МОДЕЛИРОВАНИЕ СИСТЕМ ДЛЯ ПРИКЛАДНЫХ ОБЛАСТЕЙ ЗНАНИЙ.

Пути развития системного программирования.

Открытая конференция ИСП РАН

Имени В.П.Иванникова

5.12 -6.12.2019

Доклад

доктора физ.- мат. наук, проф. МФТИ, г.н.с. ИСП РАН

Лаврищевой Е.М.

Понятие модели

С давних времен сформировались термины – образец, модель (парадигма) технических объектов. Модель отражает основные понятия, закономерности их развития и связи между ними. К таким понятиям относятся формулы, набор правил для их вычисления и соглашения о связях между ними.

Модель может отражать некоторое явление, которое определяется на множестве величин, которые надо определить. И как говорил Аль Хорезми, надо разработать численный алгоритм реализации разных величин модели. Разрабатывается алгоритм математической задачи путем декомпозиции ее на отдельные подзадачи и связи между ними.

Для проведения вычислительного эксперимента определяется математическая модель - алгоритм – программа (А.А.Самарский).

При этом алгоритм – связывающее звено между моделью и программой и отражает теорию эксперимента, численное его моделирование в виде программы реализации на компьютере.

Концептуальная модель предметной области знаний

Древнегреческие ученые Ньютон предложил численный алгоритм (метод) решения алгебраических, а Эйлер – дифференциальных уравнений.

Для проведения расчета исходная математическая модель заменяется ее аналогом, пригодным для расчета, т.е. дискретной моделью. Аналогично для любой предметной области знаний (информатика, физика, биология, медицина и др.) должна создаваться концептуальная модель (КМ) вида:

КМ = <X, R, F>,

где X – конечное множество понятий предметной области знаний;

R – конечное множество отношений между понятиями области;

F – конечное множество функций (алгоритмов) их интерпретации.

КМ модель задает информацию о понятиях и фиксирует их как знания о научной дисциплине. Одним из способов создания КМ является онтология, термин которой возник в 1613г. и означает концептуализацию знаний в виде модели.

С момента появления вычислительной техники и компьютеров понятие модели и моделирования систем широко вошли во все сферы математического, технического и программного обеспечения.

Далее приводится описание моделей Msys и MF (Model Feature), отражающие базовые понятия предметной области знаний.

Моделирование, программирование

В связи с моделированием на первых ЭВМ трансляторов, ОС, отладчиков программ возникла новое направление - *системное программирование ПО* как научная дисциплина, которая включает:

- математическое обеспечение программ и систем, в том числе линейную алгебру, аналитическую геометрию и др.;

- функции управления, диспетчирования, мониторинга системными компонентами ПО (трансляторами, отладчиками, Базами данных и т.п.);

- системные функции ОС, ПО (компиляторы, тестировщики, верификаторы и др.) и ресурсы прикладных систем, комплексов программ и сетей (специальность 05.13.11);

- анализаторы правильности описания алгоритмов программ в ЯП (Prolog, Smalltalk, C++, Python и др.) и трансформации их к промежуточному коду;

- средства взаимодействия готовых сервисных и системных ресурсов Интернет.

Анализ методов моделирования - ООП

Подходы к моделированию	Основные сущности для моделирования систем	Метод анализа и построения
ООП Гради Буча (1987)	1. Базовые элементы ОМ: - класс - общие свойства; - метод экземпляризации; - наследование в классе; - инкапсуляция – скрытие свойств; - полиморфизм – многие. 2. Формальное определение - объектов и методов; - архитектуры системы; - модели системы. 3. Процессы: - анализ требований; - программирование; - эволюции системы; - модификации, выполнения.	1. Построение - ОМ из выделенных объектов - проверка правильности объектов; - взаимодействие объектов в классе, системе; - структуры системы. 2. Операции - описания объектов; - описание классов; - взаимодействия. 3. Обработка - объектов множества; - трансформации объектов; - интеграция объектов.

Анализ подходов к моделированию – UML, CORBA..

Подходы к моделированию	Основные сущности для моделирования систем	Метод анализа и построения
UML моделирование (1994). Rational Rose, RUP	Базовые элементы: - Use Case диаграммы: - классы, состояния; - взаимодействия; - компонентов; - последовательностей, развертки, выполнения, др.	Построение: - элементов системы; - архитектуры, модели системы; - файла интеграции; - диаграмм кооперации; - управление деятельностью.
OMT, OOAD, CORBA, COM, SADT, SSADM и др. (1992,..)	Виды технологий: - OMT – техника моделирования объектов; - OOAD – ОО анализ и проектирование; - CORBA – ОМ модель и брокер ORB.	Построение - моделей ОМ, - интерфейсов, - взаимодействия объектов.
Динамическая модель IDEF0, IDEF1 (1984)	Виды моделей: IDEF0 и IDEF1 – функциональная модель, - информационная модель системы, -модель анализа элементов, - модель синтеза.	Задание - графа функций, - димодель, - интерфейс и переход к сети Петри.

Анализ подходов к моделированию – models…

Подходы к моделированию	Основные сущности для моделирования систем	Метод анализа и построения
Модели проектирования систем (1999)	**Модели проектирования систем:** MDA (Model Driven Architecture); MDD (Model Driven Development); MDE (Model Driving Engineering); SOA (Service Oriented Architecture); SCA(Service Component Architecture); PIM (Platform Independed Model); PSM (Platform Specific Model) и др.	**Управление:** - проектированием; - моделированием; - трансформацией и конфигурационной сборкой элементов моделей; - вычисление.
Средства моделирования изменяемых систем	**Языки и инструменты:** - диаграмм. характеристики – FODA, ConIPF, модель Koala, KBuild, xADL и др.; - языки OVM, VSL, OWL, ConIPF с заданием вариантных точек в артефактах, ГОР и приложениях; - CASE, ORB Corba, COM, Config (http:// www. Sap.org) , Kbuild и др.; -VAMOS-Variability Management OS.	**Формирование:** - модели MF, - Config, - Kbuild, - Linux,Intel, WebSphere и др., - верификация, - тестирование, - оценка качества и надежности ОС.

Анализ подходов к моделированию - SPLE, GDM

Подходы к моделированию	Основные сущности для моделирования систем	Метод анализа и построения
Produc t Line/ Product Family SPLE (2004)	Продуктовая линия - Model Feature (вариабельности) артефактов, приложений; - модель архитектуры продукта. - вариант ПП.	Процессы линии: - анализ; - требований; - проектирования; - реализация; -- тестирование; -управление вариа-бельностью.
Конвейерная сборка Czernetski Generative programming (2005).	Методы генерации: - моделей MF, - конфигурации; - трансформации; - конфигурационного файла системы.	Процесс Доменной инженерии и Приложения: - создание ГОР; - трансформация их к среде; - тестирование ГОР; - конфигурация ГОР. - тестирование СПС.

Анализ подходов к моделированию – Grid, ОКМ

Подходы к моделированию	Основные сущности для моделирования систем	Метод анализа и построения
Grid – Европейский проект (2002…)	**Инфраструктура ETICS:** - интеграция; - тестирование; - конфигурирование; - запуск удаленной сборки; - точки входа в портал Grid.	**Процессы:** - анализа требований; - разработки ГОР; - стандартизация ГОР; - Сохранение ГОР в Базе проект
ОКМ – Объектно-компонентный метод (2003)	**Метод ОКМ:** - четыре уровня построения модели ОМ и ПС; - математ. операции ($\cup$ (A, B), $\cap$ (A, B), - A $\oplus$ B, - A \ B и др.); - модель вариабельности Mvar, Мпс, Мспс; - процесс управления вариантами и версиями; - верификация моделей; - тестирование ПС и СПС. - выполнение и сопровождение.	**ИТК- http://dragons.ru/ru** **Обработка объектов:** - описание объектов в ЯП; - трансформация объектов к компонентам; - сборка объектов и интерфейсов - конфигурация вариантов ПСв $P_1 = O_2 \cup O_5$, link P_1 =ln O'_5 $(O_2 \cup O_5)$, $P_2 = O_2 \cup O_6$, *link* P_2 =*ln*O'_6 $(O_2 \cup O_8)$.

Интеллектуализация знаний

К современным средствам представления знаний в терминах дескриптивной логики относятся: *FaCT* (на LISP), *FaCT++* (на C++), *CEL*, KAON-2 (JAVA), *MSPASS* (C) и *Pellet* (JAVA) и др. Необходимой составляющей представления знаний являются *прикладные web-сервисы*, ориентированные на коллективное решение задач в предметных областях знаний в современных средах (problem-solving environments, PSE) в мировом сообществе.

В процессе создания *моделей знаний* предметных областей используются система KBS (Knowledge-based systems), CommonKADS и др.

Они обеспечивают построение библиотек знаний, содержащих элементы решения задач, которые затем могут повторно использоваться (reuses, assets и др.) и в других областях знаний.

Формальное описание математической задачи для ЭВМ

Первоначально алгоритм математической задачи задавался с помощью схем Ляпунова (Янова), в виде конечного ориентированного графа, в вершинах которого размещались функции математической задачи, которые задавались формальными символами, формулами и математическими операциями из множества базовых.

Для граф-схемы была определена полная система преобразований формул в последовательность выполняемых операций на ЭВМ для получения выходных значений заданных переменных. Такую схему сначала кодировали в ручную в коды ЭВМ (МЭСМ, М-20, БЭСМ, Стрела и др.), чтобы провести решение небольших математических задач.

Таким образом, программирование алгоритма включает:

- описание базовых понятий элементов программы (процедур, подпрограмм, модулей и др.);

- методы алгоритмизации задач предметных областей (математическая логика, математика, физика и др.) и описание задач формальными средствами ЯП;

- формальный аппарат проведения анализа правильности описания программ в ЯП и поиск ошибок;

- инструменты трансформации (трансляция, компиляция, интерпретация) программ к выходному коду для последующего выполнения программы на ЭВМ.

Интеллектуальные ресурсы (ИНР)

Для разных предметных областей знаний (математика, медицина, биология, генетика и др.) разрабатываются **интеллектуальные ресурсы** (ИНР), типа Reuse, отображающие знания специалистов о функциях предметной области знаний, а также накоплены в библиотеках, хранилищах, Базах знаний и в общесистемных средах (IBM, MS.Net, Unux, Intel и др.) Интернет. Каждый ресурс ИНР проверяется на правильность и работоспособность (надежность).

Reuse — это готовый ИНР, который реализует алгоритм функции в некоторой предметной области знаний и может иметь общие переменные, которые образуют классы или множества. Внешние общие переменные и методы класса задаются в интерфейсе экземпляров класса. **ИНР** формально описывается в ЯП в виде модуля (объекта, компонента, сервиса и др.) и может взаимодействовать с другими через link (CALL/RPC/RMI) с передачей данных друг другу для обработки типов данных, которые относятся к фундаментальным (FDT) и общим типам данных (GDT) стандарта ISO/IEC 11404, а также с большими объемами данных (Big Data).

Информационные ресурсы (ИР)

ИР – это данные разного структуры и объема, в том числе и Big Data, представлены в Базах Данных, файлах, каталогах, таблицах, документах и т.п. ИР обрабатываются сервисными, системными, клиентскими и серверными службами Интернет и помещаются в Хранилищах данных, БД малых и больших объемов со структурированными и неструктурированными данными.

Передаваемые данные могут быть следующих типов в языке JAVA:

строки (xsd:string);

целые числа (xsd:int, xsd:long, xsd:short, xsd:integer, xsd:decimal), **числа с плавающей запятой** (xsd:float, xsd:double);

логический тип (xsd:boolean);

последовательности байт (xsd:base64Binary, xsd:hexBinary);

дата и время (xsd:time, xsd:date, xsd:g);

любые простые типы (xsd:anySimpleType).

Типами могут быть множества, последовательности, включающие фиксированное количество переменных простых типов.

Интерфейсы ресурсов

Интерфейс — это спецификатор информационной части ИНР — КПИ, компонента, reuses для обращения к другим ресурсам и обмена данными между ними. В нем содержатся операции вызова метода или функции (RPC/RMI) с параметрами, которые управляют внешними переменными экземпляра класса (выборка значения *get-метод,* присвоение значения *set-метод*, *Home-интерфейс* в языке JAVA и др.) при их взаимодействии [24, 25]. Интерфейс описывается в языке WSDL и в общем случае содержит:

***название* функции** (метода) и *ID* — идентификатор ресурса;

***описание** (спецификацию)* функции средствами ЯП;

параметры (входные и выходные) для передачи данных другим КПИ;

***язык* описания** КПИ (C, C++, Java, Python, Ruby и др.);

необязательные атрибуты (дата, состояние, версия, право доступа, автор, срок использования и т.п.).

Системное и теоретическое программирование

Теоретическое и системное программирование, как часть математической науки, создает математические абстракции алгоритмов описания программ на формальном языке (А.П.Ершов 1974). Цель этой науки - создать математически точную формулировку понятия программы и алгоритмов реализации функций (задач) предметной области [3].

Задача и метод формулируются в рамках теории предметной области, в которой математическая модель понятий сочетается с общематематическим знанием, задающим эту модель.
Алгоритмизация программы – это отображение точного знания о задачи и способах представления операций над ее элементами для последующего выполнения в компьютерной среде.
Термин программа и программирование впервые возникли на первых ЭВМ, на которых начали реализовывать математические, физические и производственные задачи.
Программирование – это творческий процесс постановки задачи, логики описания алгоритма в виде программы для получения решения задачи на компьютере.

ОС, ЯП, системное программирование[1-3]

ОС, ЯП, системное программирование[1-3]

ОС- 68, Диспетчер 68 МГУ (Королев Л.Н., В.П. Иванников и др.), ОС ИПМ для М-220 1967г. ИПМ (И.Б. Задыхайло, С.С. Камынин и др.), Монитор - Дубна (Н.Н. Говорун и В.П. Шириков). ОС «Днепр-2» (1967).

Системы программирования и Библиотеки программ

Программирующая программа для языка операторных схем Ляпунова А.А. (1953 г.) в МГУ (Э.З. Любимский, А.П. Ершов), трансляторы с языка Алгол-60 (ТА) для работы с программами на ЯП:

ТА1– С.С. Лавров (ЛГУ, 1962);

ТА2 – М.Р. Шура-Бура и Э.З. Любимский (ИПМ, 1963);

ТА3 - (Альфа-система) русская версия языка Алгол-60 (СО АН СССР, 1964);

ТА4 – Е.Л. Ющенко для УМШН, УВК «Днепр-2» (ИК АН УССР).

ТА5 – АЛМО З.Л. Любимский З.Л., 1966.

1.4 Технологии (схемные, структурные, модульные, компиляторные, объектные и др.)

Метод сборки модулей (1975-1980) через интерфейс с помощью 64 примитивных функций библиотеки примитивов преобразования разнородных типов данных в ЯП (АПРОП) -передана в 52 организации СССР и внедрена в IBM-360. В 1996 сделан IDL и стандарт Configuration (2009).

Синтез программ, как манипулирование знаниями задач, отображающих математические закономерности функций и их доказательство (1982).

Инженерия знаний[1-4]

Знания представляются в Базе знаний (БЗ), содержащей правила вывода и информацию о человеческом опыте и знаниях (стандарт ISO/IEC 2382-1:1993)[1].

На основе фактической информации БЗ и правил вывода делается автоматическое умозаключение об имеющихся или вновь вводимых фактах и осмысленной обработки информации[2].

Современные БЗ работают совместно с системами поиска и извлечения (Mining) информации, заданной онтологией в виде набора понятий и отношений. Онтологии описываются в специальных языках OWL, FODA, ODM, DSL. Проведена разработка онтологии домена ЖЦ ISO/IEC 12207 -2007 в OWL, средствами Protégé 2.3 с выходным результатом в языке XML[3].

Подход к автоматизации ЖЦ докладывался на Международной конференции «Science and Information-2015» в Лондоне и предложено IEEE сделать патент на идею интеллектуализации ЖЦ.

Онтологический подход набирает обороты Семантик Веб Интернет (www.semantic_web.org), как инструмент построения БЗ разных доменов.

---

1. ISO/IEC 2382-1:1993, Information technology — Vocabulary — Part 1: Fundamental terms.
2. Гаврилова и др. Базы знаний интеллектуальных систем // Учебник для вузов. — СПб.: Питер, 2000.
3. Lavrischeva E.M. Ontology of Domains. Ontological Description Sofware Engineering Domain. The Standard Life Cycle, Journal of Software Engineering and Applications, July 24, 2015.
4. Федоренко Д.Г., Астраханцев Н.А. Автоматическое извлечение новых концептов предметно-специфичных терминов. Труды ИСП РАН, том 25, 2013, стр. 167-178.

Современные нанотехнологии

Нанотехнология - ориентирована на получение веществ и устройств с заранее заданной атомарной архитектурой (Э.Дрекслер).

Атом – это 10^{10} =1 нанометра (нм), а бактерии это 10^{-9} нм. Частицы от 1 до 100 нанометров называют наночастицами. Это группирование молекул, маленьких элементов самоорганизовываться, чтобы получить новый материал или устройство (комплексные структуры молекул – синтез ДНК).

В перспективе программные элементы малые по размеру будут синтезироваться к виду «маленьких частиц» с заданной структурой и функциями.

Появились нано элементы - нейроны, кванторы, графены, кубиты (МФТИ, МИСиС и др.). Кубит состоит из нескольких джозефсоновских контактных элементов, разделенных тонким слоем диэлектрика. Квантовые биты кубита способны выполнять вычисления, которые не доступны современным компьютерам.

Кроме того, маленькие элементы типа чипа, транзисторов и др. используются для создания маленьких приборов и нейро компьютеров. Из маленьких элементов типа нано собираются новые технические приборы, устройства и используются в e-science (биология, генетика, медицина, физика и др.). Они способствовать лечению и улучшению здоровья общества и жизни на земле.

Пути развития Интернет[1-3]

1. Web 4.0 будет обеспечивать взаимодействие участников сети с помощью интеллектуальных агентов.
2. Корпоративные решения для облачных технологий (PaaS, SaaS), будут сращиваться с Интернет-пространством и использоваться для управления адаптивными Web-приложениями, в которых облачные сервисы будут взаимодействовать через веб-страницы агентов.
3. Internet of Things, Smart IoT будут поддерживать конкурентные APPS через: распределенные microservices; Hypercat мобильной связи; GSM-R для цифровых дорог. Промышленный Интернет (Industrial Internet) ориентируется автоматизацию новых концептов - «умная энергетика», «умный транспорт», «умные приборы», «умная промышленность», «умные дома», «умные города и медицина» и т.п.
4. Интернет 4.2 предоставляет технологическое конфигурирование и взаимодействие с устройствами обработки больших данных для превращения Big Data в Smart Data.
5. Семантика взаимодействия информационных объектов, которые задают цифровые проекции реальных или абстрактных объектов, будут взаимодействовать через Web-сервисы в языке OWL стандарта ISO 15926 Интернета 3.0.

1. E.M. Lavrischeva, I.B. Petrov, Ways of Development of Computer Technologies to Perspective Nano, Future Technologies Conference (FTC-2017), 29-30 November 2017| Vancouver, Canada.-p.539-547.
2. Боровков А.И. Компьютерный инжиниринг Учебное пособие // Боровков А.И., Клямин О.А. , Мельникова М.П. и др. /Computing Engineering.- San-Peterburg.-2012.- Россия, Полит. ул. 29.- 93 с.
3. Пройдаков Э.А. и др. Платформа «АН2», 2016, 139с.

Лаврищева Е.М.

Метод моделирования объектных систем с помощью графа

Метод моделирования графа из объектов-функций, как КПИ, и их интерфейсов включает :

- **формальный апарат Фреге спецификации семантики объектов функций;**
- **математические операции (объединения ∪, пересечения ∩, *вычитания ⊕, симметричного вычитания* \ и др.) для формирования из объектов проекций программных структур по вершинам графа с отображением к компонентной модели (СМ);**
- **объектную и компонентную алгебры для изменения КПИ КПИ (reuses, assets, servises, artifacts);**
- **операции сборки (конфигурирования) КПИ и преобразования данных к разным форматам платформ сред (VS.Net, IBM, Corba, Eclipse и др.) с помощью примитивных функций GDT ISO/IEC 11404–2007 к FDT ЯП (и обратно);**
- **модель MF (Model Feature) Msys для создания отдельных ПС в семействе СПС из готовых КПИ и решений (assets) к требованиям, документам, тестам и др., например на линии (ProductLine, ProductFamily) или APP/FAB.**

Согласно треугольника Фреге (1951) объект из множества $O=(O_1, O_2,\ldots, O_n)$ имеет:

***$Name_i$* - имя объекта,**

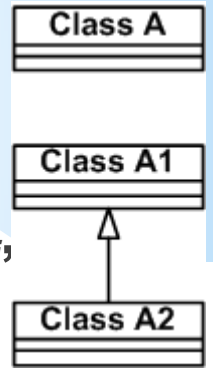
***Den_i* - денотат – сущность реальной действительности,**

***Con_i* - концепт, семантика (смысл) денотата.**

Денотаты декомпозируются -

$decds(O_i)$: $O_i \rightarrow \{O_{i1}, \ldots, O_{ik}\}$,

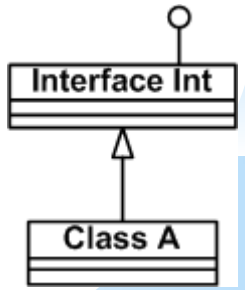
где $O_i = O_i(Name_i, Den_i, Con_i)$, $\forall\ j\ Con_{ij} = Con_i$; $Den_i = Den_{i1} \cup \ldots \cup Den_{ik}$.;



композируются - $comds(O_{i1}, \ldots, O_{ik})$: $\{O_{i1}, \ldots, O_{ik}\}$, $\rightarrow O_i$,

где $O_i = O_i(Name_i, Den_i, Con_i)$, $\forall\ j\ Con_i = Con_{ij}$; $Den_{i1} \cup \ldots \cup Den_{ik} = Den_i$.

Концепт может расширяться и сужаться специальными операциями

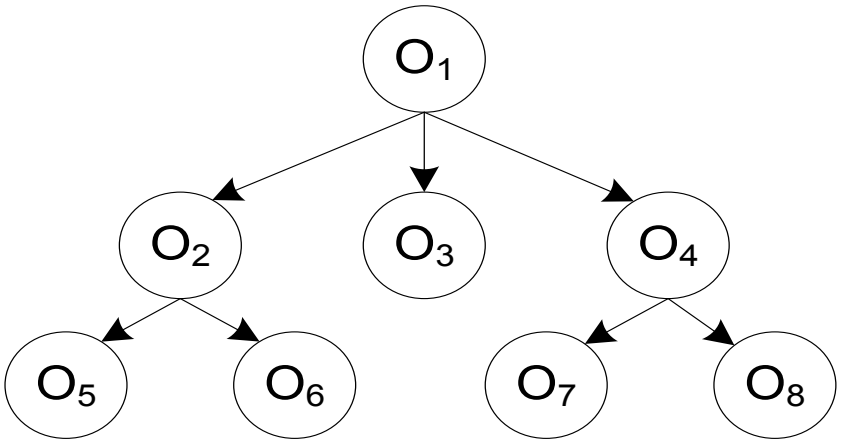


$conexp(O_i, P_t)$: $O_i \rightarrow O'_i$, где

$O_i = O_i`(Name_i, Den_i, Con_i)$,

$Con_{ij} \cup \{P_t\} = Con_i \ldots$

Объектный граф $G=(O)$



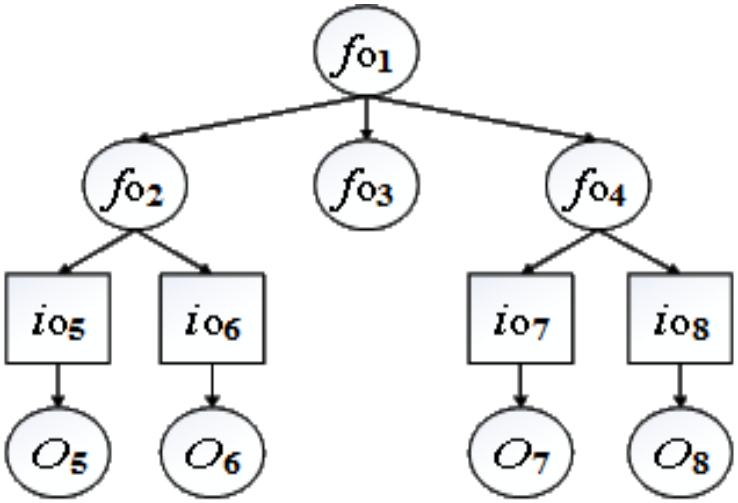
Свойства графа:

– для каждой вершины существует хотя бы одна связь с другой вершиной графа (стрелки);

– существует лишь одна вершина O_1 графа G, которая имеет статус множества объектов, отображающего предметную область знаний целом.

Каждый объект графа реализует функцию, которая задает алгоритм решения некоторой задачи предметной области.

После детализации отношений между элементами в граф добавляются интерфейсные вершины



Граф G на множестве функциональных и интерфейсных объектов

По графу G собираются программы P_0 - P_5 с использованием математической операции ∪, соответствующей операции *link*:

1) $P_0 = (P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5)$;
2) $P_1 = f_{O2} \cup f_{O5}$, *link* $P_1 = In\ i_{O5}\ (f_{O2} \cup f_{O5})$;
3) $P_2 = f_{O2} \cup f_{O6}$, *link* $P_2 = In\ i_{O6}\ (f_{O2} \cup f_{O8})$;
4) P_3 :
5) $P_4 = f_{O4} \cup f_{O7}$, *link* $P_4 = In\ i_{O7}\ (f_{O4} \cup f_{O7})$;
6) $P_5 = f_{O4} \cup f_{O8}$, *link* $P_4 = In\ i_{O8}\ (f_{O4} \cup f_{O8})$.

Программы P_0 - P_5 входят в состав графа *G* и модели ПС. Функции элементов Pi описываются в любом ЯП, а их интерфейсы в WSDL. После перевода с ЯП , они могут поступать в компонентную модель (СМ). Все элементы накапливаются в библиотеке ПС. Компонентная модель адекватна объектной в конфигурационном файле.

После 1992 года стали использоваться в основном зарубежные парадигмы - UML (Unified Modeling Language), OWL (Ontology Web Language), WSDL (Web Service Definition Language), XML, XSL, RDF, Semantic Web и др.

Новые ЯП – C++, C, C#, Basic, JAVA, Ada, Prolog, Python и др.

С 2004 года сформировались модели (**Model Variability**), on ProductLine/ ProductFamily производства программ, в Grid-системах и применяется на фабриках программ AppFab.

* Особое развитие получил научный сервис в среде Семантик Веб Интернета (www.semantic_web.org), позволяющий производить конфигурационную сборку сервисов (SOA, SCA) и Reuses в Веб-системах.

Разработаны интеллектуальные средства (Data Mining) для анализа данных, извлечения знаний (Mining knowledge) и данных в разных прикладных системах, **Cloud Computing, Big Data**, и др. размещаемых в новых глобальных хранилищах.

Обеспечение вариабельности

Вариабельность[1,2] **– это свойство (системы/ продукта) к расширению, изменению, приспособлению или конфигурированию для использования в определенном контексте и обеспечения последующей его эволюции.**

Вариантная характеристика **содержит множество видов одной характеристики и образует их коллекцию. Каждый член коллекции соответствует требованию к артефактам ПС, СПС и присоединяется к СПС в точках вариантности.**

Точка вариантности **– это место в ПС, по которой осуществляется выбор варианта по точкам в системе. Вариантная характеристика транслируется в коллекцию вариантов с количеством точек вариантности в ПС. Объекты, помеченные вариантными точками, встраиваются в контексте ПС для обеспечения версии (или варианта) продукта.**

Внешняя изменяемость **– это изменение *области артефактов,* которая понятна заказчику при выборе вариантов, которые ему требуются.**

Внутренняя изменяемость **– изменение артефактов, которые скрыты от заказчиков и находятся в пределах обязанностей stakeholders, представляющих поставщика.**

1. Лаврищева Е.М. Software Engineering компьютерных систем. Теория, Парадигмы, CASE – средства.-К.:2014.-287с.

2. Лаврищева Е.М., Петренко А.К. Моделирование семейств программных систем.-Труды ИСП РАН. Том.28, вып.6..-с.49-65.

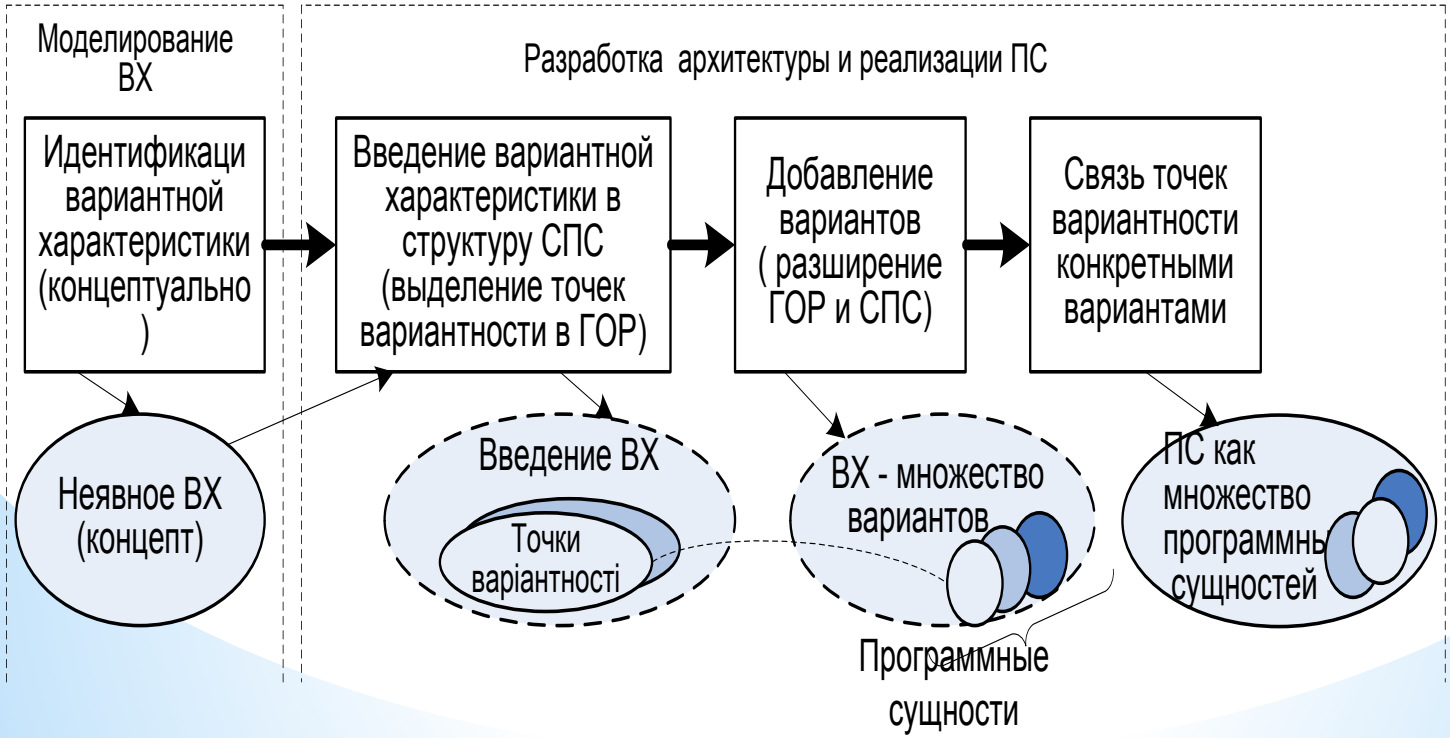
Процесс обеспечения Вариабельности

1. ***Идентификация вариабельности СПС,*** **определение вариантной характеристики и введение ее в коллекцию вариантов и точек вариации.**

2. Условия вариабельности СПС**, учитывающие текущие требования, потребности в ПС и перспективы изменения.**

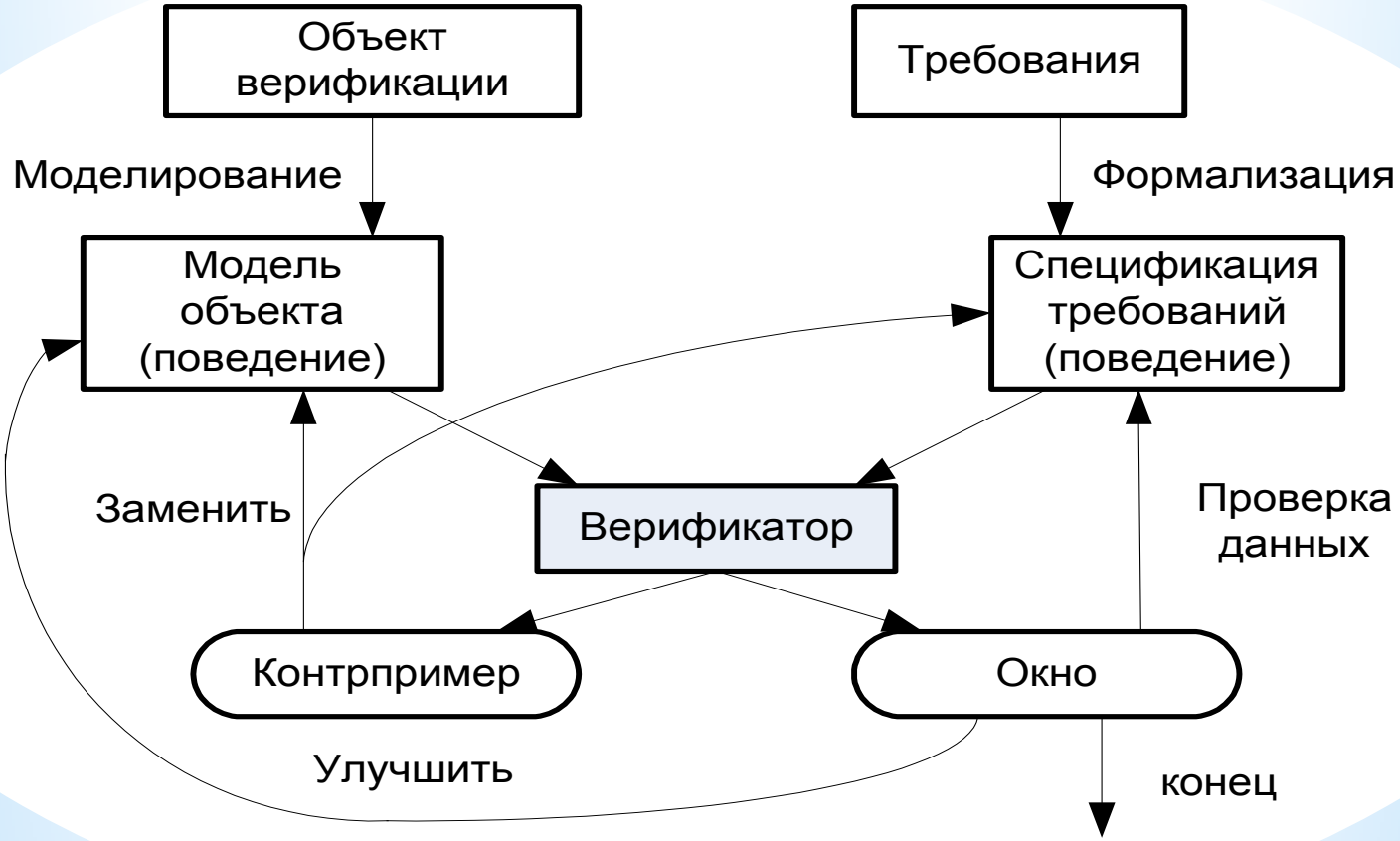
3. *Реализация вариабельности* по M_{var} и технологии реализации точек вариации в КПИ и ПС.

4. *Управление процессом вариабельности ПС и СПС.*



Верификация модели вариабельности MF

Верификатор метода model checking устанавливает соответствие спецификации модели MF объекта, требованиям и свойствам. Если соответствие удовлетворяется, то верификатор сообщает о правильности модели, в противном случае дается пояснение о возникшем несоответствии.



Тестирование ГОР, ПС и СПС

Концептуальная модель тестирования ПС и СПС из ГОР и КПИ имеет вид: $SFT = \langle TM; TD, TA, Env \rangle$,

где *TM* – подпроцесс управления тестированием;

TD и *TA* – подпроцессы тестирования артефактов и ПС;

Env – информационная среда процесса тестирования ПС.

Подпроцесс *TM* имеет унифицированное представление:

$$TM = \langle Task(TM, TD, TA), En(TM), CM(TM) \rangle,$$

$$En(TM) \cup En(TD) \cup En(TA) = Env$$

где *Task* – задачи, разрешимые при выполнении подпроцесса;

Env – информационная среда тестирования КПИ, ПС;

CM – подмодель координации операций подпроцессов.

Схема концептуальной среды *Env* задается выражением

$$Env = TG \cup SG \cup T \cup P \cup RG \cup RP,$$

где *TG* и *SG*– тесты активов (assets) и программных КПИ;

T и *P* – тесты и тестированные ПС;

RG и *RP* – отчеты о выполнении тестовых КПИ и тестов ПС.

Процесс управления вариабельностью

Функции управления вариабельностью ПС по Демингу

Plan-Do-Check-Act это:

1) F1– планирование реализации вариабельности в артефактах СПС (на уровнях инженерии ПрО и инженерии приложений);
2) F2 – реализация вариабельности в артефактах СПС;
3) F3 – мониторинг состояния СПС в аспекте архитектуры СПС;
4) F4 – актуализация СПС по результатам мониторинга.

Требования к управлению вариабельностью:

1) R1 – обоснованность – наличие объективных оснований принятия решений для F1 – F4;
2) R2 – согласованность – одинаковость способа выработки и реализации решений на всех уровнях абстракции и на этапах процесса разработки СПС ;
3) R3 – масштабность – независимость способа выработки и реализации этих решений от объема функциональных возможностей, охватываемых СПС;
4) R4 – трассирование связей вариабельности на уровнях и процессах СПС.

Переход ОМ к компонентной модели

Компонентная модель СМ задает вариантность ОМ ПС для платформы выходного кода с оценкой уровня вариабельности заданной заказчиком. Объекты и интерфейсы ОМ преобразуются к программному коду в СМ.

Модель СМ **СПС отражает реализацию методов объектов или КПИ и интерфейса между ними.**

Имеет следующий вид:

$$CM = \langle RC, In, ImC, Fim \rangle,$$

где *RC* – базовые компоненты множества компонентов С, которые соответствуют базовым объектам модели ОМ;

In **– интерфейс компонентов, среди параметров которого задается имя точки вариантности;**

ImC **– реализация базового компонента в заданной среде;**

Fim (·) **– функции преобразования входных и выходных параметров интерфейса и множество данных в сигнатуре этих функций.**

Операции над компонентами

Операция добавления компонента С к компонентой среде обозначается $\oplus$. Добавление компонента $C \oplus CE1 = CE2$ к среде выполняется согласно правил,

$CE2.NameSpace = \{C.CName\} \cup CE_1.NameSpace,$

$CE2.InRep = \{C.(CIni, CName)\} \cup CE_1.InRep,$

$CE2.ImRep = \{C.(Cj, CName)\} \cup CE_1.ImRep.$

Операция удаления компонента из компонентной среды обозначается знаком $\Diamond$ и подчиняется следующим правилам:

$CE1 \Diamond C = CE2,$

$\exists\, CName_k: CName_k \in CE1.NameSpace \;\; (CNamek = C.CName)$

$\Rightarrow CE2.NameSpace = CE1.NameSpace \setminus \{C.CName\} \;\; CE2.IntRep = CE1.IntRep.$

$\forall\, i: IntRepi.CName = C.CName)\; IntRepi\} \;\; CE2.ImpRep = CE1.ImRep \setminus$

$\forall\, j \;\&ImtRepj.CName = C.CName)\; ImRepj\}.$

Операции управления созданием ПС из компонентов в CASE-среде:

Link ($C1 \cup C2 \Rightarrow C3$) – объединение компонентов;

Conf ($C1 \cup C2 \cup C3$) – конфигурация компонентов;

Reing (Cn) – реинженерии и др.;

Prove PS (CPIi) – доказательство правильности ПС из КПИ;

Creat (Cj) – образовать класс компонентов и др.

Компонентная модель CM имеет вид:

$CM = \{CLm\{Lm1,\ldots, Lmn\}, P\{Pi, \ldots, Pm\}, CLn\{In1,\ldots, Ink\}, CDi$

где *CLm* – компоненты из множества реализаций в языках *L;*

P {*Pi, …, Pm*} – множество предикатов, определяющих процессы сборки или конфигурации КПИ, реализаций компонентов C*Lm* и интерфейсов *In*;

CLn – множество интерфейсов компонентов;

CDi – множество данных.

Внутренняя компонентная алгебра **имеет вид:**

$$\varphi 2 = \{CSet, CESet, \Omega 2\},$$

где ***Cset = {OldC, NewC}*** **– множества старых** ***OldC*** **и новых компонентов** ***NewC,*** **разработанных или преобразованных из старого к** ***NewC;***

OldC = (OldCName,OldIn, CFact, OldIm, CSer) **– интерфейсы реализации старых компонентов в серверной среде;**

NewC = (NewCName, NewIn, CFact, NewIm,CSer) **– интерфейсы реализации новых компонентов в серверной среде;**

$\Omega 2 = \{addIm, addIn, replIn, replIm\},$

где ***addim*** **– операции добавления реализации;**

addIn **– добавления интерфейса;**

replIm **– операция замещения реализации компонента,**

replIm **– замещения интерфейса.**

Алгебра эволюционного изменения имеет вид:

$$\varphi 3 = \{CSet, CESet, \Omega 3\},$$

где $\Omega 3 = \{O_{refac}, O_{Reing}, O_{Rever}\}$, в котором

O_{refac} – рефакторинг,

O_{Reing} – реинженерия,

O_{Rever} – реверсная инженерия компонентов.

Семантика этих операций состоит во внесении изменений (замены, добавления, переименования) компонентов и/или их интерфейсов и добавления новых компонентов.

Модель рефакторинга компонентов: $M_{refac} = \{O_{refac}, \{CSet = \{NewCn\}\}$,

где $(CSet, O_{refac})$ – элемент алгебры эволюции;

Orefac = {*AddOlmp, AddNlm, Replm, Addln*} – операции рефакторинга:

AddOlm – добавить старую реализацию компонента;

AddNlm – добавить новую реализация;

Addln – добавить интерфейс во множество C*Set*;

Replm – заменить реализацию компонентов в репозитории.

Модель реинженерии компонентов: $M_{Reing} = \{O_{Reing}, \{CSet = \{NewC_n\}\}$,

где $(CSet, O_{Reing})$ – элемент алгебры эволюции функциональности;

$Oreing = \{rewrite, restruc, adop, conver\}$ – операции реинженерии

Rewrite – переписать компонент;

Restruc – реструктуризировать структуру компонента;

Adop – адаптация компонента к среде выполнения;

Conver – конвертирование компонента в другой язык.

Модель реверсной инженерии компонентов:

$M_{Rever} = \{O_{Rever}, \{CSet = \{NewCn\}\}$,

где $(CSet, O_{Rever})$ – элемент компонентной алгебры эволюции;

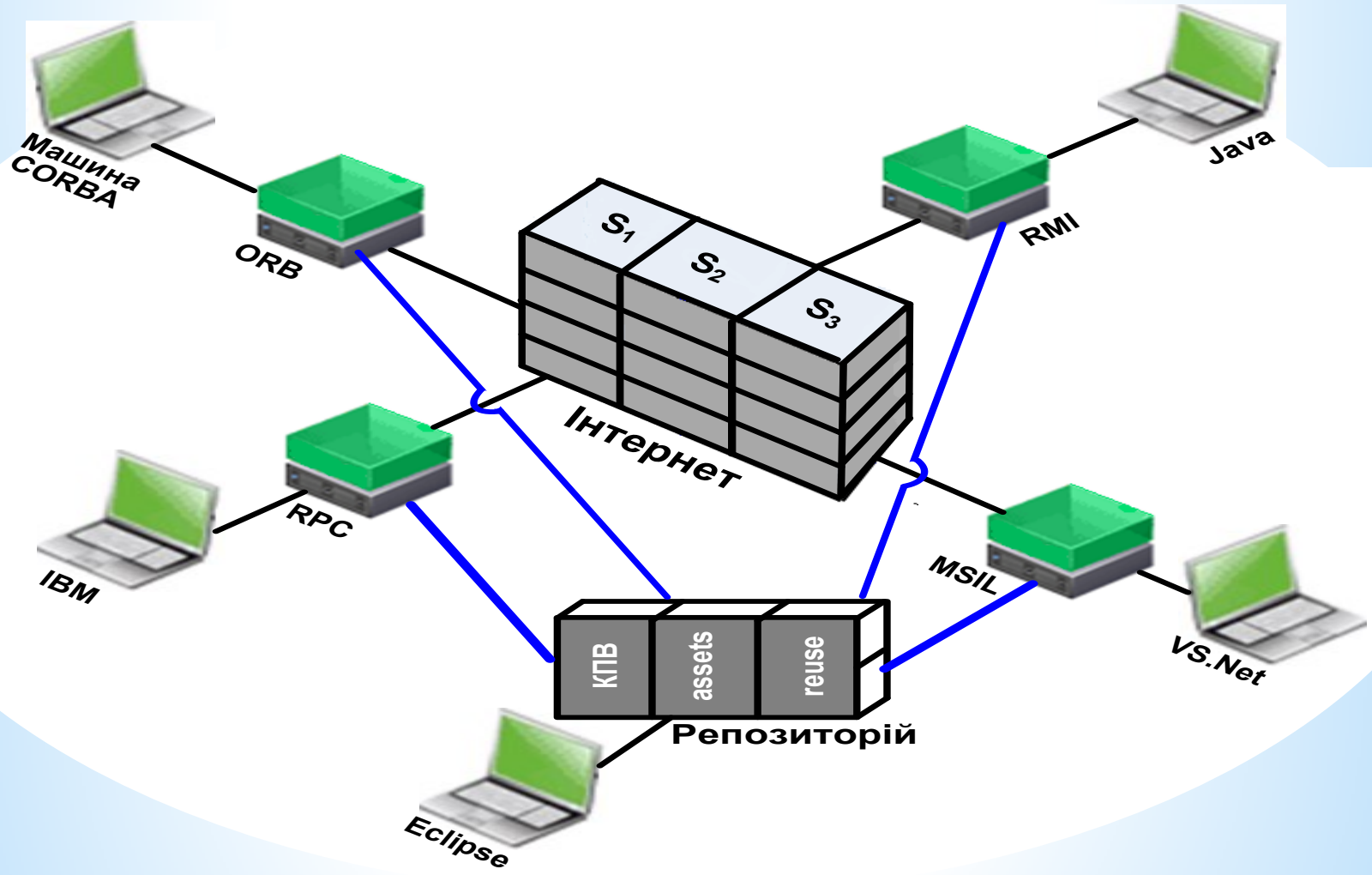
$O_{Rever} = \{restruc, designt\}$ – операции реверсной инженерии,

Design – проектирование компонента или системы;

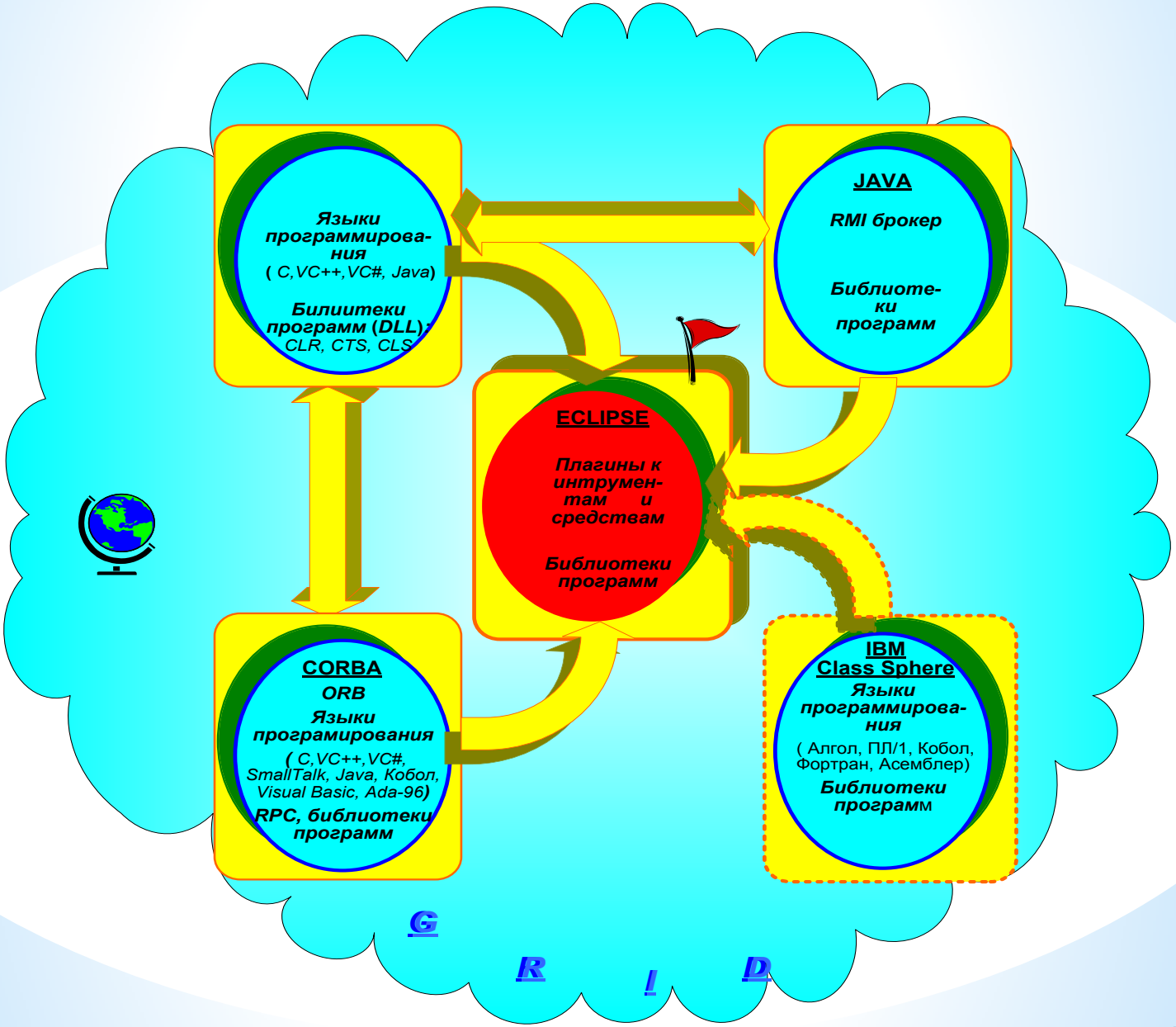
Restruc – трансформация структуры системы.

Взаимодействие систем и сред

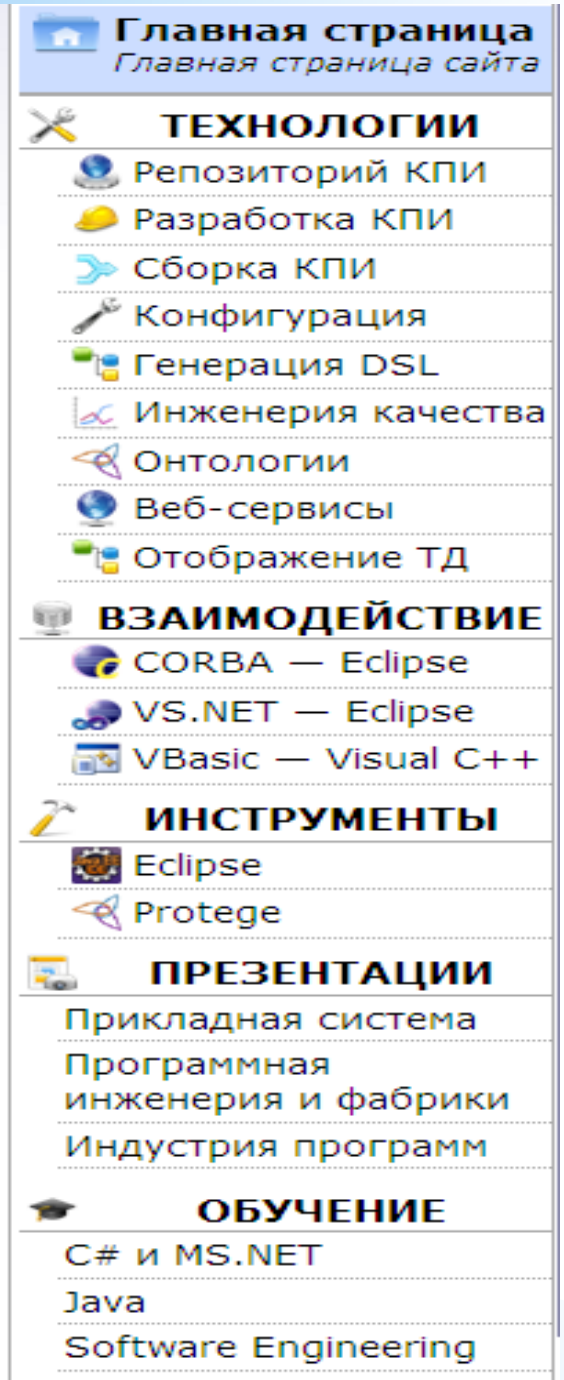
основано на интерфейсе и механизмах обработки даних, которые передаються по сети с общих и глобальных хранилищ типа Grid и Cloud.



Общая модель взаимодействия современных сред



Лаврищева Е.М.



Содержание разделов веб-сайта ИТК

ТЕХНОЛОГИИ ИТК:

Технология обслуживания репозиторию КПИ,
Технология разработки КПИ,
Технология сборки КПИ,
Технология конфигурирования КПИ,
Технология генерации описания КПИ в языке DSL,
Технология оценка затрат и качества,
Технология онтологии вычислит. геометрии, ЖЦ ISO/IEC12207
Технология веб–сервисов,
Технология генерация типов данных ISO/IEC 11404.

ВЗАИМОДЕЙСТВИЕ программ, систем и сред:

Модель Corba–Eclipse–Java,
Модель VS.Net C#–Eclipse,
Модель Basic–C++.

ІНСТРУМЕНТЫ ИТК:

Система Eclipse,
Система Protégé

ПРЕЗЕНТАЦИИ В ИТК:

Система ведения зарубежных командировок для НАНУ,
Слайды про ИТК, фабрики программ
Методологии построения ТЛ

ОБУЧЕНИЕ:

Технологии разработки программ в C# VS.Net, языка Java, электронного учебника с предмета «Программная инженерия» веб-сайту КНУ http://programsfactory.univ.kiev.ua.

Перспективные парадигмы программирования

Парадигма программирования — это совокупность понятий, математических терминов и теорий формального моделирования задач прикладной предметной области знаний.

Современные парадигмы:

Автоматная, агентная, алгебраическая, алгеброалгоритмическая парадигма, Аспектно-ориентированная, граф-схемная парадигма, Компонентная, сервисная парадигмы Композиционная, синтезирующая, сборочная.

Логическо-математическая парадигмы

Онтологическая парадигма

Объектная, семантическая, сценарная, параллельная, распределенная и др.

Agile Software Development

Это методологии:

eXtreme Programming - XP (экстремальное программирование), SCRUM, Dynamic Systems Development Method (DSDM),

Adaptive Software Development,

Crystal, Feature-Driven Development,

Pragmatic Programming.

Методологии Agile ориентированы на тесное взаимодействие команды разработчиков с пользователями на основе итеративной модели ЖЦ с приращениями и быстрой реакции на изменяющиеся требования и ProductLine подход к разработке ПО. В XP любой член группы может изменить не только свой код, но и код другого программиста и каждый модуль снабжается автономным тестом (*unit test*), обеспечивая возможность регрессионного тестирования модулей. Тесты пишут сами программисты и они имеют право написать теста для любого модуля. SCRUM — гибкая методика управления проектом с процессом разработки - анализ требований, проектирование, программирование, тестирование (http://agile.csc.ncsu.edu).

DSDM - метод разработки информационных систем консорциума DSDM.

Agile — манифест разработки ПО. www.agilemanifesto/org/iso.

Технологии фабрик программ в Интернет

1. Технология Product Line /Product Family (K. Pohl).
2. Технология генерации систем (K. Czarnecki).
3. Варианты ПС в Grid Европейского проекта №7.
4. Фабрики приложений AppFab в VS.Net, WSphereIBM и др.
5. Технология объектного моделирования ПП и ПС.
6. Фабрики потоковой сборки Дж. Гринфильда.
7. Фабрика use case Г. Ленца.
8. Фабрика программ http://programsfactory.univ.kiev.ua
9. Continious integration М. Фаулера.
10. Коммерческие фабрики программ ЕПАМ (Белоруссия).
11. Системные фабрики BEA WebL Oracle,

    SAP NetWeaver,

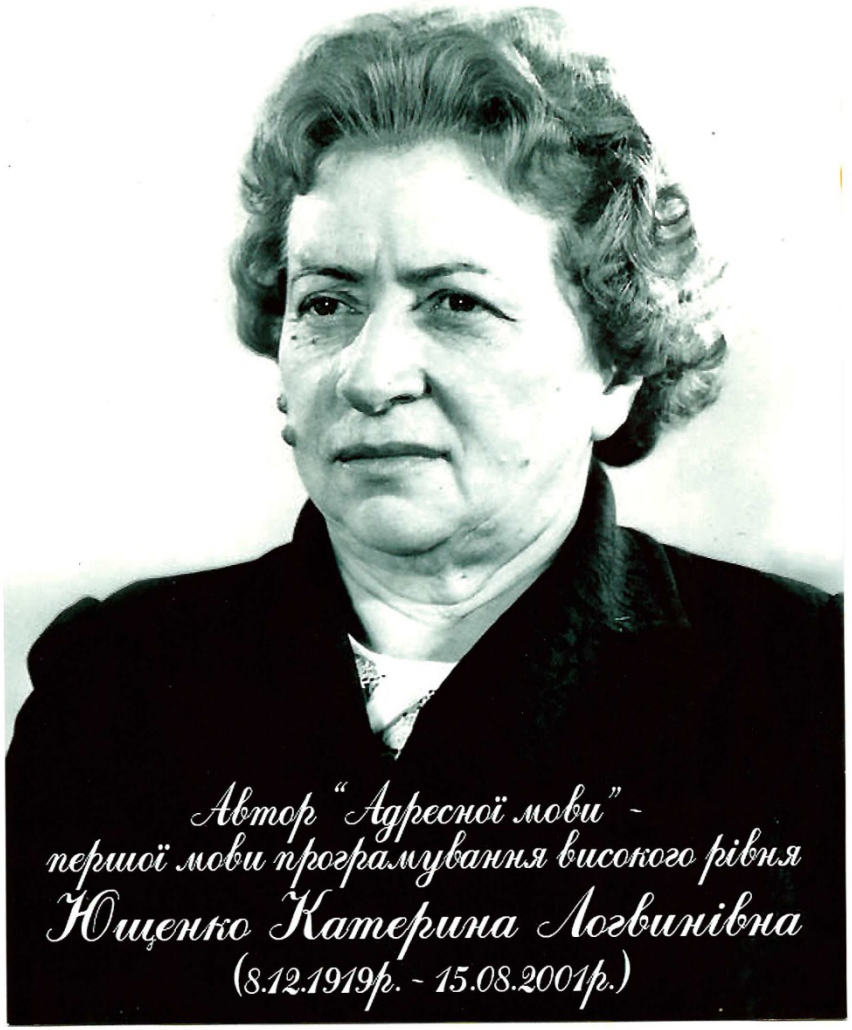
    ИВК "Юпитер и др.

БЛАГОДАРЮ ЗА ВНИМАНИЕ!

100-летие 8 декабря 2019года

1-программистке СССР

Ющенко Екатерине Логвиновне


Автор "Адресної мови" -
першої мови програмування високого рівня
Ющенко Катерина Логвинівна
(8.12.1919р. - 15.08.2001р.)

Е.Л.Ющенко с 1952 г. работала в лаборатории С. А. Лебедева - создателя первой в Европе ЭВМ "МЭСМ". И первые работы по теоретическому программированию начались на этой ЭВМ (100 ячеек оперативной и 64 ячейки односторонней сменно-наборной и разрядность каждой ячейки - 21 бит, быстродействие (100 операций/сек), электронных ламп до 6000).

В число первых задач входило разработка математическое обеспечение и математических задач, поставленных академиком М.В.Келдышем (Москва) и Гнеденко Н.Н. (Киев):

- внешняя баллистика;
- составление таблиц для статистического контроля;
- динамические задачи теории упругости;
- выбор оптимальных параметров шахтных канатов;
- определение областей устойчивости энергосистем (Куйбышевской ГЭС);
- расчет тепловых напряжений строительных конструкций;
- обработка геодезических наблюдений;
- расчет задач синтеза аммиака;
- оценка объемов земляных работ при проектировании автодорог.

Эти задачи реализовывались с помощью машинных программ. Сразу возникла идея создания ЯП "высокого уровня" и соответствующего транслятора.

Был разработан Адресный язык, со средствами аналитического преобразования выражений, дифференцирования, интегрирование выражений, анализа и обработки текстов и проверка синтаксической правильности программ. Совместно с профессором КГУ Л.И.Калужнин, читающего в 50-е годы XX века курс математической логики, был предложен формальный аппарат граф-схем программ А.А.Ляпунова. В 1955 году В.С.Королюк и Е.Л.Ющенко предложили адресный язык программирования, содержащий математические алгебраические для описания программ для ЭВМ и средства манипулирования адресами второго ранга.

Адресный язык предвосхитил появление языков программирования (ЯП) с аппаратом косвенной адресации, но и ассемблеров. На этом язык описывались трансляторы с ЯП и программы решения математических задач для компьютеров первого поколения ДНЕПР, КИЕВ, М20, УРАЛ, МИНСК. Первоначально этот язык и появившиеся ЯП (Алгол, Кобол, Фортран и ДР.) назывались алгоритмическими. Транслятоы с этих языков обеспечивали анализ правильности описания программ. Появились первые непроцедурные языки для описания механизмов обработки данных, основанной на только складывавшейся тогда файловой системе (прообраз информационных систем), для разнообразных инженерно-научных приложений на основе используемого в них математического обеспечения.

Основным достижением школы по теории программирования Ющенко Е.Л:

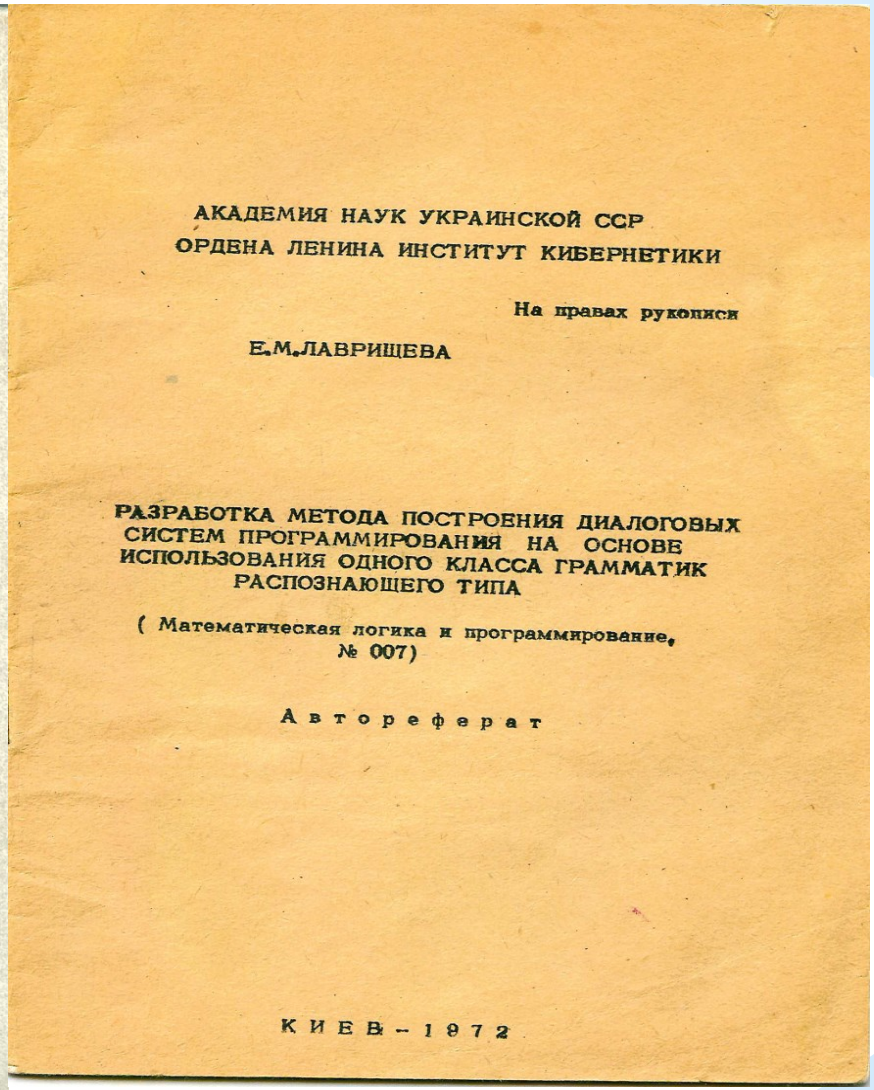
алгебро-грамматический аппарат СМ-грамматик,
системы алгоритмических алгебр,
методы анализа и генерации систем программирования и др.

Эти теории разрабатывались совместно с учениками (Г.Е.Цейтлин, Семик В.А., И.В. Вельбицкий, О.Л.Перевозчикова, Лаврищева Е.М., А.М.Халилов и др.) и представлены в главной теоретической монографии "Алгебра. Языки. Программирование" (В.М.Глушков, Г.Е.Цейтлин, Е.Л.Ющенко, 1974.-318с.). Монография издавалась три раза в СССР и в Германии.

За 40 лет научной школы ее подготовлено более 100 специалистов высшей квалификации (кандидатов и докторов физико-математических наук) с Украины и других республик СНГ (Содружество независимых государств) и дальнего зарубежья.

Под ее руководством для М-20 и других ЭВМ разработано ОМО и системы программирования. Я была главным разработчиком группы из 10 человек трансляторов АКД и АЛОЛ-60 для УВК Днепр-2 (1960-1972). Под ее руководством был разработан метод трансляции программ с ЯП Алгол, представлены на конференциях по программированию (1968-1992) и докладывался в Лондоне на конференции по ПО ЭВМ

(A method of analyzing programs based on a machine language E. M. Lavrishcheva and E. L. Yushchenko 1972, Volume 8, Number 2, Pages 219-223.).



АКАДЕМИЯ НАУК УКРАИНСКОЙ ССР
ОРДЕНА ЛЕНИНА ИНСТИТУТ КИБЕРНЕТИКИ

На правах рукописи

Е.М.ЛАВРИЩЕВА

РАЗРАБОТКА МЕТОДА ПОСТРОЕНИЯ ДИАЛОГОВЫХ СИСТЕМ ПРОГРАММИРОВАНИЯ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ОДНОГО КЛАССА ГРАММАТИК РАСПОЗНАЮЩЕГО ТИПА

(Математическая логика и программирование, № 007)

А в т о р е ф е р а т

К И Е В – 1 9 7 2