Анализ AADL моделей с помощью графического представления

Александр Страх
Институт Системного Программирования
Российской Академии Наук,
Email: strakh@ispras.ru

***Аннотация*— На ранних стадиях проектирования программно-аппаратных систем предпочтительнее использовать визуальную верификацию графического представления проектной модели, которую должен проводить специалист в предметной области. В статье описаны основные инструменты для выполнения анализа графического представления, показаны примеры выявления аномалий или потенциальных ошибок на примерах моделей AADL, обнаружение которых при традиционном подходе было бы более трудоемко или почти невозможно. В заключении делаются выводы и приводятся направления дальнейшего развития.**

Ключевые слова. AADL; графическое представление; моделирование систем; авионика.

I. Введение

При проектировании программно-аппаратных систем используются упрощенные модели реальной системы. Использование опыта и знаний экспертов предметной области может существенно увеличить эффективность выявления ошибок или аномалий на ранних этапах моделирования системы.

Для верификации в автоматическом режиме обычно используется текстовое представление модели, но оно не удобно для человека. Поэтому для визуального анализа модели экспертом требуются специализированные представления, способные акцентировать внимание на свойствах или характеристиках модели важных в конкретной предметной области. Наиболее подходящим представлением в данном случае является графическое. Подробнее о преимуществах такого представления перед текстовым рассказано в разделе V.

Статья написана на основе опыта работы с языком AADL [3] в среде MASIW[9], разрабатываемой в ИСП РАН в сотрудничестве с ГосНИИАС. Стандарт AADL разработан Сообществом Автомобильных Инженеров (SAE) для описания программно-аппаратных систем. (название SAE это исторический анохронизм, работы, которые ведутся под его эгидой выходят далеко за рамки чисто "автомобильной" тематики). Следует отметить, что визуальный анализ, описываемый в данной статье, в общем случае, может применяться и к моделям, построенным с помощью других объектно-ориентированных языков.

Визуальный анализ программно-аппаратных систем представленных в виде графической модели можно рассматривать как один из видов валидации. Последующие разделы статьи построены следующим образом. Раздел III дает кратко описывает визуальное моделировании. Раздел III знакомит читателя со стандартом AADL и основными аспектами, необходимыми для дальнейшего понимания статьи. В разделе IV приведена информация о средах для работы с языком AADL. В разделе V указаны преимущества графического представления для анализа и понимания модели AADL человеком. В разделе VI описаны примеры обнаружения ошибок в модели с помощью графического представления MASIW. Раздел VII рассказывает о текущем состоянии и планах на дальнейшее развитие графического представления среды MASIW. Раздел VIII содержит выводы.

II. Визуальное программирование

Визуальное программирование - это способ создания программы для ЭВМ, путем манипулирования графическими объектами. Языки визуального программирования можно условно разделить на два типа. Природно-визуальные языки имеют непосредственное графическое выражение, для которого изначально не предусмотрено текстовое представление (UML, G в среде LavVIEW). Визуально-преобразованный язык является текстовым языком с наложенным графическим представлением. В данной статье речь идет о визуально-преобразованном языке. Такие языки имеют представление основывающееся на отображении субъектов в виде геометрических фигур и отношений между субъектами в виде линий. Расположение фигур в пространстве, позволяет акцентировать внимание человека на ключевых субъектах (расположение в центре, самый большой размер, яркий цвет) или выделять набор наиболее похожих субъектов (группировка, отображение одним размеров, одной геометрической фигурой или цветом). Наибольшую выразительность графическое представление может иметь для предметно-ориентированных языков. В частности для языков проектирования программно-аппаратных систем. В этом случае речь уже идет о визуальном моделировании. Подробнее о нем можно прочитать в работе [11]. Таким образом подходы к проектированию сложных инженерных объектов на основе чертежей, используемых в строительстве, машиностроении, энергетики переносятся на индустрию производства программного обеспе-

чения [10]. Развитие визуального моделирование привело к созданию множества CASE систем, где внимание уделяется проектированию модели, автоматической верификации и коммуникации между специалистами. Использование диаграмм как средств коммуникаций между экспертами в различных предметных областях приводит к обнаружению проблем в модели. Такой верификации модели описывается в данной статье.

III. AADL

Основное черты AADL унаследовал от языка Meta-H, предназначенного для описания бортовых систем авионики. AADL используется для моделирования встраиваемых систем реального времени, но также может быть использован для анализа и генерации кода.

Язык AADL имеет два способа расширения:

- С помощью определения пользовательских свойств
- С помощью определения языковых расширений (language annexes)

С помощью языковых расширений были написаны и стандартизованы несколько приложений языка AADL. Наиболее известные среди них: Behavior annex; ARINC653; ANNEX Error Model v.2 [1]. Каждое расширение представляет собой узкую специализацию и может анализироваться на основе внешних знаний о конкретной предметной области, в общем случае независимой от AADL.

A. Основные аспекты описания модели на языке AADL

Вначале приведем основные сведения об элементах AADL. В AADL можно выделить три типа объектов [3]: компоненты (components), порты (features) и связи (connections, flows). Компоненты могут содержать в себе порты и другие компоненты, а связи способны соединять порты и компоненты друг с другом. В свою очередь компоненты можно разделить на программные, аппаратные и композитные:

- Примеры программных: процесс (process), подпрограмма (subprogram), поток (thread), данные (data)
- Примеры аппаратных: процессор (processor), шина (bus), память (memory), устройство (device)
- Примеры композитных: системы (system)

С помощью компонентов AADL описывается архитектурная модель программно-аппаратной целевой системы. Так, система с двумя процессами и портами в текстовом виде будет выглядеть следующим образом:

```
PACKAGE test1

PUBLIC

  PROCESS inproc
    FEATURES
      inprt: IN DATA PORT;
  END inproc;

  PROCESS outproc
    FEATURES
      outprt: OUT DATA PORT;
  END outproc;

  SYSTEM tsys
  END tsys;

  SYSTEM IMPLEMENTATION tsys.isys
    SUBCOMPONENTS
      subinproc: PROCESS inproc;
      suboutproc: PROCESS outproc;
    CONNECTIONS
      connx: PORT suboutproc.outprt -> subinproc.
          inprt;
  END tsys.isys;

END test1;
```

В коде описан тип системы tsys и ее реализация tsys.isys. В свою очередь реализация системы tsys.isys содержит в себе подкомпоненты (SUBCOMPOENTS): процесс inproc и процесс outproc. Процессы inproc и outproc содержат в себе порты, которые описаны в соответствующих секциях FEATURES. А в секции CONNECTIONS системы tsys.isys описано соединение между портами процессов.

B. Графическая нотация

Стандарт AADL помимо текстовой нотации частично описывает и графическую. В стандарте показано как должны выглядеть основные компоненты, соединения, порты. Этого описания не достаточно для построения полноценного графического представления и отображения всего того, что может быть описано в текстовом виде. Однако, этого вполне достаточно, чтобы человеку, работающему с редакторами на базе стандарта, не приходилось тратить время на переобучение переходя от одной среды AADL к другой.

IV. Графические среды для работы с моделями AADL

На сегодняшний день существует немало инструментов выполняющих ту или иную задачу для работы с AADL. Мы будем рассматривать только те инструменты, которые удовлетворяют следующим требованиям:

- имеют текстовый редактор
- имеют графический редактор
- имеют средства автоматического анализа
- расширяемы

Среди открытого ПО следует выделить три среды, которые отвечают этим требованиям: TOPCASED [8], OSATE [7] и MASIW [9]. В таблице I приведено сравнение графических представлений AADL модели сред OSATE и MASIW. На момент написания статьи TOPCASED не имел стабильной версии графического представления, поэтому не будет представлен в сравнении. В таблице I видно, что по указанным характеристикам выигрывает графическое представление среды

Таблица I. Сравнение графических представлений MASIW и OSATE

Параметр	OSATE	MASIW
Просмотр	+	+
Редактирование	+	+
Открытие без подготовки модели	-	+
Открытие отдельного компонента	+	+
Подсветка ошибок	-	+
Переход к декларации	+	+
Автоматическая раскладка компонентов	+/-	+/-
Автоматический роутинг соединений	-	+/-
Поддержка работы с соединениями	+	+
Поддержка работы с биндингами	-	+
Поддержка работы с потоками	+	+/-
Просмотр AADL свойств	+	+
Редактирование AADL свойств	-	+
Поддержка режимов	+	-

MASIW. Поэтому мы будем его использовать в дальнейшем как наиболее удобное и информативное для анализа целевой системы в "собраном" виде.

A. MASIW

MASIW - Modular Avionics System Integrator Workspace. Проект разработки инструментальных средств поддержки интегрированной модульной авионики на базе стандарта AADL. Проект разрабатывается ИСП РАН совместно с ГосНИИАС. Основными задачами проекта являются: развитие методов моделирования и верификации комплексных программно-аппаратных систем и разработка инструментария для проектировщиков и интеграторов. На сегодняшний день средства MASIW позволяют решать задачи создания, редактирования, анализа и управления моделями на языке AADL, а также задачи синтеза и генерации исходного кода. В состав входят два графических представления:

- Графический редактор модели AADL
- Специализированное графическое представление AFDX расширения AADL

V. Преимущества графического представления AADL модели перед текстовым

Под "ручным"анализом модели AADL подразумевается анализ текстового или графического представления человеком. Графическое представление AADL модели имеет ряд преимуществ перед текстовым.

Стандарт AADL предоставляет возможность структурировать информацию о модели по пакетам. Среда работы с AADL также предоставляют возможность структурировать информацию по файлам и папкам. Помимо этого в проекте могут быть использованы сторонние библиотеки. Таким образом исчерпывающая информация, необходимая для анализа модели в ручном режиме, может находиться в разных местах, что сильно усложняет анализ модели как единого целого. Графическое представление позволяет показать пользователю всю информацию о модели как единую целую систему, что значительно упрощает восприятие и анализ человеком.

AADL описывает модель в декларативном виде с использованием механизмов наследования и переопределения. Такой способ описания позволяет представить модель в компактном виде и избежать дублирования кода. Но с точки зрения анализа текстового представления в ручном режиме этот подход имеет серьезный недостаток. Отсутствует представление компонента в "собранном"виде. Графическое представление решает эту проблему, отображая конечный вид компонента, в котором учитываются все наследования и переопределения.

Графическое представление не содержит лишнюю для восприятия человеком информацию. К примеру, текстовое представление содержит информацию, необходимую для интерпретатора. Эта информация усложняет анализ модели человеком.

Для анализа графического представления нет необходимости иметь навыки программирования. Таким образом, анализ структуры на соответствие документации может производить человек, не знакомый со стандартом AADL.

Графическое представление позволяет отображать модель в гораздо более компактном виде, чем в текстовом. К примеру, модель, показанная на рис. 1 в текстовом виде занимает 1208 строк. Как видно из рис. 1 графическое отображение условно умещается на одном экране монитора.

С точки зрения инженера, графическое представления визуально более близко к реальной модели, чем текстовое. Так, например, связи всегда отображаются линиями, что делает похожим модель соединения на кабель реальной модели. Наличие такой дополнительной информации существенно облегчает визуальный поиск компонентов человеком и определение наличия тех или иных компонентов еще до запуска автоматической верификации.

Возможность представить "сквозную"информацию о компонентах в одном месте. Так например, стандарт AADL описывает особый вид связи, называемый "поток". Потоки могут проходить через последовательность компонентов системы, которые в текстовом виде описаны в разных местах. В графическом представлении поток может быть отображен в виде одной линий, проходящей через последовательность компонентов.

Графический редактор имеет возможность уже на этапе проектирования модели физически запретить создавать некорректные компоненты.

Безусловно, графическое представление имеет и недостатки, которые усложняют процесс ручного анализа модели. Перечислим некоторые из них. Отсутствие удобного интерфейса и четких требований построения представления для конкретного эксперта. Быстрота создания нового представления. Стоит отметить, что графическое представление не стандартизовано, в отличие от текстового. Поэтому графический анализ одной и той же модели разными инструментами может усложняться. Трудность определения "правильной"или удобной раскладки с точки зрения пользова-

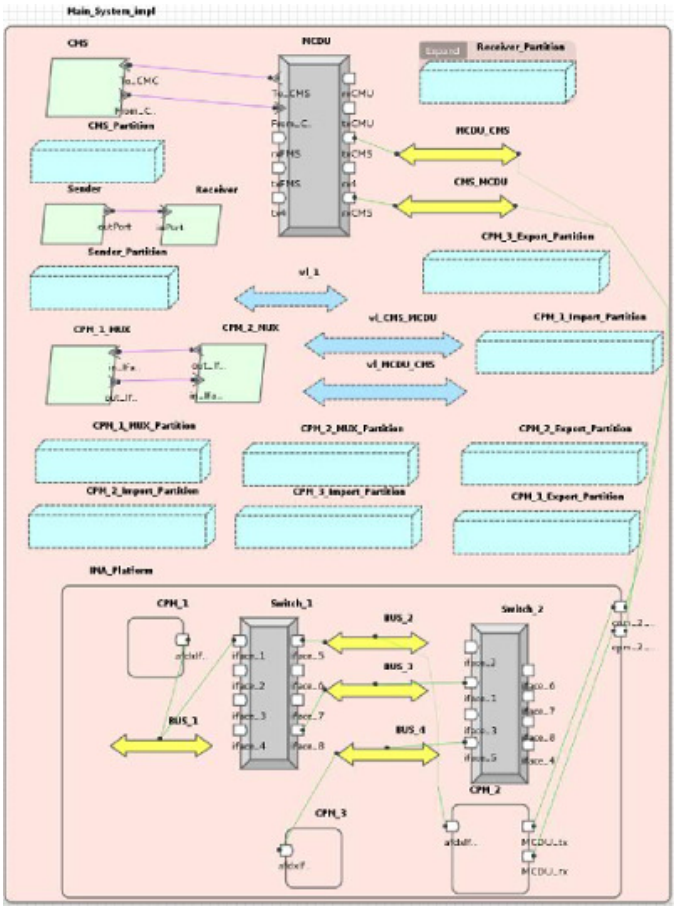

Рис. 1. Отображение модели AADL, текстовое представление которой занимает 1208 строк

теля в автоматическом режиме. Проблема восприятия диаграммы, которая не умещается на экране монитора. На момент написания статьи, графические представления в различных инструментах присутствовали в сыром для человеческого восприятия виде. Эти и другие недостатки препятствуют повсеместному использованию графического представления.

VI. Применение графического редактора MASIW для обнаружения проблем в моделях AADL

Типы ошибок, выявляемые с помощью графического представления разнообразны, поскольку могут быть определены на разных этапах проектирования системы. Приведем несколько примеров. Во всех случаях использовалось графическое представление модели, созданное с помощью графического редактора AADL модели среды MASIW. Раскладка компонентов выполнялась вручную при отсутствии каких-либо экспертных знаний о модели, за исключением случаев, указанных явно.

Предположим, что модель уже написана на языке AADL. И написана семантически корректно. Перед переходом на следующий этап моделирования инженер может посмотреть на графическое представление описанной модели исходя из собственных знаний о реальной системе и выявить различного рода несоответствия. В данном случае инженер является экспертом, а проблемы могут быть обнаружены исходя из его знаний о своей предметной области.

В открытом доступе существует библиотеки общих компонентов AADL. Человек, который их использует, часто обладает достаточным объемом знаний в предметной области и способен выполнить предварительные проверку. Так в библиотеке AADLib описан пример модели FMS (Flight Managment System [4]). FMS - это прибор управления полетом, который стоит в кабине летательного аппарата, отображает различную информацию о плане полета и имеет интерфейс состоящий из клавиатуры и дисплея, как изображено на рисунке 2.

Рис. 2. Прибор управления полетом в кабине пилота

Инженер, проектирующий системы авионики обязан знать, как устроен прибор FMS. Поэтому при открытии модели прибора FMS из библиотеки AADLib см рис. 3 он обнаружит, что у дисплея (display) отсутствуют соединение с контроллером BUS_CAN. Тем более это очевидно, исходя из симметрии графического представления модели. У клавиатуры, с другой стороны, присутствует соединение с шиной контроллера. Это

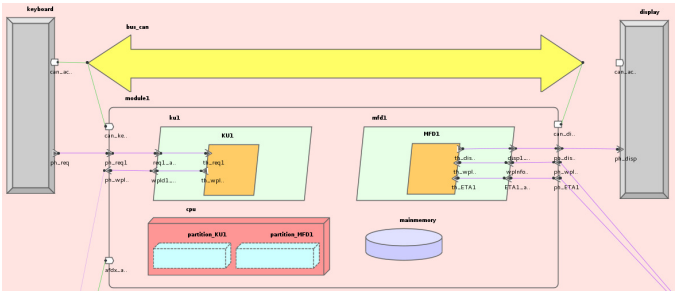

Рис. 3. Отсутствует соединение дисплея с шиной контроллера BUS_CAN

пример реальной ошибки, которая впоследствие была подтверждена и исправлена автором библиотеки. После исправления модель FMS выглядит, как показано на

рис. 4. Видно, что у клавиатуры и дисплея из портов

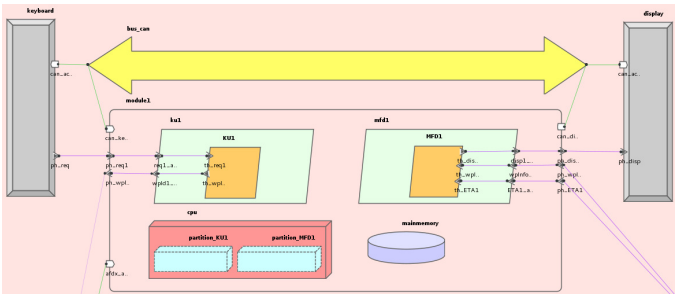

Рис. 4. Модель системы управления полета из библиотеки AADLib после исправления

под названиями "can_access"идут соединения к шине контроллера BUS_CAN. Человек, который обнаружил ошибку, имел общие знания в области устройств авионики и знания AADL, но обнаружил ошибку только после того как открыл модель в графическом представлении. В текстовом представлении полное описание модели состоит из 10 файлов суммарно 802 строчки, что значительно замедляет поиск ошибки и практически исключает обнаружение методом перебора тестовых вариантов.

Даже при отсутствии знаний о реальной системе в графическом представлении можно выполнить формальную инспекцию модели нацеленную на проверку полноты. В результате могут быть выявлены "аномалии"модели. Это может быть подозрительное отсутствие связей между компонентами, отсутствие самих компонентов, там, где есть, казалось бы, все условия для того, чтобы компонент или связь была. На рис. 5 показана система, которая состоит из процессора и памяти. Никаких других компонентов нет. При этом у памяти и процессора есть порты для связи. Логично предположить, что отсутствие соединений в этой системе скорее всего является ошибкой.

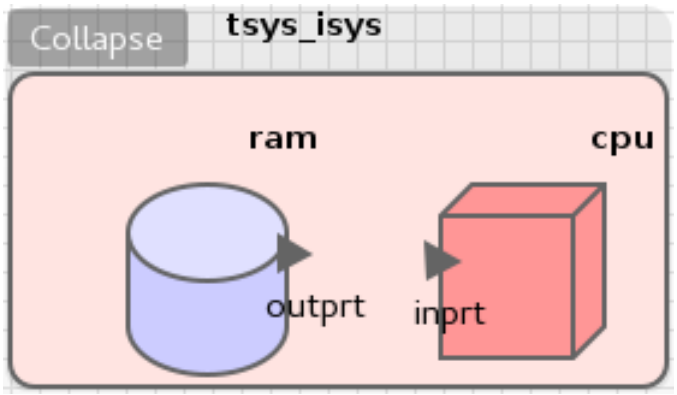

Рис. 5. Отсутствие связи между процессором и памятью может быть ошибкой

Рассмотрим более очевидный случай. В системе уже присутствуют компоненты с похожей связью, а наличие идентификаторов неявно указывают на присутствие ошибки рис. 6.

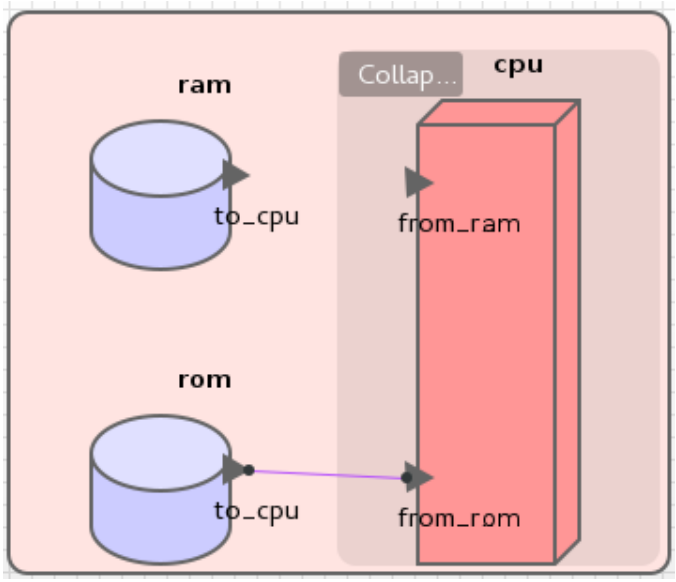

Рис. 6. Отсутствие второй связи

Иногда, характерными признаками наличия ошибки является отличие одного компонента от остального набора идентичных компонентов. Или несоответствие отображения компонента и информации в идентификаторе этого компонента. Так на рис. 7 показан набор процессоров, представление одного из которых заметно отличается от остальных без особых на то очевидных причин, при этом идентификатор "выделяющегося"компонента говорит о том, что компонент должен бы быть отображен как процессор, а отображается как процесс. Если человек, который анализирует графическое представление с рис. 7 знает язык AADL, то он поймет, в чем могла быть ошибка. Декларация процессора и процесса отличается на две последние буквы. И в этом случае при написании модели очень легко допустить ошибку, которая в текстовом виде практически

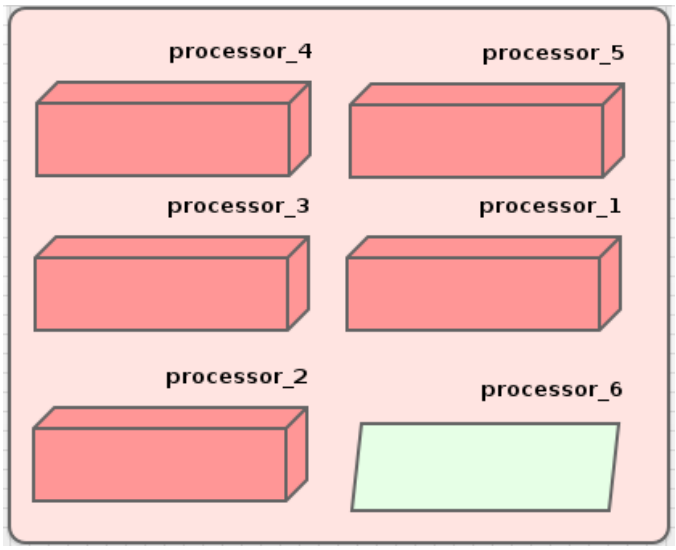

Рис. 7. Не верная декларация шестого процессора

не видна, а в графическом становится очевидной.

Иногда для того чтобы быстро определить, есть ли в структуре модели ошибки, достаточно визуально сравнить реальную систему и графическое представление модели.

Предварительный анализ графического представления может упростить обнаружение ошибок в "байндингах". "Байндинг" это связь аппаратной и программной части модели. Так, например, если в системе существует процесс и процессор, то с помощью "байндинга"мы указываем, какой процесс, на каком процессоре будет выполняться. Дело в том, что "байндинги" в AADL спецификации определяются как свойства компонентов. Анализируя текст визуально очень трудно определить корректность, отсутствие или присутствие "байндинга". В графическом представлении мы можем отобразить "байндинги" удобным нам способом. Так в MASIW

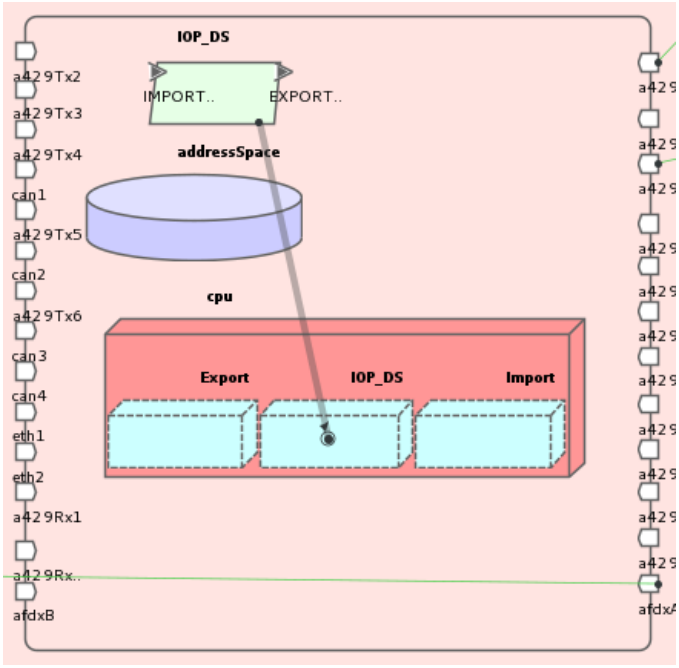

Рис. 8. "Байндинг"процесса

"байндинги" отображаются в виде линий специального стиля. На рисунке рис. 8 показан серой толстой линией баиндинг процесса IOP_DS на виртуальный процессор IOP_DS (внутри CPU). Т.е. процесс IOP_DS будет выполняться на виртуальном процессоре IOP_DS процессора CPU.

В инструменте MASIW реализовано специализированное графическое представление AADL моделей AFDX сетей. Алгоритм работы и представления компонентов редактора AFDX практически ничем не отличается от редактора AADL. За исключением того, что в AFDX присутствуют связи по крайней мере тройного уровня вложенности. Так связь самого верхнего уровня должна состоять из непрерывной последовательности соединений более низкого уровня. Графическое представление имеет возможность включать или отключать связи разных уровней, что позволяет обнаружить отсутствие промежуточных связей. Так на рис. 9 изображена AFDX модель на которой показываются связи всех уровней. Если мы отключим все уровни соедине-

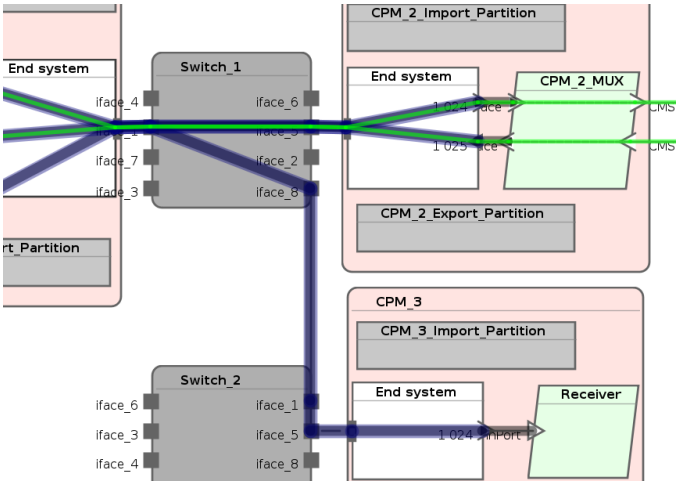

Рис. 9. AFDX модель - включены все слои соединений

ний кроме самого низкого, то увидим, что этих соединений не достаточно для того, чтобы провести соединения более высокого уровня. Так, например, в компонентах End system, Switch1 и Switch2 отсутствуют внутренние соединения вообще (см. рис. 10), что является недоработкой модели. AFDX представление является случаем

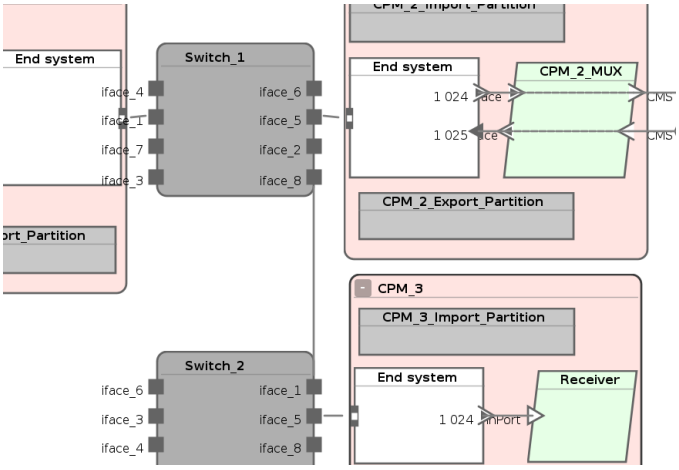

Рис. 10. AFDX модель - отображается самый низкий уровень связей

представления модели в виде, удобном для анализа экспертом AFDX сетей. Специализация помогает эффективно применять знания предметной области для выявления потенциальных проблем.

Из примеров видно, что часть потенциальных ошибок можно обнаружить практически сразу, а часть после сравнения с реальной системой. Безусловно большинство из этих проблем можно определить и с использованием формального анализа. Но на практике инструменты описания требований и автоматической верификации либо не слишком гибкие и не позволяют

описать все требования, либо требуют знаний в области программирования и самого инструмента. Поэтому, часто простой поиск проблем полноты системы будет быстрее выполнить с помощью визуального анализа.

VII. Текущее состояние и перспективы дальнейшего развития

На сегодняшний день состояние разработок графических редакторов AADL модели позволяет полноценно отображать основные сущности модели, а также почти полностью редактировать и создавать модели. Но все же есть некоторые общие для всех разработок слабые места, препятствующие визуальному восприятию AADL модели. Наиболее важные из них описаны ниже.

При открытии AADL модели в графическом редакторе компоненты могут накладываться друг на друга или находится в неудобных для восприятия местах. Поэтому необходима качественная раскладка компонентов. Наиболее известные редакторы для раскладки компонентов используют базовый функционал одного и того же фреймворка GEF [6], [12]. Однако, такая решение годится только для того чтобы избавиться от первичного наложения компонентов друг на друга. В остальном задача перекладывается на пользователя, который вручную назначает расположение компонентов.

Еще одно слабое место - это маршрутизация соединений. Без хорошего алгоритма маршрутизации соединения модели могут пересекать друг друга и компоненты. При большом количестве соединений графическое представление становится практически нечитаемым. Частичное решение также предоставляется базовым функционалом фреймворка GEF [6], [12], а в остальном перекладывается на пользователя.

Некоторые расширения AADL стандартизованы, активно используются и могут описывать модель системы из более узкой предметной области. Только эксперт в такой области может заметить аномалии. Поэтому представляется важным реализовать возможность создания узкоспециализированных графических представлений. А анализ информации был бы более эффективен, если представление можно было бы менять в зависимости от специфики анализа. По этой причине в дальнейших планах — предоставить возможность быстро создавать собственное графические представление, с минимальным участием программиста. Работы в этом направлении уже начаты в рамках проекта MASIW в ИСП РАН.

VIII. Выводы

В данной статье на небольших примерах было показано, что использование графического представления, как дополнение к формальному анализу, помогает находить потенциальные ошибки или аномальные ситуации на этапе создания модели. Особенно эффективно использовать анализ графического представления там, где четкие формальные требования еще не определены или требуется проверка модели нацеленная на определение полноты. Показано, что в большинстве примеров потенциальные ошибки могут быть найдены без знаний в области программирования, а порой и без глубокого изучения AADL стандарта, что делает такой метод анализа более доступным, чем анализ текстового представления или формальный анализ.

Список литературы

[1] Д.В.Буздалов, С.В. Зеленов, Е.В. Корныхин, А.К. Петренко, А.В. Страх, А.А. Угненко, А.В. Хорошилов. *Инструментальные средства проектирования систем интегрированной модульной авионики.* Труды Института Системного Программирования, Том 26-1, 2014, с. 201-230.

[2] В. В. Кулямин, Н. В. Пакулин, О. Л. Петренко, А. А. Сортов, А. В. Хорошилов. *Формализация требований на практике.* Препринт. М.: ИСП РАН, 2006.

[3] SAE International. *Architecture Analysis & Design Language (AADL)*, SAE International Standards document AS5506B, Nov 2004, Revised Mar 2012.

[4] http://en.wikipedia.org/wiki/Flight_management_system

[5] http://www.aadl.info/aadl/currentsite/aadlannex.html

[6] http://www.eclipse.org/gef

[7] https://wiki.sei.cmu.edu/aadl/index.php/Osate_2

[8] http://www.topcased.org

[9] http://www.masiw.ispras.ru

[10] А.Павлинов, Д.Кознов, А.Перегудов, Д. Бугайченко, А. Казакова, Р. Чернятчик, Т. Фесенко, А. Иванов. *Комплекс средств разработки проблемно-ориентированных визуальных языков.* Вестник Санкт-петербургского университета, Серия 10, Информатика, № 2, 2007. С. 86-96.

[11] Кознов Д.В. *Визуальное моделирование компонентного ПО.* Канд. дис. СПб.: 2000, 82 с.

[12] А.Сорокин, Д. Кознов. *Обзор проекта Eclipse Modeling Project* // Сб. Системное программирование./ Вып. 5, под ред. А.Н.Терехова и Д.Ю.Булычева. СПб.: Изд. СПбГУ, 2010. С. 6-31.