УДК 004.056

Ю.Н. ПРОХОРОВА

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина

ИСПОЛЬЗОВАНИЕ МЕТОДА АНАЛИЗА ДЕРЕВЬЕВ ОТКАЗОВ ДЛЯ СОЗДАНИЯ СПЕЦИФИКАЦИЙ ФУНКЦИОНАЛЬНО БЕЗОПАСНЫХ СИСТЕМ В EVENT-B

Предложен метод, который позволяет объединить метод анализа деревьев отказов (FTA) и формальную спецификацию в Event-B. Использование методики формальной разработки гарантирует корректность внедрения требований по функциональной безопасности в спецификацию систем критического применения. Предложенный подход основан на пошаговом внедрении результатов анализа по принципу «сверху вниз» на каждом этапе детализации спецификации, начиная с абстрактной и заканчивая реализацией. В статье также приведены измененные модели жизненного цикла разработки программного обеспечения информационно-управляющих систем в соответствии с предлагаемым методом. В качестве примера внедрения FTA в формальную спецификацию была использована противообледенительная система самолета АН-140М.

Ключевые слова: *методы формальной спецификации и верификации, Event-B, функциональная безопасность, FTA.*

Введение

При создании спецификаций информационно-управляющих систем (ИУС) критического применения важным аспектом является гарантия того, что система из безопасного состояния не перейдет в опасное, а также то, что сама спецификация не содержит ошибок проектирования. В основе Event-B лежит описание свойств – инвариантов и событий системы и проверка путем математического доказательства того, что выполнение событий (в любых допустимых состояниях) не нарушит указанные свойства. Таким образом, в процессе детализации создается полная и непротиворечивая спецификация системы [1]. Для обеспечения функциональной безопасности формальных спецификаций систем критического применения могут быть использованы методы формального анализа надежности, такие как анализ деревьев отказов (FTA) [2] и анализ видов и причин отказов (FMEA) [3]. Использование методики формальной разработки в сочетании с методами анализа надежности может гарантировать корректность внедрения требований по функциональной безопасности в Event-B спецификацию.

Наиболее широко известным подходом для объединения методов анализа надежности и формальной спецификации является ForMoSA подход, описанный в [4]. Этот подход объединяет в себе инженерную практику, формальные методы и математику: традиционный анализ надежности, временную логику, верификацию и оптимизацию. Несмотря на то, что авторы подробно описывают процесс формализации FTA и приводят пример практической реализации, данный подход не может быть применен в Event-B без изменений, поскольку он был разработан для другой формальной системы (ITL). Вопросы внедрения результатов FTA и FMEA в формальную спецификацию были рассмотрены также для таких формальных систем, как Action Systems [5, 6], B Method [7], а также FME(C)A для Event-B [8]. Однако, предложенные в [5, 6] методы не нашли практического применения и имеют только теоретический характер, а в статье [8] описан общий подход к решению поставленных задач, но не рассмотрены технические детали его реализации. Подход, описанный в [7], демонстрирует механизмы внедрения результатов анализа надежности, обнаружения ошибок и восстановления в формальные спецификации, созданные с помощью классического B метода. Примером практического использования этого подхода является промышленная рабочая станция Fillwell™, предназначенная для управления жидкостями. Данный подход был взят за основу при разработке метода внедрения результатов анализа деревьев отказов в Event-B спецификацию.

Таким образом, целью данной статьи является разработка метода создания спецификаций ИУС в Event-B, где механизмы обеспечения функциональной безопасности являются неотъемлемой частью поведения системы, а также демонстрация того, что путем пошаговой детализации спецификации и внедрения результатов анализа деревьев отказов

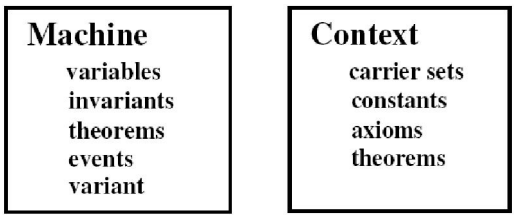
можно получить конечную спецификацию, удовлетворяющую требованиям по функциональной безопасности и не содержащую ошибок проектирования.

1. Метод формальной спецификации и верификации Event-B

B-метод, разработанный Жаном-Раймондом Абриалем (Jean-Raymond Abrial) [1], применяется для создания высоконадежных систем. Примерами успешного использования B-метода для создания систем критического применения могут служить проекты, описанные в [9]. Event-B – это расширение B-метода [10] для моделирования параллельных, распределенных и реактивных систем. Event-B использует Abstract Machine Notation (AMN) для создания и верификации моделей систем.

Основными преимуществами формальной спецификации в Event-B являются применение процесса детализации (уточнения) для представления системы на различных уровнях абстракции и использование математических доказательств для проверки логичности (последовательности) между уровнями детализации.

Формальная спецификация в Event-B [11] представляет собой совокупность машины (рис. 1, а) и контекста (рис. 1, б) на каждом уровне детализации (рис. 2). Контекст – необязателен, он содержит константы, множества, а также аксиомы и теоремы, которые определяют типы и ограничения заданных констант и множеств. Машина является обязательным элементом спецификации и, в общем случае, включает в себя список переменных, событий, которые выполняются в соответствии с определенными условиями, и инвариантов. Инварианты определяют свойства системы, которые всегда выполняются [12].



а б

Рис. 1. Машина и контекст в Event-B:
а – машина; б – контекст

Процесс детализации подразумевает постепенный переход от абстрактного описания к более точному, с постепенным добавлением деталей системы. При этом указанные на ранних стадиях свойства системы сохраняются на всех этапах детализации.

Сущность формализации заключается в том, что событиям дается формальное математическое определение и при добавлении свойств системы осуществляется доказательство их соответствия инвариантам. Таким образом, гарантируется корректное поведение системы на всех этапах ее разработки.

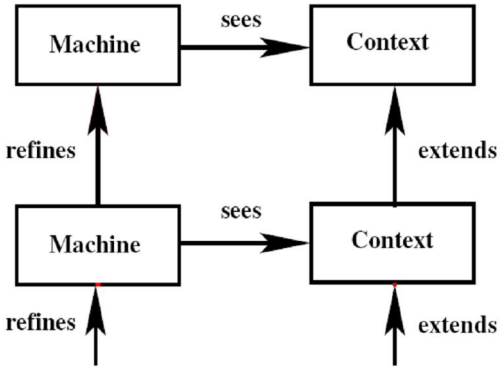


Рис. 2. Взаимодействие машины и контекста

Такой подход нашел широкое применение при разработке программного обеспечения. В этом случае в Event-B создается абстрактная спецификация, а затем осуществляется одна или несколько промежуточных детализаций. Последняя детализация называется реализацией. Эта реализация может вызывать операции других машин (импорт). После полной проверки разработанной реализации возможна генерация кода программы на языке высокого уровня.

Таким образом, на протяжении всего жизненного цикла разработки программного обеспечения используется математический аппарат для доказательства соответствия разрабатываемой системы техническим требованиям.

2. Модели жизненного цикла разработки ИУС с использованием формальных методов

На рис. 3 представлена модель жизненного цикла (ЖЦ) разработки ИУС в соответствии со стандартом IEC 61508 [13]. Пунктирной линией обозначены этапы, которые видоизменяются при использовании методов формальной спецификации и верификации, в частности Event-B метода.

В отличие от традиционного ЖЦ разработки систем при использовании формальных методов нет необходимости разрабатывать архитектуру программного обеспечения, проектировать модули, а также осуществлять их тестирование и проверку модульной интеграции [9]. Разработка происходит следующим образом (рис. 4): на первом шаге в соответствии со спецификацией требований строится формальная абстрактная спецификация системы, далее осуществляется детализация спецификации (refinement). Этапов детализации может быть несколько, при переходе на каждые последующий этап средствами математического доказательства гаран-

тируется сохранение всех инвариантных свойств системы. Конечная формальная спецификация автоматическими средствами транслируется в программный код. Полученное программное обеспечение проходит процедуру подтверждения соответствия.

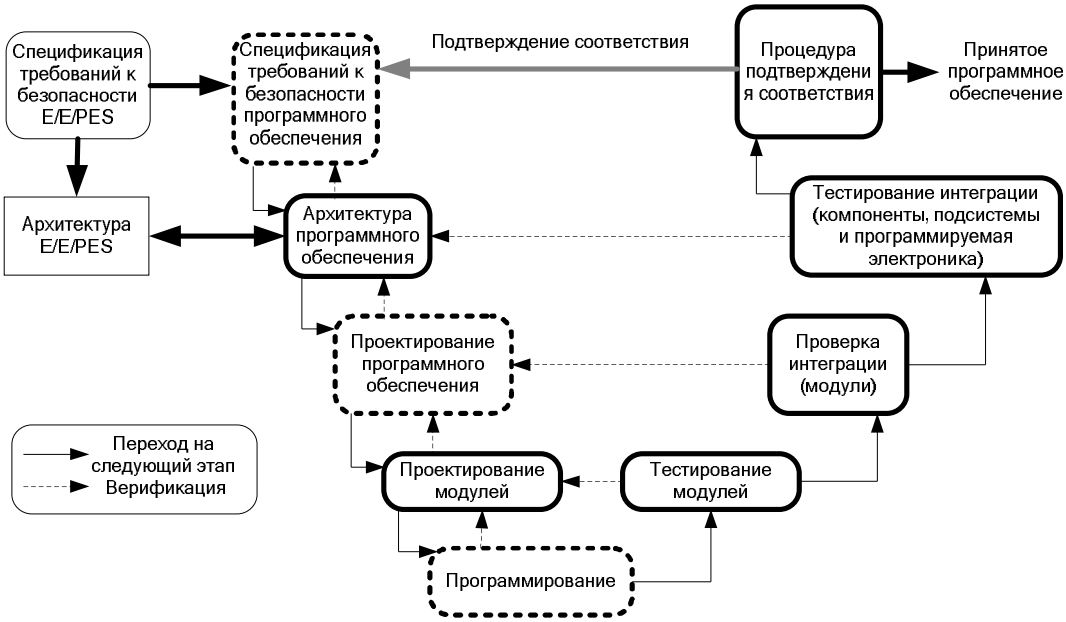


Рис. 3. Модель ЖЦ разработки ИУС

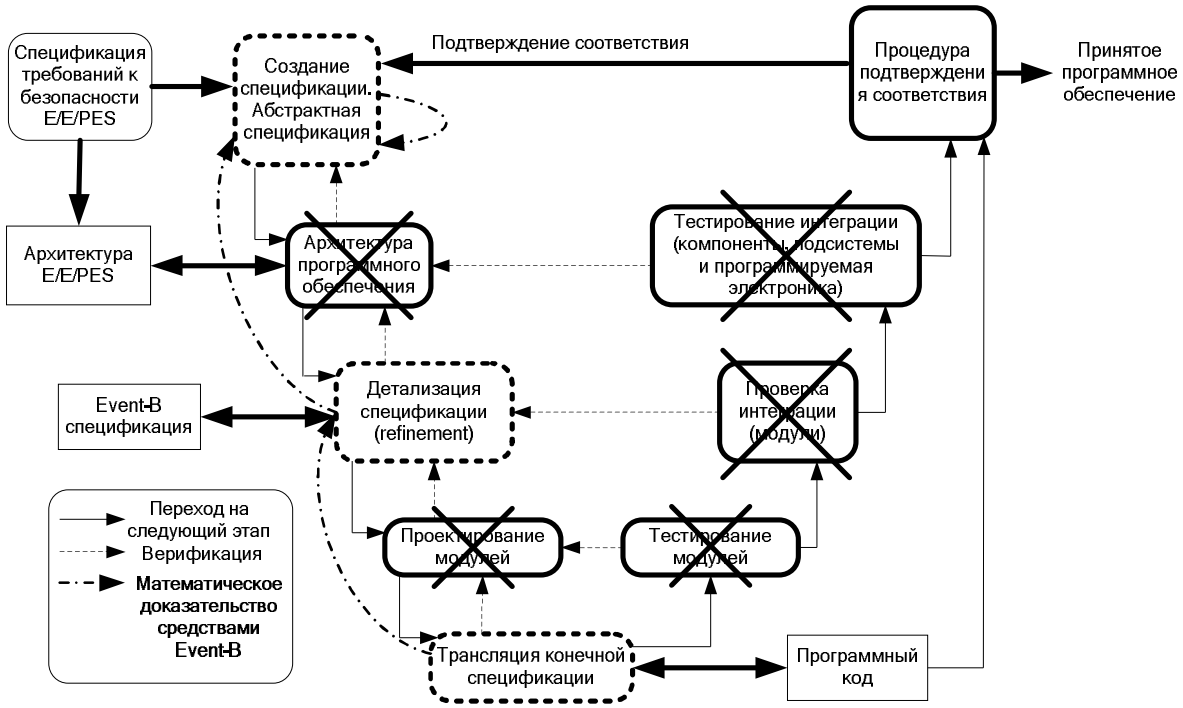


Рис. 4. Модель ЖЦ разработки ИУС с использованием Event-B

На рис. 5 показана модель ЖЦ разработки ИУС с использованием FTA и Event-B. В этом случае, в соответствии с требованиями к системе, строится дерево отказов, полученные результаты анализа дерева внедряются как в абстрактную спецификацию, так и во все последующие детализированные спецификации в Event-B. Использование методов анализа надежности в сочетании с методикой формальной разработки гарантирует корректность внедрения требований по функциональной безопасности, а также позволяет уменьшить затраты на тестирование, ограничившись только этапом подтверждения соответствия.

Применение инвариантного подхода и математического доказательства при разработке систем в Event-B позволяет создавать спецификации функционально безопасных ИУС не содержащие ошибок проектирования.

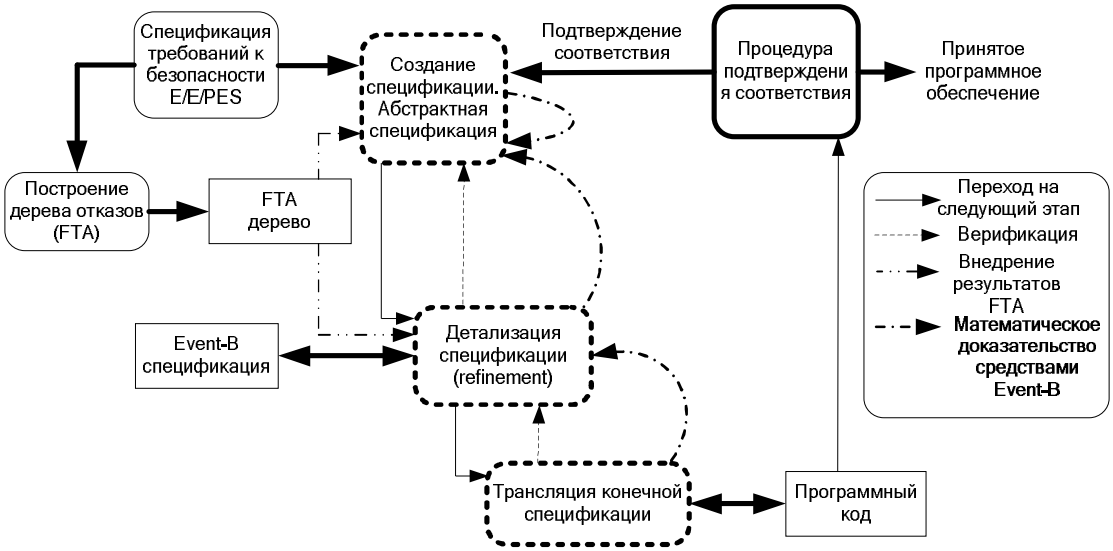


Рис. 5. Модель ЖЦ разработки ИУС с использованием метода анализа деревьев отказов и Event-B

3. Представление дерева отказов в Event-B спецификации

Рассмотрим пример обеспечения функциональной безопасности на основе анализа деревьев отказов. Анализируя дерево отказов уровень за уровнем, пошагово внедряется детализированное представление отказов и моделируется реакция системы в спецификации (рис. 6).

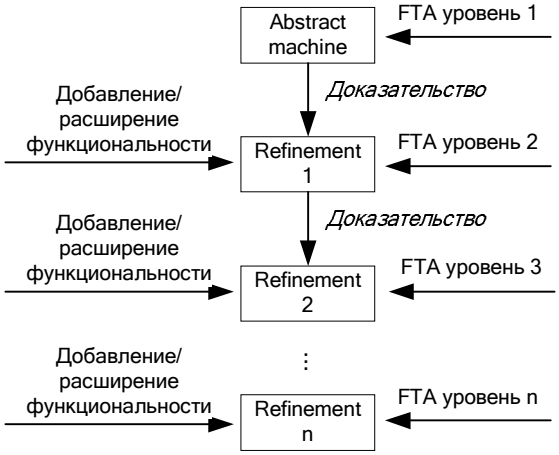


Рис. 6. Внедрение FTA в Event-B спецификацию

Результатом является спецификация системы, в которой специфицированы отказы и реакции системы на них. Кроме того, отказы обрабатываются в соответствии с их критичностью, которая определяется уровнем дерева отказов. На рис. 7 показан пример дерева отказов, где E (i,j) – событие характеризующее отказ, i – уровень дерева отказов, j - номер события.

В данном случае, возникновение события E(1,1), является самым критичным и приводит к отказу системы. На втором уровне детализации событие E(1,1) может возникнуть в случае возникновения события E(2,1) или E(2,2) в соответствии с законом E(1,1) = E(2,1) ∨ E(2,2). Третий уровень детализации характеризует принцип возникновения события E(2,1), где E(2,1) = E(3,1) ∧ E(3,2), т.е. событие E(2,1) возникнет при условии, что одновременно возникнут события E(3,1) и E(3,2).

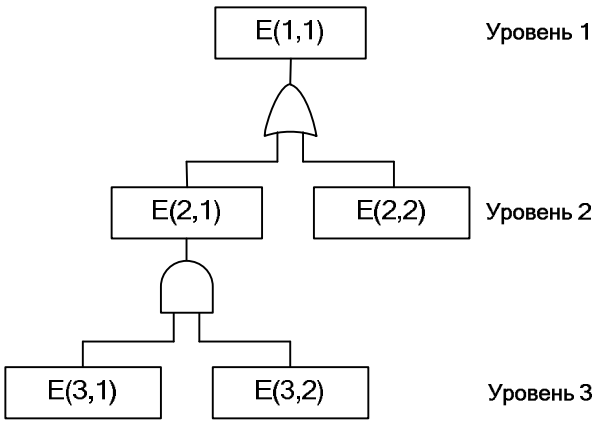


Рис. 7. Пример дерева отказов

Согласно предложенному методу абстрактная спецификация в Event-B, соответствующая первому уровню дерева отказов, (рис. 8) может иметь следующие события (events): **Operation** – корректная работа системы; **Reaction** – в случае возникновения отказа система должна быть выключена, переменная `Stop_system` принимает значение `TRUE`; **Detection** – событие, используемое для моделирования отказов в системе.

Осуществив детализацию абстрактной спецификации, мы получаем уточненную спецификацию (рис. 9), соответствующую второму уровню FTA. Зависимость E(1,1) от E(2,1) и E(2,2) задается инвариантом

@inv3 (E11 = TRUE) ⇔ (E21 = TRUE ∨ E22 = TRUE).

Для формализации операции OR дерева отказов в событии **Reaction** вводится предусловие

@grd1 E21 = TRUE ∨ E22 = TRUE.

```
machine Level_1_M
variables E11 Stop_system
invariants
 @inv1 E11 ∈ BOOL
 @inv2 Stop_system ∈ BOOL
events
 event INITIALISATION
  then   @act1 E11 ≔ FALSE
         @act2 Stop_system ≔ FALSE  end
 event Operation
  where  @grd1 E11 = FALSE
         @grd2 Stop_system = FALSE
  then   выполнение заданных функций end
 event Reaction
  where   @grd1 E11 = TRUE
         @grd2 Stop_system = FALSE
  then    @act1 Stop_system ≔ TRUE  end
 event Detection
  where  @grd1 Stop_system = FALSE
  then    @act1 E11 :∈ BOOL
 end end
```

Рис. 8. Абстрактная спецификация (Уровень 1)

```
machine Level_2_M refines Level_1_M
variables E11 Stop_system E21 E22
invariants
 @inv1 E21 ∈ BOOL
 @inv2 E22 ∈ BOOL
 @inv3 (E11 = TRUE) ⇔
       ⇔ (E21 = TRUE ∨ E22 = TRUE)
events
 event INITIALISATION extends
 INITIALISATION
   then   @act3 E21 ≔ FALSE
          @act4 E22 ≔ FALSE  end
 event Operation refines Operation
  where @grd1 ¬ (E21 = TRUE ∨ E22 = TRUE)
          @grd2 Stop_system = FALSE
  then   выполнение заданных функций end
 event Reaction refines Reaction
  where @grd1 E21 = TRUE ∨ E22 = TRUE
          @grd2 Stop_system = FALSE
  then   @act1 Stop_system ≔ TRUE  end
 event Detection refines Detection
  where @grd1 Stop_system = FALSE
  then   @act1 E21 :∈ BOOL
          @act2 E22 :∈ BOOL
 end end
```

Рис. 9. Уточненная спецификация (Уровень 2)

Спецификация, соответствующая третьему уровню дерева отказов и формализующая операцию AND, показана на рис. 10.

Зависимость E(2,1) от E(3,1) и E(3,2) и E(1,1) от E(2,1) и E(2,2) задается инвариантом

@inv3 (E21 = TRUE) ⇔ (E31 = TRUE ∧ E32 = TRUE).

Поскольку возникновение единичного отказа E(3,1) или E(3,2) не приводит к отказу системы, спецификация содержит дополнительные события **Rescue_E31** и **Rescue_E32**. Если требования к системе содержат алгоритмы парирования данных отказов, они могут быть реализованы в этих событиях.

```
machine Level_3_M refines Level_2_M
variables E11 Stop_system E21 E22 E31 E32
invariants
 @inv1 E31 ∈ BOOL
 @inv2 E32 ∈ BOOL
 @inv3 (E21 = TRUE) ⇔
       ⇔ (E31 = TRUE ∧ E32 = TRUE)
events
 event INITIALISATION extends
 INITIALISATION
  then   @act5 E31 ≔ FALSE
         @act6 E32 ≔ FALSE  end
 event Operation refines Operation
  where  @grd1 E31 = FALSE
         @grd2 E32 = FALSE
         @grd3 E22 = FALSE
         @grd4 Stop_system = FALSE
  then   выполнение заданных функций  end
 event Rescue_E31 refines Operation
  where  @grd1 E31 = TRUE
         @grd2 E32 = FALSE
         @grd3 E22 = FALSE
         @grd4 Stop_system = FALSE
  then   выполнение алгоритма парирования
отказа, приведшего к возникновению события
E31  end
 event Rescue_E32 refines Operation
  where  @grd1 E31 = FALSE
         @grd2 E32 = TRUE
         @grd3 E22 = FALSE
         @grd4 Stop_system = FALSE
  then   выполнение алгоритма парирования
отказа, приведшего к возникновению события
E32  end
  event Reaction refines Reaction
  where  @grd1 (E31 = TRUE ∧ E32 = TRUE) ∨
          ∨ E22 = TRUE
         @grd2 Stop_system = FALSE
  then    @act1 Stop_system ≔ TRUE  end
 event Detection refines Detection
  where  @grd1 Stop_system = FALSE
  then    @act2 E22 :∈ BOOL
          @act3 E31 :∈ BOOL
          @act4 E32 :∈ BOOL
 end end
```

Рис. 10. Уточненная спецификация (Уровень 3)

4. Event-B спецификация ПОС

Примером использования предлагаемого метода служит противообледенительная система (ПОС) самолета АН-140М [14].

На рис. 11 представлено упрощенное дерево отказов ПОС, состоящее из четырех уровней детализации.

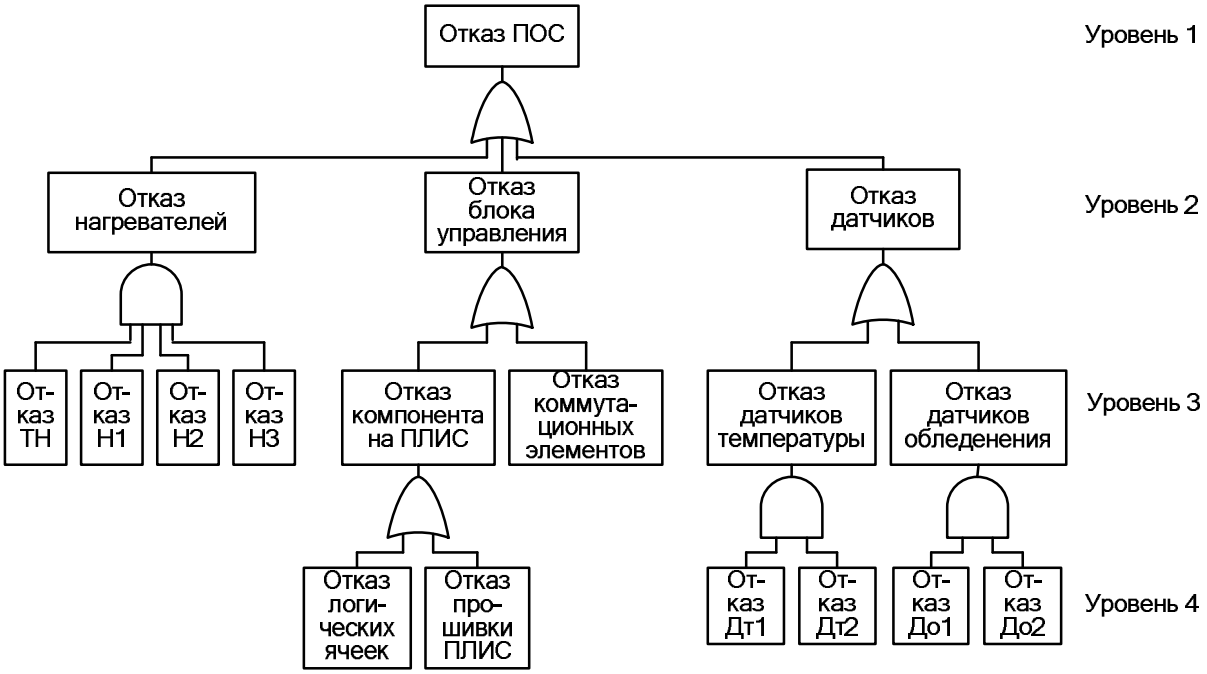


Рис. 11. Дерево отказов ПОС

Примеры спецификации ПОС для 1-3 уровней дерева отказов представлены на рис. 12-14. Все переменные спецификации, которые обозначают наличие отказа, имеют тип BOOL и инициализируются в событии **INITIALISATION** значением FALSE.

События **Operation**, **Reaction** и **Detection** являются аналогичными для всех шагов детализации.

Событие **Operation**, корректная работа системы, произойдет в случае, если в системе нет ни одного отказа, и будет выполнять основной алгоритм работы системы. Событие **Reaction** является реакцией на возникновение любого отказа, приводящего к отказу системы, а также формирует диагностический сигнал об отказе системы. **Detection** – событие, используемое для моделирования отказов в системе.

По аналогии с примером, описанным в разделе 3, внедрение результатов анализа дерева отказов начинается с первого уровня, в данном случае это «Отказ ПОС» (рис. 11). Переменная IPS_failure в абстрактной спецификации системы (рис. 12) принимает значение TRUE в случае возникновения данного отказа, а переменная IPS_failure_D является диагностическим сигналом информирующем об отказе системы.

На втором уровне детализации (рис. 13) «Отказ ПОС» возникнет, если произойдет один из следующих отказов: «Отказ нагревателей» - переменная Heaters_fault, «Отказ блока управления» - переменная Control_unit_fault, «Отказ датчиков» - переменная Sensors_fault. В формальной спецификации данная зависимость будет представлена инвариантом:

@inv4 (IPS_failure = TRUE) ⇔ (Heaters_fault = TRUE ∨ Control_unit_fault = TRUE ∨ Sensors_fault = TRUE).

```
machine IPS_L_1_M
variables IPS_failure IPS_failure_D
invariants
  @inv1 IPS_failure ∈ BOOL
  @inv2 IPS_failure_D ∈ BOOL
events
  event Operation
    where  @grd1 IPS_failure = FALSE
           @grd2 IPS_failure_D = FALSE
    then   выполнение алгоритма работы end
  event Reaction
    where  @grd1 IPS_failure = TRUE
           @grd2 IPS_failure_D = FALSE
    then   @act1 IPS_failure_D ≔ TRUE  end
  event Detection
    when   @grd1 IPS_failure_D = FALSE
    then   @act1 IPS_failure :∈ BOOL  end end
```

Рис. 12. Абстрактная спецификация ПОС (Ур. 1)

Третий уровень дерева отказов характеризуется наличием операции AND. «Отказ нагревателей» произойдет только в том случае, если откажут все нагревательные элементы, т.е. произойдет отказ теплового ножа и трех нагревателей. В формальной

спецификации (рис. 14) - это переменные Heater_knife_fault, Heater1_fault, Heater2_fault, и Heater3_fault.

```
machine IPS_L_2_M refines IPS_L_1_M
variables IPS_failure IPS_failure_D Heaters_fault
Control_unit_fault Sensors_fault
invariants
  @inv4 (IPS_failure = TRUE) ⇔ (Heaters_fault =
TRUE ∨ Control_unit_fault = TRUE ∨ Sen-
sors_fault = TRUE)
events
 event Operation refines Operation
   where  @grd1 ¬(Heaters_fault = TRUE ∨ Con-
trol_unit_fault = TRUE ∨ Sensors_fault = TRUE)
          @grd2 IPS_failure_D = FALSE  end
 event Reaction refines Reaction
   where  @grd1 Heaters_fault = TRUE ∨ Con-
trol_unit_fault = TRUE ∨ Sensors_fault = TRUE
          @grd2 IPS_failure_D = FALSE
   then   @act1 IPS_failure_D ≔ TRUE  end end
```

Рис. 13. Уточненная спецификация ПОС (Ур. 2)

```
machine IPS_L_3_M refines IPS_L_2_M
variables IPS_failure IPS_failure_D Heaters_fault
Control_unit_fault Sensors_fault Hea-
ter_knife_fault Heater1_fault Heater2_fault Hea-
ter3_fault FPGA_fault Switchers_fault Sen-
sor_temp_fault Sensor_ice_fault Heaters_fault_D
invariants
  @inv5 (Heaters_fault = TRUE) ⇔ (Hea-
ter_knife_fault = TRUE ∧ Heater1_fault = TRUE ∧
Heater2_fault = TRUE ∧ Heater3_fault = TRUE)
  @inv8 (Control_unit_fault = TRUE) ⇔
(FPGA_fault = TRUE ∨ Switchers_fault = TRUE)
  @inv11 (Sensors_fault = TRUE) ⇔ (Sen-
sor_temp_fault = TRUE ∨ Sensor_ice_fault =
TRUE)
events
 event Not_All_Heaters_Fault refines Operation
   where  @grd1 FPGA_fault = FALSE
          @grd2 Switchers_fault = FALSE
          @grd3 Sensor_temp_fault = FALSE
          @grd4 Sensor_ice_fault = FALSE
          @grd5 ¬ (Heater1_fault = TRUE ∧
Heater2_fault = TRUE ∧  Heater3_fault = TRUE ∧
Heater_knife_fault = TRUE)
          @grd6 ¬ (Heater1_fault = FALSE ∧
Heater2_fault = FALSE ∧  Heater3_fault = FALSE
∧ Heater_knife_fault = FALSE)
          @grd7 IPS_failure_D = FALSE
   then   @act1 Heaters_fault_D ≔ TRUE
          выполнение алгоритма
          парирования отказов end end
```

Рис. 14. Уточненная спецификация ПОС (Ур. 3)

Инвариант **@inv5 (Heaters_fault = TRUE) ⇔ (Heater_knife_fault = TRUE ∧ Heater1_fault = TRUE ∧ Heater2_fault = TRUE ∧ Heater3_fault = TRUE)** описывает данную зависимость, а в спецификации добавляется новое событие **Not_All_Heaters_Fault**, которое может быть использовано для внедрения алгоритма парирования отказов. В этом случае формируется сигнал об отказе нагревательных элементов, но система продолжает функционировать. Аналогичным образом создается спецификация следующего уровня детализации.

Следует отметить, что с ростом сложности системы растет количество и сложность событий, а также сложность самой спецификации. Для борьбы с этим недостатком может быть применена концепция coordinated atomic actions (CAAs) [15].

Заключение

В статье предложены модели ЖЦ разработки ИУС с учетом использования методов формальной спецификации и верификации и метода анализа деревьев отказов. Предложен метод пошагового внедрения результатов FTA в Event-B спецификацию, использование которого позволяет создавать спецификации функционально безопасных систем критического применения не содержащие ошибок проектирования. На примере противообледенительной системы самолета АН-140М показано практическое использование предлагаемого подхода к разработке информационно-управляющих систем.

В рамках дальнейших исследование для повышения эффективности и масштабируемости предложенного метода для более сложных ИУС планируется использование концепции CAAs.

Литература

1. Abrial J. R. The B Book: Assigning Programs to Meanings / J.R. Abrial. - Cambridge University Press, 1996.

2. Fault Tree Analysis Information [Электронный ресурс] – Режим доступа http://www.fault-tree.net/papers/ericson-fta-tutorial.pdf.

3. FMEA Info Centre [Электронный ресурс] – Режим доступа http://www.fmeainfocentre.com/.

4. Ortmeier F. Combining formal methods and safety analysis – the ForMoSA approach / F. Ortmeier, A. Thums, G. Schellhorn, W.Reif // Integration of Software Specification Techniques for Applications in Engineering. Springer LNCS 3147, 2004. – P. 474-493.

5. Sere K. Safety analysis in formal specification / K. Sere, E. Troubitsyna // FM'99 – Formal Methods. Proceedings of World Congress on Formal Methods in the Development of Computing Systems, vol. II, Lecture Notes in Computer Science 1709. – 1999. - P. 1564-1583.

6. Troubitsyna E. Integrating Safety Analysis into Formal Specification of Dependable Systems / E. Troubitsyna // Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03). – 2003. – P. 215.2.

7. Troubitsyna E. Elicitation and Specification of Safety Requirements / E. Troubitsyna // Proceedings of the Third International Conference on Systems (ICONS 2008). – 2008. - P. 202-207.

8. Tarasyuk O. Principles of Formal Methods Integration for Development Fault-Tolerant Systems: Event-B and FME(C)A / O. Tarasyuk, A. Gorbenko, V. Kharchenko // MASAUM Journal of Computing. - Vol. 1. - No. 3. – 2009. – P. 423-429.

9. ClearSy System Engineering [Электронный ресурс] – Режим доступа http://www.clearsy.com/pdf/ClearSy-Industrial_Use_of_%20B.pdf.

10. Abrial J. R. Extending B without changing it (for developing distributed systems) / J. R. Abrial in H. Habiras edit // First Conference on the B method, IRIN. - 1996. – P. 169-190.

11. Abrial J. R. A Summary of the Event-B Modeling Notation. Presentation / J. R. Abrial // http://deploy-eprints.ecs.soton.ac.uk/53/1/sld_evtb.pdf. - 2008. - 34 p.

12. Schneider S. The B-Method: An Introduction / S. Schneider // Palgrave. Cornerstones of Computing series, 2001. – 354 p.

13. IEC 61508 "Functional safety of electrical/electronic/programmable electronic safety-related systems".

14. Локазюк В.Н. Отказоустойчивые встроенные системы на программируемой логике. Лекционный материал / В.Н. Локазюк, С.Б. Остроумов, О.В. Поморова, Ю.Н. Прохорова, А.А. Ушаков, В.С. Харченко; под ред. В.С. Харченко // МОН Украины, Национальный аэрокосмический университет «ХАИ», 2008. – 264 с.

15. Gallina B. Coordinated Atomic Actions for Dependable Distributed Systems: the Current State in Concepts, Semantics and Verification Means / B. Gallina, N. Guelfi, A. Romanovsky // Proceedings of the The 18th IEEE International Symposium on Software Reliability. – 2007. - P. 29-38.

Поступила в редакцию 12.02.2010

Рецензент: д-р техн. наук, проф., зав. кафедрой компьютерных систем и сетей В.С. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

ВИКОРИСТАННЯ МЕТОДУ АНАЛІЗУ ДЕРЕВ ВІДМОВ ДЛЯ СТВОРЕННЯ СПЕЦИФІКАЦІЙ ФУНКЦІОНАЛЬНО БЕЗПЕЧНИХ СИСТЕМ У EVENT-B

Ю.М. Прохорова

Запропоновано метод, що дозволяє об'єднати метод аналізу дерев відмов (FTA) та формальну специфікацію у Event-B. Використання методики формальної розробки гарантує коректність впровадження вимог до функціональної безпеки до специфікації систем критичного застосування. Запропонований підхід базується на поетапному впровадженні результатів аналізу за принципом «зверху вниз» на кожному етапі деталізації специфікації, починаючи з абстрактної та закінчуючи реалізацією. В статті також наведено модифіковані моделі життєвого циклу розробки програмного забезпечення інформаційно-керуючих систем згідно зі запропонованим методом. Як приклад впровадження FTA в формальну специфікацію було використано протикригову систему літака АН-140М.

Ключові слова: методи формальної специфікації та верифікації, Event-B, функціональна безпека, FTA.

AN APPLICATION OF FAULT TREE ANALYSIS TECHNIQUE TO CREATE SPECIFICATIONS OF SAFE SYSTEMS IN EVENT-B

Yu.N. Prokhorova

In this paper we propose the approach which allows to combine fault tree analysis (FTA) method with Event-B specification. An application of formal development techniques ensures the correctness of safety requirements implementation into safety-critical systems specification. Our approach is based on a stepwise injection of FTA results by the use of "top down" approach at every refinement step starting from abstract level and finishing at realization one. This approach causes changes in life cycle model of software development for information management systems. We describe the modified development model in the paper. As an industrial example of FTA incorporation into formal specification we use the ice protection system of airplane AN-140M.

Key words: formal methods, Event-B, safety analysis, FTA.

Прохорова Юлия Николаевна – аспирант кафедры компьютерных систем и сетей Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: J.Prokhorova@csac.khai.edu.