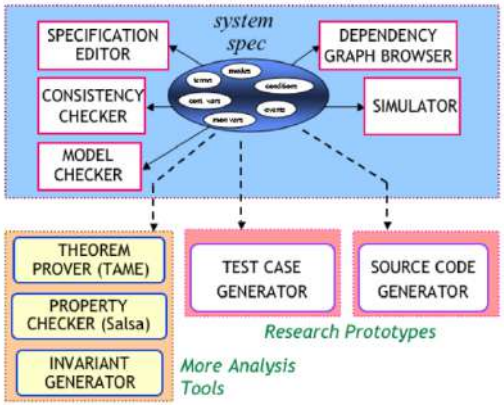

Рисунок 2 SCRToolset

Во втором проекте, инженеры Rockwell авиации использовали инструменты SCR для обнаружения 24 ошибок в спецификации требований к системе управления полетом [13]. Ошибки были обнаружены в построении спецификации, в управлении проверки согласованности, а также в выполнении симулятора. В третьем проекте, NRL использует SCR инструменты, для спецификации группы управления оружием (WCP - WeaponsControlPanel) военной системы. Инструменты обнаружили ряд ошибок, в том числе нарушение безопасности.

Более 200 организаций из научных кругов, промышленности и правительства скачали инструменты SCR. Инструменты были использованы на практике такими компаниями, как LockheedMartin в течение многих лет. Совсем недавно, инструменты были использованы для обеспечения свидетельства в орган сертификации SecurityCritical, модуль встроенного программного обеспечения устройства обеспечивает разделение данных [14] по предварительно заданным требованиям трех безопасностей программных модулей систем NASA[15]. Ниже кратко описаны десять инструментов, которые составляют SCRToolset. Четыре из пяти инструментов, показанных в окне в верхней части рисунка 2 были получены внешними организациями. Пятый инструмент, модель проверки отжима [16], получено от Жерар Хольцмана из Лаборатории реактивного движения NASA. Для получения дополнительной информации о SCRToolset [10].

Редактор спецификации. Чтобы создать, изменить или отобразить техническое задание, пользователь использует редактор спецификации SCR [10]. В SCR, каждая спецификация состоит из словарей и таблиц.

Контролер последовательности SCR [6], [10] проверяет свойства полученной из модели требований SCR. Этот инструмент обнаруживает синтаксис, ошибки типа, неполноту переменных, нежелательный недетерминизм. Когда ошибка обнаружена, контролер последовательности обеспечивает детализированную обратную связь, чтобы помочь

пользователю исправить ошибку. Проверка последовательности - форма статического анализа, так как это достигнуто, не используя спецификацию или выполняя анализ достижимости [17].

Для проверки спецификации, пользователь может запустить симулятор SCR [10] и анализировать результаты, чтобы убедиться, что спецификация захватила намеченное поведение.

IV. Выводы. Использование языка, таких как SCR имеет несколько преимуществ. Во-первых, из-за его табличной записи, разработчики считают, что техническое задание SCR относительно легче понять и применить в разработке требований. Во-вторых, в связи с формальной семантикой SCR, спецификация требуемого поведения является недвусмысленным и точным. Спецификация SCR обеспечивает прочную основу для использования формальных методов и инструментов, чтобы проверить спецификацию для свойств, представляющих интерес. Инструменты SCR предназначены для разработчиков программного обеспечения, которые не имеют математическую подготовку и навыков для доказательства теорем. Таким образом, разработчики могут использовать инструменты для выполнения относительно сложных формальных анализов технических требований. Язык и инструменты SCR обеспечивают практический формальный метод для построения высококачественного технического задания и использования этой спецификации для автоматического создания исходного кода и тестов.

Список использованных источников

1. E.Clarke, O.Grumberg and D.A.Peled, *ModelChecking*. Cambridge, Massachusetts: MITPress,1999.

2. N.Shankar, S.Owre,J. M.Rushby and D.W.J.Stringer - Calvert, “PVSProver Guide, Version2.4, ”Computer Science Lab,SRIInternational, MenloPark, CA,Tech.Rep., Nov.2001.

3. T.Ball, E. Bounimova, B. Cook, V.Levin, J. Lichtenberg, C. M.Garvey, Ondrusek, S.Rajamani, and A.Ustuner, “Thorough static analysis of deviced rivers,”in *European Systems Conference*, 2006.

4. B.Beckert,T.Hoare,R.Hahnle,D.R.Smith,C.C.Green,S.Ranise,Tinelli,T.Ball,andS.K.Rajamani,“Intelligent systems and formal methods in software engineering, ”*IEEE Intelligent Systems*, vol.21,no.6,pp.73–85,2006.

5. K.L.Heninger,“Specifyingsoftwarerequirementsforcomplexsystems:Newtechniquesandtheirapplication,”*IEEETrans.Softw.Eng.*,vol.SE-6,no.1,pp.2–13,Jan.1980.

6. C.L.Heitmeyer,R.D.Jeffords,andB.G.Labaw,“Automatedconsis-tencycheckingofrequirementsspecifications,”*ACMTransactionsonSoftwareEng.andMethodology*,vol.5,no.3,pp.231–261,1996.

7. C.Heitmeyer,J.Kirby,B.Labaw,M.Archer,andR.Bharadwaj,“Usingabstractionandmodelcheckingtodetectsafetyviolationsinrequirementsspecifications,”*IEEETrans.onSoftw.Eng.*,vol.24,no.11,Nov.1998.

8. D.L.ParnasandP.C.Clements,“Arationaldesignprocess:Howandwhytofakeit,”*IEEETrans.Softw.Eng.*,vol.12,no.2,pp.251–257,1986.

9. C.Heitmeyer,J.Kirby,Jr.,B.Labaw,andR.Bharadwaj,“SCR*:Atoolsetforspecifyingandanalyzingsoftwarerequirements,”in*Proc.Computer-AidedVerification,10thAnnualConf.(CAV’98)*,Vancouver,Canada,1998.

10. C.Heitmeyer,M.Archer,R.Bharadwaj,andR.Jeffords,“Toolsforconstructingrequirementsspecifications:TheSCRtoolsetattheageoften,”*ComputerSystemsScienceandEngineering*,vol.20,no.1,pp.19–35,Jan.2005.

11. D.L.ParnasandJ.Madey,“Functionaldocumentationforcomputersystems,”*ScienceofComputerProgramming*,vol.25,no.1,pp.41–61,Oct.1995.

12. S.M.Easterbrook, R. R. Lutz, R. Covington, J. Kelly,Y.Ampo,andD.Hamilton,“Experiencesusinglightweightformalmethodsforrequirementsmodeling.”*IEEETrans.SoftwareEng.*,vol.24,no.1,pp.4–14,1998.

13. S.Miller,“Specifyingthemodelogicofaflightguidancesysteminco REandSCR,”in*Proceeding softhe9th2ndWorkshoponFormalMethodsinSoftwarePractice(FMSP’98)*,1998.
14. C.L.Heitmeyer,M.Archer,E.I.Leonard,andJ.McLean,“Formalspecificationandverificationo fdataseparationinaseparationkernelforanembeddedsystem,”in*Proc.13thACMConf.onComp.andCom m.Sec.*,2006.
15. C.HeitmeyerandR.Jeffords,“Applyingaformalrequirementsmethodtothree NASAsystems:L essonslearned,”in*Proc.2007IEEEAerospaceConf.*,2007.
16. G. J. Holzmann, *DesignandValidation ofComputer Protocols*. Prentice-Hall,1991.
17. R.BharadwajandC.Heitmeyer,“Modelcheckingcompleterequire-mentsspecificationsusingabstraction,”*AutomatedSoftwareEngineer-ing*,vol.6,no.1,Jan.1999.

ӘӨЖ 004.9

ПРОГРАММАЛЫҚ ҚАМСЫЗДАНДЫРУДЫ ИТЕРАТИВТІ ӨҢДЕУ

Султанова Г.А.
GASultanova@mail.ru
Оқытушы, Е.А. Бөкетов атындағы Қарағанды мемлекеттік университеті,
Қарағанды, Қазақстан

Итеративті өңдеу-бұл программалық жүйелерді құруға бағытталған техникалық амал. Бірыңғайландырылған процесс - бұл OOA/D негізінде жобаны өңдеудің итеративті процессі. Негізінде, жоғарыда айтылғандай, программалық жабдықтауды өңдеу процессі (software development process) құрамында құрылуы, жазылуы және жүйені қолдау кіреді. Бірыңғайландырылған процесс UP(United Process) - бұл кеңінен пайдаланылатын объектіге-бағытталған жүйелерді өңдеу барысы. Дербес жағдайда, барлық кезеңдердің неғұрлым жеке-жеке өңдеуін қамсыздандыратын RUP (Rational United Process) бірыңғайландырылған процесі кеңінен танымал болды. Бірыңғайландырылған процессте жүйелерді өңдеу, әсіресе, итеративті өмірлік цикл және қауіп-қатерді бағалау кеңінен қолданылуда.

Бірыңғайландырылған процесстің құрамына бірнеше маңызды идеялар біріктірілген, бірақ оның бірі, шынымен, анықталатыны - бұл итеративті өңдеу (iterative development). Бұл амал шеңберінде өңдеу итерация (iteration) деп аталатын қысқа-жоба түрінде орындалады. Әрбір итерация талаптарды талдаудың, жобалаудың, іске асырудың өзіне лайықты кезеңдерін қосады және тестілік тексеріспен, интеграциямен және жұмыс атқаратын жүйені құрумен аяқталады.

Итеративті өмірлік цикл, белгілі негізге қосылатын модульдерге үйрену және периодты кері байланыспен, бірнеше итерация барысында жүйенің үздіксіз толықтырылуына және кеңейтілуіне негізделеді. Жүйе біртіндеп кеңейтіледі, сондықтан да мұндай амалды кейде итеративті және инкрементальды (iterative and incremental development) деп атайды.

Итеративті процесстің алғашқы идеялары «Серіппе жобалау және эволюциялық өңдеу» деп аталған. Әрбір итерацияның нәтижесінде әлі «Сатуға шығару үшін жарамсыз» толық функционалды емес, бірақ жұмыс атқаратын жүйе шығады. Жүйе тек 10 немесе 15 итерациядан соң ғана толық тауар түріне келеді. Әрбір итерацияның нәтижесі эксперименталды прототип емес және итеративті толық жүйенің бір бөлігінің қорытылған версиясы болып табылады[1].

Ереже бойынша, әрбір итерацияда жаңа талаптар анықталып, жүйе біртіндеп кеңейтілсе де кейбір итерациялар толығымен белгілі программаны толықтыруына арналуы мүмкін. Мысалы, бір итерация жаңа функцияны қосу үшін емес, ішкі жүйенің жұмысын арттыруға қажет болуы мүмкін. Өзгерістердің болжауы: кері байланысқа пен адаптацияға ұшырайды.

Үлгінің ақырғы, өзгеріссіз нұсқасын құруға, дұрыс қалыптастыруға, бекітуге, программалық іске асыруға дейін үлгіні және талаптар жиынын «қатыруға» тырысудың орнына, жүйені өңдеу барысында оны толықтырып отырудың маңыздылығын түсіну және