УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию, законспектируйте основные сведения о технологиях JAVA и .NET

Ответы на вопросы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 30.01.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция 4

Тема: Категории современных инструментальных средств разработки программ.

Цель: Ознакомится с категориями современных инструментальных средств.

Введение

В процессе разработки программного обеспечения используется большое количество самого разностороннего программного обеспечения (ПО). Данный курс лекций рассматривает когда и что используется на протяжении всего этапа разработки приложений.

Прежде чем приступить к рассмотрению средств разработки, которые могут быть применены для создания программ, необходимо определиться с основными понятиями, терминами, которые будут использоваться в статье. В соответствии с тематикой статьи базовым термином для нас, конечно же, является «средства разработки программ». Применительно к области разработки программного обеспечения данное определение может звучать следующим образом:

Основные классы инструментальных сред разработки и сопровождения программных средств

В настоящее время выделяют три основных класса инструментальных сред разработки и сопровождения ПС (рис. 16.1):

- инструментальные среды программирования,
- рабочие места компьютерной технологии,
- инструментальные системы технологии программирования.

Рис. 1. Основные классы инструментальных сред разработки и сопровождения ПС.

Инструментальная среда программирования предназначена в основном для поддержки процессов программирования (кодирования), тестирования и отладки ПС. Она не обладает рассмотренными выше свойствами комплексности, ориентированности на конкретную технологию программирования, ориентированности на коллективную разработку и, как правило, свойством интегрированности, хотя имеется некоторая тенденция к созданию интегрированных сред программирования (в этом случае их следовало бы называть *системами программирования*). Иногда среда программирования может обладать свойством специализированности. Признак же ориентированности на конкретный язык программирования может иметь разные значения, что существенно используется для дальнейшей классификации сред программирования.

Рабочее место компьютерной технологии ориентировано на поддержку ранних этапов разработки ПС (системного анализа и спецификаций) и автоматической генерации программ по спецификациям [16.1, 16.4]. Оно существенно использует свойства специализированности, ориентированности на конкретную технологию программирования и, как правило, интегрированности. Более поздние рабочие места компьютерной технологии обладают также свойством комплексности ([16.4]). Что же касается языковой ориентированности, то вместо языков программирования они ориентированы на те или иные формальные языки спецификаций. Свойством ориентированности на коллективную разработку указанные рабочие места в настоящее время, как правило, не обладают.

Инструментальная система технологии программирования предназначена для поддержки всех процессов разработки и сопровождения в течение всего жизненного цикла ПС и ориентирована на коллективную разработку больших программных систем с продолжительным жизненным циклом. Обязательными свойствами ее являются комплексность, ориентированность на коллективную разработку и интегрированность. Кроме того, она или обладает технологической определенностью или получает это свойство в процессе расширения (настройки).

Значение признака языковой ориентированности может быть различным, что используется для дальнейшей классификации этих систем.

Инструментальные среды программирования

Инструментальная среда программирования включает, прежде всего, текстовый редактор, позволяющий конструировать программы на заданном языке программирования, а также инструменты, позволяющие компилировать или интерпретировать программы на этом языке, тестировать и отлаживать полученные программы. Кроме того, могут быть и другие инструменты, например, для статического или динамического анализа программ. Взаимодействуют эти инструменты между собой через обычные файлы с помощью стандартных возможностей файловой системы.

Различают следующие классы инструментальных сред программирования (см. рис. 16.2):

- среды общего назначения,
- языково-ориентированные среды.

Инструментальные среды программирования *общего назначения* содержат набор программных инструментов, поддерживающих разработку программ на разных языках программирования (например, текстовый редактор, редактор связей или интерпретатор языка целевого компьютера) и обычно представляют собой некоторое расширение возможностей используемой операционной системы. Для программирования в такой среде на каком-либо языке программирования потребуются дополнительные инструменты, ориентированные на этот язык (например, компилятор).

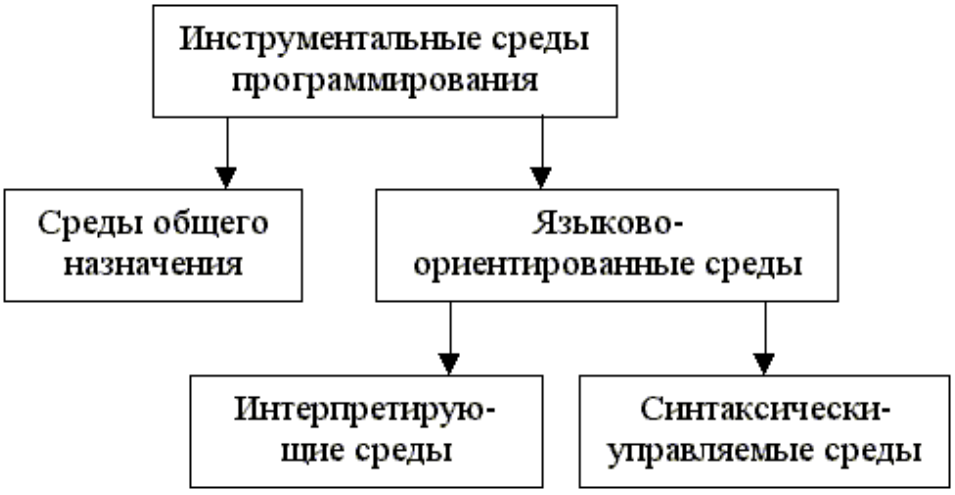

Рис.2. Классификация инструментальных сред программирования.

Языково-ориентированная инструментальная среда программирования предназначена для поддержки разработки ПС на каком-либо одном языке программирования и знания об этом языке существенно использовались при построении такой среды. Вследствие этого в такой среде могут быть доступны достаточно мощные возможности, учитывающие специфику данного языка. Такие среды разделяются на два подкласса:

- интерпретирующие среды,
- синтаксически-управляемые среды.

Интерпретирующая инструментальная среда программирования обеспечивает интерпретацию программ на данном языке программирования, т.е. содержит, прежде всего, интерпретатор языка программирования, на который эта среда ориентирована. Такая среда необходима для языков программирования интерпретирующего типа (таких, как Лисп), но может использоваться и для других языков (например, на инструментальном компьютере). *Синтаксически-управляемая* инструментальная среда программирования базируется на знании синтаксиса языка программирования, на который она ориентирована. В такой среде вместо текстового используется синтаксически-управляемый редактор, позволяющий пользователю использовать различные шаблоны синтаксических конструкций (в результате этого разрабатываемая программа всегда будет синтаксически правильной). Одновременно с программой такой редактор формирует (в памяти компьютера) ее синтаксическое дерево, которое может использоваться другими инструментами.

Понятие компьютерной технологии разработки программных средств и ее рабочие места

Имеются некоторые трудности в выработке строгого определения CASE-технологии (компьютерной технологии разработки ПС). CASE - это абревиатура от английского Computer-Aided Software Engineering (Компьютерно-Помогаемая Инженерия Программирования). Но без помощи (поддержки) компьютера ПС уже давно не разрабатываются (используется хотя бы компилятор). В действительности, в это понятие вкладывается более узкий (специальный) смысл, который постепенно размывается (как это всегда бывает, когда какое-либо понятие не имеет строгого определения). Первоначально под CASE понималась инженерия ранних этапов разработки ПС (определение требований, разработка внешнего описания и архитектуры ПС) с использованием программной поддержки (программных инструментов). Теперь под CASE может пониматься и инженерия всего жизненного цикла ПС (включая и его сопровождение), но только в том случае, когда программы частично или полностью генерируются по документам, полученным на указанных ранних этапах разработки. В этом случае CASE-технология стала принципиально отличаться от ручной (традиционной) технологии разработки ПС: изменилось не только содержание технологических процессов, но и сама их совокупность.

В настоящее время *компьютерную технологию* разработки ПС можно характеризовать использованием

- программной поддержки для разработки графических требований и графических спецификаций ПС,
- автоматической генерации программ на каком-либо языке программирования или в машинном коде (частично или полностью),
- программной поддержки прототипирования.

Говорят также, что компьютерная технология разработки ПС является "безбумажной", т.е. рассчитанной на компьютерное представление программных документов. Однако, уверенно отличить ручную технологию разработки ПС от компьютерной по этим признакам довольно трудно. Значит, самое существенное в компьютерной технологии не выделено.

На наш взгляд, главное отличие ручной технологии разработки ПС от компьютерной заключается в следующем. Ручная технология ориентирована на разработку документов, одинаково понимаемых разными разработчиками ПС, тогда как компьютерная технология ориентирована на обеспечение семантического понимания (интерпретации) документов программной поддержкой компьютерной технологии. Семантическое понимание документов дает программной поддержке возможность автоматически генерировать программы. В связи с этим существенной частью компьютерной технологии становится использование формальных языков уже на ранних этапах разработки ПС: как для спецификации программ, так и для спецификации других документов. В частности, широко используются формальные графические языки спецификаций. Именно это позволяет рационально изменить и саму совокупность технологических процессов разработки и сопровождения ПС.

Из проведенного обсуждения можно определить *компьютерную технологию разработки ПС* как технологию программирования, в которой используются программные инструменты для разработки формализованных спецификаций программ и других документов (включая и графические спецификации) с последующей автоматической генерацией программ и документов (или хотя бы значительной их части) по этим спецификациям.

Теперь становятся понятными и основные изменения в жизненном цикле ПС для компьютерной технологии. Если при использовании ручной технологии основные усилия по разработке ПС делались на этапах собственно программирования (кодирования) и отладки (тестирования), то при использовании компьютерной технологии - на ранних этапах разработки ПС (определения требований и функциональной спецификации, разработки архитектуры). При этом существенно изменился характер документации. Вместо целой цепочки неформальных документов, ориентированной на передачу информации от заказчика (пользователя) к различным категориям разработчикам, формируются прототип ПС, поддерживающий выбранный пользовательский интерфейс, и формальные функциональные спецификации (иногда и формальные спецификации архитектуры ПС), достаточные для автоматического синтеза (генерации) программ ПС (или хотя бы значительной их части). При этом появилась возможность автоматической генерации части документации, необходимой разработчикам и пользователям. Вместо ручного программирования (кодирования) - автоматическая генерация программ, что делает не нужной автономную отладку и тестирование программ: вместо нее добавляется достаточно глубокий автоматический семантический контроль документации. Появляется возможность автоматической генерации тестов по формальным спецификациям для комплексной (*системной*) отладки ПС. Существенно изменяется и характер сопровождения ПС: все изменения разработчиком-сопроводителем вносятся только в спецификации (включая и прототип), остальные изменения в ПС осуществляются автоматически.

С учетом сказанного *жизненный цикл ПС для компьютерной технологии* можно представить следующей схемой (рис. 16.3).

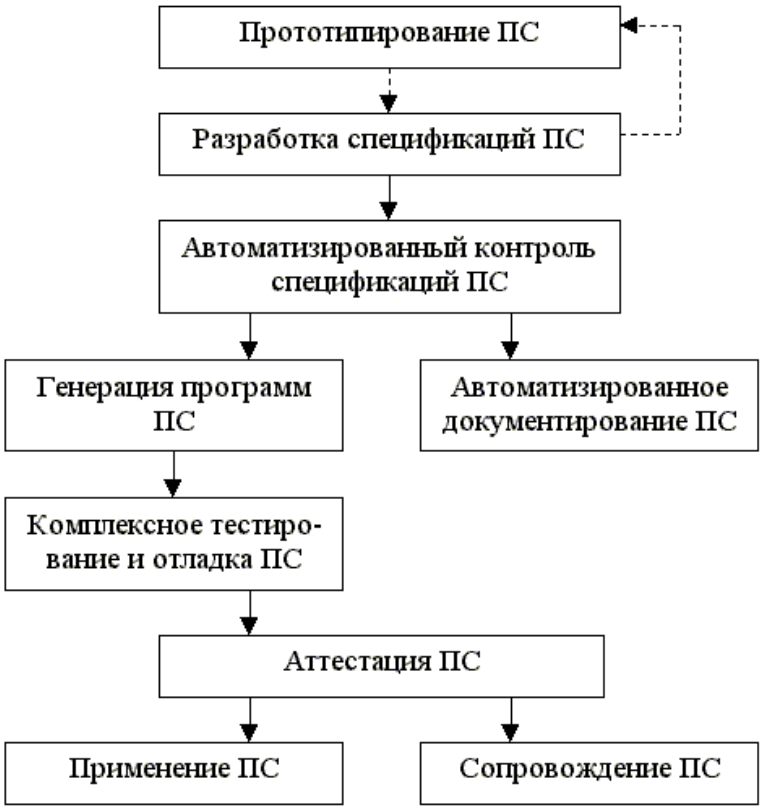

Рис. 3. Жизненный цикл программного средства для компьютерной технологии.

Прототипирование ПС является необязательным этапом жизненного цикла ПС при компьютерной технологии, что на рис. 16.3 показано пунктирной стрелкой. Однако использование этого этапа во многих случаях и соответствующая компьютерная поддержка этого этапа является характерной для компьютерной технологии. В некоторых случаях прототипирование делается после (или в процессе) разработки спецификаций ПС, например, в случае прототипирования пользовательского интерфейса. Это показано на рис. 16.3 пунктирной возвратной стрелки. Хотя возврат к предыдущим этапам мы допускаем на любом этапе, но здесь это показано явно, так как прототипирование является особым подходом к разработке ПС (см. лекцию 3). Прототипирование пользовательского интерфейса позволяет заменить косвенное описание взаимодействия между пользователем и ПС при ручной технологии (при разработке внешнего описания ПС) прямым выбором пользователем способа и стиля этого взаимодействия с фиксацией всех необходимых деталей. По существу, на этом этапе производится точное описание пользовательского интерфейса, понятное программной поддержке компьютерной технологии, причем с ответственным участием пользователя. Все это базируется на наличие в программной поддержке компьютерной технологии настраиваемой оболочки с обширной библиотекой заготовок различных фрагментов и деталей экрана. Такое прототипирование, по-видимому, является лучшим способом преодоления барьера между пользователем и разработчиком.

Разработка спецификаций ПС распадается на несколько разных процессов. Если исключить начальный этап разработки спецификаций (определение требований), то в этих процессах используются методы, приводящие к созданию формализованных документов, т. е. используются формализованные языки

спецификаций. При этом широко используются графические методы спецификаций, приводящие к созданию различных схем и диаграмм, которые определяют структуру информационной среды и структуру управления ПС. К таким структурам привязываются фрагменты описания данных и программ, представленные на алгебраических языках спецификаций (например, использующие операционную или аксиоматическую семантику), или логических языках спецификаций (базирующихся на логическом подходе к спецификации программ). Такие спецификации позволяют в значительной степени или полностью автоматически генерировать программы. Существенной частью разработки спецификаций является создание словаря именованных сущностей, используемых в спецификациях.

Автоматизированный контроль спецификаций ПС использует то обстоятельство, что значительная часть спецификаций представляется на формальных языках. Это позволяет автоматически осуществлять различные виды контроля: синтаксический и частичный семантический контроль спецификаций, контроль полноты и состоятельности схем и диаграмм (в частности, все их элементы должны быть идентифицированы и отражены в словаре именованных сущностей), сквозной контроль сбалансированности уровней спецификаций и другие виды контроля в зависимости от возможностей языков спецификаций.

Генерация программ ПС. На этом этапе автоматически генерирует скелеты кодов программ ПС или полностью коды этих программ по формальным спецификациям ПС.

Автоматизированное документирование ПС. Предполагает возможность генерации различных форм документов с частичным заполнением их по информации, хранящейся в репозитории. При этом количество видов документов сокращается по сравнению с традиционной технологией.

Комплексное тестирование и отладка ПС. На этом этапе тестируются все спецификации ПС и исправляются обнаруженные при этом ошибки. Тесты могут создаваться как вручную, так и автоматически (если это позволяют используемые языки спецификаций) и пропускаются через сгенерированные программы ПС.

Аттестация ПС имеет прежнее содержание.

Сопровождение ПС существенно упрощается, так как основные изменения делаются только в спецификациях.

Рабочее место компьютерной технологии разработки ПС представляет собой инструментальную среду, поддерживающую все этапы жизненного цикла этой технологии. В этой среде существенно используется репозиторий. В репозитории хранится вся информация, создаваемая в процессе разработки ПС (в частности, словарь именованных сущностей и все спецификации). По существу, рабочее место компьютерной технологии является интегрированным хотя бы по пользовательскому интерфейсу и по данным. Основными инструментами такого рабочего места являются:

- конструкторы пользовательских интерфейсов;
- инструмент работы со словарем именованных сущностей;
- графические и тестовые редакторы спецификаций;
- анализаторы спецификаций;
- генератор программ;
- документаторы.

Инструментальные системы технологии программирования

Инструментальная система технологии программирования - это интегрированная совокупность программных и аппаратных инструментов, поддерживающая все процессы разработки и сопровождения больших ПС в течение всего его жизненного цикла в рамках определенной технологии. Выше уже отмечалось, что она помимо интегрированности обладает еще свойствами комплексности и ориентированности на коллективную разработку. Это означает, что она базируется на согласованности продукции технологических процессов. Тем самым, инструментальная система в состоянии обеспечить, по крайней мере, контроль полноты (комплектности) создаваемой документации (включая набор программ) и согласованности ее изменения (версионности). Поддержка инструментальной системой фазы сопровождения ПС, означает, что она должна обеспечивать *управление конфигурацией ПС*. Кроме того, инструментальная система поддерживает управление работой коллектива и для разных членов этого коллектива обеспечивает разные права доступа к различным фрагментам продукции технологических процессов и поддерживает работу *менеджеров* по управлению коллективом разработчиков. Инструментальные системы технологии программирования представляют собой достаточно большие и дорогие ПС, чтобы как-то была оправданна их инструментальная перегруженность. Поэтому набор включаемых в них инструментов тщательно отбирается с учетом потребностей предметной области, используемых языков и выбранной технологией программирования.

С учетом обсужденных свойств инструментальных систем технологии программирования можно выделить три их основные компоненты:

- репозиторий,
- инструментарий,
- интерфейсы.

Инструментарий - набор инструментов, определяющий возможности, предоставляемые системой коллективу разработчиков. Обычно этот набор является открытым и структурированным. Помимо минимального набора (*встроенные инструменты*), он содержит средства своего расширения (*импортированными инструментами*). Кроме того, в силу интегрированности по действиям он состоит из некоторой общей части всех инструментов (*ядра*) и структурных (иногда иерархически связанных) классов инструментов.

Интерфейсы разделяются на пользовательский и системные. *Пользовательский* интерфейс обеспечивает доступ разработчикам к инструментарию. Он реализуется *оболочкой* системы. *Системные* интерфейсы обеспечивают взаимодействие между инструментами и их общими частями. Системные интерфейсы выделяются как архитектурные компоненты в связи с открытостью системы - их обязаны использовать новые (*импортируемые*) инструменты, включаемые в систему.

Самая общая архитектура инструментальных систем технологии программирования представлена на рис. 16.4.

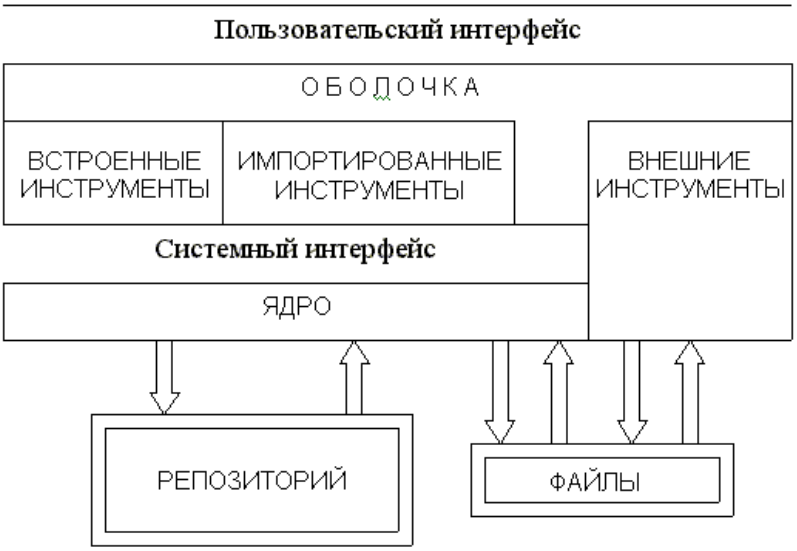

Рис. 4. Общая архитектура инструментальных систем технологии программирования.

Различают два класса инструментальных систем технологии программирования: инструментальные системы поддержки проекта и языково-зависимые инструментальные системы.

Инструментальная система поддержки проекта - это открытая система, способная поддерживать разработку ПС на разных языках программирования после соответствующего ее расширения программными инструментами, ориентированными на выбранный язык. Набор инструментов такой системы поддерживает разработкой ПС, а также содержит независимые от языка программирования инструменты, поддерживающие разработку ПС (текстовые и графические редакторы, генераторы отчетов и т.п.). Кроме того, он содержит инструменты расширения системы. Ядро такой системы обеспечивает, в частности, доступ к репозиторию.

Языково-зависимая инструментальная система - это система поддержки разработки ПС на каком-либо одном языке программирования, существенно использующая в организации своей работы специфику этого языка. Эта специфика может сказываться и на возможностях ядра (в том числе и на структуре репозитория), и на требованиях к оболочке и инструментам. Примером такой системы является среда поддержки программирования на Аде.