Enterprise Content Management: Bridging the Academia-Industry Gap

Dr. Sergey V. Zykov, Ph.D.
TEKAMA Ltd.

Software Engineering Conference (Russia) (SEC(R)- 2007)
November 1–2, 2007, Moscow, Russia

Heterogeneous enterprise software integration – a challenging SE task

- The methodology combines formal model and SDK for class-level association-based relationships.
- Problem domain features:
- - high object classes complexity
- - incomplete information on the structure of certain instantiations of the classes;
- - the set of class attributes and operations can be determined rigorously.
- **Reasons for methodology application:**
- - variety of heterogeneous classes,
- - importance of association-based inter-class relationships
- - class inference possible even under certain % of weak-structured class instances

The methodology vs. ontology-based approaches (OBA):

- OBA (e.g. Cyc) efficiency is comparable only under a total class-level uncertainty, which is a different problem domain than ECM
- Thesaurus needed for the OBA to meet the relevance required
- The methodology uses similar foundations and tools as OBA (UML and XML-based tools, predicate calculus-based CycL, “conceptual model” etc.) for data modelling and integration
- OBA lack a balanced combination of formal models and industry-level SDKs (incl. visualization) for ECM lifecycle, resulting in low scalability and non-suitability for the major enterprise-level tasks

Objective, tasks, theoretical background

Objective: to make a software development methodology, which supports entire lifecycle of the enterprise software in the global computational environment

Tasks:

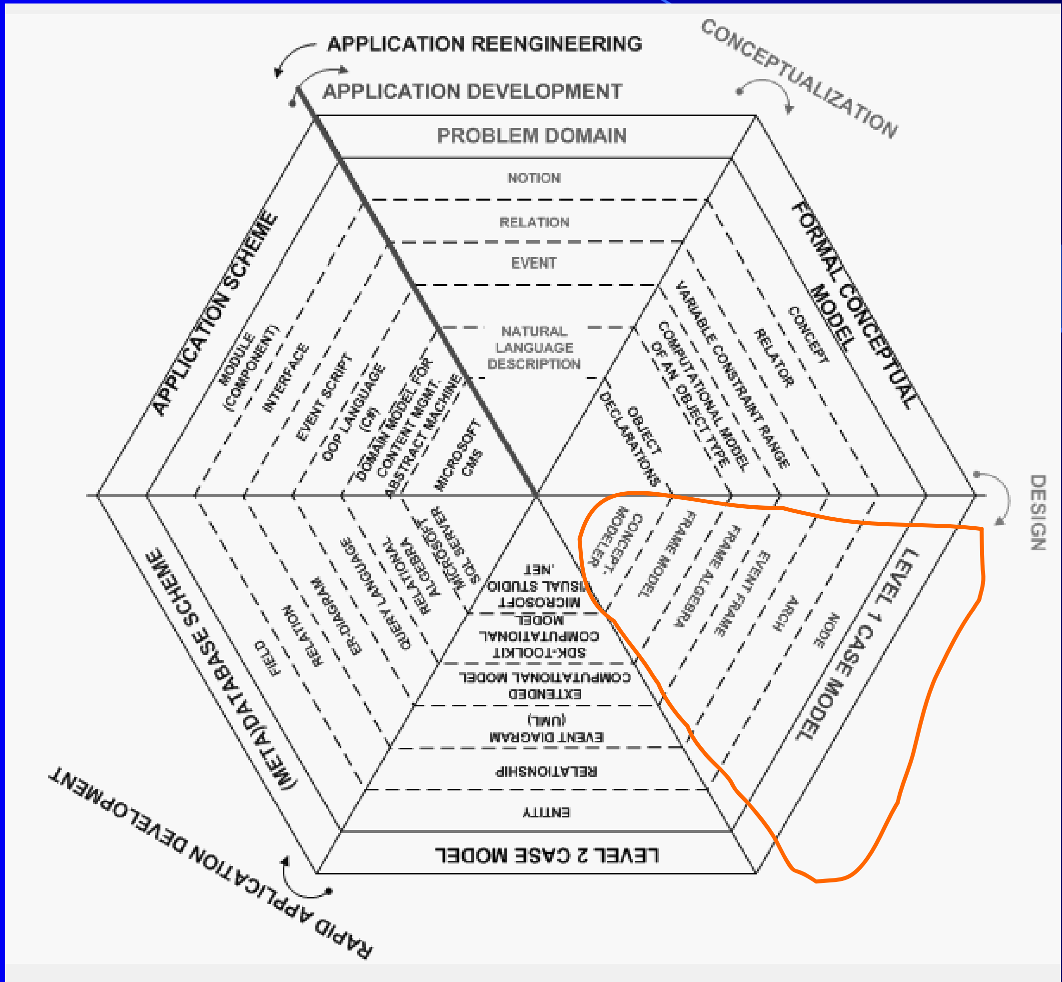
- formalizing stages and levels of the methodology;
- mathematical modeling;
- creating CASE- and RAD-tools ;
- implementing the methodology (prototype, full-scale).

Background: finite sequence, category, computation (D.Scott), semantic networks.

Innovations – the integrate methodology includes :

1. a set of data models for problem domain objects and for computational environment (CM, AMCM);
2. algorithm of the new component integration into the software implemented;
3. personalization procedure for enterprise content access;
4. SDKs: ConceptModeller, Content Management System

Enterprise software lifecycle support by the methodology



Problem domain modeling

Data object modeling:

"class → object → value"

Class – collection of data objects of the integrated problem domain;

Object – class instantiation by CMS template (metadata partial evaluation);

Value – static HTML page generated by CMS (full evaluation).

Benefits:

- evolves from the object-oriented approach;
- develops the existing models ([V.E.Wolfengagen's CM] et al.) in relation to global computational environment

Modeling classes of data objects

Classes C of problem domain data objects are modeled by domains:

$C = Iw{:}[D]\ \forall v{:}D\ (w(v) \leftrightarrow \Delta) = \{v{:}D | \Delta\}$, where:

1) C and D are in a partial order relation (C ISA D);
2) Δ is a criterion of data object w belonging to class C from the viewpoint of a problem domain expert.

Class of "n-dimensional" data objects is modeled by an n-arity relation:

$R^n = Iw{:}\ [V_1,..,V_n]\ \forall v_1{:}V_1\ \dots\ \forall v_n{:}V_n\ (w\ [v_1, \dots, v_n] \leftrightarrow \Gamma)$
$= \{[\ v_1{:}V_1,\ \dots,\ v_n{:}V_n] \mid \Gamma\}$, where:

Γ – "n-dimensional" criterion of data object w belonging to class R^n

<u>*Class*</u> is a collection of ordered pairs (v_i, V_i), where v_i is its *i*-th attribute (either of data or of metadata); V_i – attribute type.

From problem domain to computational environment (1)

Under class C instantiation with assignment a_1 and template Δ_k of CMS HTML page, evaluation of the template collection M sets into "true" value its element m_i, which index (k) equals the template number:

$$M = (m_1, \ldots, m_k, \ldots, m_N), \ \forall i=1,\ldots,N \ m_i \in \{0,1\};$$

$$[M|\Delta_k] = (m_1^*, \ldots, m_k^*, \ldots, m_N^*), \quad где \quad m_i^*=1, i=k \quad и \quad m_i^*=0, i \neq k.$$

Certain attributes of class metadata $v_1, \ldots, v_n$ are evaluated according to t_i conditions of Γ *template*:

$$[(v_1:V_1, \ldots, v_n:V_n)]t_i = ([v_1]|\Gamma(t_1), \ldots, [v_n]|\Gamma(t_n)) = (v_1':V_1', \ldots, v_n':V_n'),$$

причем V_1' ISA $V_1, \ldots, V_n'$ ISA V_n.

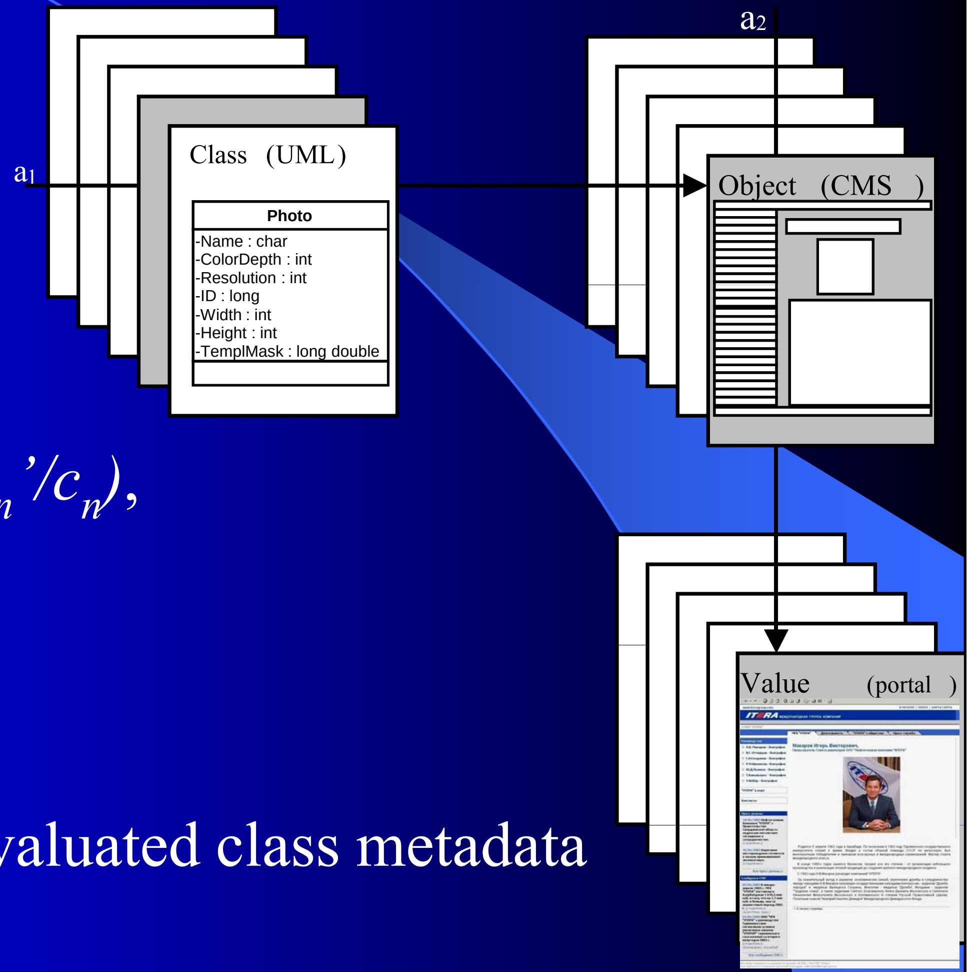
From problem domain to computational environment (2)

The second assignment a_2 instatiates non-evaluated template elements $(v_1',...,v_n')$ of CMS HTML-page by content values $(c_1,..., c_n)$:

$$[(v_1':V_1',...,v_n':V_n')]c = (v_1'/c_1,..., v_n'/c_n),$$

where $c_1:C_1,..., c_n:C_n$, and C_1 ISA $V_1',..., C_n$ ISA V_n'.

C_i class template is $T^i = (i,(t_1,...,t_n))$, where $(t_1,...,t_n)$ is the vector of the evaluated class metadata



Formal syntax of the CMS abstract machine

- Let us collect all the CMS abstract machine language identifiers into *Ide* domain, commands – into *Com* domain, and expressions – into *Exp* domain:

- *Ide ={I | I – identifier};*
- *Com ={C | C – command};*
- *Exp ={E | E – expression}.*

Formal semantics of the CMS abstract machine (1)

Order of construction:

- *standard* domains (most often used);
- *finite* domains (including explicitly enumerable elements);
- *domain constructors* – operations of building new domains out of existing ones;
- composite domain formalization based on standard domains and domain constructors.

Domain constructors :

- functional space: $[D_1 \rightarrow D_2]$;
- Cartesian product: $[D_1 \times D_2 \times \ldots \times D_n]$;
- sequence: D^*;
- disjunctive sum: $[D_1 + D_2 + \ldots + D_n]$.

Formal semantics of the CMS abstract machine (2)

- $State = Memory \times Input \times Output;$
  $Memory = Ide \rightarrow [Value + \{unbound\}];$
- $Input = Value^*;$
- $Output = Value^*;$
- $Value = Type1 + Type2 + \ldots$

- *Constant denotate:* *<variable, value>*

- *Identifier denotate:*
- *<variable_in_memory, identifier, state>*

Formal semantics of the CMS abstract machine (3)

Semantic function for expression:

$\boldsymbol{E}: Exp \rightarrow [\ State \rightarrow [[Value \times State] + \{error\}]];$

Semantic function for command:

$\boldsymbol{C} : Com \rightarrow [State \rightarrow [State + \{error\}]].$

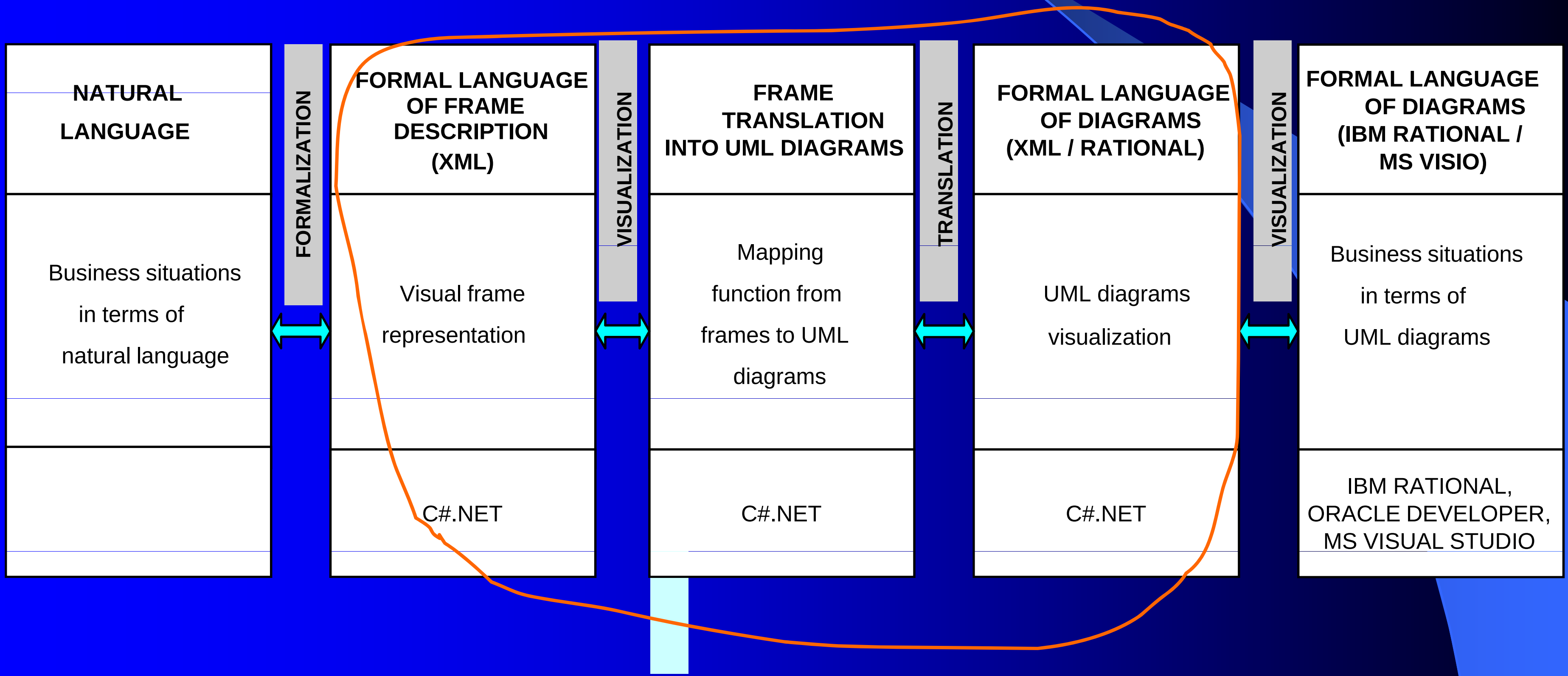
Semantic statement for identifier:

$\boldsymbol{E}\ [I]\ s = (m, I = unbound)\ error, \rightarrow (m, I, s).$

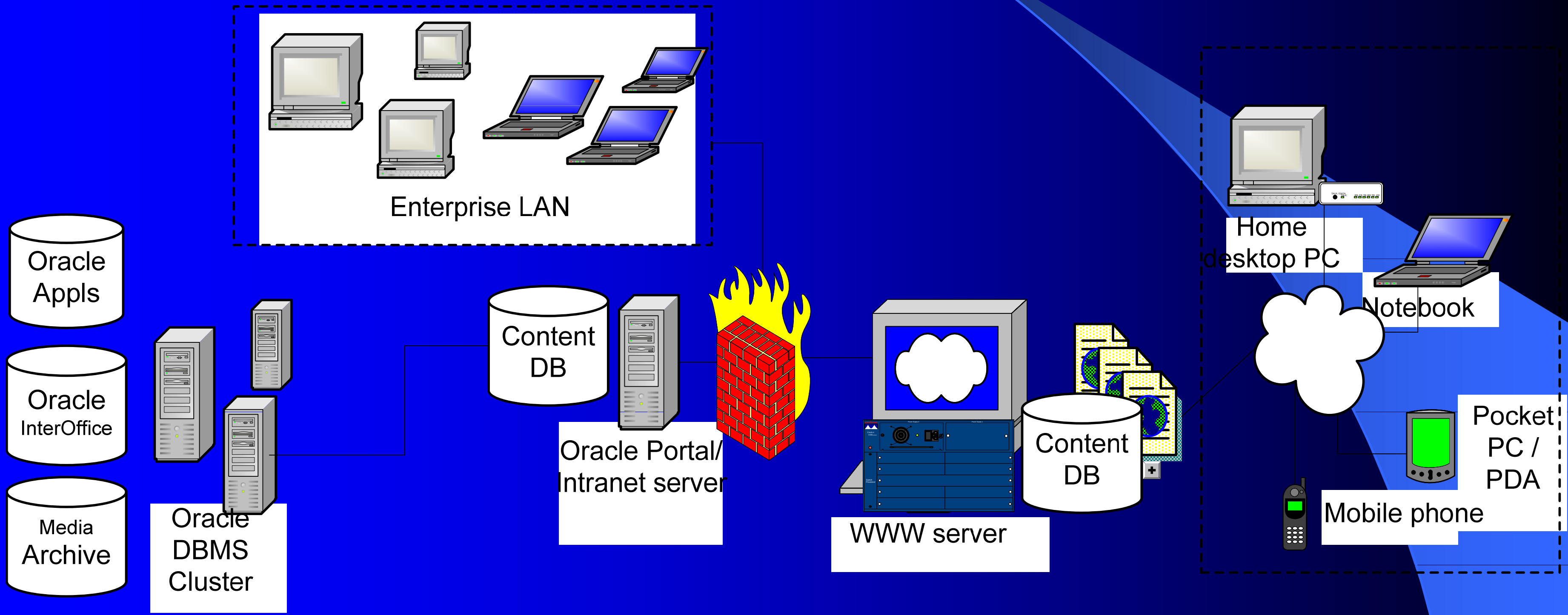
Semantic statement for assignment command:

$\boldsymbol{C}\ [I=E] = \boldsymbol{E}\ [E] * \lambda v\ (m\ , i, o)\ .\ (m\ [v/I], i, o).$

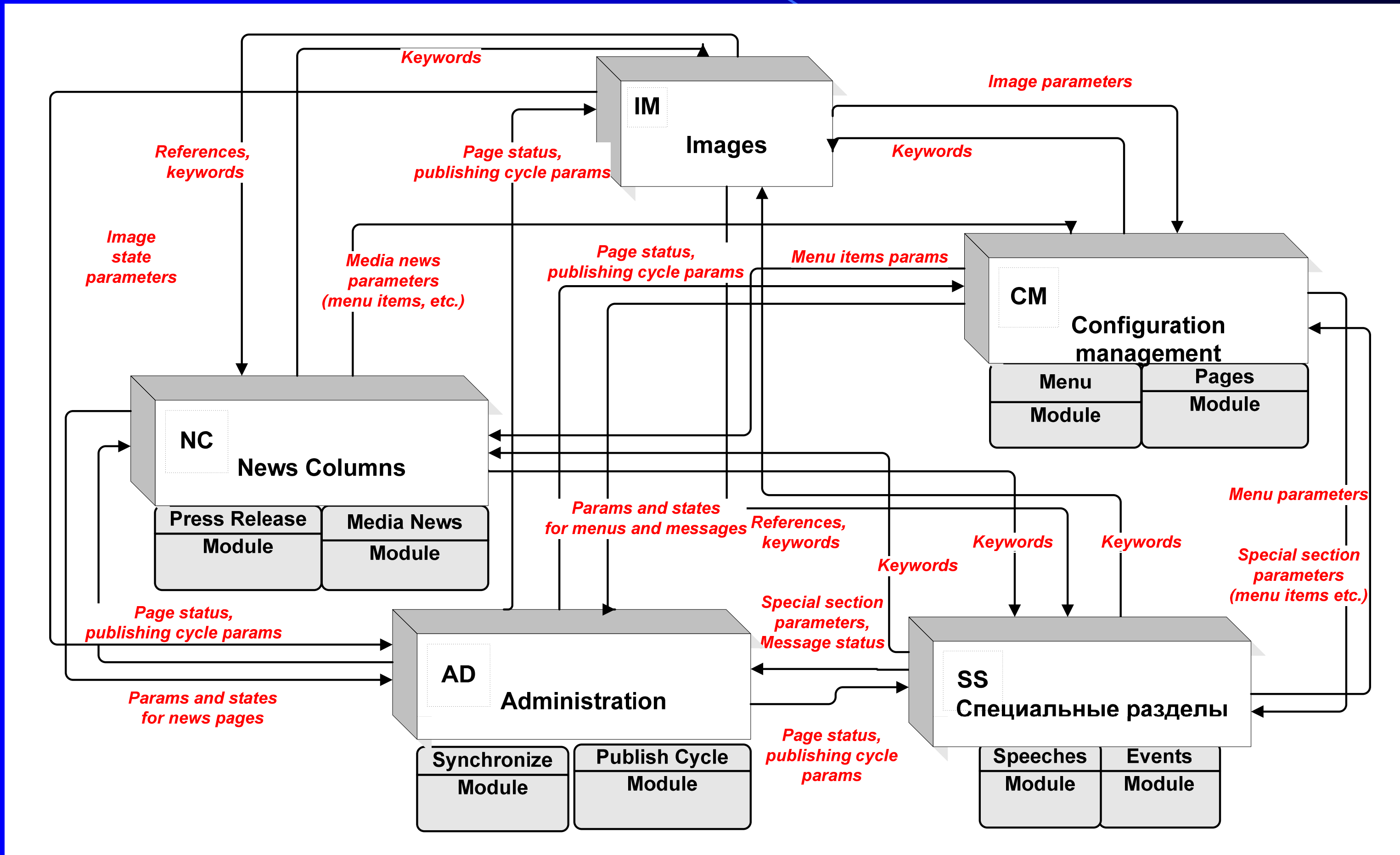
Bi-directional software development in ConceptModeller CASE-toolkit



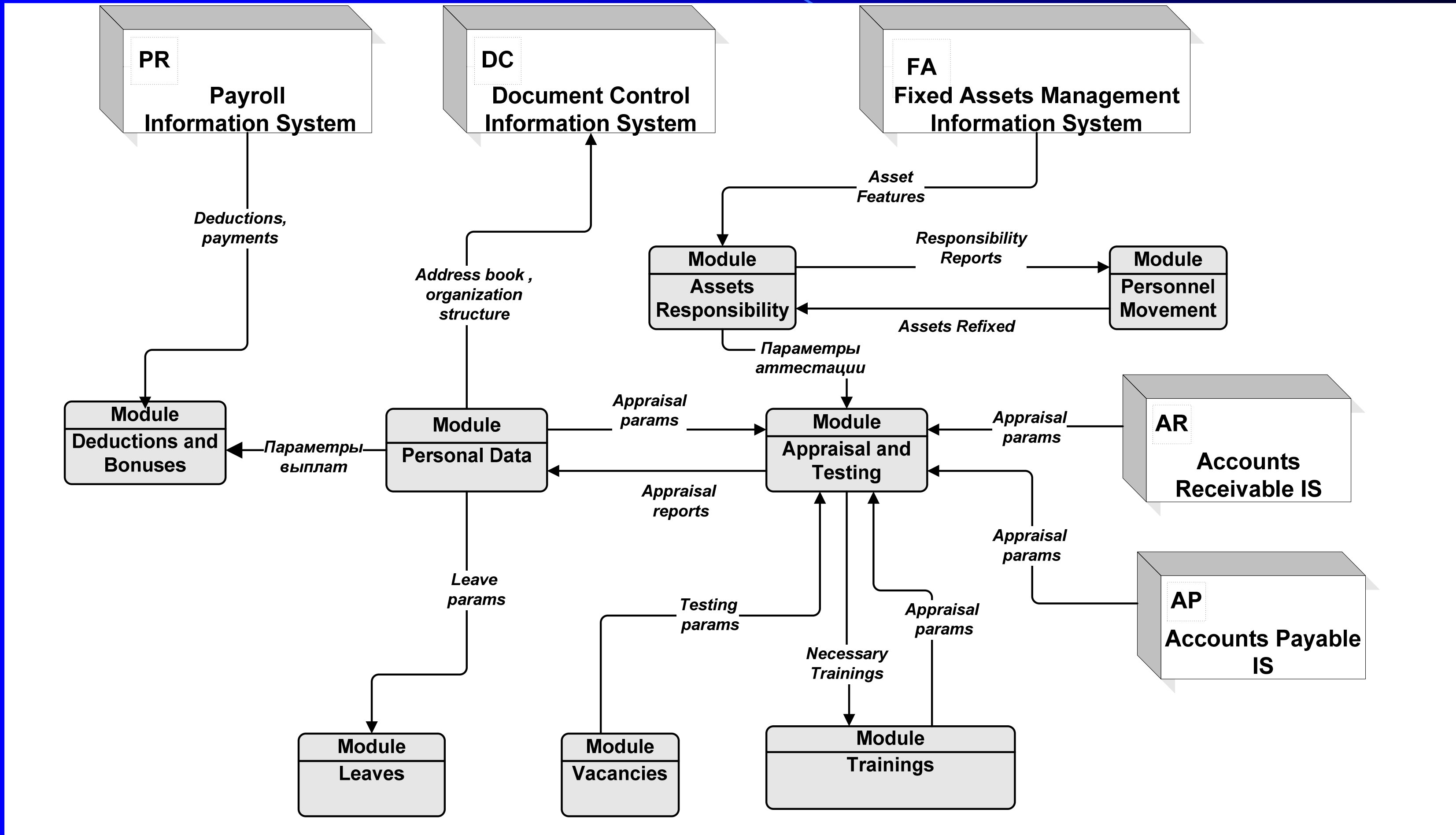
Software Solution Arcitecture



CMS logical structure



Structure of the integrated enterprise program system



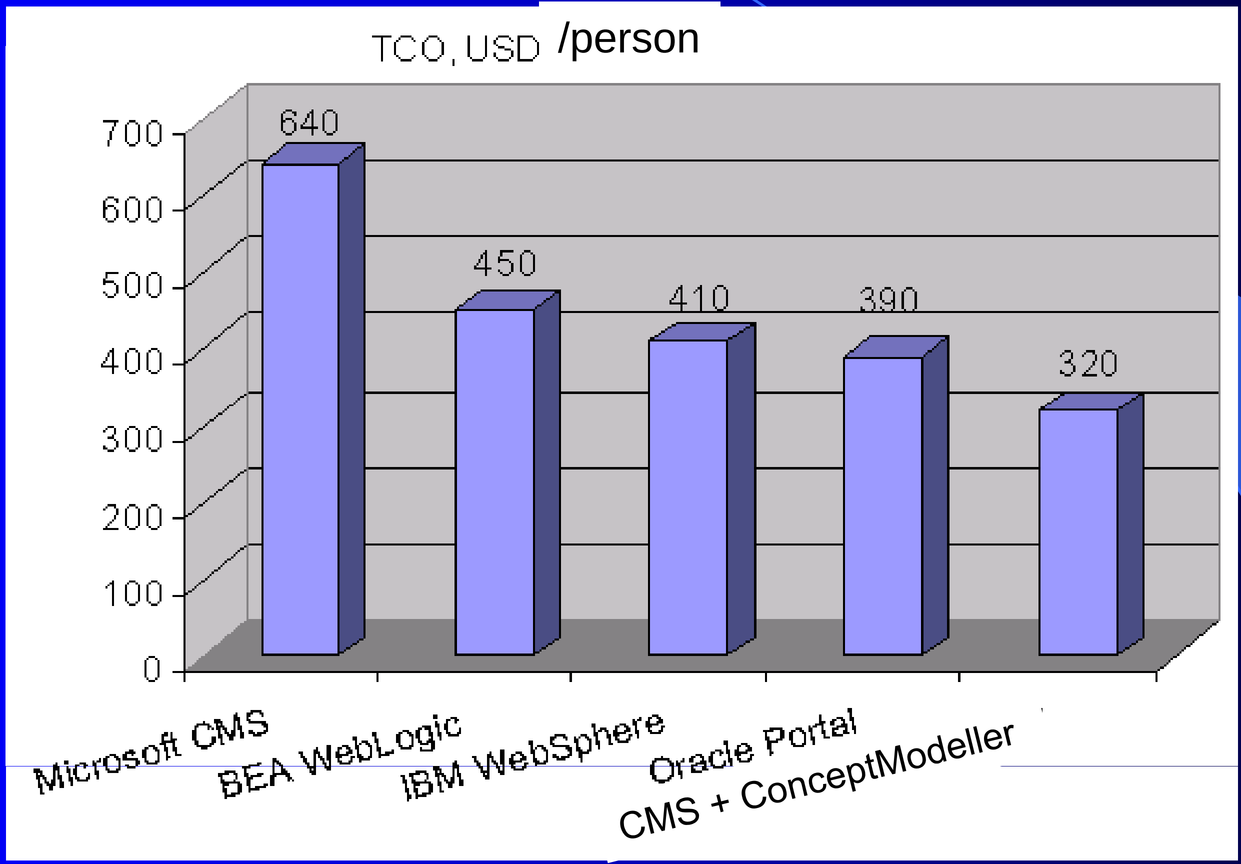
Comparing the software development methodology to the commercial methodologies available

	Methodology	Mathematical model	"Lower" CASE	"Upper" CASE	RAD	BPR
1	IBM RUP	-	-	+	+	+
2	Oracle CDM	-	-	+	+	+
3	Microsoft MSF	-	-	+	+	+
4	BEA Solution	-	-	+	+	+
5	Integrated methodology	+!	+!	+	+	+

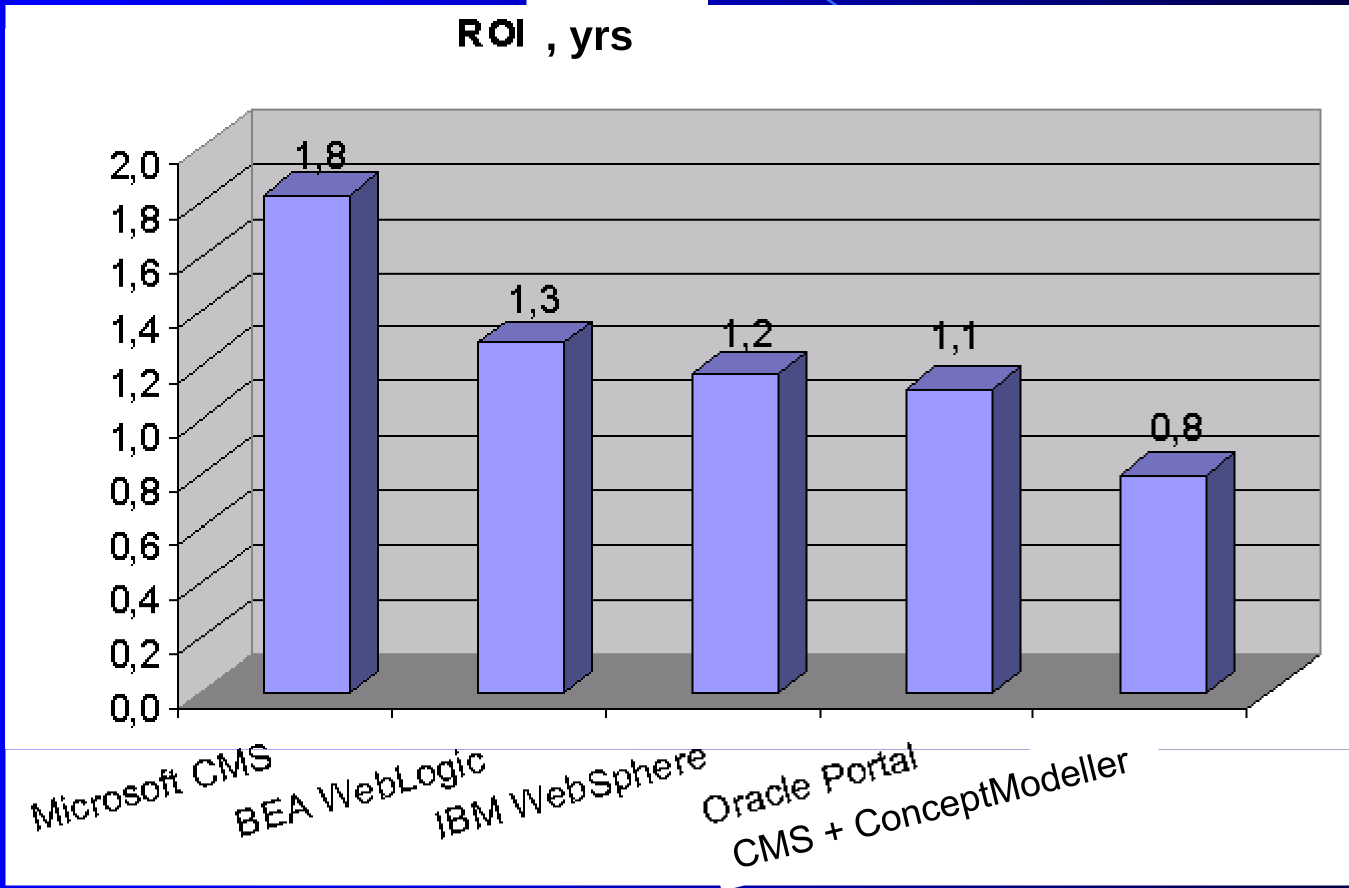
Implementation features comparison

	Software	Multi-language publication	Java servlets	.NET web services	UML diagrams	WYSIWYG-mapping	Integrated ERP information system reports	Integration with legacy information systems	Smart, template-based design	Complex data object embedding
1	IBM WebSphere	+	+	-	+	+	±	±	±	±
2	Oracle Portal	+	+	-	+	+	±	-	±	±
3	Microsoft CMS	+	-	+	+	+	-	-	±	±
4	BEA WebLogic	+	+	-	+	+	±	±	±	±
5	ITERA CMS	+	+	±	+	+	+!	+!	+!	+!

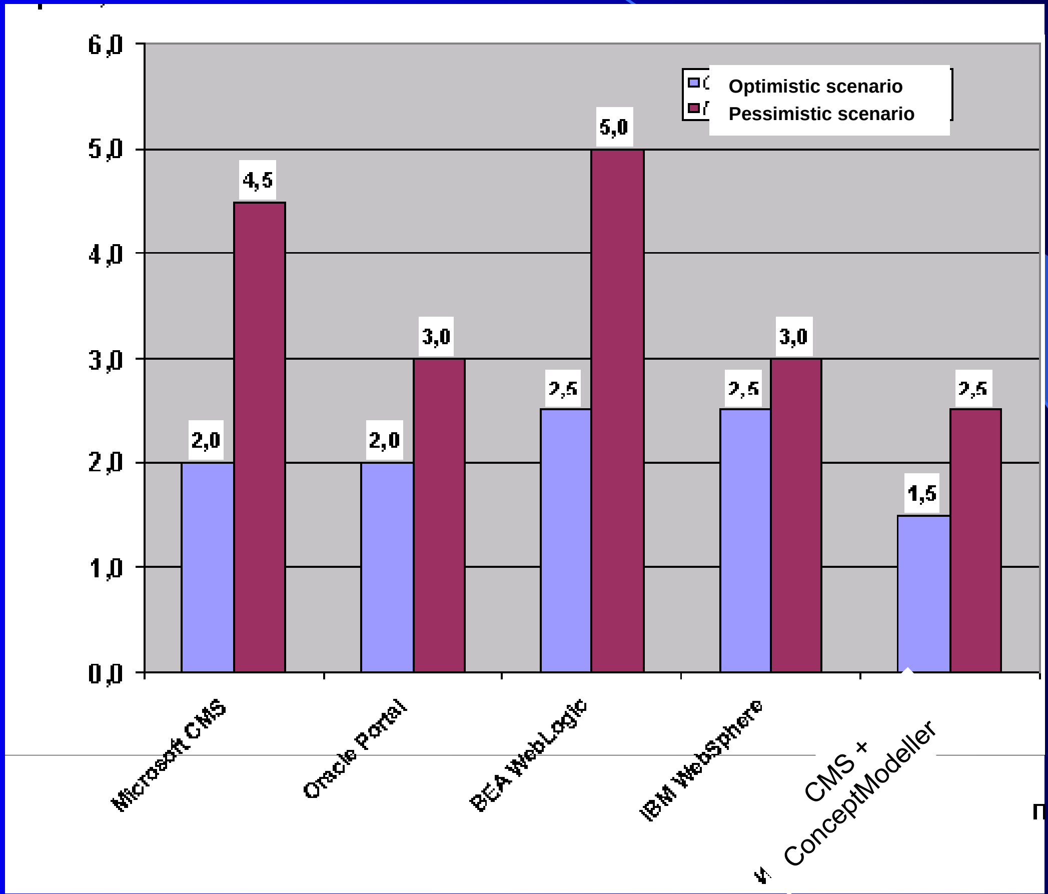
TCO comparison results



ROI comparison results



Implementation terms comparison results



Theoretical results:

1) A system of formal models for problem domain and computational environment (rigorous semantics, entire lifecycle support, content management orientation);

2) Algorithm of integrating new components to the enterprise software system (problem-oriented, heterogeneous software architecture support);

3) personalization procedure for accessing enterprise content (flexible, reliable)

Engineering results:

1) CASE- and RAD-toolkits:
   a) ConceptModeller (rigorous semantics; compatible to up-to-date CASE-tools, ERP and и legacy systems; re-engineering; XML/BPR/UML standard support);

   b) ITERA CMS (rigorous semantics; rapid publishing of complex content; WYSIWYG interface; office products integration).

2) Architecture (environment unification of heterogeneous enterprise applications; role personalization with situation dynamics)

Practical value of the results obtained:

1) implementation term-and-cost reduction (TCO, ROI) as compared to commercially available software by 30% (average);

2) major enterprise software features improvement:
- scalability;
- reliability;
- ergonomics.

Research results approbation:

Over 30 presentations on international conferences, over 50 papers (incl. 4 books).

Research grants from MSR (2002-2003) and RFBR (1996-2006).

ITERA implementation (150 companies, 10,000 employees): CMS (2002); Internet-portal (2003); Intranet-portal (2004)

Other implementations: ICP (RAS), *Sterkh* Foundation, Ashihara Karate Association, Russian Orthodox Church, etc.

Curricula (MEPhI, MSUFI, INTUIT, LANIT, SoftLine) – over 3000 graduates