МОДЕЛЬНО-УПРАВЛЯЕМОЕ ПРОЕКТИРОВАНИЕ АСУТП

В.И. Клепиков (ФГУП "ИТМиВТ"), С.П. Пашков (ОАО "НПП "ЭГА")

Рассматривается концепция модельно-управляемого подхода (МУП) к проектированию АСУТП энергоустановками на базе газотурбинных двигателей (АСУТП ГТЭ). В основу концепции положен язык моделирования подчиненных сетей Петри (ПСП), обеспечивающий удобство и лаконичность описания процессов, доступность для использования всеми участниками проекта, реализуемость в специализированных и универсальных инструментальных средах.

Концепция модельно-управляемого проектирования АСУТП

Благодаря прогрессу в области проектирования и производства аппаратных средств АСУТП возможности микропроцессорной техники практически не отстают от потребностей заказчика. В то же время производительность труда проектировщиков и разработчиков ПО отстает от аналогичного показателя на 10...15% в год и сейчас технологический разрыв составляет более 7 лет [1]. Программная индустрия в целом и индустрия программирования встроенных систем нуждается в настоящее время в разработке и внедрении новых концепций, позволяющих сократить имеющееся отставание.

Одним из путей, активно развиваемых в настоящее время и позволяющих сократить имеющееся отставание, является МУП к разработке встроенных систем управления [1, 2]. Ключевым моментом модельно-управляемого подхода является создание на ранних стадиях проектирования системы ее "работающей" модели. При этом как спецификация функциональных требований к системе, так и ее поведенческое описание выполняются в единой инструментальной среде и в единой для всех участников проекта графической нотации. Это позволяет проектировщикам и заказчикам на всех стадиях проекта анализировать поведение и свойства будущей системы, проверять полноту и непротиворечивость требований, подготавливать и выполнять наборы тестов. Заказчик освобождается от необходимости вникать в детали реализации на том или ином языке программирования, а проектировщик алгоритмов получает возможность формализовано или даже автоматически генерировать код исполняемой программы, лишенной ошибок, привнесенных на этапе кодирования.

МУП фокусируется на проблеме наиболее раннего исключения ошибок из проектных документов. Именно эта проблема считается главной в том, что только около 30% проектов по разработке встроенных систем заканчиваются успешно, остальные выполняются с урезанной функциональностью или выходят за отведенные рамки по срокам и стоимости. Сложность и масштабность проектов, слабая методологическая и инструментальная поддержка процессов жизненного цикла приводят к тому, что до 40% проектных ошибок, включая ошибки программирования, алгоритмические и ошибки в требованиях, обнаруживаются на этапах интеграции и внедрения систем [1].

Преимущества МУП по отношению к традиционному подходу к разработке ПО проиллюстрированы на рис. 1. При традиционном подходе основная активность процесса (в смысле трудоемкости и ответственности за качество системы) приходится на этап программной и аппаратно-программной интеграции системы. Именно на этом этапе проектировщики убеждаются, что разработанное ПО функционирует в соответствии с заложенными алгоритмами, а заказчик убеждается в том, что система выполняет все поставленные требования. Обнаружение несоответствующего поведения системы влечет за собой поиск места возникновения ошибки, ее исправление, повторную интеграцию и тестирование. Наибольшие неприятности вызывают ошибки, обнаруженные в алгоритмическом обеспечении и связанные с некорректностью описания требований к системе.

Применение МУП смещает пик активности процесса проектирования на начальные стадии, когда, не дожидаясь момента реализации программы, алгоритмист и заказчик имеют возможность убедиться в полноте и непротиворечивости поставленных требований, а также в возможности реализации системы и ее тестируемости. В результате снижаются общая трудоемкость и сроки внедрения системы.

Применение автоматической кодогенерации управляющих программ, снижая риск появления ошибок программирования, вызывает отрицательный побочный эффект, связанный с увеличением объема

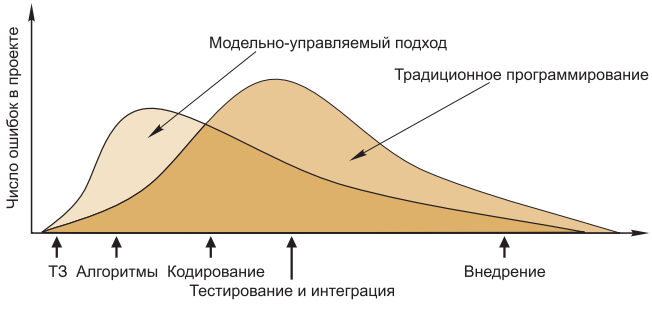

Рис. 1. Распределение числа ошибок по стадиям проекта

программы и повышением времени ее исполнения. Это ужесточает требования к аппаратным средствам системы в части объемов памяти и быстродействия процессоров, что, однако, является оправданной платой за повышение качества системы и снижение сроков ее внедрения, что в конечном счете означает снижение стоимости проекта в целом.

Развитие МУП происходит в нескольких направлениях, отражающих реальные потребности и проблемы различных областей применения встроенных систем. Это, прежде всего, разработка: подходов и методов к модельно-управляемому проектированию систем различного класса; инструментальных сред проектирования прикладных систем; автоматической кодогенерации для различных аппаратных платформ; методологий и инструментов поддержки процессов жизненного цикла и др. Среди основных направлений развития МУП можно выделить концепции "исполняемого UML", исполняемых спецификаций и модельно-управляемого тестирования [1-3].

Применительно к области АСУТП в настоящий момент не разработано эффективных и доступных инструментальных средств проектирования, основанных на концепции МУП. Основными препятствиями на этом пути являются, с одной стороны, сложность и обширность языка UML, что требует выработки определенных диалектов, ограничивающих набор используемых языковых конструкций для определенного класса систем. С другой – изначальная ориентированность UML на работу только с логическими данными и событиями вызывает проблемы по интеграции UML-спецификаций с системами и процессами, ориентированными на последовательную обработку данных (Data Flow Charts).

Наиболее полно указанные проблемы решены в комплексе продуктов компании MathWorks – Matlab/Simulink/Stateflow, позволяющем в единой инструментальной среде выполнить графическое описание и моделирование одновременно событийно-управляемых процессов, систем и процессов, ориентированных на последовательную обработку данных. Встроенные средства кодогенерации позволяют непосредственно из спроектированных моделей получать код исполняемой программы. Средства управления жизненным циклом обеспечивают специфицирование и трассируемость требований из текстовых документов в исполняемые модели. Автоматическое и полуавтоматическое тестирование позволяет оценить глубину и полноту результатов выполнения тестов.

Однако прямое применение комплекса Matlab/Simulink/Stateflow ограничено областью экспериментальных работ по моделированию и исследованию поведения систем и их компонентов. Практическое использование данного пакета для промышленной разработки ПО АСУТП ограничено отсутствием инструментария для генерации управляющих программ в стандарте IEC:61131-3, слабой поддержкой UML-нотаций для спецификации требований к системе, отсутствием средств управления изменениями в процессе проектирования, трудоемкостью организации аппаратно-программных комплексов тестирования.

На основе анализа достигнутых теоретических и практических результатов в области проектирования ПО встроенных систем можно сформулировать общие требования к методологии и инструментальным средствам проектирования АСУТП:

- для обеспечения требуемого качества системы и снижения сроков и стоимости проектов проектирование программно-алгоритмического обеспечения АСУТП должно строиться на основе МУП;
- для обеспечения взаимодействия заказчиков, проектировщиков и разработчиков системы на всех стадиях ее жизненного цикла должен использоваться единый инструментальный язык, отражающий особенности построения АСУТП;
- инструментальные средства разработки должны поддерживать работу как с событийно управляемыми процессами, так и процессами, ориентированными на последовательную обработку данных;
- методология и применяемые инструментальные средства должны обеспечивать автоматическую или строго формализованную процедуру получения исполняемых программ для требуемых аппаратных средств АСУТП, включая программы на языках стандарта IEC:61131-3;
- сопровождение жизненного цикла программно-алгоритмического обеспечения должно поддерживаться комплексом методических программных и аппаратных средств.

Сети Петри (ПСП) как язык логического проектирования и моделирования

Качество описания процессов АСУТП, трудоемкость реализации, отладки, модификации и сопровождения системы в большой степени зависят от того, насколько явно в выбранных моделях процессов отражены их свойства и наглядно выбранная модель допускает ее программную реализацию. Опыт использования универсальных методологий и соответствующих инструментальных средств показывает, что каждая специфическая область автоматизации требует привнесения в набор универсальных средств описания и моделирования процессов соответствующих специфических особенностей, наиболее полно и четко отражающих поведение данного класса систем и процессов.

Основами практически всех существующих языков моделирования дискретных систем являются конечные автоматы и сети Петри [4]. Широкий выбор теоретических платформ и инструментальных средств проектирования и моделирования подтверждает тот факт, что для решения специфических проблем конкретных областей приложений в общетеоретическую модель необходимо введение различных расширений и дополнений.

В ОАО НПП "ЭГА" при проектировании программно-алгоритмического обеспечения систем управления авиационных газотурбинных двигателей (САУ ГТД) и АСУТП энергоустановок на их основе в течение ряда лет

используется язык моделирования, получивший название подчиненных ПСП и являющийся подмножеством структурированных нейроподобных сетей [5]. В основе ПСП лежит идея о необходимости явного моделирования свойства подчиненности различных подсистем, входящих в общую иерархическую структуру системы управления. Свойство подчиненности проявляется, когда наряду с информационно-осведомительными сигналами и потоками данных, которые обрабатываются отдельными компонентами системы по своему внутреннему алгоритму, в системе циркулируют командно-управляющие потоки, которые переводят отдельные компоненты системы в определенные состояния, независимо от внутренней логики работы той или иной компоненты. Такая ситуация возникает, когда процесс функционирует относительно самостоятельно, но при этом находится под контролем обслуживающего персонала, других процессов и подсистем, или когда нормальный ход процесса нарушается изменениями во внешней среде, изменениями режимов работы, отказами оборудования. Так в САУ ГТД процессы запуска двигателя, розжига камер и другие имеют свою внутреннюю логику работы, но постоянно находятся под контролем пилота, систем обеспечения газодинамической устойчивости, систем управления полетом. В АСУТП процессы могут быть приостановлены, прекращены, модифицированы действиями оператора, системами диагностики, смежными или вышестоящими подсистемами.

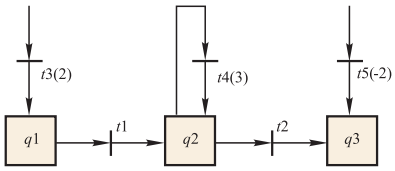

Рис. 2. Иллюстрация работы переходов ПСП

При использовании классических конечно-автоматных моделей или моделей сети Петри свойство подчиненности реализуется путем задания реакции проектируемого процесса на все возможные команды и сигналы со стороны других процессов и подсистем во всех возможных состояниях данного процесса. То есть для получения модели проектируемого процесса разработчик алгоритма должен знать, какие окружающие процессы и системы и каким образом могут скорректировать поведение данного процесса. Следствием этого является усложнение описания, увеличение сроков проектирования, повышение трудоемкости отладки и модификаций системы.

Для явного описания и реализации свойства подчиненности процессов в ПСП наряду с обычными переходами дополнительно вводятся [5] переключающие, удерживающие и блокирующие переходы. Каждый переход снабжается весовым коэффициентом, соответствующим его приоритету. Переходы активизируются только тогда, когда активны все их входные разрешающие состояния и неактивны все запрещающие.

Переключающий переход, когда он активен и его приоритет выше, чем у других активных переходов, обеспечивает активизацию состояний, являющихся для него выходными, независимо от того, существуют ли переходы в данное состояние из других активных в данный момент состояний.

Удерживающий переход, когда он активен и его приоритет выше, чем у других активных переходов, обеспечивает удержание процесса в текущем активном состоянии, даже если из этого состояния имеются другие активные переходы с меньшими приоритетами.

Блокирующий переход, когда он активен и его приоритет выше, чем у других активных переходов, запрещает активизацию своей выходной позиции, даже если существуют другие активные переходы с меньшими приоритетами, пытающиеся данную позицию активизировать.

Рис. 2 иллюстрирует работу переходов ПСП. Здесь переходы, нагруженные условиями $t1$ и $t2$, являются обычными переходами сети Петри, их вес по умолчанию равен 1. Переход, нагруженный условием $t3$, является переключающим, его вес равен 2. Переход $t4$ – удерживающий, его вес равен 3, а переход с условием $t5$ и весом равным 2 – блокирующий.

Если в данной сети переходы $t3$, $t4$, $t5$ неактивны, то ПСП работает как обычная сеть Петри. То есть, если активна позиция $q1$ (на рисунке это отмечено черным квадратиком – маркером) и выполняется условие $t1$, то активной становится позиция $q2$. Далее, если истинным становится условие $t2$, то активизируется соответствующий переход и активной становится позиция $q3$.

Однако, если в момент, когда активной является позиция $q2$, истинным окажется условие $t4$, активизируется удерживающий переход, имеющий вес 3, и даже истинность условия $t2$ не позволит сети перейти в состояние $q3$. Аналогичная ситуация возникает, когда активной является позиция $q2$ и истинными оказываются условия $t2$ и $t5$. Так как блокирующий переход, нагруженный условием $t5$, имеет больший по абсолютной величине вес, но с отрицательным знаком, то переход $t2$ не сможет сработать и перевести процесс в состояние $q3$. Но, если переход $t2$ успел перевести процесс в состояние $q3$ (т.е. условие $t5$ стало истинным позже, чем условие $t5$), то блокирующий переход $t5$ уже не сможет вывести процесс из состояния $q3$. Переключающий переход $t3$ переводит процесс в состояние $q1$ из состояния $q3$ при любой комбинации активности всех других переходов, и из состояния $q2$ – при условии, что не будет активным удерживающий переход $t4$, так как вес последнего выше, чем у перехода $t3$.

Преимущества ПСП перед классическим конечно-автоматным описанием и описанием сетью Петри выражаются в минимизации числа переходов и условий их нагружающих. В плане преимуществ проектирования ПСП позволяет "развязать" описание собственной логики работы процесса и логики его взаимодействия со смежными процессами и подсистемами. Так в рассматриваемом примере переключающий и блокирующий переходы задают командно-управляющие воздействия, поступающие из смежных подсис-

тем. Проектировщик, разрабатывая и отлаживая логику работы данного процесса, может не рассматривать влияние указанных переходов до момента интеграции процесса со смежными системами.

ПСП благодаря наглядности и выразительной эффективности с успехом применяются в НПП "ЭГА" в течение ряда лет в качестве инструмента описания алгоритмов. В зависимости от выбранной аппаратной платформы реализации системы ПСП алгоритмы формализовано переводятся на язык контекстно-свободных грамматик (GRACE), либо в типовые конструкции инструментального пакета Matlab/Simulink/Stateflow, либо на язык выбранной SCADA-системы.

Список литературы

1. UML Solutions for Embedded Systems with limited Resources // Willert Software Tools. Herminenstabe. 2005.
2. *Keller B., Matthes J., Hekker W., Schonecker H.* Reliable and efficient Software Development by Model Driven Architecture // Proceedings embedded world Conference. 2005.
3. *Долинский М.* Тенденции развития методов и средств автоматизации проектирования встроенных цифровых систем по материалам DATE '2002 //Компоненты и технологии. 2002. № 8.
4. *Питерсон Дж.* Теория сетей Петри и моделирование систем: Пер. с англ. М.: Мир. 1984.
5. *Клепиков В.И., Сосонкин В.Л.* Структурированные нейроподобные сети как средство моделирования дискретных процессов // Автоматика и телемеханика. 1998. № 1.

Клепиков Владимир Иванович *– канд. техн. наук, руководитель проектов ФГУП "ИТМиВТ им.С.А.Лебедева" РАН ,*
Пашков Сергей Петрович *– начальник отдела АСУТП ОАО "НПП "ЭГА".*
Контактный телефон (916) 213-05-71. E-mail: viklepikov@mail.ru

Сервис-ориентированная архитектура как инструмент интеграции информационного обеспечения в гетерогенной распределенной АСДУ ЕСГ России

В.А. Швечков, С.А. Сарданашвили (РГУ нефти и газа имени И.М. Губкина)

Рассмотрены современные компьютерные технологии, позволяющие интегрировать разнородные компоненты информационного обеспечения автоматизированных систем диспетчерского управления (АСДУ) газотранспортных обществ (ГТО). Показаны варианты создания открытых интерфейсов к внешним информационным системам с использованием технологий распределенных вычислений (CORBA, DCOM, Java RMI, .NET) и сервис-ориентированной архитектуры (SOA). Предложены структурная схема организации распределенной вычислительной сети и структуры вычислительной сети гетерогенной распределенной АСДУ ЕСГ (единая система газоснабжения). Сформулированы рекомендации разработчикам программного обеспечения (ПО), функционирующего в распределенной структуре АСДУ ЕСГ.

Действующее в настоящий момент информационное обеспечение АСДУ ГТО представлено большим набором разнообразного и неоднородного ПО, распределенного по серверам и рабочим станциям диспетчерских служб (различные ОС, SCADA-системы, СУБД, расчетные вычислительные комплексы и прикладное ПО). Такое многообразие порождает целый комплекс проблем: непереносимость и несовместимость ПО; плохая масштабируемость; закрытость результатов работы ПО; различие в структуре хранения информации, методах доступа к ней и последующей обработке; несовместимость результатов режимно-технологических расчетов при одинаковых исходных данных в различных ГТО и на уровне центрального производственного диспетчерского департамента (ЦПДД) в целом. Одним из путей решения этих проблем является применение современных компьютерных технологий, позволяющих интегрировать разнородные компоненты информационного обеспечения АСДУ ГТО.

Начальным этапом в решении задачи интеграции ПО должно быть определение и детальное описание информационных потоков между существующими уровнями иерархической системы диспетчерского управления ЕСГ. Перед разработчиками встает задача создания открытых интерфейсов к информационным системам и обеспечения взаимодействия между ними.

В диспетчерских службах используются распределенные вычислительные сети на базе аппаратных и программных решений компаний SUN Microsystems, Microsoft, Novell. Использование сетевых ОС не решает задачи взаимодействия различных компонент программного комплекса или информационных систем, установленных на распределенных серверах и рабочих станциях диспетчерских служб. ОС и существующие каналы связи можно рассматривать только лишь как единую коммуникационную среду.

Современные распределенные системы обычно создаются в виде ПО промежуточного уровня для нескольких платформ, организованных согласно какой-либо из имеющихся парадигм, базирующихся или на распределенных системах объектов, или на файловых системах, или использующих системы документов и согласования [1]. Рассмотрим наиболее распространенные архитектуры и модели организации промежуточного программного обеспечения.

Обобщенная архитектура брокеров объектных запросов Common Object Request Broker Architecture (CORBA) разработана Object Management Group (OMG), в которую входит более 800 промышленных компаний (www.omg.org/technology/documents/corba_spec_catalog.htm). CORBA является не столько распределенной системой, сколько ее спецификацией, изложенной на более чем 700 страницах, а также