На правах рукописи

Шошмина Ирина Владимировна

Метод разработки формальных контекстных требований для верификации программных систем логического управления

05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Санкт-Петербург – 2015

Работа выполнена в *федеральном государственном автономном образовательном учреждении высшего образования «Санкт-Петербургский политехнический университет Петра Великого»*.

Научный руководитель:	*доктор технических наук,* *профессор,* *Карпов Юрий Глебович*
Официальные оппоненты:	*Ломазова Ирина Александровна,* *доктор физико-математических наук,* *профессор,* *зав. лаб. «Международная научно-учебная лаборатория процессно-ориентированных информационных систем»,* *национальный исследовательский университет «Высшая школа экономики»,* *Ивановский Сергей Алексеевич,* *кандидат технических наук,* *доцент,* *зав. кафедрой математического обеспечения и применения ЭВМ,* *Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)*
Ведущая организация:	*ФГБУН «Санкт-Петербургский институт информатики и автоматизации Российской академии наук»*

Защита состоится «____» _____ 2015 г. в ____ часов на заседании диссертационного совета *Д 212.229.18* при *ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого»*, расположенном по адресу: *195251, г. Санкт-Петербург, ул. Политехническая, д. 29*

С диссертацией можно ознакомиться в библиотеке *ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого»* и на сайте www.spbstu.ru.

Автореферат разослан «____» _____ 2015 г.

Ученый секретарь
диссертационного совета,
кандидат технических наук, доцент *Васильев Алексей Евгеньевич*

Общая характеристика работы

Актуальность темы исследования. Программные системы логического управления контролируют функционирование самолетов, кораблей, атомных станций и т. д. Ошибки в таких системах могут привести к тяжелейшим последствиям. Поэтому проблема повышения их качества является одной из актуальных областей научных исследований.

Программные системы логического управления относятся к классу реагирующих систем (reactive systems). Ошибки в них выражаются в нарушении желаемых последовательностей управляющих событий. Формальный метод верификации — метод проверки модели (model checking) — позволяет выявить подобные ошибки. Однако при применении метода проверки модели на практике возникает проблема: как выразить функциональность программной системы логического управления в требованиях к поведению.

Степень разработанности темы исследования. Согласно стандартам IEEE 830-1998, ИСО/МЭК 25010:2011, функциональные требования определяют то, что система должна делать. Именно эти требования отражают пожелания заказчика. Для программных систем обработки данных, ошибки которых связаны с неправильным преобразованием входной информации, известно несколько методов разработки функциональных требований [Джексон, 1999; Антон, 1996; Ченг и Атли, 2007]. Для реагирующих систем в работах [Лэмпорт, 1977; Манна и Пнуэли, 1991] сформулированы требования к поведению, исключающие ошибки, типичные для любых взаимодействующих параллельных процессов (безопасность, живость, взаимная блокировка процессов, отсутствие справедливости). Работы [Этесами и Вилке, 1996; Блюм и др., 1999; Куцера, Стрейцек, 2002; Черна, Пеланек, 2003; Айснер, Фишман, 2006] посвящены исследованиям выразительной мощности классов подформул формальных языков [Этесами и Вилке, 1996; Блюм и др., 1999; Куцера, Стрейцек, 2002; Черна, Пеланек, 2003; Айснер, Фишман, 2006]. В работах [Двайер и др., 1999; Ламсверде, 2003; Мондрагон и др., 2005; Конрад и Ченг, 2005; Гранске, 2008; Клебанов, Степанов, Шалыто, 2010; Пост и др., 2011; Саламах и др., 2012; Рамезани и др., 2013] рассматривается проблема перехода от описания на естественном языке к описанию на формальном языке для некоторых часто встречающихся типов требований. Практические подходы, в основном, сосредоточены на задании частных сценариев поведения системы из пожеланий заказчика [Кнапп и др. 2002; Друсинский, 2006; Конрад и др., 2006; Баранов, Котляров, 2011; Иванников, Камкин, Петренко и др., 2007; Коннов, Подымов, Захаров и др, 2014; Паронджанов, 2001]. Анализ литературы и практики разработки показывают, что не существует систематического метода, позволяющего выявить требования к поведению систем логического управления, описывающие их функциональность.

Цели и задачи исследования. Цель диссертационной работы — разработать метод, позволяющий формировать функциональные требования к поведению программных систем логического управления на языке линейной тем-

поральной логики из неформального описания и их впоследствии верифицировать. Для достижения этой цели в работе решены следующие задачи:

1. определение класса требований к поведению систем, описывающих их функциональность на языке линейной темпоральной логики (LTL). Этот класс назван классом контекстных требований;
2. разработка способа выявления контекстных требований из неформального описания программных систем;
3. разработка алгоритма трансляции формул LTL, выражающих контекстные требования, в автомат Бюхи, используемый в методе проверки модели;
4. интеграция отдельных этапов в метод разработки формальных контекстных требований на языке LTL из неформального описания программных систем.

Научную новизну составляют следующие положения:

- подход к определению функциональных требований к программным системам логического управления в терминах поведения, задаваемого через причинно-следственные отношения между событиями на интерфейсах модулей этих систем с учетом источника событий. Эти требования названы в работе контекстными;
- масштабируемые шаблоны LTL-формул, описывающие контекстные требования при различных условиях и режимах функционирования систем;
- метод разработки формальных контекстных требований из неформального описания программных систем;
- алгоритм трансляции LTL-формул, выражающих контекстные требования, в автомат Бюхи, используемый в методе проверки модели.

Теоретическая и практическая значимость работы определяется тем, что выделенный класс требований, предложенный метод выявления и формализации требований этого класса, а также разработанный эффективный алгоритм проверки выполнения таких требований расширяют возможности верификации и способствуют повышению качества программных систем логического управления. Практические результаты состоят в наборе приёмов выявления, формализации контекстных требований и реализации разработанного алгоритма проверки требований к поведению программной системы, позволяющих учитывать при верификации систем важный класс контекстных требований. Предложенный метод внедрен в ЗАО “НПЦ Электродвижение судов” и в учебный процесс СПбПУ, о чем получен акт о внедрении.

Положения, выносимые на защиту:

- новый класс требований к поведению программных систем логического управления, названных контекстными. Он получен систематизацией темпоральных причинно-следственных зависимостей между поведением системы и поведением ее окружения (контекста ее функционирования);
- метод выявления контекстных требований из неформального описания программной системы на основе анализа разных типов отношений причины и следствия между событиями, инициируемыми системой и ее окружением;
- эффективный алгоритм трансляции в автомат Бюхи LTL-формул, описыва-

ющих контекстные требования к системе. Этот алгоритм является этапом метода проверки модели;

- метод разработки контекстных требований на языке LTL из неформального описания программной системы.

Степень достоверности и апробация результатов. Основные результаты диссертационной работы докладывались и обсуждались на следующих конференциях и семинарах: IV международная конференция МГУ "Современные информационные технологии и ИТ-образование 2009"; семинар "Program Semantics, Specification and Verification PSSV 2010"; конференция "Parallel Computing Technologies PaCT-2011"; XIV и XVI Всероссийская научно-методическая конференция "Фундаментальные исследования и инновации в технических университетах" (2010, 2012); семинар "Verification of Embedded Systems VES 2013" конференции "Computer-Aided Verification CAV 2013"; заседание рабочей группы IFIP WG 2.3 "Programming Methodology 2013".

Публикации. Материалы диссертации опубликованы в 6 печатных работах, из них 2 статьи в российских журналах из перечня ВАК [1, 2], 1 статья в зарубежных изданиях, входящих в международные индексы цитирования из перечня ВАК [3], 3 статьи в сборниках международных конференций [4–6].

Личный вклад автора. Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованные работы. При подготовке к публикации полученных результатов с соавторами вклад диссертанта был определяющим. Все представленные в диссертации результаты получены лично автором.

Структура и объем диссертации. Диссертация состоит из введения, 4 глав, заключения и библиографии. Общий объем диссертации составляет 228 страниц, из них 134 — основного текста, включая 21 рисунок, объем приложений — 70 стр. Библиография включает 114 наименований на 14 страницах.

Содержание работы

Во введении показана актуальность диссертационной работы, сформулирована цель и обоснована научная новизна исследований, показана практическая значимость полученных результатов, представлены выносимые на защиту научные положения.

Первая глава посвящена задаче разработки требований к поведению программных систем логического управления. Эта задача относится к области инженерии требований. Основным результатом главы является формулировка нового класса требований на языке линейной темпоральной логики (LTL). В п. 1.1 дан обзор существующих подходов к разработке темпоральных требований. В п. 1.2 описывается постановка задачи разработки требований для темпоральных причинно-следственных отношений между событиями в системе и ее окружении, определяются необходимые формализмы. В п. 1.3 вводится класс таких

требований, названных контекстными, и дается их формальное представление в LTL. В п. 1.4 даны описания систем управления, используемых в работе для демонстрации подхода.

Обзор существующих работ по разработке требований к поведению программных систем логического управления показал, что не существует систематического метода, позволяющего выявлять и формализовать подобные требования, описывающие функциональность этих систем.

Предлагаемое в данной работе решение основано на следующем наблюдении. Задачей реагирующей системы является поддержание определенных шаблонов взаимодействия с окружением. Поэтому значительная часть функций системы связана именно с обеспечением реакций на поведение окружения.

Формальное описание требований к поведению удобно задать формулами LTL, которые строятся на множестве атомарных высказываний $p \in AP$ при помощи булевых операций и темпоральных операторов $\mathcal{U}$ (Until), $\mathcal{X}$ (NextTime) в соответствии с грамматикой:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathcal{X}\varphi \mid \varphi \mathcal{U} \varphi \tag{1}$$

Для сокращения записи в LTL вводятся дополнительные булевы операции и темпоральные операторы — оператор будущего $\mathcal{F}$, глобальный оператор $\mathcal{G}$ и оператор освобождения $\mathcal{R}$, слабый Until $\mathcal{W}$: $\mathcal{F}\varphi = \top U \varphi$; $\mathcal{G}\varphi = \neg\mathcal{F}\neg\varphi$, $\varphi \mathcal{R} \psi = \neg(\neg\varphi \mathcal{U} \neg\psi)$; $\varphi \mathcal{W} \psi = \mathcal{G}\varphi \vee \varphi \mathcal{U} \psi$, где $\top = p \vee \neg p$. Семантика формул LTL задается на бесконечных вычислениях системы.

Взаимодействие системы с ее окружением в общем случае может быть самым разнообразным. В данной работе вводится класс требований, основанных на двух типах причинно-следственных отношений между событиями, названных в работе отношением отклика и отношением предшествования. Через эти отношения и описываются соответствующие функции: функция отклика обеспечивает специфическую реакцию на внешнее воздействие, функция предшествования гарантирует отсутствие реакций в отсутствии внешнего воздействия.

Простейшее отношение описывает ситуацию, в которой по событию s, возникшему в окружении, когда-нибудь в будущем система должна обеспечить реакцию p (*после s наступит p*). Оно названо отношением безусловного отклика. Соответствующая LTL-формула обозначена $Resp(s, p)$:

$$Resp(s, p) = s \Rightarrow \mathcal{F}\, p. \tag{2}$$

Примером требования с отношением безусловного отклика является требование обработки вызова к системе управления лифтом: *"Если лифт вызван, то когда-нибудь в будущем он обязательно прибудет на этаж вызова"*. Отношение отклика в форме (2) хорошо известно, впервые оно введено в работе [Pnueli, 1977].

Отношение безусловного предшествования (*до p наступило s*) описывает

ситуацию, когда *реакция p вызвана возникшим ранее стимулом s*:

$$Prec(s, p) = \neg p \wedge \mathcal{F}\, p \Rightarrow \neg p \mathcal{U}\, s. \tag{3}$$

Пример требования этого типа — отсутствие некорректных реакций в системе управления пожаротушением: *"Если система пожаротушения подводной лодки включилась, то ранее на это была выдана санкция дежурного офицера"*.

На интервале между событием получения стимула до наступления реакции система должна как-то "помнить" факт получения стимула. Обобщим это наблюдение, введя понятие условного отношения отклика (*после s наступит событие p, до которого непрерывно выполняется условие ожидания t*). Такую формулу обозначим $Resp(s, p, t)$:

$$Resp(s, p, t) = s \Rightarrow t \mathcal{U}\, p. \tag{4}$$

Аналогично введем условное отношения предшествования (*до p наступит событие s, которое установит условие ожидания t*). Формула $Prec(s, p, t)$ показывает, что реакции p предшествовало наступление события s, что было запомнено в условии, фиксирующем событие t:

$$Prec(s, p, t) = \neg p \Rightarrow (t \mathcal{U}\, p \Rightarrow \neg p \mathcal{U}\, s). \tag{5}$$

Безусловные отношения (2) и (3) получаются из условных при $t = \top$. При формировании контекстных требований важно отслеживать источник каждого события, поскольку система не может "заставить" окружение выполнить то или иное действие.

Формулы (4) и (5) являются локальными контекстными требованиями между одной причиной и одним следствием, т. е. определенными в конкретном состоянии системы. Чтобы задать требование к любому поведению системы, определим временные области, характеризующие режим (фазу) работы системы, где должны выполняться введенные выше отношения. В данной работе выбраны четыре временные фазы поведения: *начальная* фаза, *глобальная* и *регулярная* фазы, *финальная* фаза.

В *глобальной* фазе контекстное требование к поведению должно выполняться во всех состояниях системы. В *финальной* фазе, в которой система обычно по некоторым параметрам переходит в невозвратный режим, контекстное требование должно выполняться после установления некоторого события. В *начальной* фазе контекстное требование должно выполняться во всех состояниях до наступления заданного события. *Регулярная* фаза — это режим работы системы, при котором контекстное требование начинает выполняться после некоторого специфического события, но ограничено другим событием. Аналогичные фазы для выполнения события предложены в [Dwyer, 1999]: событие должно выполняться глобально, до события, после события, между двумя событиями.

Формализуем на языке LTL контекстные требования для каждой фазы:

$$\begin{array}{rcl} global(s,p,t;\varphi) & = & \mathcal{G}\,\varphi(s,p,t), \\ fin(s,p,t;q;\varphi) & = & \mathcal{F}\,\mathcal{G}\,q \Rightarrow \mathcal{F}\,global(s,p,t;\varphi), \\ start(s,p,t;r;\varphi) & = & \neg r \Rightarrow \varphi(s,p,t)\,\mathcal{W}\,r, \\ reg(s,p,t;q,r;\varphi) & = & \mathcal{G}\,(q \Rightarrow start(s,p,t;r;\varphi)), \end{array} \tag{6}$$

Здесь φ обозначает формулу либо $Resp$, либо $Prec$, формула $global$ определяет отношения в глобальной фазе, fin, $start$ и reg — соответственно в финальной, начальной и регулярной фазах. Событие q задает начало финальной фазы, r — конец начальной фазы, либо начало и конец регулярной фазы. Формулы (6) построены в предположении, что события s, p, q, r, t независимы. В общем случае события могут задаваться булевыми выражениями и быть зависимыми.

В рамках предложенной структуры контекстные требования легко масштабируются на случай, когда стимул и/или реакция задаются цепочками событий: $\vec{s} = \{s_1, s_2, \ldots, s_m\}, \vec{p} = \{p_1, p_2, \ldots, p_n\}$ с наборами условий $\vec{v} = \{v_1, v_2, \ldots, v_{m-1}\}$, $\vec{t} = \{t_1, t_2, \ldots, t_n\}$, ограничивающих стимулы и реакции соответственно. Формальное задание отношений отклика и предшествования изменится следующим образом:

$$\begin{array}{rcl} \mu(\vec{s},\vec{v}) & = & s_1 \wedge v_2\,\mathcal{U}\,(\ldots s_{m-1} \wedge (v_m\,\mathcal{U}\,s_m)\ldots), \\ \chi(\vec{p},\vec{t}) & = & t_1\,\mathcal{U}\,(p_1 \wedge \ldots t_{n-1}\,\mathcal{U}\,(p_{n-1} \wedge (t_n\,\mathcal{U}\,p_n))\ldots), \\ Resp(\vec{s},\vec{p},\vec{v},\vec{t}) & = & \mu(\vec{s},\vec{v}) \Rightarrow v_2\,\mathcal{U}\,(s_2 \wedge \ldots (v_m\,\mathcal{U}\,(s_m \wedge \chi(\vec{p},\vec{t})))\ldots), \\ Prec(\vec{s},\vec{p},\vec{v},\vec{t}) & = & \neg p_1 \Rightarrow \big(\chi(\vec{p},\vec{t}) \Rightarrow \neg p_1\,\mathcal{U}\,\mu(\vec{s},\vec{v})\big) \end{array} \tag{7}$$

Структура самих требований при этом останется прежней и задается по шаблонам, представленным в (6), в конкретной фазе функционирования системы. Требование к системе управления лифтом "*всегда, если был вызов, то лифт проедет мимо этажа вызова не более одного раза*" является примером контекстного требования с цепочкой откликов в глобальной фазе: $global(\vec{s},\vec{p},\vec{v},\vec{t}) = \mathcal{G}\,\big(req \Rightarrow (\neg atfloor \wedge \neg dooropen)\,\mathcal{U}\,\big((atfloor \wedge \neg dooropen)\,\mathcal{U}\,(atfloor \wedge dooropen)\big)\big)$; $\vec{s} = \{req\}, \vec{p} = \{\top, atfloor \wedge dooropen\}$, $\vec{v} = \emptyset$, $\vec{t} = \{\neg atfloor \wedge \neg dooropen, atfloor \wedge \neg dooropen\}$.

Определенные выше требования заданы относительно конечных цепочек событий с отношениями отклика и предшествования. Требования в условиях бесконечного поведения окружения должны быть представлены формулами LTL следующего вида:

$$\psi \Rightarrow \varphi, \tag{8}$$

где φ задает контекстное требование в конкретной фазе функционирования, а ψ — это требование к бесконечному поведению окружения. В частности, классические требования справедливости в нашей терминологии являются разновидностью темпоральных требований с бесконечным поведением окружения.

Итак, формально на языке LTL контекстные требования задаются формулами отношений отклика и предшествования (7) в разных в временных фазах (6) и требований с бесконечным поведением окружения (8).

Предложенный в данной работе метод разработки контекстных требований был использован для верификации двух программных систем: системы управления лифтом, описанной Д. Кнутом в книге “Искусство программирования”, и системы управления энергоснабжением судна (СУЭС), разработанной одним из российских производителей. Для этих систем были доступны как техническое задание, так и полный текст программного кода. В обоих случаях программный код был тщательно протестирован разработчиками систем. Для обеих систем в данной работе были построены контекстные требования к их поведению и формально проверено наличие ошибок в коде программ.

В работе подробнее представлена СУЭС. Оригинальная реализация системы на C++ составляет около 20000 строк кода, не учитывая внешних библиотек. Несмотря на то, что СУЭС была разработана и протестирована, требования к ней задавались в самом общем виде, а потому задача разработки формальных требований возникла практически в том же виде, что и при проектировании новой системы.

Основная цель СУЭС состоит в обеспечении электроэнергией потребителей и оборудования судна. Для этого СУЭС подключает и отключает судовые электростанции (ЭС) в зависимости от нагрузки. В каждую из двух электростанций входят дизель, генератор, генераторный автомат (ГА). На рис. 1 жирной рамкой обведены модули СУЭС: контроллеры дизелей, генераторов, ГА, ЭС, пульт и шкаф управления. Модули СУЭС работает независимо, параллельно и асинхронно. Они синхронизируются друг с другом через асинхронные каналы. Основные объекты окружения, взаимодействующие с СУЭС, изображены вне рамки. Состояние аппаратных объектов отслеживается при помощи датчиков и управляется сигналами.

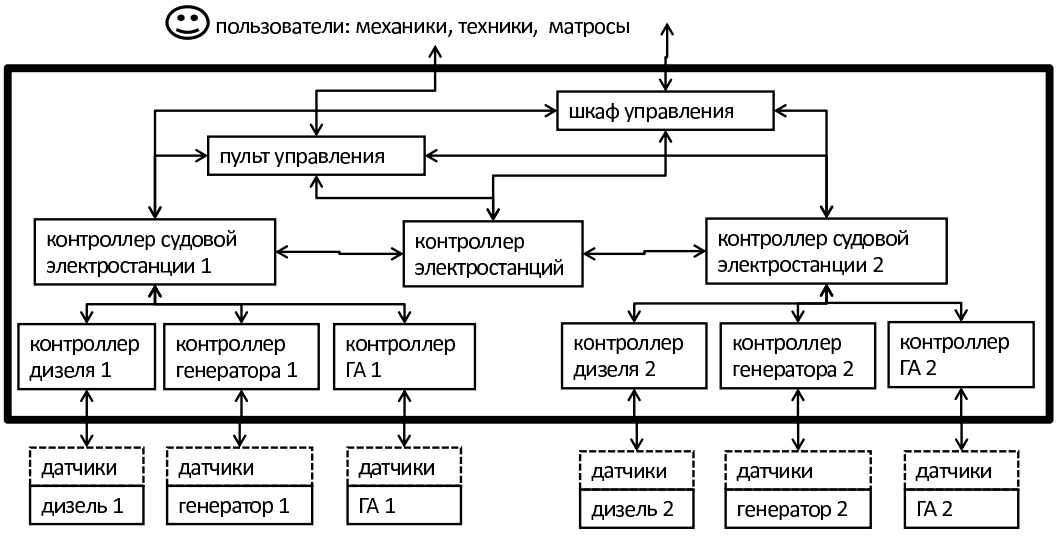


Рис. 1. Состав системы управления энергоснабжением судна

Вторая глава посвящена задачам выявления и формализации контекстных требований к системе из неформального описания. В качестве основы для решения задачи выявления требований в работе предложено использовать метод схем целей (Problem Frame Approach), разработанный М. Джексоном для программ обработки данных. Основным результатом главы является модификация этого метода для выявления контекстных требований к программам логического управления и формализация их на языке LTL. В п. 2.1 анализируются проблемы, возникающие при применении метода М. Джексона при формировании темпоральных требований. В п. 2.2 предложена модификация этого метода для разработки контекстных требований. В п. 2.3 представлена атрибутная семантика для преобразования контекстных требований, описанных на языке схем целей, в формулы LTL.

В методе М. Джексона требования формируются, исходя из целей будущей программной системы. Задача метода состоит в структуризации неформальной информации о системе так, чтобы выявить цели программы. Согласно М. Джексону, все разнообразие целей программ сводится к ограниченному набору целей-прототипов. Метод поддерживается графическим языком схем. Процесс разработки целей и соответствующих требований — итеративный. Изначально сведения о целях программы формулируются в общем виде и не совпадают с целями–прототипами, затем они последовательно уточняются. Текущие сведения о целях компактно собираются на схемах. Для каждой цели–прототипа определена так называемая элементарная схема. Структура таких схем фиксирована. При совпадении цели с одним из прототипов процесс уточнения этой цели завершается. Метод М. Джексона не рассматривает цели, связанные с временными зависимостями: схемы целей с разными последовательностями одинаковых множеств событий здесь не различаются. Поэтому формулировка темпоральных требований средствами оригинального метода невозможна.

Для выявления контекстных требований в работе введены две новых цели-прототипа. Цель 1 называется *гарантией локального отношения*: “Система должна гарантировать выполнение темпорального отношения (отклика или предшествования) между событиями”. Цель 2 устанавливает *отношение во временной фазе*: “Система должна гарантировать выполнение цели 1 в (конкретной) временной фазе функционирования”. Чтобы различать события стимула и реакции в отношении, задавать условные отношения, определять последовательности событий, язык схем целей был расширен новыми графическими элементами. С учетом этих графических элементов в работе разработаны два новых типа элементарных схем для целей 1 и 2: схемы отношений (отклика и предшествования) и схемы временной фазы (глобальной, начальной, финальной, регулярной).

Модификация метода М. Джексона в части итеративного уточнения целей системы касается выявления только контекстных требований, она состоит в следующем. Сначала собирается информация о событиях на интерфейсе системы. Для каждого события, инициируемого окружением системы, определяется га-

рантируемое локальное отношение (отклика или предшествования), сведения о котором фиксируется на соответствующей схеме отношения. Потом анализируется, в какой временной фазе отношение должно выполняться, и составляется схема фазы. Как только на схемах отношения и фазы собрана вся необходимая информация, процесс уточнения контекстного требования останавливается. Если событие окружения не вошло ни в одно из контекстных требований, то, возможно, это свидетельствует о противоречивости или неполноте неформального описания системы. Поэтому роль этого события в функционировании системы необходимо выяснить у заказчика.

Предложенный в работе модифицированный язык схем целей отличается от известных графических языков (MSC, UCM, TLCharts и др.), также позволяющих задавать последовательности событий, единообразием представления информации о причинных зависимостях событий, привязкой их к фазам функционирования системы, поддержкой не полностью определенных требований.

Продемонстрируем схемы отношения и временной фазы, а также элементы модифицированного языка схем целей на примере формирования контекстного требования к запуску СУЭС. Процесс запуска ЭС многоэтапный. Он включает сложные проверки всего электрооборудования. Запуск ЭС без предшествующего сигнала оператора свидетельствует о некорректном управлении. Сформулируем требование, не допускающее такое поведение:

Требование 1 (отсутствие неуправляемых запусков ЭС1) *“В дистанционном автоматизированном режиме с нейтральным приоритетом всегда, если лампа “ДГ1 Пуск” не горит ровным светом, но будет гореть ровным светом в будущем, то до этого поступит сигнал старта”.*

Событие “запустить ЭС1” поступает от оператора и является стимулом. Событие “включить лампу “ДГ1: Пуск” ровным светом” устанавливается системой на пульте управления и является реакцией. Таким образом, в отношении между этими событиями участвуют три объекта: “Пользователь”, “Пульт управления”, и “Машина запуска ЭС1”. Объекты изображаются на схеме прямоугольниками (см. рис. 2, *а*). Объект “Машина запуска ЭС1” отражает функцию системы в конкретной схеме, такой объект называется машиной. Машина выделяется на схеме прямоугольником с двумя вертикальными чертами. Машина на схеме может быть только одна.

Согласно требованию 1, реакция системы “включить лампу “ДГ1: Пуск” ровным светом” не должна происходить без подачи стимула “запустить ЭС1”. Следовательно, события связаны отношением предшествования. На рис. 2, *а* приведена схема, построенная по элементарной схеме цели 1 гарантии отношения предшествования с одним стимулом и одной реакцией. Расположение объектов, стимула и реакции на схеме фиксирован. События требования 1 сокращенно зафиксированы на интерфейсах объектов в формате <Объект>:<событие> (например, “МЗС1: включить лампу”). Интерфейсы изображаются на схемах

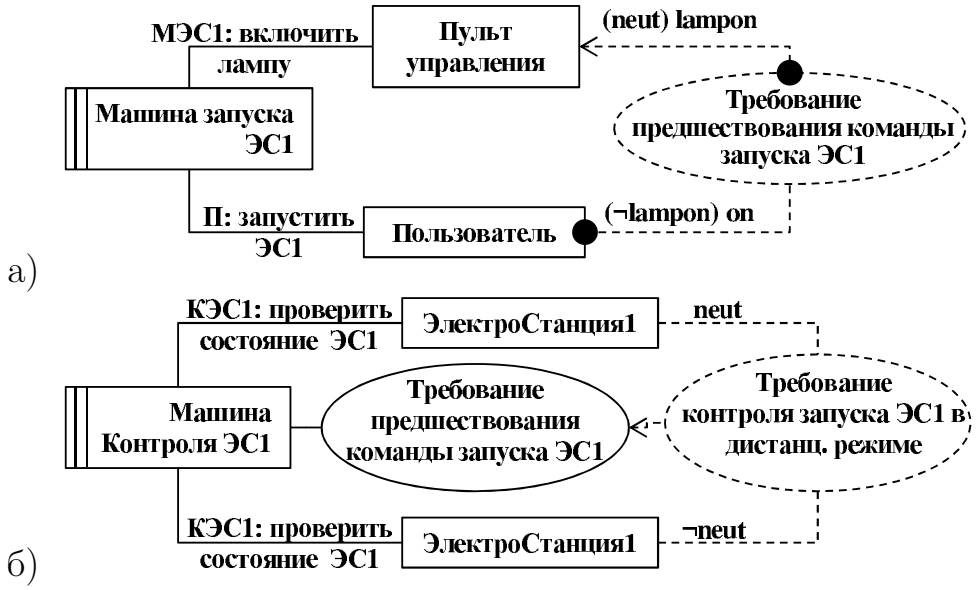


Рис. 2. Схемы требования отсутствия неуправляемых запусков ЭС1: а) гарантии локального предшествования команды запуска, б) гарантия отношения на рис. 2, *а* в дистанционном автоматизированном режиме с нейтральным приоритетом.

прямыми сплошными линиями. Требование также кратко определяется на естественном языке в овале с пунктирной границей, например, “Требование предшествования команды запуска ЭС1”.

Конкретные параметры, характеризующие события, задаются на штриховых прямых линиях, связывающих объекты и требования. Эти параметры и входят в формальное описание требования. Штриховая линия с точкой соответствует событию-стимулу. Оно определяется на рис. 2, *а* булевой переменной on. Булево выражение в скобках до переменной задает последовательность событий $\neg lampon$, ограничивающих возникновение стимула. Штриховая стрелка с точкой обозначает реакцию $lampon$. Эта реакция ограничена условием нахождения системы в дистанционном автоматизированном режиме с нейтральным приоритетом $neut$.

Отношение предшествования команды запуска ЭС1 должно выполняться каждый раз в дистанционном автоматизированном режиме с нейтральным приоритетом, что соответствует регулярной фазе функционирования. Схема на рис. 2, *б* построена по элементарной схеме регулярной фазы. В основном, состав схем с фазами подобен схеме отношения, описанной выше: в них присутствует машина, объекты, требование, интерфейсы, события и параметры. К особым элементам схем с фазами относится ссылка на элементарную схему отношений. Она обозначается овалом со сплошной границей (рис. 2, *б*).

После того, как информация о контекстном требовании собрана при помощи модифицированного метода схем целей, соответствующее требование транслируется в LTL-формулу по правилам трансляции, разработанным в диссертационной работе. В частности, по схеме на рис. 2, *а* однозначно определяются стимул и реакция, а также ограничивающие их события. Следовательно, имеется вся информация для построения формулы $Prec(on, lampon, auto)$ из (5).

Аналогично, сведения на схеме рис. 2, *б* позволяют составить LTL-формулу $reg(s, p, t; Prec)$ из (6):

$$\mathcal{G}\,(neut \Rightarrow (\neg lampon \wedge neut\,\mathcal{U}\,lampon \Rightarrow \neg lampon\,\mathcal{U}\,on)\,\mathcal{W}\,\neg neut). \qquad (9)$$

Теоретически доказано, что сложность трансляции LTL-формулы в автомат Бюхи, используемый в методе проверки модели, экспоненциальна относительно сложности LTL-формул. Из формулировки контекстных отношений для цепочек событий (7) очевидно, что формулы LTL, выражающие контекстные требования, могут быть весьма сложными. Современные алгоритмы трансляции LTL-формулы в автомат Бюхи (назовем их алгоритмами LTLBA) не справляются с такими формулами. Например, по формуле, соответствующей контекстному требованию с бесконечным поведением окружения (8), $\Phi(n) = \neg\big((\wedge_1^n \mathcal{G}\,\mathcal{F}\,r_i) \Rightarrow \mathcal{G}\,(q \Rightarrow \mathcal{F}\,p)\big)$ при $n = 6$ стандартным алгоритмом, встроенным в средство верификации SPIN, автомат Бюхи построить не удается.

Глава 3 посвящена проблеме эффективной трансляции LTL-формул, выражающих контекстные требования, в автоматы Бюхи. Предлагаемое решение использует на промежуточном этапе обобщенные автоматы Бюхи с допускающими переходами (ОАБ) и символьное кодирование состояний и переходов всех промежуточных автоматов, реализуемый при помощи упорядоченных редуцированных бинарных решающих диаграмм (BDD). ОАБ можно эффективно построить по LTL–формуле. Символьный подход с BDD также хорошо себя зарекомендовал на практике. Во всех известных из литературы алгоритмах с ОАБ символьное кодирование если и применяется, то лишь частично, поскольку существуют трудности построения функции переходов обобщенного автомата и корректного формирования его допускающих условий в символьном виде. Именно эти две задачи и решены в данной главе. Результатом главы является новый символьный алгоритм LTLBA, названный нами SymLTL2BA. В п. 3.1 рассматриваются недостатки наиболее эффективного среди известных к настоящему времени алгоритмов LTLBA, возникающие при трансляции LTL-формул, описывающих контекстные требования, формулируется постановка задачи разработки нового эффективного алгоритма. В п. 3.2 описывается символьный алгоритм трансляции LTL-формул в обобщенный автомат Бюхи с допускающими переходами. В п. 3.3 решается задача поиска сокращенной ДНФ положительной булевой функции, представленной BDD. В п. 3.4 излагается алгоритм SymLTL2BA. Автомат Бюхи, полученный при трансляции формулы, переводится во входной язык SPIN. В п. 3.5 приводятся результаты сравнения реализации SymLTL2BA и нескольких известных алгоритмов LTLBA. В главе используются общепринятые определения автомата Бюхи и BDD.

Определение 1. *Обобщенный автомат Бюхи с допускающими переходами (ОАБ) есть кортеж* $G = (Z, \Sigma, z^0, \Delta, T)$, *такой что:*

- Z — *конечное множество состояний;*

- Σ — *конечный алфавит;*
- $z^0 \in Z$ — *начальное состояние;*
- $\Delta : Z \times \Sigma \to 2^Z$ — *функция переходов;*
- $T = \{T_1, \cdots, T_m\}$ — *множество подмножеств допускающих переходов по Бюхи,* $T_i \subseteq Z \times \Sigma \times Z, 1 \leq i \leq m$.

Автомат G допускает бесконечное слово w по Бюхи, если его вычисление, индуцируемое w, включает хотя бы по одному переходу из каждого подмножества T_i бесконечное количество раз.

Состояния автомата G_φ, который строится по LTL-формуле φ, размечаются комбинациями пометок ξ из множества элементарных формул $el(\varphi)$, образованном из подформул темпоральной формулы φ:

$$\begin{aligned} el(\varphi) = \ & \{\mathcal{X}(\psi_1 \,\mathcal{U}\, \psi_2) \mid \psi_1 \,\mathcal{U}\, \psi_2 \in subf(\varphi)\} \cup \{\mathcal{X}\varphi\} \cup \{\mathcal{X}\top\} \cup \\ & \{\mathcal{X}(\psi_1 \,\mathcal{R}\, \psi_2) \mid \psi_1 \,\mathcal{R}\, \psi_2 \in subf(\varphi)\} \cup \{\mathcal{X}\psi \mid \mathcal{X}\psi \in subf(\varphi)\}, \end{aligned}$$

где $subf(\varphi)$ — множество всех подформул формулы φ. Каждую формулу $\xi \in el(\varphi)$ можно рассматривать как темпоральное обязательство: автомат "обязуется" выполнить все такие подформулы в будущем. Функция распространения обязательств $sat : subf(\varphi) \to 2^\Sigma \times 2^Z$ строится рекурсивно по структуре φ с использованием правил расширения темпоральных операторов.

Переходы автомата направляются в минимальное подмножество состояний, помеченных обязательствами, полученными по функции sat. В работе показано, что задача поиска такого подмножества сводится к поиску сокращенной дизъюнктивной нормальной формы (ДНФ) по BDD положительной булевой функции (БФ), которой представляется функция переходов G_φ. Однако стандартный алгоритм построения ДНФ БФ по BDD не приводит к сокращенной ДНФ.

Наше решение строится на основе того факта, что минимальная конъюнкция сокращенной ДНФ входит в любую ДНФ положительной булевой функции. Т. о., чтобы построить минимальную ДНФ по BDD положительной булевой функции, надо находить и последовательно извлекать минимальные конъюнкции из ДНФ. Алгоритм поиска минимальных конъюнкций построен на основе алгоритма топологической сортировки вершин BDD. Сложность алгоритма составляет $\mathcal{O}(|V|+|E|)$, где $|V|$ — количество вершин, $|E|$ — количество дуг BDD. Реализация этого алгоритма включена в свободно распространяемую библиотеку BuDDy 2.4.

Другая задача состоит в формировании допускающих групп переходов ОАБ. Допускающие условия на переходах автомата G_φ связаны с состояниями, помеченными оператором $\mathcal{U}$. Для каждой формулы $f = \psi_{1,f} \,\mathcal{U}\, \psi_{2,f}, \mathcal{X}f \in el(\varphi)$ формируются множества допускающих переходов:

$$T_f = \{(z, \alpha, z') \mid f \notin \lambda(z') \text{ или } (\alpha, z') \in sat(\psi_{2,f})\}. \tag{10}$$

Это переходы в такие состояния, которые не помечены f или которые принадлежат функции распространения обязательств подформулы $\psi_{2,f}$.

Условие (10) в совокупности с процедурой построения функции позволяет корректно транслировать большинство LTL-формул в автоматы G_φ. Однако существует некоторые чрезвычайно редко встречающиеся при спецификации поведения LTL-формулы, язык которых таким образом задать не удается. Такие формулы содержат подформулы вида $\mathcal{X}(\psi_1 \mathcal{U} \psi_2)$ во множестве подформул $subf(\varphi)$ формулы φ. Причина недостаточности условия (10) в том, что для таких формул минимальное допускающее множество пометок состояния больше, чем минимальное множество, получаемое по алгоритму.

Теорема 1. *Обобщенный автомат Бюхи G_φ, построенный по LTL-формуле φ без подформул вида $\mathcal{X}(\psi_1 \mathcal{U} \psi_2) \in subf(\varphi)$ с допускающим условием (10), допускает тот же язык, что и формула φ.*

В работе показано, что любую LTL-формулу можно привести к виду, не содержащему подформул $\mathcal{X}(\psi_1 \mathcal{U} \psi_2) \in subf(\varphi)$. Такая форма представления названа модифицированной положительной нормальной формой (МПНФ). Все синтаксически правильные LTL–формулы φ в МПНФ описываются грамматикой:

$$\begin{array}{lcl} \varphi & ::= & p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathcal{X}\psi \mid \varphi \mathcal{U} \varphi \mid \varphi \mathcal{R} \varphi \\ \psi & ::= & p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathcal{X}\psi \mid \varphi \mathcal{R} \varphi \end{array} \qquad (11)$$

Итак, основные этапы разработанного алгоритма SymLTL2BA таковы. На этапе синтаксического анализа транслируемая LTL-формула представляется в МПНФ. По синтаксическому дереву строится функция распространения обязательств *sat* и формируются допускающие условия ОАБ (10). Далее строится функция переходов ОАБ при помощи алгоритма поиска сокращенной ДНФ по BDD положительной булевой функции, получающейся из *sat*. Трансляция ОАБ в искомый автомат Бюхи осуществляется по известному алгоритму [Giannakopoulou и др., 2002]. Размер ОАБ и временная сложность его построения оцениваются, как $\mathcal{O}(2^{N+M})$, где $N = |AP|$, M — количество темпоральных операторов в формуле φ. Размер автомата Бюхи при этом оценивается $\mathcal{O}(M \cdot 2^{N+M})$.

Алгоритм SymLTL2BA был разработан в 2011 году [4]. Он сравнивался с наиболее эффективной реализацией алгоритма LTLBA — LTL2BA из работы [Gastin и др., 2001 г.]. Результаты сравнения реализаций для двух классов формул, значимых для проверки контекстных требований, приведены на рис. 3: Φ — отрицание формулы из класса (8), Ψ — отрицание формул, образуемых контекстными отношениями (7). Эти эксперименты показывают существенное (на 2 и более порядка) преимущество алгоритма SymLTL2BA.

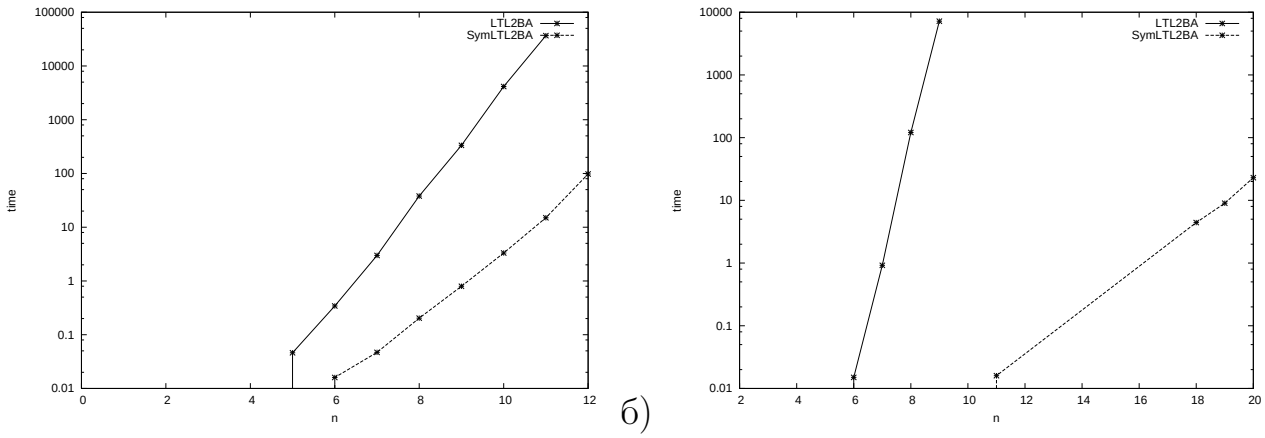
В последнее время было разработано несколько алгоритмов, направленных на снижение сложности трансляции отдельных классов LTL-формул: LTL3BA в

работе T. Babiak и др., 2012, LTL2TGBA, реализованный A. Duret-Lutz в 2011 г. Последние реализации этих алгоритмов также используют BDD для операций над конечными множествами данных. A. Duret-Lutz разработал систему тестов SPOT для сравнения реализаций алгоритмов LTLBA. В частности, в тесте ltl2tgba при помощи реализаций строятся автоматы Бюхи для LTL-формул из некоторого заранее подобранного набора. В наборе known (табл. 1) наибольшее количество формул описывают контекстные требования. Тест ltl2tgba фиксирует суммарное время построения всех автоматов в секундах (на табл. 1 — time), их суммарный размер (states, edges, transitions). Кроме того, при тестировании проверяется корректность работы алгоритмов. Для этого по полученным автоматам Бюхи проверяется выполнимость LTL-формул, а также пустота пересечения языков, задаваемых LTL-формулой и ее отрицанием. По столбцу count из табл. 1 видно, что алгоритм SymLTL2BA успешно прошел проверку корректности для всех 184 формул набора known, а также что среднее время трансляции им LTL-формул из этого набора в автоматы Бюхи сравнимо с другими современными алгоритмами LTLBA.

tool	count	time	states	edges	transitions
`ltl2ba`	184	12.36	1017	3385	30237
`ltl3ba`	184	11.63	891	2722	25511
`ltl2tgba`	184	16.93	673	1645	10373
`symltl2ba`	184	12.29	953	2865	15737

Таблица 1. Результаты теста ltl2tgba из пакета SPOT на формулах набора known

Четвертая глава посвящена представлению окончательной формулировки метода разработки формальных контекстных требований и его применению при верификации СУЭС. В п. 4.1 метод разработки формальных контекстных требований интегрируется из шагов, описанных в работе ранее. В п. 4.2 излагается процесс разработки контекстных требований к СУЭС из представленной документации, проведенный в соответствии со сформулированным методом. По



а) б)

Рис. 3. Сравнение по времени работы реализаций алгоритма LTL2BA и SymLTL2BA. а) $\Phi = \neg\big((\wedge_1^n \mathcal{G}\,\mathcal{F}\,p_i) \Rightarrow \mathcal{G}\,(q \Rightarrow \mathcal{F}\,r)\big)$, б) $\Psi = \neg(p_1\,\mathcal{U}\,(p_2\,\mathcal{U}\,\ldots\,(p_{n-1}\,\mathcal{U}\,p_n)\ldots))$

программному коду СУЭС, полученному от разработчиков, построена модель на языке Promela и проведена верификация этой модели с помощью верификатора SPIN с подключенным алгоритмом SymLTL2BA. В п. 4.3 приводятся результаты верификации СУЭС.

При разработке контекстных требований из описания СУЭС было выявлено 71 событие (9 событий — общих для обеих станций, по 31 событию — для каждой станции). Всего было составлено 36 контекстных требований. Из них 20 требований основывались на отношении предшествования, остальные — на отношении отклика. В LTL-формулы с отношением предшествования в среднем входило 4-5 событий, с отношением отклика — 13-14 событий.

Найденные при верификации ошибки подтверждены трассами в программном коде системы. Почти каждое из найденных контекстных требований нарушалось при верификации. Причинами тому были, как правило, легко исправляемые ошибки, связанные с последовательностями проверок в отдельных модулях реализации СУЭС. Три найденные ошибки были критическими. Одна из них относилась к реакции системы на неполное срабатывание генераторного автомата в некоторых режимах. В результате СУЭС включалась независимо от текущих действий оператора. Вторая ошибка состояла в неконтролируемом выходе СУЭС из состояния защиты. Как следствие, на оборудование мог подаваться ток при производстве на нем ремонтных работ. Третья ошибка состояла в отсутствии подключения запрашиваемой резервной электростанции. При этой ошибке СУЭС теряла управление. Основной причиной этих и многих других ошибок было то, что некоторые события из-за асинхронного чередования действий независимых модулей системы не обрабатывались должным образом. В процессе ходовых испытаний судна, проводившихся параллельно и независимо от нашего анализа его СУЭС, некоторые ошибки, обнаруженные при верификации, проявились и привели к аварийным ситуациям. Благодаря формальной проверке контекстных требований удалось систематизировать взаимодействия модулей системы, упростить алгоритмы работы отдельных модулей, повысить прозрачность поведения системы для конечных пользователей.

В заключении обобщены основные результаты работы.

Основные результаты работы

1. Определен новый класс контекстных требований к поведению программных систем логического управления, описывающих их функциональность через расширенные темпоральные причинно-следственные отношения между событиями на интерфейсах системы в зависимости от источника событий на языке линейной темпоральной логики (LTL).
2. Предложен модифицированный метод схем целей для выявления контекстных требований из неформального описания программных систем логического управления, расширяющий известный метод, используемый для другого

класса программных систем.

3. Разработан и реализован алгоритм трансляции LTL-формул в автомат Бюхи на основе обобщенных автоматов Бюхи с допускающими переходами и символьного представления данных, реализуемого при помощи бинарных решающих диаграмм. Эффективность алгоритма значительно превосходит существующие решения на отдельных классах LTL-формул. Алгоритм включен в систему верификации SPIN.
4. Предложен метод разработки контекстных требований на языке LTL из неформального описания программных систем логического управления.
5. Предложенный метод разработки использован для построения формальной спецификации требований из технического задания реальной системы управления энергоснабжением судна. При верификации этой системы был выявлен ряд критических ошибок, которые другими методами повышения качества управляющих программ определить не удавалось.
6. Сформулированы рекомендации по использованию метода разработки формальных контекстных требований при верификации реальных программных систем логического управления.

Список публикаций

1. Шошмина И. В. Проектирование программных бортовых систем управления с поддержкой верификации // Модел. и анал. информ. систем. 2010. Т. 17, № 4. С. 125–136. (из перечня ВАК).
2. Шошмина И. В. Методика составления контекстных требований к программным системам логического управления // Информационно-управляющие системы. 2014. № 3. С. 68–77. (из перечня ВАК).
3. Shoshmina I., Belyaev A. Symbolic algorithm for generation Büchi Automata from LTL formulas // Parallel Computing Technologies - 11th International Conference PACT'2011 / Ed. by V. Malyshkin. Vol. 6873 of LNCS. Springer-Verlag, 2011. P. 98–109.
4. Shoshmina I. Distributed Embedded Control System Design with Verification Support // Automatic Control and Computer Sciences. 2011. Vol. 45, no. 7. P. 437–443.
5. Шошмина И. В., Карпов Ю. Г. Технология проектирования и верификации распределенных бортовых систем // Материалы международного семинара PSSV / Под ред. В. А. Непомнящего, В. А. Соколова. Казань: 2010. С. 88–94.
6. Шошмина И. В. Технология проектирования и верификации распределенных встроенных бортовых систем // IV Международная научно-практическая конференция "Современные информационные технологии и ИТ-образование" / Под ред. В. Сухомлина. М.:ИНТУИТ.РУ, 2009. — декабрь. С. 464–471.