Web Based Problem Solving Process [2]

(A 2020 Computer User)

Teodor Rus
The University of Iowa,
Iowa City, IA 52242, USA

September 19, 2011

[2] Talk given at the KAUST Workshop, February 2011

Alain Chesnais, ACM President

Are you ready for 2020?

Topics

- Computer Based Problem Solving Process;
- Web Based Problem Solving Process;
- Cloud Computing Supporting WBPSP;
- NLD System: a Cloud Model for WBPSP;
- Example NLD application.

Credit

- Research I am presenting here evolved from my students effort developing Application Driven Software.
- Tools we use for ADS development are:
  1. TICS Compiler Construction Technology. Among many I mention John Knaack, Tom Halverson, Erick Van Wyks.
  2. ADS for Chemistry, ADS for Linear Algebra (Don Curtis, using Python)
  3. Teaching High School Algebra (currently under development by Cung Bui, using Protege, Axis, JAXB, Java).
  4. Many course projects (Compiler Constriction, Parallel Programming, Computational Linguistic, Web Programming).

Time to Change

What do we change and why ?

- Need to address the future of Computer Science as a SCIENCE;
- Need to address the future of Computer Technology, as a BUSINESS.

Note: CS users are CS experts (CSE), CT users are problem domain experts (PDE).

CSE and PDE are not necessarily the same!

Teach people how computer works!

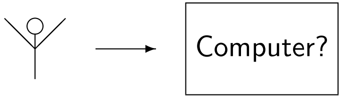


Figure:

In other words: take the problem to the computer!

Problem: There is a potential infinite number of problem domains and each problem domain may raise a potential infinite number of problems!

Software complexity:

Computer is used by one-pattern-fits-all

The implementation of the mapping:

$\infty \ \#PD^{\infty \# problems} \longrightarrow 1$

requires infinite many translators.

Resulting software complexity threatens CT!

CT solution:

Gadgets dedicated to PD/problem

Problems are solved by clicking gadget's buttons!

- Buttons are labeled by concepts that implement solutions;
- The same concept clicked on different gadgets may solve different problems!
- Syllogism and induction are universals but logics that use them may be different.

That is, CBPSP by one-pattern-fits-all (i.e. programming) needs to be changed!

WBPSP

Web Based Problem Solving Process

Teach computers how people work!

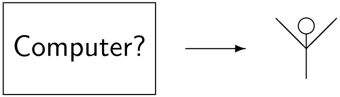


Figure:

In other words: take the computer to the problem!

Problem: thinking inertia!

Implementation:
Fighting thinking inertia

Formalize the process of gadget development by Computational Emancipation of the Application Domain (CEAD):

1. identify primitive concepts of the domain;
2. organize them using an ontology;
3. associate concepts in the ontology with web services implementing them;
4. develop a gadget that transform natural language algorithms into processes performing them on the Web.

Problem Domain (PD) In 2020

will be computationally emancipated

- PD will be specified by a Domain Ontology (DO);
- Concepts in DO are associated with Web Services that implement them;
- PD will be provided with a Domain Dedicated Virtual Machine (DDVM);
- DDVM instructions are computer processes performing Wed Services that implement domain concepts.

DDVM Formally:

$DDVM = \langle CC, Execute, Next \rangle$

where:

1. CC is a concept counter that runs on DO;
2. Execute executes the computer process specified by WS associated with (CC);
3. Next determines next concept of the algorithm performed by DDVM.

Web Based Problem Solving Process:

Polya methodology + DDVM-interpreter

Assumption: problem solver has access to the DDVM:

- Formulate the problem;
- Develop a solution algorithm;
- Input the algorithm to DDVM which performs:
  1. CC = First Concept of the algorithm;
  2. While(CC is not End of the algorithm)
     Execute(WS(CC)); CC := Next(CC);
  3. If CC is End of the algorithm DDVM outputs the solution.

Note: WS(CC)-s are standalone and composeable, hence, interoperable!

Contrast:

WBPSP is a service oriented business

- Computer user does not write programs!
  Hence, she is not required to be a computer expert.
- Problem solving is a DDVM-based business.
- The problem solver work is completely reusable; (algorithm developed by problem solver is executed by DDVM).
- Problem domain evolves with PSP (DO is expanded with concepts executed by DDVM). This mimics human learning process.

Conclusion:

Computational Emancipation of Application Domain (CEAD) transforms the computer from a

number-crunching tool

into a

domain dedicated cognitive machine.

What is CEAD?

CEAD is similar to the process of domain formalization performed by mathematical applications:

- Domain formalization transforms domain concepts into mathematically well-defined abstractions;

  CEAD transforms domain concepts into computer artifacts;

- Domain formalization develops the "mathematical language of the domain" used by mathematicians to prove theorems.

  CEAD develops the "computational language of the domain" used by domain experts to express problems and algorithms.

Examples CEAD

Texts embedding hyperlinks

1. Dragon book where each concept is associated with na hyperlink pointing to itsn implementation;
2. Linear algebra where concepts are associatedn with their implementations

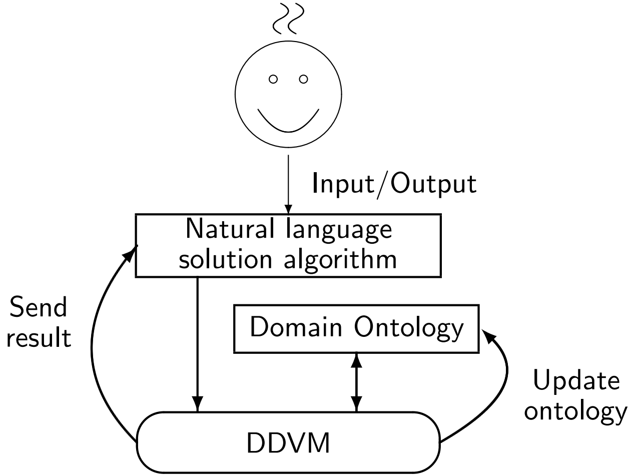
Natural Language of the Domain

- Algorithms performed by DDVM are language expressions used by domain experts during problem solving process.

  Example: solve equation
  $ax^2 + bx + c = 0; a, b, c : real; a \neq 0;$

- Computational language of a domain is formally defined by CEAD(Domain) and is further referred to as the Natural Language of the Domain, (NLD).

Using DDVM


Input/Output
Natural language solution algorithm
Send result
Domain Ontology
Update ontology
DDVM

Questions

1. Does WBPSP imply that CBPSP disappear?
2. What are the technological implications of WBPSP?
3. Who CEAD(Domain) and how?
4. Who develop the DDVM and how?
5. Do we have examples of WBPSP-s?

Does programming as usual disappear?

By the contrary:

Programming as usual receives a new dimension with WBPSP!

WBPSP sets bases for a new mechanism of interdisciplinary collaboration which bridges the semantic gap between PDE and CSE!

Implications of WBPSP

WBPSP generates an unlimited and unrestricted domain of new Computer Technology:

- Each domain needs to be provided with its own DDVM.
- WBPSP makes the universality of conventional computer match the diversity of human universe of discourse!
- Problem solving by One-Pattern-Fits-All (computer programming) is no longer valid! Hence:
- WBPSP may resolve problems raised by software complexity!
- System Software receive new dimensions: dedication to problem domains, evolution with problem solving process, etc.

Who manages WBPSP?

WBPSP is managed by Cloud Computing!

Cloud Computing: is a wonderland populated by:

1. computer networks (clients and servers), and
2. computer experts (people who develop and sell web services!)

The Web Service (WS)

A WS is a stand-alone and composeable program provided with three new mechanisms:

1. A WSDL expression that describes WS's functionality as a stand-alone program run on a node in the network;
2. A SOAP expression that describes the WS's composition mechanism;
3. A UDDI registry that allow people interested to discover and integrate the WS in their applications.

Examples of WBPSP

- Service Oriented Architecture (SOA) provides best examples of WBPSP!
- BPEL (Business Process Execution Language) is a concrete example;
- NLD System provides a model of cloud business that allows people to buy DDVM-s dedicated to their domains.

Note: with SOA and BPEL in addition to programming as usual, computer user needs to develop WSDL, SOAP, UDDI too!

What is NLD System?

NLD System is a work in progress. It consists of three components that are integrated and run in the cloud:

1. NLD, used by computer user to develop domain algorithms.
2. OWL file representing the CEAD-ed Domain Ontology.

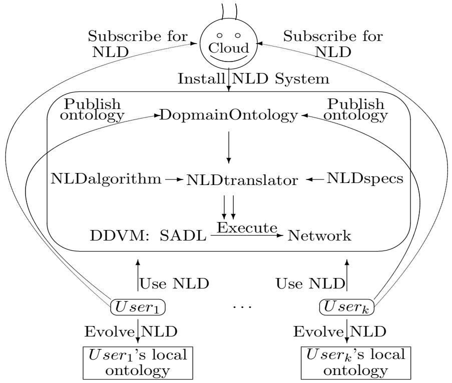
   CEAD(Domain) is an RDF file automatically generated by Tools (Protege, Axis, JAXB) from a description logic expression of the domain.
3. A translator that map NLD algorithms into SADL, the language of DDVM. SADL interpreter runs SADL expressions on the Web.

Using NLD System

Computer users can use the NLD system by the following pattern:

1. Get a Cloud subscription (Subscribe2NLD) to use the NLD system. Cloud manager install NLD system on user laptop!
2. The user uses the NLD system (UseNLD) by developing and running NLD algorithms.
3. The user evolves the NLD with new concepts by EvolveNLD which generate a local OWL File for the new concepts;
4. The user can remove a concept from her ontology by RemoveConcept ;
5. NLD users can share the ontology concepts they develop by publishing their local ontologies (PublishOWL).

NLD System Pictorial



Tools Used for NLD Implementation

- protégé (`http://protege.stanford.edu/`) is an Editor and Knowledge Acquisition System;
- OWL (web ontology language), for ontology development and RDF (Resource Description Framework) as the Ontology Reasoning Tool.
- Apache extensible interactive system (axis) is the Apache server used to implement software services interoperability.

Java Architecture for XML Binding (JAXB) is a newer and more convenient way to process XML content using Java objects by binding its XML schema to Java representation.

Example: High School Algebra

Problem Domain: Solving equations.

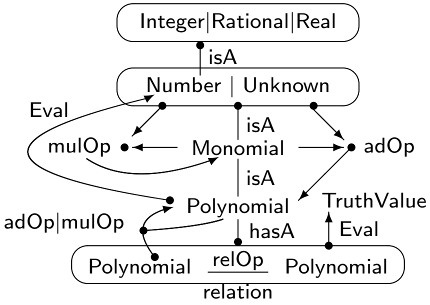
Domain concepts:

- Numbers (integers, real)
- Unknowns (variables denoted by letters)
- Monomials
- Polynomials
- Relations

Domain Ontology

A highgraph (D. Harel, Comm. ACM 31, #5) whose nodes represent concepts and edges represent conceptual relationships.

Assumption: concepts, properties, and actions used in the ontology are associated with web services implementing them.



Example: solving second-degree equations

- Problem formalization:
  equation: $ax^2 + bx + c = 0;\ a, b, c \in R;\ a \neq 0;$
- Solution algorithm:
  solver: $x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a};\ x_2 = \frac{-b-\sqrt{b^2-4ac}}{2a};$
- Solving equations:
  (1) input solution algorithm to DDVM;

  (2) call the DDVM and engage in the DDVM dialog.
- Evolving the domain:
  add equation and solver to the domain ontology.

Demos

1. The CEAD-ed Arithmetic Ontology;
2. Evolving Arithmetic Ontology by NLD reimplementation:
   adding new data (complex type)
   adding new web services (square root);
3. Using Arithmetic Ontology:
   equation solver (SolverR, SolverC);
4. Adding, using, removing concepts:
   add2Ont SolverR, add2Ont SolverC
   use SolverR, use SolverC, remove SolverR, remove SolverC
5. Evolving Arithmetic Ontology to a Vector space:
   adding abstractions to NLD (vector algebra)