УДК 681.51:629.78

АВТОМАТИЗИРОВАННАЯ СПЕЦИФИКАЦИЯ, ВЕРИФИКАЦИЯ И СИНТЕЗ УПРАВЛЯЮЩИХ ПРОГРАММ НА ОСНОВЕ ЛОГИЧЕСКОГО И АЛГЕБРАИЧЕСКОГО ПОДХОДОВ

2007 А. А. Тюгашев

Самарский государственный аэрокосмический университет

Рассматривается комплекс проблем спецификации, верификации и синтеза управляющих программ реального времени, исполняемых на борту космического аппарата. Анализируются подходы к решению этих задач на основе временн*о*й логики и применения расширенной алгебры управляющих процессов. Описывается структура основанного на приведенном подходе инструментального программного комплекса разработки управляющих программ для космических аппаратов.

Разработка надежного и качественного программного обеспечения является актуальной и сложной задачей. Размер современных разрабатываемых комплексов программного обеспечения достигает миллионов и десятков миллионов команд, программное обеспечение становится столь сложной системой, что появляется острая необходимость нахождения таких способов организации процессов ее жизненного цикла (проектирование, разработка, тестирование, эксплуатация и др.), чтобы свести к минимуму вероятность появления неприемлемых ошибок.

Согласно российским и международным стандартам (ИСО 9126, ГОСТ 28195-89) качество программной системы характеризует ряд базовых показателей:

- корректность, то есть соответствие программы спецификации;
- надежность (отсутствие ошибок, восстанавливаемость и др.);
- эффективность (в том числе временн*а*я эффективность);
- сопровождаемость (в том числе удобство проведения анализа и простота внесения изменений).

Как правило, оценка степени соответствия программы предъявляемым к ней требованиям производится на основании тестирования. В то же время в случае, когда какое-либо из свойств системы (в том числе из перечисленного выше списка) может быть записано на некотором формальном языке, анализ на соответствие этому свойству может быть произведен при условии наличия адекватной и соответствующей принимаемой формальной системе модели программы методами верификации программ. В ряде случаев путем использования строгих логических рассуждений удается гарантированно доказать тот факт, что программа имеет или не имеет то или иное важное интересующее исследователя свойство. Тестирование же, как правило, не дает подобной гарантии в связи с тем, что провести полную проверку функционирования на всех потенциальных наборах исходных данных и для всех вариантов (логических ветвей) не представляется возможным в силу размера современных программных комплексов.

При использовании формальных методов доказательства свойств программ рассуждения обычно проводятся следующим образом. Сначала формулируется на некотором языке спецификация программы. Используемый язык должен:

1) обладать полнотой и достаточной выразительной силой для записи всех функциональных требований и ограничений, накладываемых на программу;

2) быть непротиворечивым;

3) позволять записывать значимые свойства программ лаконично и удобно;

4) быть понятным и удобным для человека.

Далее, в рамках подхода Model Checking, для проведения верификации программа заменяется некоторой отражающей ее поведение в интересующих аспектах (семантику) моделью и доказывается или опровер-

методами формального логического вывода наличие тех или иных свойств. Возможен также подход, когда каждый из операторов (базовых конструктов) программы характеризуется на языке так называемых пред- и постусловий [1, 2], а также описывается влияние композиции этих конструктов на истинность и ложность данных условий каждого оператора. В этом случае исследование свойств программы проводится путем логического вывода, прослеживающего последовательность операторов, в рамках полученного таким образом исчисления программ.

Еще более привлекательной представляется возможность синтеза программы, гарантированно удовлетворяющей поставленным в спецификации требованиям, на основе некоторой формальной процедуры. В этом случае фактически процесс синтеза программы сродни процессу логического вывода в некоторой теории (очевидна параллель с конструктивизмом в математике).

Логический подход к спецификации, верификации и синтезу программ

Традиционный вычислительный алгоритм представляет собой преобразователь некоторых исходных данных, имеющихся в начальный момент до исполнения алгоритма, в выходные данные, получаемые по окончании работы. Данные хранятся в памяти компьютера. При этом в процессе работы исходные данные не меняются. В отличие от этого, если рассматривать управляющий алгоритм, выполняющийся, например, в бортовой вычислительной системе космического аппарата (КА), то он имеет дело с объектом управления – как правило, динамической системой, состояние которой постоянно изменяется. В ряде случае при этом вообще не подразумевается какого-то останова системы, и она должна функционировать непрерывно. Реакция системы управления должна укладываться в определяемые физическими характеристиками управляемых процессов временные рамки.

Выполняя формальные спецификации и наблюдая за поведением описываемой системы, можно проверить, реализуют ли спецификации предполагаемые функциональные требования или нет. Например, можно проверить, не возникают ли определенные нежелательные последовательности событий при некоторых критических обстоятельствах. Таким образом, производится тестирование на ранней фазе процесса разработки. К сожалению, традиционная временная логика [3] плохо подходит для описания свойств систем управления реального времени. Неадекватность классической временной логики проблемам спецификации систем реального времени основана на том факте, что ее операторы обеспечивают только качественное представление времени, таким образом, обеспечивая способы выражения таких свойств, как предшествование, возможность или постоянство во времени. Только наличие оператора «следующий момент времени» приближает временную логику к количественному представлению времени: например, с его применением можно постулировать, что данное свойство будет сохраняться на протяжении k «тиков» (тактов) времени, начиная с текущего момента.

Другой подход описывается интервальной логикой, которая предоставляет «прерывающийся» оператор для конкатенации интервалов и разрешает мгновенное описание последовательностей событий и свойств, сохраняющихся в различимых, смежных интервалах. Интервальная логика имеет, однако, те же самые ограничения, что и классическая временная логика.

Впоследствии временные логики были развиты и сформулированы метрическая временная логика, временная интервальная логика реального времени, временная логика реального времени. При этом в качестве семантической модели часто применяются таймированные автоматы.

Алгебраический подход (алгебры процессов)

В основу алгебраического подхода были положены не программы, а процессы. При этом термин «процесс» определяет поведение некоторой системы. Существенным для характеристики исследуемых систем является факт, что, как правило, процессы в них носят параллельный и/или распределенный характер. В рамках рассматриваемого подхода для описания поведения используется алгеб-

раический/аксиоматический подход. При этом становится возможным проведение рассуждений о таких системах, используя алгебру, то есть уравнения. Посредством операций с уравнениями можно производить верификацию, то есть проверку того, что система удовлетворяет требуемым свойствам. В [4] был сформулирован базовый набор аксиом относительно операций, производимых над процессами.

Синтез логического и алгебраического подходов при проектировании алгоритмов управления КА

При анализе бортовой аппаратуры (БА) современного КА можно выделить ряд функциональных систем, определяющих в совокупности его целевые свойства. Каждая из этих систем, в свою очередь, имеет достаточно сложную структуру и состоит из подсистем, приборов, агрегатов, датчиков и др. Элементы систем БА соединяются между собой и совместно должны функционировать в рамках решения поставленных перед КА целевых задач.

При этом для современных КА характерным является применение бортовых цифровых вычислительных машин (БЦВМ) для решения задач управления БА [5]. Даже на микро- и наноспутниках использование управляющего бортового компьютера (с бортовой информационной системой) сейчас стало нормой [6].

При этом по ряду причин, включающих, в частности, простоту коррекции бортового программного обеспечения (БПО) и добавления в него дополнительных задач, возможность оперативного дистанционного изменения состава решаемых БПО задач, более эффективную загрузку вычислительных ресурсов для сложных многофункциональных комплексов БПО, в которых моменты начала и окончания решения задач могут меняться в широких пределах в зависимости от временных разбросов работы БА и исходных данных, передаваемых с Земли, предпочтительно использование приоритетной динамической асинхронной организации вычислительного процесса [5]. Данная дисциплина организации вычислительного процесса характеризуется тем, что управление работой бортовой вычислительной системы (БВС) осуществляется бортовой операционной системой (БОС) реального времени, которая обеспечивает параллельное исполнение ряда задач на одной или нескольких БЦВМ с поддержкой системы прерываний как по сигналам от БА, так и от специального устройства отсчета времени – таймера.

КА создается для решения заданного набора целевых задач (ЦЗ) в зависимости от типа аппарата, параметров орбиты и т. д. При этом особенности космической баллистики, а также другие факторы обуславливают тот факт, что каждая из задач верхнего уровня (базовых ЦЗ) привязывается к опорным моментам времени.

Обозначим набор базовых ЦЗ КА как

$g_1, g_2, \ldots, g_N$.

Привязанные к опорным моментам шкалы времени T_{on}, целевые задачи образуют пары:

$(g_1, T_{on2}), (g_1, T_{on2}), \ldots, (g_N, T_{onN})$.

Обратим внимание на следующее обстоятельство: многие целевые задачи (например, работа двигательной установки для сообщения требуемого импульса) имеют определенную длительность выполнения.

В свою очередь, реализация каждой ЦЗ верхнего уровня требует, как правило, согласованной работы нескольких систем, приборов, агрегатов, датчиков и других элементов БА, каждый из которых при этом должен обеспечить выполнение набора функциональных задач $f_1, f_2, \ldots, f_N$ в некоторые моменты времени $t_1, t_2, \ldots, t_N$. Моменты времени зависят от времени выполнения основной ЦЗ и привязываются, таким образом, к T_{on}. Аналогично основным ЦЗ функциональные задачи также имеют определенные длительности исполнения - $t_1, t_2, \ldots, t_N$.

С точки зрения БВС, выполнение функциональной задачи есть выдача некоторой команды или посылка сигнала той или иной системе, тому или иному прибору или агрегату БА или же выполнение некоторой программы из комплекса БПО, результаты работы которой (информационная связь) затем используются при выполнении иных функциональных задач.

Для описания спецификации (требований к логике управления) на вербальном уровне часто применяются такие выражения, как «За 5 минут до *X* необходимо включить режим *P*», «В момент срабатывания *X* подаются команды f_2, f_5», «Операции f_2, f_3 должны завершиться к началу работы системы *C*», «В случае *Y* с интервалом в 1 с выдаются команды f_1, f_2, f_3», «Не менее чем через 7 мксек после *X* обмен данными может быть возобновлен путем выдачи команды f_{10}».

Таким образом, для спецификации требований к времени выполнения может быть использована как временная (темпоральная) логика [3], а в процессе конструирования комплексного управляющего алгоритма - аппарат алгебры процессов реального времени [4].

При этом в процессе проектирования бортового комплекса управления и соответствующего БПО важно знать временные характеристики программ, начиная с самых ранних этапов проектирования БПО и КА в целом.

Поэтому представляется целесообразным и наиболее эффективным синтез логического и алгебраического подходов. При этом, с одной стороны, на каждом шаге проектировщику предоставляется возможность самостоятельно, применяя алгебраические операции, конструировать все более и более сложные управляющие алгоритмы, а с другой стороны, рассуждать об их свойствах и степени соответствия спецификации с применением формальной логической теории. Более того, совмещение алгебраического и логического подходов позволяет строго утверждать на каждом шаге конструирования точное соответствие создаваемого программного продукта спецификации на основе выведенных свойств операций алгебры во временной логике.

Предлагается адаптированная к проблематике проектирования управляющего БПО и расширенная по сравнению с имеющимися формальная система, обладающая свойствами временной логики, которая позволяет учитывать длительности процессов, происходящих на борту КА, и специфицировать на точно определенном языке логику управления подсистемой и системой на основе логики управления отдельным прибором.

В [7] предложена и исследована алгебра управляющих алгоритмов (УА) реального времени, включающая операции во временном пространстве:

1. Операция совмещения по началу (СН), означающая задание общего (одинакового) времени запуска некоторых управляющих алгоритмов.

2. Операция совмещения по концу (СК), означающая задание общего времени окончания выполнения управляющих алгоритмов.

3. Операция следования $\rightarrow$, означающая запуск некоторого УА сразу после окончания выполнения другого УА.

Также введена операция в логическом пространстве =>, смысл которой заключается в обуславливании выполнения той или иной функциональной задачи истинностью или ложностью некоторого логического условия. Эта операция позволяет точно специфицировать альтернативную композицию в терминах текущей бортовой ситуации, то есть в данном случае она специфицирует теорему расширения алгебры процессов в понимании [4].

Отметим также, что с точки зрения классических алгебр процессов, операции СН и СК являются несущими физический смысл уточнениями операции параллельной композиции процессов, а операция $\rightarrow$ – последовательной композиции. При этом, естественно, сохраняются истинными все базовые аксиомы алгебр процессов.

В качестве основного множества (носителя) алгебры предлагается набор кортежей следующего вида:

$$УА\ РВ = \left\{ < f_i, t_i, t_i, \overline{l_i} > \right\},\ \overline{i = 1, N},$$

где f_i – функциональная программа (действие); t_i - момент начала исполнения действия (целое неотрицательное число); t_i - длительность действия (целое неотрицательное число); $\overline{l_i}$ - логический вектор, обуславливающий действие. Нетрудно видеть, что графическим образом (за исключением логического компонента) представленной математической модели будет циклограмма.

Использование данного подхода позволяет уйти от проблемы сложности (числа со-

стояний) традиционно используемых в качестве модели семантик временных логик и алгебр процессов автоматов и систем переходов.

Состояние системы в приведенной модели в некоторый момент времени описывается, с точки зрения вычислительного процесса:

- выполняющимися в данный момент функциональными задачами;

- признаками – завершены ли к данному моменту те или иные функциональные задачи.

В конечном счете, все представление текущей ситуации может быть сведено к значениям логических переменных, образующих вектор.

В качестве дополнительных компонент этого вектора могут использоваться определяемые для каждой f_i два предиката – предикат (функция) выполнимости $a_{fi}(t)$, отражающий тот факт, что функциональная задача (ФЗ) *fi* выполняется в момент времени *t*, и предикат $a_{zfi}(t)$, говорящий о том, что к моменту времени *t* выполнение данной ФЗ было завершено.

Расширенная модель алгебраической системы управляющих алгоритмов допускает истолкование присутствующих в алгебре [7] символов СН, СК, → как предикатов (утверждений) о соотношении управляющих процессов:

1. f_1 СН f_2 означает, что управляющие алгоритмы начинаются в одно и то же время (t_1=t_2).

2. f_1 СК f_2 означает, что управляющие алгоритмы заканчиваются в одно и то же время (t_1+t_1=t_2+t_2).

3. $f_1 \rightarrow f_2$ означает, что момент старта алгоритма f_2 совпадает (равен) моменту завершения выполнения алгоритма f_1 (t_1+t_1=t_2).

Расширенная модель также вводит дополнительные символы << и ≤, не задающие жестких связок между процессами, а накладывающих некоторые ограничения, в рамках которых конкретное время выполнения операций может меняться.

4. Предикат $f_1 << f_2$, означает, что момент окончания выполнения алгоритма f_1 предшествует (строго меньше) моменту старта алгоритма f_2 (t_1+t_1<t_2).

5. Предикат $f_1 \le f_2$ означает, что момент запуска управляющего алгоритма f_1 предшествует моменту запуска управляющего алгоритма f_2($t_1 < t_2$).

При этом теория интерпретируется на том же самом множестве, что и алгебра УА. Очевиден при этом тот факт, что в случае применения операций алгебры УА (СН, СК, →) тождественно истинными будут одноименные им предикаты.

Временная логика позволяет проводить доказательства соответствия сгенерированной программной реализации управляющего алгоритма его спецификации, что означает правильность выполнения функциональных задач и корректность работы КА в целом.

Список литературы

1. Логика и компьютер. Моделирование рассуждений и проверка правильности программ / А. М. Анисов, П. И. Быстров, В. А. Смирнов и др. – М.: Наука, 1990.

2. Хоор Ч. А. З. Непротиворечивые взаимодополняющие теории семантики языков программирования // В сб.: Семантика языков программирования. - М.: Мир, 1980.

3. Ben-Ari M., Manna, Z., Pnueli A.: The Temporal Logic of Branching Time. Proc. 8th Annual Symposium on Principles of Programming Languages, 1981, ACM Press, Williamsburg, p. 164-176. Springer-Verlag, 1992.

4. Baeten J. C. M., Bergstra J. A.: Real time Process Algebra. Formal Aspects of Computing, 3, p.142-188, 1991.

5. Управление космическими аппаратами зондирования Земли: Компьютерные технологии / Д. И. Козлов, Г. П. Аншаков, Я. А. Мостовой, А. В. Соллогуб. - М.: Машиностроение, 1998.

6. Ю. М. Урличич, А. С. Селиванов, В. М. Вишняков, Ю. М. Тучин. Предварительные результаты летных испытаний технологического наноспутника ТНС-0 // Тезисы докладов 10-й международной конференции «Системный анализ, управление и навигация». – М.: Изд-во МАИ. - 2005. - С.24-25.

7. А. А. Калентьев. Автоматизированный синтез алгоритмов асинхронного управления технологическими системами с множеством дискретных состояний / Самар. аэрокосм.ун-т. – Самара. - 1998.