

ТЕОРИЯ МОДЕЛИРОВАНИЯ СИСТЕМ И СЕМЕЙСТВ СИСТЕМ

Лаврищева Е.М., Петренко А.К.

Открытая конференция ИСП РАН 2016

1.12 -2.12.2016

Доклад

доктора физ.- мат. наук, проф. МФТИ, гнс ИСП РАН
Лаврищевой Е.М.

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 – скрывание 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: классов, состояний, взаимодействия, компонентов, последовательностей, развертки, выполнения, др.	Построение - архитектуры, - модели системы, - файла конфигурации.
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 Driving Architecture); MDD (Model Driving 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, GOP 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
Product 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 вари- абельностью.
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), $- A \oplus B$, $- A \setminus B$ 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 $P_1 = O_2 \cup O_5,$ $\text{link } P_1 = \text{In } O'_5 (O_2 \cup O_5),$ $P_2 = O_2 \cup O_6,$ $\text{link } P_2 = \text{In } O'_6 (O_2 \cup O_8).$

Метод моделирования объектных систем ОКМ

ОКМ - метод проектирования ОМ (2007) систем из объектов, функции которых реализуются программными компонентами и интерфейсами.

ОКМ включает:

- метод Фреге спецификации объектов;
- математические операции (объединения $\cup$, пересечения $\cap$, вычитания $\ominus$, симметричного вычитания $\setminus$ и др.) для формирования сложных объектов и отображения ОМ к компонентной модели (СМ);
- объектную и компонентную алгебры для изменения ГОР и 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 линии (ProductLine, ProductFamily) создания отдельных ПС в семействе СПС из готовых КПИ, ГОР и решений (assets) – требований, документов, тестов и др.

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

$Name_i$ - имя объекта,

Den_i - денотат – сущность реальной действительности,

Con_i - концепт, семантика (смысл) денотата.



Денотаты декомпозируются -

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

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

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

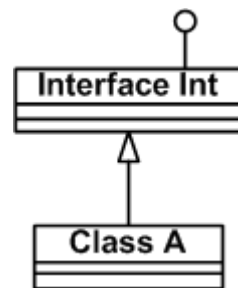
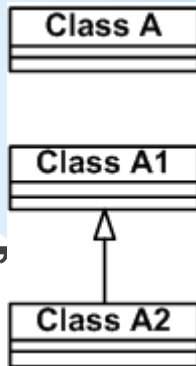
где $O_i = O_i(Name_i, Den_i, Con_i)$, $\forall j Con_i = Con_{ij}$; $Den_{i1} \cup \dots \cup Den_{ik} = Den_i$.

Концепт может расширяться и сужаться специальными операциями

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

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

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



Моделирование систем в ОКМ – это

представление домена в виде множества объектов со свойствами и характеристиками, которые необходимы и достаточны для их определения и идентификации, а также описания их поведения в рамках системы понятий и абстракций.

Объекты в ОКМ определяются на **четырёх уровнях** моделирования с привлечением логико-математических формализмов и уточнения функций, свойств и характеристик объектов.

К формальным уровням моделирования ОМ из объектов относятся:

1. **Обобщающий уровень**
2. **Структурный уровень**
3. **Характеристический уровень**
4. **Поведенческий уровень**

Уровни логико-математического моделирования

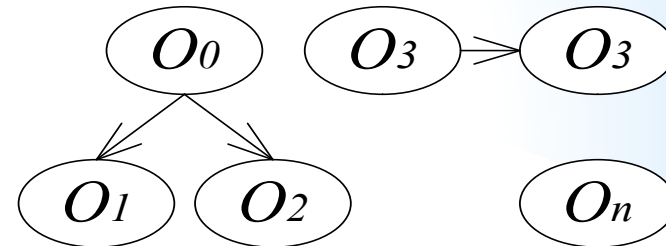
I. Обобщающий уровень

Определение множества объектов



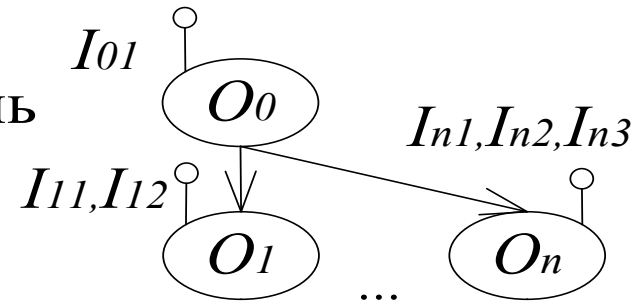
II. Структурный уровень

Определение иерархии объектов



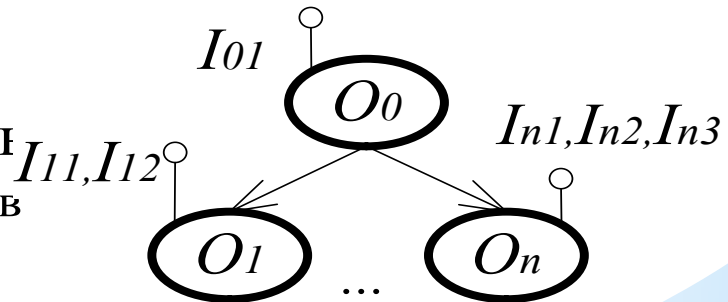
III. Характеристический уровень

Определение интерфейсов
объектов I_n



IV. Поведенческий уровень

Определение поведения объектов



На обобщающем уровне

выделяются объекты предметной области (ПрО) с применением аксиоматической теории Фреге и множеств Геделя-Бернаиса):

$$O = (O_0, O_1, \dots, O_n), \text{ где}$$

O – множество объектов ПрО,

O_0 – собственно предметная область,

выполняется условие $\forall i \exists j [(i > 0) \& (j \geq 0) \& (i \neq j) \& (O_i \in O_j)]$.

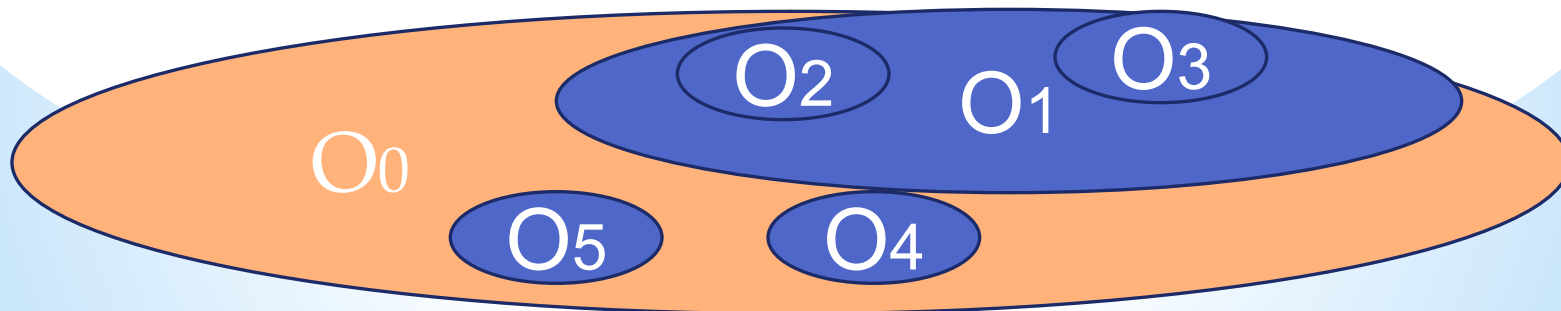
На структурном уровне

исключается элемент O_0 из множества O , получается новое множество

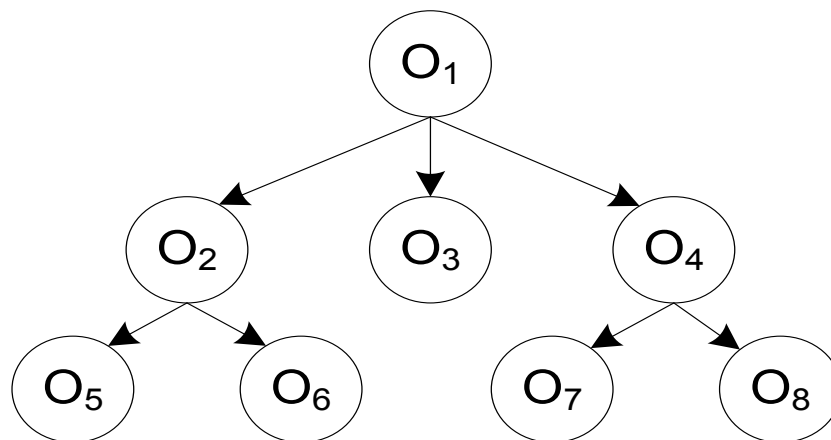
$$O' = (O_1, \dots, O_n).$$

$$\forall i \exists j [(i > 0) \& (j \geq 0) \& (i \neq j) \& (O_i \in O'_j)].$$

На множестве объектов O' определена алгебраическая система $\Sigma = (O', \Omega)$, где Ω – множество операций ($\cup, \cap, /, \diamond, \oplus, -$ и др.).



На структурном уровне построен объектный граф $G=(O)$



Этот граф задает:

множество вершин графа $G=(O)$, которыми являются объекты обобщенного и структурного уровней описания предметной области.

Свойства графа:

- для каждой вершины существует хотя бы одна связь (структурная) с другой вершиной графа (стрелки);
- существует лишь одна вершина O_1 графа G , которая имеет статус множества объектов, отображающего предметную область в целом.

На поведенческом уровне

для атрибутов объектов и их значений задаются **состояния объектов** с помощью диаграмм переходов состояний.

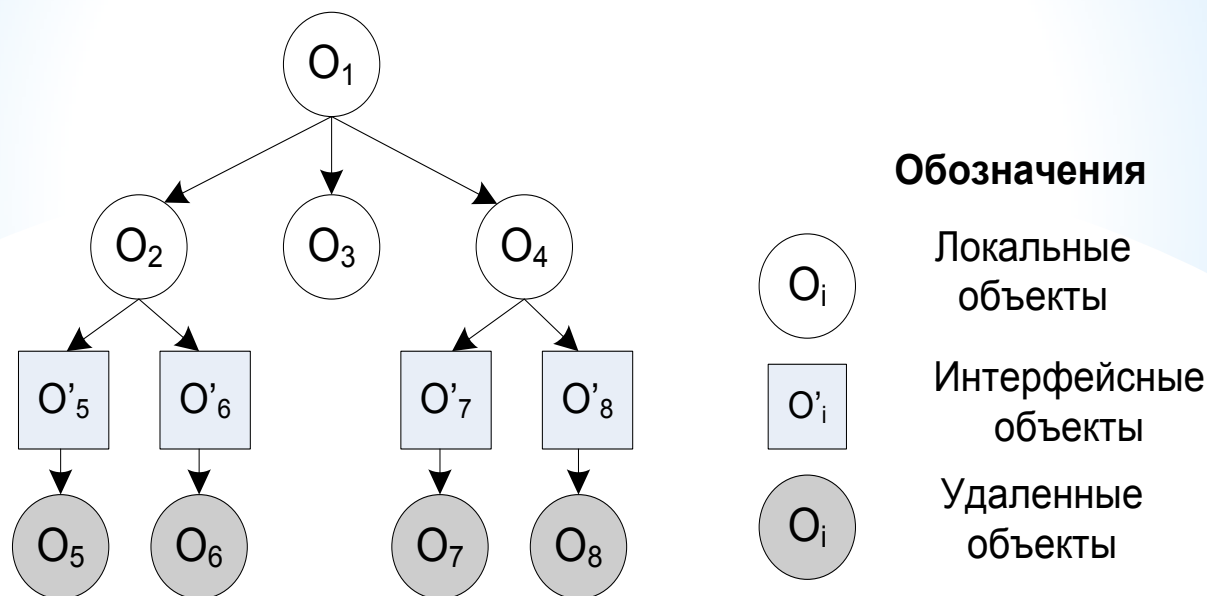
Взаимосвязи между объектами задаются бинарными предикатами свойства объектов O , которые детализируют взаимосвязи между состояниями объектов.

Параметр **времени** t (*Timer*) вносится в ОМ для рассылки специальных сообщений в текущий момент времени.

Каждый объект содержит метод, который анализирует полученное значение и выполняет переход к другому состоянию или оставляет его без изменений. Состояние имеет статический и/или динамический атрибут, который может формировать время и свойства объектов.

Эти состояния задаются в описании объектов на данном уровне графовой модели ОМ.

Граф объектной модели



Результат связи двух объектов O_2 , O_5 интерфейсный объект (например, O'_5), у которого множество входных интерфейсов O_2 совпадает с множеством выходных интерфейсов O_5 и наоборот

Аксиома. Сборка объектов является корректной, если объект-передатчик O_2 полностью обеспечивает интерфейсный сервис, который необходим объекту-приемнику O_5 для выполнения функций объекта.

Объекты могут унаследовать интерфейсы других объектов, тогда последние предоставляют сервис для всего множества исходных интерфейсов.

Описание вариантов программ по графу G

В данном графе G заданы:

- $O_1, O_2, O_3, O_4, O_5, O_6, O_7, O_8$ - функциональные объекты/КПИ;
- O'_5, O'_6, O'_7, O'_8 - интерфейсные объекты, которые размещаются в репозитории интерфейсов, а дуги соответствуют связям между видами объектов.

Элементы графа $O_1 - O_8$ описываются в ЯП (Fortran, Basic, C++ и др.), а интерфейсные объекты $O'_5 - O'_8$ - в языке IDL (Interface Definition Language). По графу G можно построить программы $P_1 - P_5$ с использованием мат. операций $\cup$ объединения *link* :

$$P_1 = O_2 \cup O_5, \text{ link } P_1 = \text{In} O'_5 (O_2 \cup O_5);$$

$$P_2 = O_2 \cup O_6, \text{ link } P_2 = \text{In} O'_6 (O_2 \cup O_8);$$

$$P_3 :$$

$$P_4 = O_4 \cup O_7, \text{ link } P_4 = \text{In} O'_7 (O_4 \cup O_7);$$

$$P_5 = O_4 \cup O_8, \text{ link } P_4 = \text{In} O'_8 (O_4 \cup O_8);$$

$$P_0 = (P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5).$$

Конечный **граф G модели OM** имеет вид:

$$OM = \langle G_{t1}, G_{t2}, G_{t3}, G_{t4} \rangle,$$

где G_{t1} – граф объектов ПрО на обобщающем уровне ($t=1$);

G_{t2} – граф характеристического уровня ($t=2$);

G_{t3} – граф структурного уровня ($t=3$);

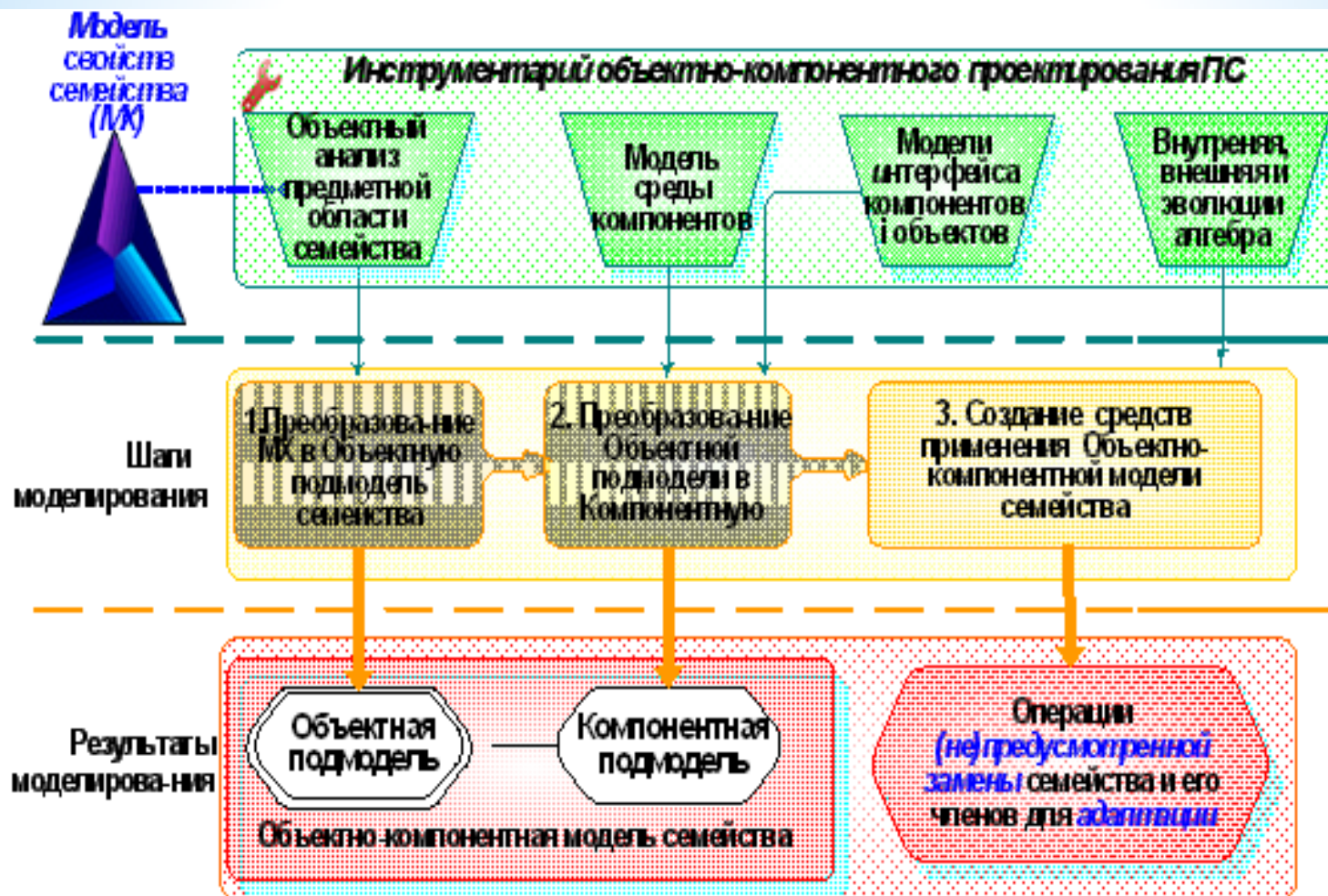
G_{t4} – граф интерфейсов для взаимосвязи объектов на поведенческом уровне ($t=4$).

Объектам функций G_{t1} и их характеристикам соответствуют методы и данные (уровня 2 и 3), необходимые для реализации отдельных систем или семейства СПС с обеспечением их взаимодействия.

Определены операции над элементами класса:

$Oclass_i = \{ClassName_i, Method_i, Field_i\}$, где $Imethod_i = Method_i \cup \{get\langle Pfieldn_i \rangle\} \cup \{set\langle Pfieldn_i \rangle\}$ и ему сопоставляется интерфейс $Ifunc_i$, состоящий из прототипов методов, входящих в $Imethod_i$.

Реализация ОКМ



Обеспечение вариабельности

Вариабельность - это свойство (системы/продукта) к расширению, изменению, приспособлению или конфигурированию для использования в определенном контексте и обеспечения последующей его эволюции (K.Pohl in SPLE).

Точка вариантности – это место в ПС, по которой осуществляется выбор варианта в системе.

Вариантная характеристика артефакта или приложения транслируется в коллекцию вариантов ПС или ПП.

Модель варибельности ПС

Модель варибельности M_{var} системы ПС это:

$$M_{var} = (SV; AV) \text{ задает}$$

1) уровень изменяемости продуктов ПС с учетом требований

$$SV = \langle G_1; \langle G_t, TR_t \rangle, t=2, \dots, 4; Con; Dep \rangle$$

$G_t = (F_t, L_t)$ – граф структуры уровня $t=1, \dots, 4$ модели MF ПС семейства ПС/ПП,

TR_t – двусторонние связи трассирования артефактов уровней $t-1$ и t ,

Con, Dep – предикаты на $\otimes F_t$, задающие ограничения и зависимости артефактов;

2) степень изменяемости артефактов в вариантных точках ГОР и ПС

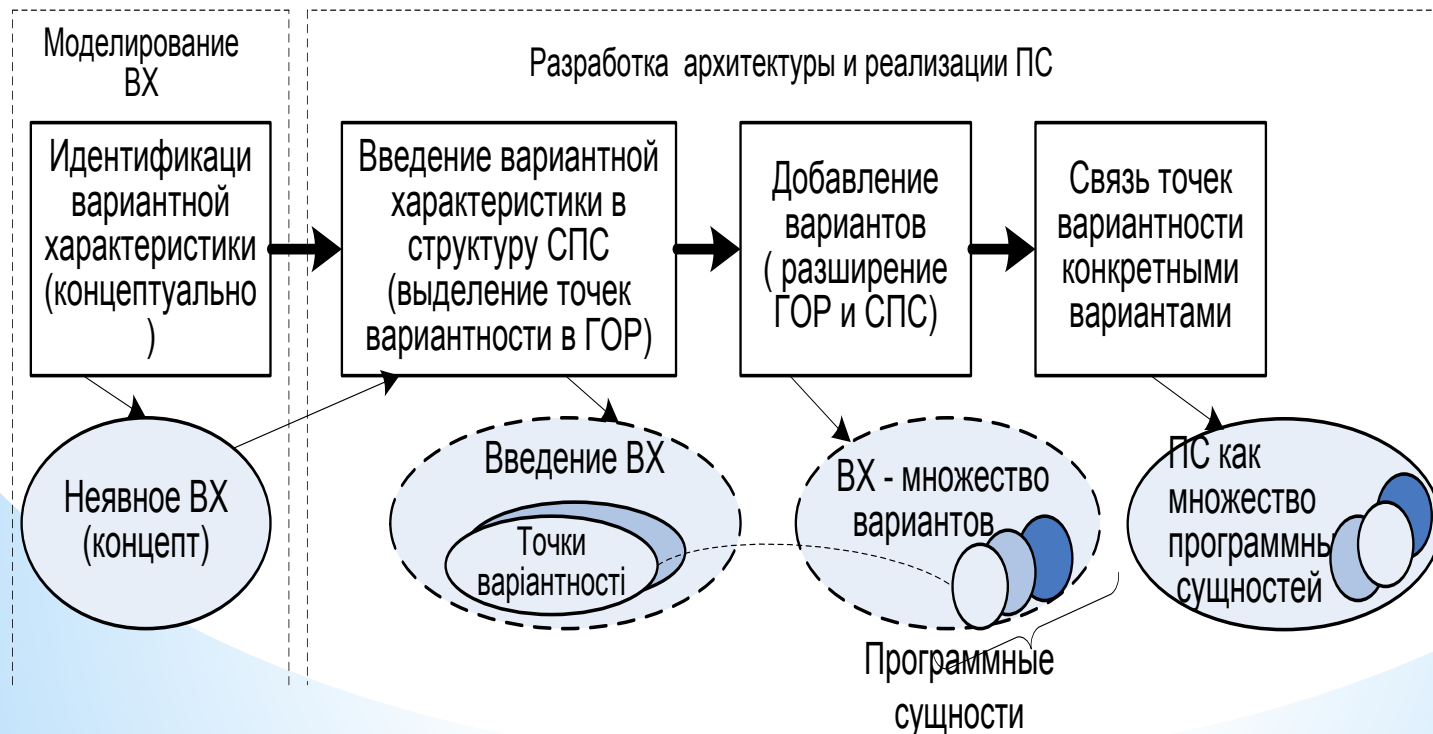
$$AV = \langle g_1; \langle g_t, tr_t \rangle \langle p_t, tr_t \rangle \rangle, t = 2, \dots, m,$$

где g_t и p_t – подграфы G_t графа G с артефактами типа m с идентификатором id_m реализации ПС/ПП;

Управление варибельностью ПС (в артефактах и структуре) - основа разработки ПС из ГОР.

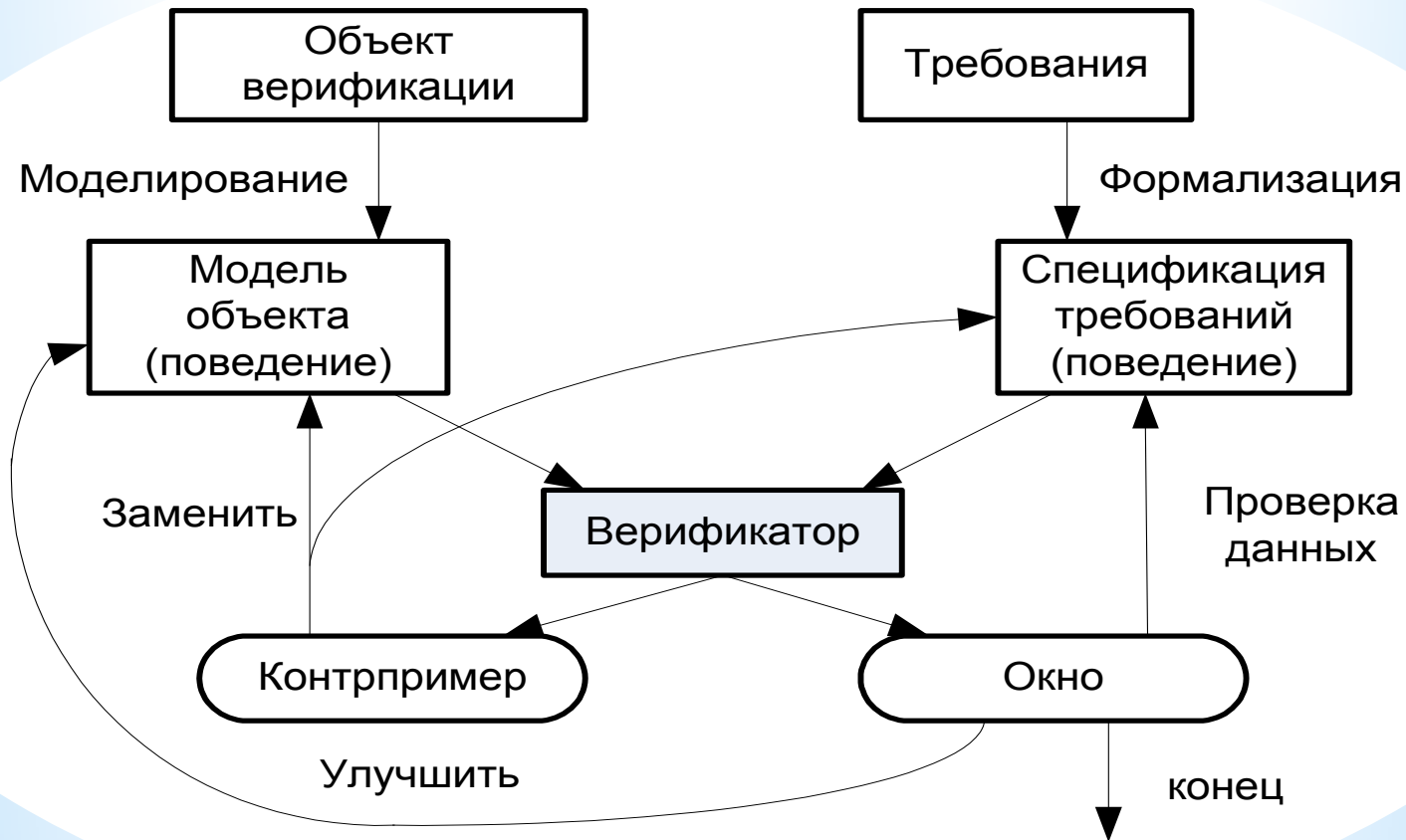
Процесс обеспечения Вариабельности

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 тестируемые ПС;

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

Оценка оптимального времени тестирования

Оптимальное время тестирования определяется по формуле :

$$\Delta R(t_0|t_e) = C_m (\mu(t_0) - \mu(t_0 + t_e) + \mu(t_e)), \text{ где}$$

$$C_m = \sum_{j=J}^n [P(S_j) \sum_{j=J}^r P(H_j | S_j) \cdot C_j] - \text{взнос модуля в риск отказов ПС;}$$

$\mu(t)$ – функция роста надежности;

$\lambda(t) = d\mu(t)/dt$ – **интенсивность отказов** модуля;

$P(S_i)$ – вероятность того, что при реализации сценария S_i ($i=1, 2, \dots, n$)
будет выполняться данный модуль;

$P(H_j|S_i)$ – условная вероятность того, что при реализации сценария S_i
причиной возникновения угрозы H_j будет отказ данного модуля;

C_j – стоимость последствий реализации угрозы H_j ($j=1, 2, \dots, r$);

$C(t_e)$ – полная стоимость тестирования в течение времени t_e ;

Прибыль от тестирования определяется по формуле:

$$K(t_0 | t_e) = \Delta R(t_0 | t_e) - C(t_e) = C_m (\mu(t_0) - \mu(t_0 + t_e) + \mu(t_e)) - c_1 t_e - c_2 \mu(t_e).$$

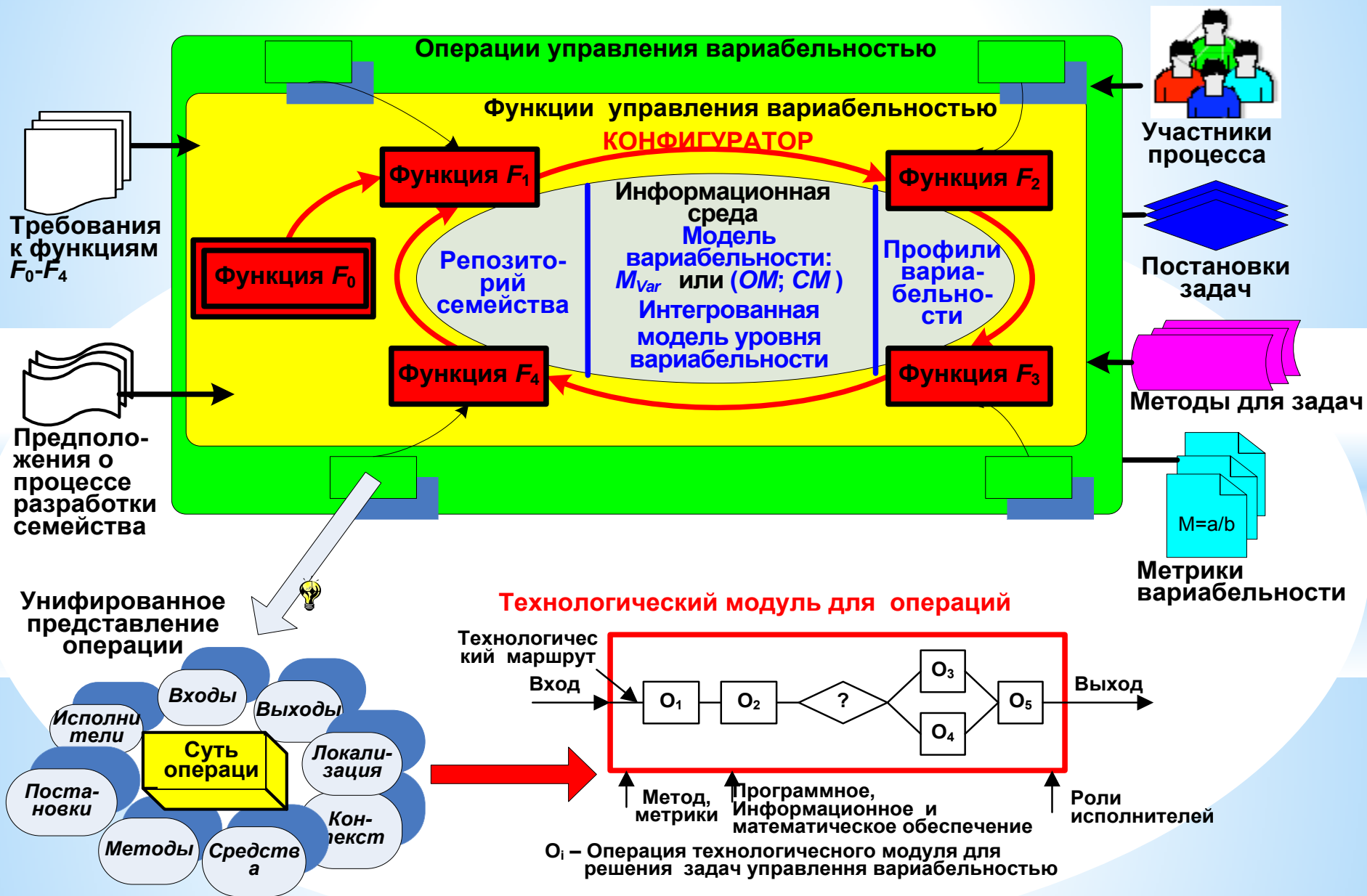
Походная функции $K(t_0 | t_e)$ по t_e равняется:

$$K'(t_0 | t_e) = C_m (\lambda(t_e) - \lambda(t_0 + t_e)) - c_1 - c_2 \lambda(t_e).$$

Значение оптимального времени t_e^* получается как решение уравнения

$$K'(t_0 | t_e) = 0 \text{ в зависимости от функции } \lambda(t).$$

Схема управления варибельностью СПС



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 *C*, 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.

Компонентные модели

Модель компонента задает типовые решения, касающиеся функциональной сущности компонента, его структуры, свойств и характеристик и имеет вид.

$$M_{Comp} = (C_{name}, C_{in}, C_{fact}, C_{im}, C_{ser}),$$

где C_{Name} – уникальное имя компонента;

$C_{In} = \{C_{Ini}\}$ – множество интерфейсов компонента;

C_{Fact} – управление экземплярами компонента;

$C_{Im} = \{C_{Inj}\}$ – множество реализаций компонента;

$C_{Ser} = \{C_{Serr}\}$ – множество системных сервисов.

Множество $C_{in} = C_{Ini} \cup C_{Ino}$ состоит из входных C_{ini} и выходных C_{ino} интерфейсов.

Модель интерфейса компонента имеет вид:

$CIn = (InName, InFun, CIn, InSpec),$

где *InName* – имя интерфейса;

InFun – функциональность (метод) интерфейса;

CIn – интерфейс управления экземплярами компонента;

InSpec – спецификация интерфейса (описания типов, констант, сигнатур методов и т. д.).

Интерфейс задает операции управления экземплярами:

$CIn = \{Locate, Create, Remove\},$

где *Locate* - поиск и определение экземпляра компонента;

Create - создание экземпляра компонента;

Remove - удаление экземпляра компонента.

Компонентная среда имеет вид:

$CE = (NameSpace, InRep, ImRep, CSer, CSerIm),$

где $NameSpace$ – множество имен компонентов среды;

$InRep = \{InRep_i\}$ – репозиторий интерфейсов;

$ImRep = \{ImRep_j\}$ – репозиторий реализаций;

$CSer = \{CSer_k\}$ – системные сервисы;

$CSerIm = \{CSerImr\}$ – реализации сервисов.

Компонентная среда – это множество серверов приложений, где разворачиваются компоненты, контейнеры, экземпляры которых обеспечивают реализацию функциональности компонента.

Каркас компонентной среды

- это совокупность имен компонентов, интерфейсов и реализаций как пустых множеств $FW = (\emptyset, \emptyset, \emptyset, CSer, CSerIm)$.

Если каркасы $FW_1 = (\emptyset, \emptyset, \emptyset, CSer_1, CSerImp_1)$ и $FW_2 = (\emptyset, \emptyset, \emptyset, CSer_2, CSerIm_2)$, то FW_1 совместим с FW_2 , если существует отображение $SMap: CSer_1 \rightarrow CSer_2$ такое, что $SMap(CSer_1) \subseteq CSer_2$.

В $CSer_1, CSer_2$ входит сервис именования и между этими сервисами должна существовать такая связь, при которой сервис первой среды – это именование объектов среды и наоборот.

Любая компонентная среда использует те же сервисы, а их связи определяют их совместимость. Взаимодействие компонентов, расположенных в разных серверах, поддерживают сервисы этих серверов. Если совместимость между серверными сервисами существует, то такие компоненты могут входить в состав общей среды компонента.

Операции над компонентами

Операция добавления компонента C к компонентой среде обозначается $\oplus$. Добавление компонента $C \oplus CE_1 = CE_2$ к среде выполняется согласно правил,

$$CE_2.NameSpace = \{C.CName\} \cup CE_1.NameSpace,$$

$$CE_2.InRep = \{C.(CIni, CName)\} \cup CE_1.InRep,$$

$$CE_2.ImRep = \{C.(Cj, CName)\} \cup CE_1.ImRep.$$

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

$$CE_1 \diamond C = CE_2,$$

$$\exists CName_k: CName_k \in CE_1.NameSpace \ (CName_k = C.CName)$$

$$\Rightarrow CE_2.NameSpace = CE_1.NameSpace \setminus \{C.CName\} \quad CE_2.IntRep = CE_1.IntRep.$$

$$\forall i: IntRep_i.CName = C.CName) \ IntRep_i\} \quad CE_2.ImpRep = CE_1.ImRep \setminus$$

$$\forall j \ \&ImtRep_j.CName = C.CName) \ ImRep_j\}.$$

Операция замены компонента задается знаком "–" и имеет вид: $CE.NameSpace(C1) - C2 = (CE \diamond C1) \oplus C2$.

Операция объединения ($\cup$) компонентных сред выполняется согласно следующих правил:

$$CE1 \cup CE2 = CE3 ,$$

$$CE3.NameSpace = CE1.NameSpace \cup CE2.NameSpace,$$

$$CE3.InRep = CE1.InRep \cup CE2.InRep,$$

$$CE3.ImRep = CE1.ImRep \cup CE2.ImRep.$$

Теорема. Операция объединения сред ассоциативна:

$$(CE1 \cup CE2) \cup CE3 = CE1 \cup (CE2 \cup CE3).$$

Доказательство

$$\forall C \in (CE1 \cup CE2) \cup CE3 \Rightarrow C \in CE1 \vee C \in CE2 \vee C \in CE3;$$

$$\forall C \in CE1 \cup (CE2 \cup CE3) \Rightarrow C \in CE1 \vee C \in CE2 \vee C \in CE3.$$

Операция объединения сред коммутативна

$$CE1 \cup CE2 = CE2 \cup CE1.$$

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 (CPi) – доказательство правильности ПС из КПИ;

Creat (Cj) – образовать класс компонентов и др.

Компонентная модель CM имеет вид:

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

где CLm – компоненты из множества реализаций в языках L ;

$P\{Pi, \dots, Pm\}$ – множество предикатов, определяющих процессы сборки или конфигурации КПИ, реализаций компонентов CLm и интерфейсов In ;

CLn – множество интерфейсов компонентов;

CDi – множество данных.

Многоосновная компонентная алгебра

$$\Sigma = \{\varphi_1, \varphi_2, \varphi_3\},$$

где $\varphi_1 = \{CSet, CSet, \Omega_1\}$ – внешняя алгебра,

$\varphi_2 = \{CSet, CSet, \Omega_2\}$ – внутренняя алгебра,

$\varphi_3 = \{Set, CSet, \Omega_3\}$ – алгебра эволюции компонентов.

Внешняя компонентная алгебра включает операции над компонентами среды:

$$\varphi_1 = \{CSet, CSet, \Omega_1\},$$

где $CSet$ – множество компонентов;

$CSet$ – множество компонентных сред;

Ω_1 – множество операций:

$CSet_2 = Cset \oplus CSet_1$ - инсталляция (развертывание);

$CSet_3 = CSet_1 \cup CSet_2$ - объединение сред ;

$CSet_2 = CSet_1 \setminus CSet$ удаления компонента из среды.

Внутренняя компонентная алгебра имеет вид:

$$\Phi_2 = \{CSet, CSet, \Omega_2\},$$

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,

$replIn$ – замещения интерфейса.

Алгебра эволюционного изменения имеет вид:

$$\Phi_3 = \{CSet, CSet, \Omega\},$$

где $\Omega = \{O_{refac}, O_{Reing}, O_{Rever}\}$, 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;

$O_{refac} = \{AddOIm, AddNIm, ReplIm, AddIn\}$ – операции рефакторинга:

$AddOIm$ – добавить старую реализацию компонента;

$AddNIm$ – добавить новую реализация;

$AddIn$ – добавить интерфейс во множество $CSet$;

$ReplIm$ – заменить реализацию компонентов в репозитории.

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

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

$O_{Reing} = \{rewrite, restruc, adop, conver\}$ – 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;

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

Design – проектирование компонента или системы;

Restruc – трансформация структуры системы.

Доказательство свойств компонентов

Члены множества S компонентов хранятся в репозитории на разных языках, с их именами и интерфейсами. Они могут изменяться и заменяться новыми компонентами для получения разных вариантов продуктов ПС.

Аксиома. Два компонента $C1$ и $C2$ являются тождественными (равными), если тождественны их соответствующие составные.

Как следствие, замена $C1$ на $C2$ не влияет на компонентную модель, к которой принадлежит компонент $C1$.

Средства описания КПИ:

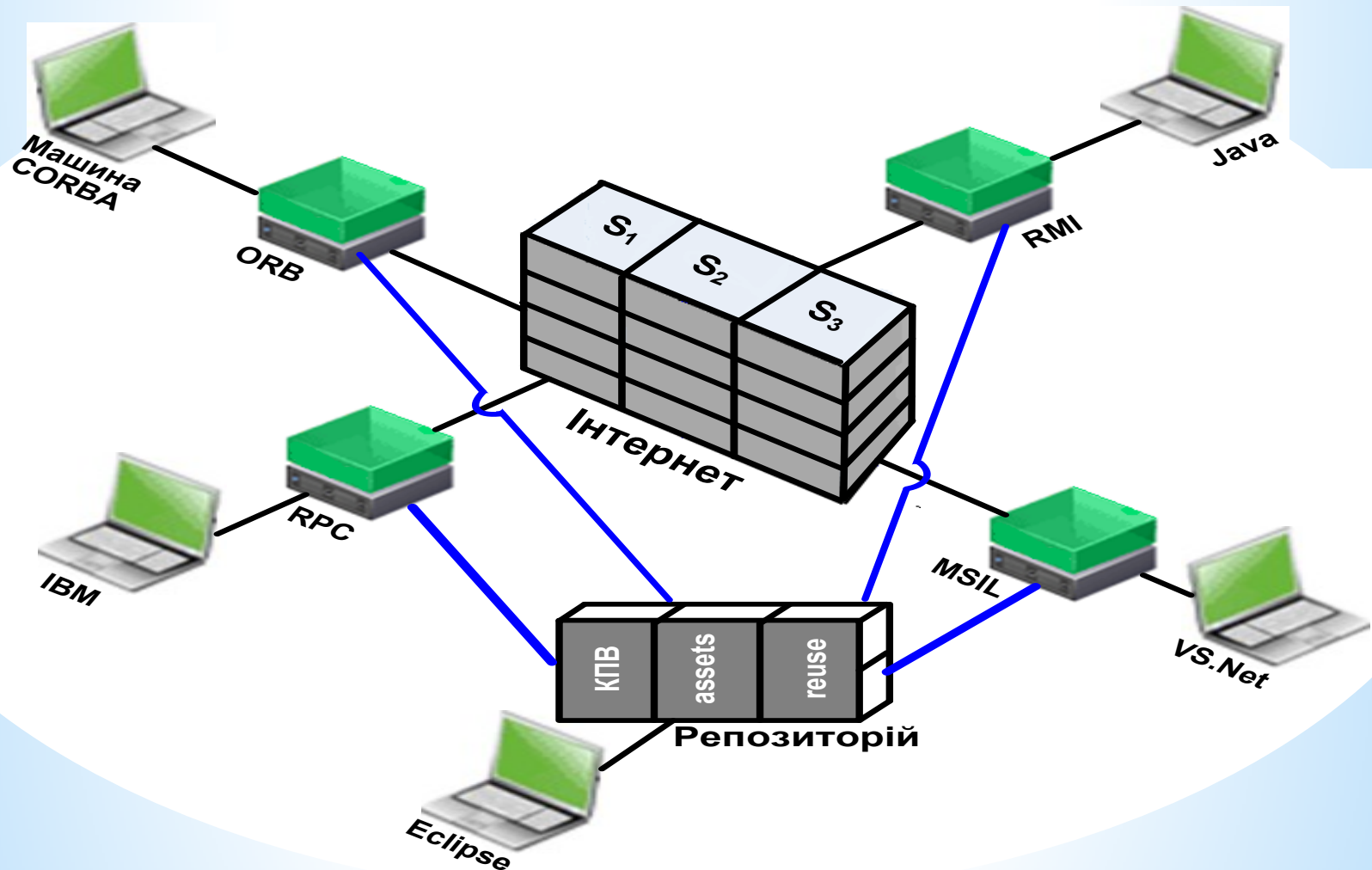
- язык спецификации интерфейсов КПИ (IDL, API, OSWL, XML)
- сборка компонентов и КПВ в приложения, домены, семейства программ и систем;
- для представления КПИ в репозитории для разных предметных областей.

Поддержка технологии ОКМ

- Репозиторий КПИ и онтологии доменов ЖЦ и вычислительной геометрии;
- Компонентная алгебра сборки КПИ;
- Системы преобразования ТД $GDT \leftrightarrow FDT$;
- Модели программной системы, семейства систем;
- Модель вариантной ПС и СПС;
- Модель качества ПС и СПС;
- Модель взаимодействия и вариабельности ПС, СПС;
- Методы индустрии ПС и СПС из КПИ;
- Монография 2 издание «Сборочное программирование»;
- Инструментально-технологический комплекс (ИТК) изготовления ПС и СПС из КПИ.

Взаимодействие систем и сред

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



Взаимодействие – это взаимосвязь двух и больше объектов

Модель взаимодействия имеет вид:

$M_{вз} = \{M_{пр}, M_{сис}, M_{сред}\}$,

где $M_{пр} = \{Com, Int, Pro\}$ – модель программы,

$M_{сис} = \{FPC, Int, Pro\}$ – модель системы,

$M_{сред} = \{Envir, Int, Pro\}$ – модель среды,

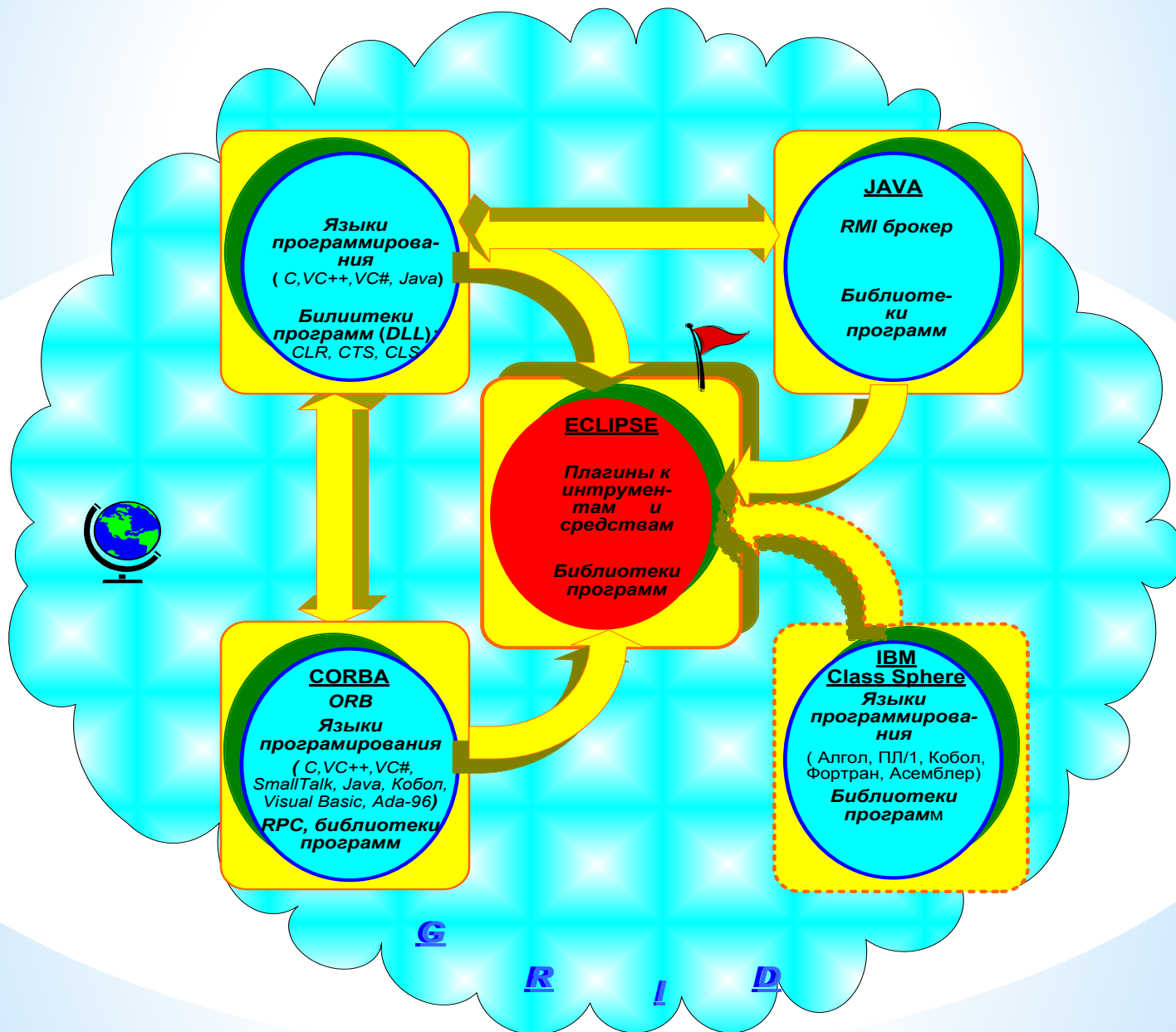
Int, Pro отображают совокупность интерфейсов и протоколов, которые передают через сеть данные между программами разных сред.

$M_{вз}$ по отношению к стандартной модели открытых систем OSI есть модель верхнего уровня, которая реализуется сервисом Eclipse.

Модель Visual Studio ↔ Eclipse

Модель CORBA ↔ Visual Studio ↔ Eclipse

Общая модель взаимодействия современных сред



Membership ID#513442 considering your paper “Object-Component Development of Application and Systems Theory and Practica.

I am writing with reference to your research paper titled, “Object-Component Development of Application and Systems Theory and Practice” and have acknowledged it worthy of commendation. I found your research work to be exceedingly impressive and impactful. This research can indeed prove to be significant for fellow researchers and scientists working in the same domain.

*** Taking note of your research interests which matches with our journal domain, I would like to welcome you to associate with us. To follow this, our Editorial Board has agreed to recognise you under "Quarterly Franklin Membership" (Membership ID#513442|UK) of **London Journals Press**. Also, we encourage you to have your upcoming research articles/papers published in, London Journal of Research in Computer Science and Technology (LJCST). As a part of this honorary membership, APC/Membership fee is fully waived off for you.**

	Главная страница Главная страница сайта
	ТЕХНОЛОГИИ
	Репозиторий КПИ
	Разработка КПИ
	Сборка КПИ
	Конфигурация
	Генерация DSL
	Инженерия качества
	Онтологии
	Веб-сервисы
	Отображение ТД
	ВЗАИМОДЕЙСТВИЕ
	CORBA — Eclipse
	VS.NET — Eclipse
	VBasic — Visual C++
	ИНСТРУМЕНТЫ
	Eclipse
	Protege
	ПРЕЗЕНТАЦИИ
	Прикладная система
	Программная инженерия и фабрики
	Индустрия программ
	ОБУЧЕНИЕ
	C# и MS.NET
	Java
	Software Engineering

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++.

ИНСТРУМЕНТЫ ИТК:

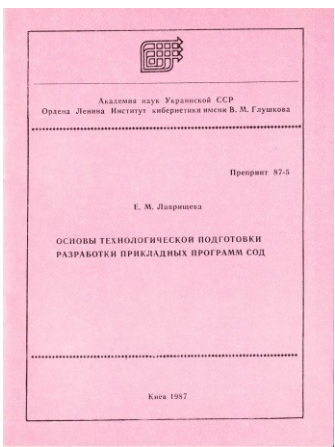
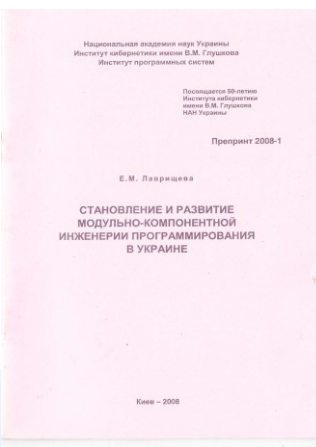
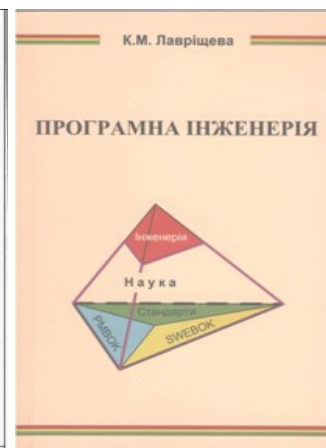
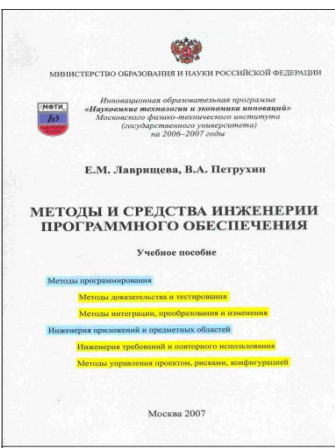
Система Eclipse,
Система Protégé

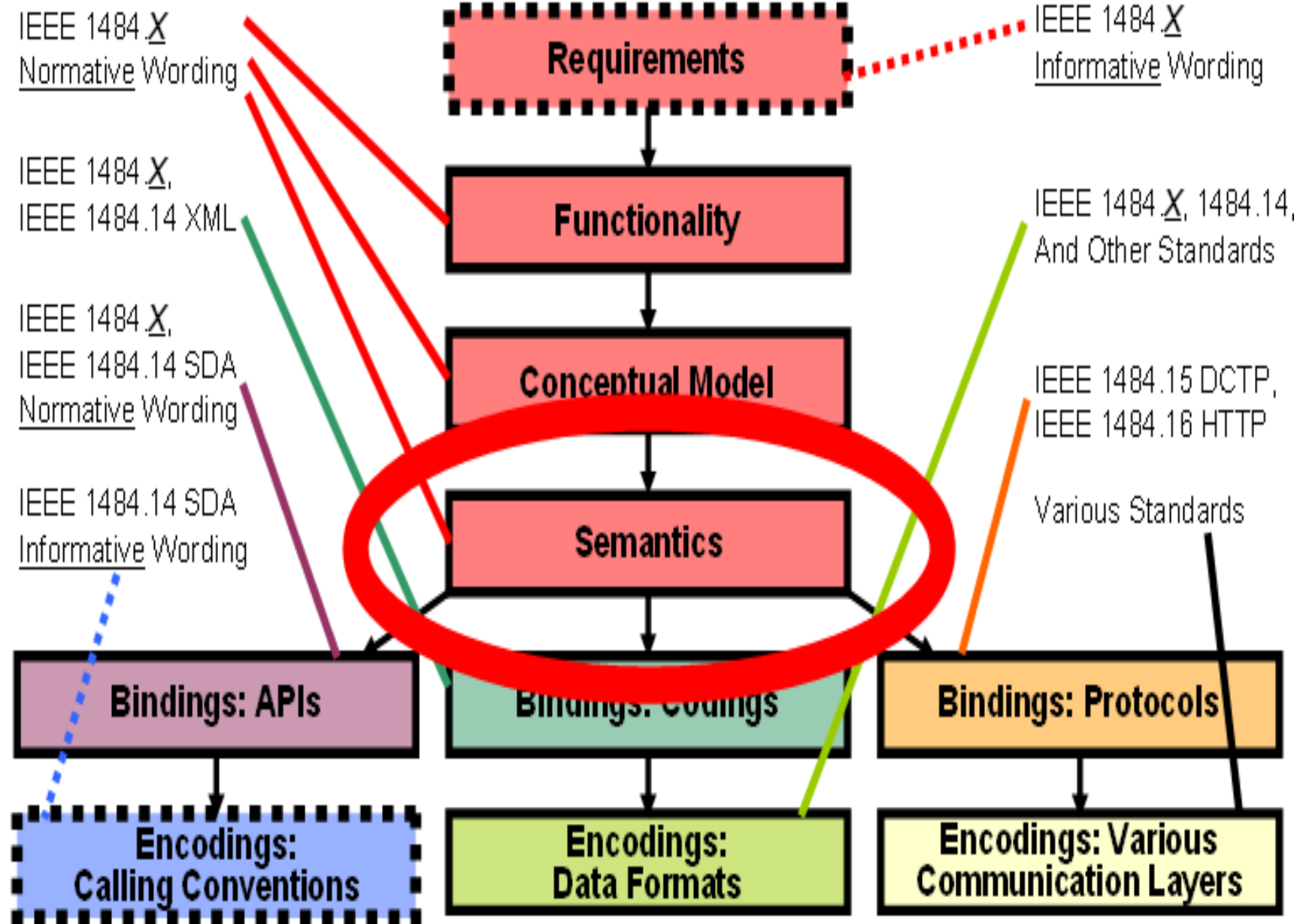
ПРЕЗЕНТАЦИИ В ИТК:

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>.





БЛАГОДАРЮ ЗА ВНИМАНИЕ!