УДК 681.51:629.78

Калентьев А.А., Тюгашев А.А.

CALS-ТЕХНОЛОГИИ ДЛЯ БОРТОВЫХ АЛГОРИТМОВ УПРАВЛЕНИЯ КОСМИЧЕСКИМИ АППАРАТАМИ

В настоящее время при производстве сложных высокотехнологичных изделий промышленности все большее применение находят CALS-технологии, которые приходят на смену обычному автоматизированному проектированию. Ситуация в сфере проектирования и производства наукоемкой продукции развивается в сторону полного перехода на основанные не на бумажных документах, чертежах, технологических спецификациях и т.п., а на электронных моделях и представлениях методы проектирования, изготовления и сбыта. Происходит постепенный переход к ситуации, когда невозможно становится продать на внешнем рынке машиностроительную продукцию без соответствующей международным стандартам безбумажной электронной документации. Ситуация складывается таким образом, что некоторые зарубежные компании даже рассматривают работу в этом направлении как действенное средство ограничения доступа на международный рынок наукоемкой продукции тех стран, которые не сумеют своевременно освоить соответствующую безбумажную электронную технологию.

Впервые концепция CALS возникла в середине 70-х годов в оборонном комплексе США в связи с необходимостью повышения эффективности управления и сокращения затрат на информационное взаимодействие в процессах заказа, поставок и эксплуатации средств вооружения и военной техники (ВВТ). Движущей силой явилась естественная потребность в организации "единого информационного пространства", обеспечивающего оперативный обмен данными между заказчиком -- федеральными органами, производителями и потребителями ВВТ.

Первоначальное и до сей поры наиболее развитое применение (несмотря на активное внедрение в других отраслях, например, в судостроении), в силу сложности и высокой наукоемкости CALS-технологии получили в аэрокосмическом комплексе. Характерно, что ведущие корпорации, такие, как Boeing, заключают субконтракты на поставку комплектующих изделий только с теми фирмами, которые осуществляют разработку и производство изделий в режиме CALS-технологий.

Аббревиатура CALS при ее появлении сначала расшифровывалась как Computer Aid Logistics Support (автоматизация или компьютерное обеспечение логистики). Сейчас наиболее распространенной расшифровкой следует считать Continuous Acquisition and Lifecycle Support (компьютерное сопровождение и поддержка жизненного цикла изделий), хотя используются и другие (Computer Aided Acquisition and Lifecycle Support, компьютеризированные поставки и поддержка, и даже Commerce At Light Speed, бизнес в высоком темпе).

Под CALS-технологиями, таким образом, понимается принципиально новая компьютерная система электронного описания процессов проектирования, разработки, технологической подготовки производства, выпуска, модернизации, эксплуатации и сервисного обслуживания продукции военного, гражданского и двойного назначения [1]

В русском языке в настоящее время для обозначения комплекса данных технологий принято обозначение ИПИ (информационная поддержка изделий), также встречается КСП (компьютерное сопровождение и поддержка жизненного цикла изделий) [2].

Главная задача создания и внедрения CALS-технологий - обеспечение единообразия описания и интерпретации данных независимо от времени и места их получения в общей системе, имеющей масштабы вплоть до глобальных.

Как правило, в применяемых на сегодняшний день в промышленности информационных технологиях автоматизируются отдельные фазы и этапы жизненного цикла (ЖЦ) продукции (конструирование, разработка технологии, проектирование, планирование производства и др.). Внедрение CALS-технологий позволяет интегрировать в одну систему комплекс информационных потоков, существующих на всех этапах ЖЦ.

По оценкам специалистов США, подтвержденным первым опытом использования CALS-технологий, внедрение системы CALS в полном объеме позволяет ускорить на 30-40 % выполнение НИОКР и снизить на величину до 30% издержки при производстве и эксплуатации высокотехнологичной продукции [1].

Следует отметить, что для современного этапа развития информационных технологий характерен переход к зрелости от той ситуации, когда программирование являлось уделом избранных одиночек, искусством, что нашло свое отражение в том числе в классическом труде Д. Кнута "Искусство программирования" [3], к промышленному характеру разработки программного обеспечения. Можно рассуждать о том, насколько это хорошо или плохо с точки зрения отдельного талантливого программиста, однако этот переход является практически свершившимся процессом. В связи с этим, несмотря на продолжающиеся в профессиональной среде дискуссии, все более и более актуальными становятся параллели между про-

цессами создания программ и традиционными процессами разработки сложных технических (аппаратных) средств. В этой связи появился и стал вполне адекватным термин Software Engineering или инженерия программных средств, подчеркивающий родственность подходов и методик в промышленности и в современном программировании.

Итак, при создании сложных технических изделий и комплексов в промышленности сейчас наступила эпоха перехода от информационных технологий, поддерживающих какой-то один этап жизненного цикла изделия (автоматизация проектирования CAD/САПР, технологической подготовки производства CAM, автоматизации производства CAE/ГАП и др.), к концепции комплексной информационной поддержки изделия от эскизного проекта до списания и утилизации. Что же имеется в сфере производства программного обеспечения?

В силу специфики того, что программы для ЭВМ сами изначально предназначались для автоматизации тех или иных сторон деятельности человека, практически одновременно с появлением первых ЭВМ стали предприниматься попытки автоматизировать процесс создания программ, их эксплуатации и доработки. Эти попытки привели, с одной стороны, к появлению операционных систем (ОС) ЭВМ, с другой – к созданию инструментальных программ, таких, как ассемблеры, трансляторы, интерпретаторы, компоновщики, отладчики, загрузчики.

Постепенно стало наиболее типичным объединение в рамках одного интегрированного комплекса таких программ, предназначенных для поддержки процессов, связанных с программированием, как редактора текстов, компилятора, компоновщика и отладчика. Такие программные комплексы получили название IDE (интегрированная среда разработки, Integrated Development Environment).

Вместе с тем, переход к созданию сложных программных комплексов, состоящих из сотен тысяч и даже миллионов строк программного текста, требовал обратить особое внимание на вопросы тщательного проектирования, обоснованного выбора архитектурных решений, корректного анализа предметной области и постановки задачи, которую необходимо решить. Это привело к появлению специализированных так называемых CASE-средств. CASE применительно к программным средствам расшифровывается как Computer Aided Software Engineering, а после обобщения данного подхода на произвольные сложные системы стали говорить более широко о Computer Aided System Engineering. CASE-средства прошли непростой период становления, и сейчас уже можно говорить о том, что они стали общепризнанным стандартом. Прежде всего здесь нужно назвать такие языки и стандарты, как SADT (IDEF0), ERD, DFD, UML.

CASE-средства играют огромную роль на этапе анализа задачи, сбора требований разработке проекта и архитектуры программной системы. Однако, они достаточно долго рассматривались как дополнительный, внешний инструмент по отношению к традиционной IDE в которой большую часть времени проводит программист при создании программы. Однако аналогично тому, как это происходило для сложных промышленных изделий, постепенно стало появляться понимание необходимости тесной интеграции различных процессов жизненного цикла программных средств. Одним из пионеров этого направления стала корпорация Rational (США), которая разработала комплексную методологию процесса проектирования и разработки программ, названную RUP -- Rational Unified Process. При этом практически все отдельные стадии разработки поддерживаются специфическими автоматизирующими программными средствами, тесно интегрированными между собой. Позднее в направлении интеграции и комплексной автоматизации различных стадий разработки стали разворачиваться такие гиганты программной индустрии, как Borland и Microsoft, предложившие свои методологии (Borland Core SDP и Microsoft Solutions Framework). Подобного рода решения (называемые Application Lifecycle Management) можно считать CALS-технологиями применительно к программным системам, рассматриваемым как сложные технические изделия. Однако, у программных систем есть свои существенные особенности и отличия от технических средств.

Итак, рассмотрим более подробно особенности жизненного цикла программной продукции по сравнению со сложными техническими (аппаратными) комплексами. Основные этапы жизненного цикла промышленной продукции таковы (по Р 50.1.031-2001)

1. Обоснование необходимости разработки.
2. Техническое задание.
3. Проектирование.
4. Подготовка производства (технологическая подготовка).
5. Производство (изготовление).
6. Реализация продукции (поставка).
7. Эксплуатация (включая ремонт и техобслуживание).
8. Утилизация.

В противоположность приведенному, у программных средств в силу физической сущности (информация) практически вырожденной является стадия утилизации. Также следует отметить такую особенность, как отсутствие необходимости в ремонтах и снабжении запасными частями -- процессе, который играет важную роль в CALS-технологиях промышленных

изделий. В то же время, процесс технического обслуживания для программных систем может иметь место. Например, к нему может относиться необходимое регламентное архивное сохранение данных, построение индексов и т.д. Для программных средств в процессе эксплуатации важную роль могут иметь процессы исправления ошибок, выявленных на протяжении использования программы, и проведение доработок (выпуск новых версий продукта).

К настоящему моменту в индустрии разработки программного обеспечения нет единого общепринятого понимания понятия "жизненный цикл программного средства (ПС)". Существует набор стандартов, регламентирующих описание основных процессов жизненного цикла программных комплексов. Разработаны стандарты различных уровней утверждения – как национальные, так и международные, а также существуют стандарты корпораций, крупных организаций, имеющих значительный опыт разработки и сопровождения программного обеспечения.

Впервые формализованный стандарт жизненного цикла ПС (как, обратим внимание, и концепция CALS) был утвержден в 1985 г. (уточнен в 1988 г.) для программных систем военного назначения по заказам министерства обороны США -- стандарт DOD-STD 2167 A. Этим стандартом регламентированы восемь фаз (этапов) при создании сложных ПС:

1. Создание концепции, выработка общих требований к системе.
2. Детализация требований к программной системе.
3. Предварительное проектирование.
4. Детальное проектирование.
5. Разработка компонентов (модулей).
6. Комплексирование (интеграция модулей).
7. Тестирование.
8. Испытания ПС в составе целевой системы

Согласно RUP компании Rational, жизненный цикл программного средства включает в себя фазы и итерации. Фаза -- это промежуток времени между двумя важными опорными точками процесса, в которых должны быть достигнуты четко определенные цели, подготовлены те или иные артефакты и принято решение о том, следует ли переходить к следующей фазе:

1. Начало -- определение бизнес-целей проекта.
2. Исследование – разработка плана и архитектуры проекта.
3. Построение – постепенное создание системы.
4. Внедрение -- поставка системы пользователям.

Итерация представляет собой полный цикл разработки, завершенный этап, в результате которого выпускается версия продукта для внутреннего или внешнего использования. Итерация включает так называемые рабочие процессы:

1. Моделирование.
2. Сбор требований.
3. Анализ и проектирование.
4. Реализация.
5. Тестирование.
6. Развертывание.

Исторически большое внимание уделялось этапам математической постановки исходной задачи, решаемой затем на ЭВМ, и выработки алгоритма ее решения. Это нашло свое отражение, например, в таком российском стандарте, как ГОСТ 19.102-77 ЕСПД "Стадии разработки", в котором формализуются следующие стадии и этапы разработки программ и программной документации:

1. Обоснование необходимости разработки программы (в том числе постановка задачи; выбор и обоснование критериев эффективности и качества разрабатываемой программы).

2. Научно-исследовательские работы (в том числе определение структуры входных и выходных данных; предварительный выбор методов решения задач; обоснование целесообразности применения ранее созданных программ; определение требований к техническим средствам; обоснование принципиальной возможности решения поставленной задачи).

3. Разработка и утверждение технического задания (в том числе определение требований к программе; определение стадий, этапов и сроков разработки программы и программной документации; согласование ТЗ).

4. Разработка эскизного проекта.

5. Утверждение эскизного проекта.

6. Разработка технического проекта (в том числе уточнение структуры входных и выходных данных; разработка алгоритма решения; определение форм представления данных; разработка структуры программы; окончательное уточнение комплекса технических средств).

7. Утверждение технического проекта (в том числе составление плана мероприятий по разработке и внедрению программы).

8. Разработка программы (в том числе программирование и отладка).

9. Разработка программной документации в соответствии с ГОСТ 19.101.

10. Испытания программы (включая разработку и согласование программы и методики испытаний; проведение приемо-сдаточных и других видов испытаний; корректировка программы и программной документации по результатам испытаний).

11. Внедрение – подготовка и передача программы.

Нетрудно видеть параллели с подходами к проектированию и разработке технических комплексов. Детализированы, особенно с точки зрения создаваемой документации, этапы составления технических задания и проектов. Одной строкой здесь проходит разработка, включая программирование и отладку. При этом в отдельный этап выделены "испытания" программы, которые более логично объединить с тестированием и отладкой. Следует отметить, что в 1977 году – году принятия упомянутого стандарта, понятие "жизненного цикла программы" еще не использовалось.

Позднее был выпущен стандарт ГОСТ 34.601-90, определяющий стадии разработки автоматизированной системы (АС), в состав которой могут входить программные средства, и согласно ему, выделяются такие стадии жизненного цикла, как

1. Формирование требований к автоматизированной системе.
2. Разработка концепции АС.
3. Разработка эскизного проекта.
4. Разработка технического проекта
5. Разработка рабочей документации.
6. Ввод в действие.

В соответствии с международным стандартом ИСО/МЭК 12207-99, модель жизненного цикла ПС представляет собой структуру, состоящую из процессов, работ и задач, включающих в себя разработку, эксплуатацию и сопровождение, то есть всю "жизнь" программного средства: от предъявления требований к нему до снятия с эксплуатации.

Структура жизненного цикла программного средства базируется на трех группах процессов:

А. Основные процессы жизненного цикла – приобретение, поставка, разработка, эксплуатация, сопровождение.

Б. Вспомогательные процессы, обеспечивающие выполнение основных процессов (подготовка и выпуск документации, конфигурационное управление, обеспечение качества, верификация, аттестация, оценка, аудит и др.).

В. Организационные процессы – управление проектами, создание обеспечивающей инфраструктуры, обучение и др.

К основным процессам согласно данному стандарту относят:

1. Процесс приобретения.

2 Процесс разработки.

3. Процесс эксплуатации.

4. Процесс сопровождения.

Нетрудно видеть, что данный стандарт акцентирован на процессах приобретения и использования программного средства пользователем, а не на технологических вопросах.

По версии корпорации Borland, выделяются такие стадии ЖЦ программного продукта как:

1. Определение требований.

2. Проектирование.

3. Разработка.

4. Тестирование.

5. Поставка.

По представлениям Microsoft (Microsoft Solutions Framework), ключевыми являются стадии:

1. Создание общей картины.

2. Планирование.

3. Разработка.

4. Стабилизация.

5. Развертывание.

Отметим, что здесь для тестирования и фиксации работоспособной версии программной системы используется специфический термин – "Стабилизация".

Как видно, существует ряд подходов к определению стадий жизненного цикла программной продукции. В то же время, несмотря на некоторые различия, в основном выделяются такие крупные фазы, как:

1. Анализ, постановка задачи и формулировка требований.

2. Проектирование.

3. Разработка, включая создание документации.

4. Тестирование (испытания) и отладка.

5. Внедрение (поставка и установка).

6. Эксплуатация и сопровождение.

Это достаточно универсальные стадии, подходящие практически для любых программных комплексов, но все-таки больше ориентированные на программные системы об

го пользования, предназначенные для исполнения на универсальных ЭВМ. В то же время, если вести речь о специализированном управляющем программном обеспечении реального времени для космических аппаратов (КА), то встает вопрос о наличии важных особенностей, которые могут повлиять на спецификацию отдельных этапов и стадий жизненного цикла. В первую очередь, разработка управляющих программ тесно увязана в единый процесс с жизненным циклом самого КА, который включает следующие стадии [4].

1. Формирование требований к КА (внешнее проектирование).
2. Проектирование (внутреннее проектирование), в том числе:
– разработка технических предложений;
– эскизное проектирование;
– рабочее проектирование.
3. Создание опытного образца КА.
4. Испытания опытного образца.
5. Серийное производство КА.
6. Эксплуатация КА.
7. Увод КА с рабочей орбиты и затопление (при необходимости).

На стадии разработки технических предложений выбирается конструктивная схема, производится структурная и функциональная декомпозиция КА, включая разделение функций аппаратной и программной частей систем КА, определяются принципы управления КА. Осуществляется также временная декомпозиция задач, решаемых КА. Подготавливаются технические задания на системы КА. В процессе эскизного проектирования выпускаются исходные данные на отдельные алгоритмы КА, определяется структура бортового программного обеспечения (БПО) и принципы мультипрограммного решения задач.

Рабочий проект содержит всю техническую документацию, необходимую для изготовления, сборки, испытаний агрегатов и систем, а также КА в целом. На этапе выпуска рабочего проекта проводятся экспериментальные исследования и отработка систем КА, включая БПО бортовой вычислительной системы (БВС), на специальных моделирующих стендах с реальными БВС и программами. Разработка программ, автономная отладка и сборка программного комплекса БПО производится также на этом этапе.

Особенности жизненного цикла управляющих программ для БЦВМ вызываются набором факторов, которые необходимо учитывать, в том числе [4]:

1. промышленный характер разработки, производства и эксплуатации БПО;
2. отсутствие возможности прямой коррекции БПО в процессе эксплуатации;

3. невозможность остановки выполнения управляющих программ без остановки функ-ционирования изделия в целом;

4. высокие требования к надежности, так как ошибки БПО могут приводить к катастрофическим последствиям и потере дорогостоящих технических изделий;

5. необходимость независимой разработки отдельных программ БПО различными разработчиками с последующим объединением их в единый программный комплекс;

6. необходимость управления изменениями, вносимыми в БПО;

7. необходимость технологической защиты информации в процессе разработки и эксплуатации БПО;

8. необходимость модификаций программ БПО в процессе эксплуатации изделий.

Согласно разработкам отечественных ученых и конструкторов [4], выделяют следующие этапы жизненного цикла для комплекса БПО:

1. Проектирование БПО и алгоритмов, моделирование алгоритмов.

2. Программирование БПО (кодирование).

3. Автономная отладка программ БПО.

4. Комплексная отладка БПО.

5. Сопровождение БПО.

Нетрудно видеть, что особенности жизненного цикла комплекса управляющих программ для КА близки к общей постановке задачи и истокам возникновения CALS-технологий. В связи с этим, вопросы создания комплексной интегрированной методики разработки, отвечающей CALS-стандартам, и поддерживающих ее программных средств являются особенно актуальными для данной разновидности программных средств.

Такого рода методика и поддерживающий ее инструментальный программный комплекс - ГРАФКОНТ/ГЕОЗ разрабатываются на протяжении ряда лет сотрудниками ГНПРКЦ "ЦСКБ-Прогресс", НТЦ "Наука", Самарского государственного аэрокосмического университета. Программный комплекс ГРАФКОНТ/ГЕОЗ автоматизирует следующие процессы жизненного цикла:

1. Выделение элементарных функциональных задач, на которые разбивается сложный управляющий алгоритм.

2. Формулировка требований по временным и логическим зависимостям между функциональными программами с помощью специального интерактивного визуального языка.

3. Автоматическая генерация управляющей программы на языке ассемблера целевой БЦВМ.

4. Формирование технической документации на управляющий алгоритм и программу (включая временные диаграммы и блок-схемы).

5. Формирование таблицы вариантов исполнения (отладочных вариантов) и отладочных заданий для отладки программы на специальном наземном стенде.

В программной системе широко используются такие перспективные и современные методы, как визуальное программирование. Таким образом, происходит синтез двух технологий: CALS и визуального программирования.

При этом система ГРАФКОНТ/ГЕОЗ не отменяет, а органично дополняет на новом уровне используемую и успешно апробированную в течение ряда лет на практике методологию разработки и отладки управляющих программ.

При необходимости проведения доработки программы, система предоставляет возможность автоматической перегенерации "на лету" управляющей программы после внесения изменений в описание управляющего алгоритма на интерактивном графическом языке или после изменения функциональной задачи.

При этом система дает пользователю-программисту возможность работы в интегрированной среде, которая позволяет не только вызывать отдельные модули системы ГРАФКОНТ/ГЕОЗ, но и готовить данные в форматах других, используемых при проектировании программ, например, AutoCAD, и производить их вызов. Все это позволяет говорить о выполнении программной системой ГРАФКОНТ/ГЕОЗ функций управления данными об изделии (PDM) применительно к комплексу управляющих программ для КА.

Таким образом, в соответствии с базовым принципом CALS, данные отделяются от прикладных программных средств.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Проблемы продвижения продукции и технологий на внешний рынок. Специальный выпуск, 1997 г.

2. Норсиков И.П., Кузьмик П.К. Информационная поддержка наукоемких изделий. CALS-технологии. – М.: Изд-во МГТУ им. Баумана, 2002.

3. Кнут Д. Искусство программирования. В 3-х томах. – М.: Наука, 1990.

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