5643??

Ninth International Workshop on Program Comprehension (IWPC 2001)

May 12-13, 2001 - Toronto, Ontario, Canada - Westin Harbour Castle Hotel

Co-located with ICSE 2001

CALL FOR PAPERS

General chair

Ric Holt
Dep. of Computer Science
University of Waterloo, Canada
E-mail: holt@plg.uwaterloo.ca

Program co-chairs

Kostas Kontogiannis
Dep. of Electrical and Computer Engineering
University of Waterloo, Canada
E-mail: kostas@swen.uwaterloo.ca

Andrea De Lucia
Faculty of Engineering
University of Sannio, Benevento, Italy
E-mail: delucia@unisannio.it

Sponsored by:

IEEE, IEEE Computer Society, IEEE Computer Society TCSE

In cooperation with:

- University of Waterloo, Ontario, Canada
- University of Sannio, Benevento, Italy
- University of Toronto, Ontario, Canada
- University of Victoria, British Columbia, Canada

Conference location

IWPC 2001 will be held in Toronto, Ontario, Canada at the Westin Harbour Castle Hotel, co-located with ICSE 2001. Please, see the ICSE 2001 web page (http://www.csr.uvic.ca/icse2001/) for more information.

Submissions deadlines

- Papers: November 15, 2000
- Tool demo proposals: December 15, 2000
- Working session proposals: December 15, 2000

Acceptance notification: January 15, 2001

Camera ready copy due: February 20, 2001

Electronic submission guidelines

To submit a paper, a tool demo, or a working session proposal, please send an e-mail to iwpc2001@swen.uwaterloo.ca including the type of the submission (technical paper, tool demo proposal, working session proposal), the title, the list of authors, the abstract, and a contact phone number. Please, do not include the paper in the message. By return mail, you will receive instructions on how to ftp your submission.

Web sites: http://serg.ing.unisannio.it/iwpc2001
http://www.swen.uwaterloo.ca/~iwpc2001

Theme

Comprehending programs is one of the core software engineering activities. Program comprehension is needed when one maintains, reuses, migrates, reengineers, inspects, or enhances software systems. This workshop will gather practitioners and researchers from academia, industry, and government, to review the current state of the practice, to report on program understanding experiments, and to present issues and solutions in the general area of program comprehension.
The workshop program will include technical papers, tool demos, and working sessions. Authors are expected to present an accepted paper or proposals at the workshop in Toronto.

Technical Papers

Papers related, but not limited, to the following topics are invited:

- Theories and models for software comprehension;
- Program comprehension processes and strategies;
- Cognitive models in program comprehension;
- Experiments with comprehension models;
- Knowledge, program representation forms and repositories for program comprehension;
- Tools facilitating software comprehension;
- Comprehension during large scale maintenance, reengineering, and migration;
- Reuse reengineering and concept assignment processes;
- Reverse engineering strategies to support comprehension;
- Innovative technologies for reverse engineering;
- Case studies in reverse engineering, reengineering, and maintenance with focus to program understanding;
- Guidelines for facilitating program comprehension (based on observation and experimentation);
- Computer Supported Collaborative Understanding (CSCU);
- Understanding systems built using distributed object technology;
- User interfaces, software visualization and animation;
- Understanding product line systems.

Papers should be original work, approximately 10 proceeding pages or 6000 words. Papers must not have been previously published nor have been submitted to, or be in consideration for, any journal, book, or conference. The workshop proceedings will be published by IEEE Computer Society Press.

Tool demos

IWPC 2001 final program will include demo sessions of tools for program comprehension. We invite two page proposal submissions in IEEE format to be included in the workshop proceedings. Proposals for tools demonstrations should include a description of the tool or environment, its applicability to program comprehension, and a brief description of the proposed demonstration. Specify the proposal as "Research Prototype" or "Vendor Tool". Authors of accepted demos are also invited to exhibit their tool at the ICSE 2001 exhibits and tools fair.

Working sessions

We also invite proposals for working sessions (90 minutes each) on any of the topics areas mentioned above. Working sessions are designed around a specific theme and should be more interactive than a panel session. Working session proposals (maximum one page in IEEE proceedings format) should include the leader of the session (working session chair) and the list of co-authors with their short debate statements.

Research Prototype: Automated Analysis of Scientific & Engineering Semantics

Mark E. M. Stewart
Dynacs Engineering, Inc.
Cleveland, OH 44135, USA
Mark.E.Stewart@grc.nasa.gov

Abstract

Physical and mathematical formulae and concepts are fundamental elements of scientific and engineering software. These classical equations and methods are time tested, universally accepted, and relatively unambiguous. The existence of this classical ontology suggests an ideal problem for automated comprehension. This problem is further motivated by the pervasive use of scientific code and high code development costs.

To investigate code comprehension in this classical knowledge domain, a research prototype has been developed. The prototype incorporates scientific domain knowledge to recognize code properties (including units, physical, and mathematical quantity). Also, the procedure implements programming language semantics to propagate these properties through the code. This prototype's ability to elucidate code and detect errors will be demonstrated with state of the art scientific codes.

1. Tool description

From a user's perspective, this procedure involves taking a user's existing scientific or engineering code (1), adding semantic declarations, and viewing/querying the analysis results (Figure 1). Semantic declarations (distinguished by "C?") identify primitive program variables using standardized technical terms (ie. *mass*, *acceleration*).

```
C?  MA == mass
C?  ACC == acceleration
    FF = MA*ACC                    (1)
```

From the analysis perspective, this procedure involves three elements: a scientific semantics analysis procedure, facilities for programming language semantics, and a graphical user interface (GUI).

1.1 Scientific semantics analysis procedure

Classical mathematics emphasizes equations—lexical, sequential expressions that quantify physical or mathematical concepts. A parser is not only an effective way of representing a large set of these physical equations, but a parser can also efficiently recognize these equations in program expressions.

In particular, (1) may be translated into expressions of code properties, including, a physical quantity expression (2), and a physical dimensions expression (3).

```
mass * acceleration          (2)
(M) * ( L*T**-2 )            (3)
```

Parsers recognize formulae in these translated phrases. For example, a dynamics expert parser would include the rule (4), be able to recognize the phrase (2) as Newton's law, and annotate the data structure.

$$\text{force} \Leftarrow \text{mass} * \text{acceleration} \qquad (4)$$

The units expert parser can reduce (3) and verify units. These properties (Table 1) reflect the different aspects of program statements that scientists and engineers analyze.

Parsers represent and recognize equations effectively and easily for many properties. However, recognizing mathematical rules is harder. For example, the discrete difference, $\Delta\phi$, may involve many different physical quantities in place of ϕ, and this generality precludes recognition by lexical pattern matching alone. In fact, parser recognition must be supplemented with additional tests (5).

$$\Delta\phi \Leftarrow \phi - \phi \quad \{\text{ Additional Tests }\} \qquad (5)$$

Quantity-Physical	force $\Leftarrow$ mass * accel	3
Quantity-Math	$\Delta\phi \Leftarrow \phi - \phi$	5
Value / Interval	$[1,50] \Leftarrow [0,49] + 1$	2
Grid Location	$\phi_i \Leftarrow \phi_{i+1} + \phi_{i-1}$	4
Vector Analysis	$\phi \cdot \phi \Leftarrow \phi_x^2 + \phi_y^2 + \phi_z^2$	1
Non-Dimensional	$\phi/A \Leftarrow \chi/A + \varphi/A$	1
Dimensions	$L \Leftarrow (L/T) * T$	1
Units	m $\Leftarrow$ m/s * s	1

Table 1: Scientific semantic properties analyzed by the procedure including sample equations, and number of parsers.

1.2 Programming language semantics

These analyzed properties (Table 1) reside in data structures and must be propagated through a code as the programming language dictates. For example, a result's assignment to an array involves a transfer of properties which must respect array organization and instruction sequence. Other language issues include array reference recognition, loop analysis, basic block termination, subroutine call tree ordering, and external variables. The parser paradigm of section 1.1 is not a particularly appropriate solution here; instead, non-general coding is required.

1.3 User Interface

The GUI allows the user control of these analyses, and displays the user's code as well as the analysis results (Figure 1). The user queries the analysis results by selecting text from the displayed code; the properties of variables, expressions, and arrays are displayed, with derivations. A dictionary provides definitions of technical terms. Semantic concepts may be searched for; navigation tools assist in discovering results.

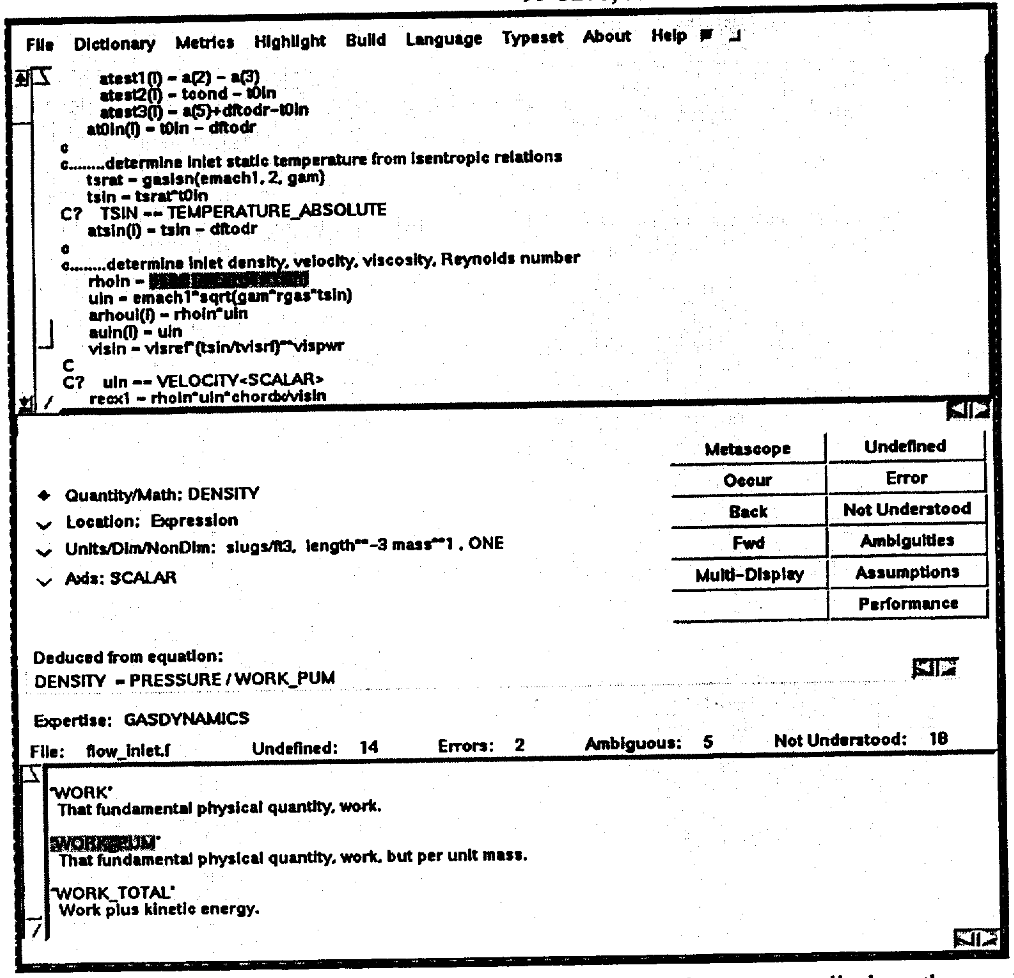
2. Demonstration test cases

The demonstration will involve test cases from a suite of ten state of the art, practical, scientific and engineering codes including an experimental data reduction code, one-, two-, and three-dimensional computational fluid dynamics (CFD) codes for turbomachinery problems, and a chemically reacting fluid flow code. The size of these codes is typically 3-8k loc.

The test case results show 20-30% recognition[1] and establish the feasibility of these analyses. Yet, additional effort is required to reach the degree of recognition required for a practical tool.

3. Bibliography

[1] M.E.M. Stewart, S. Townsend, "An Experiment in Automated, Scientific-Code Semantic Analysis," AIAA-99-3276, June 1999.



```
      atest1(l) = a(2) - a(3)
      atest2(l) = tcond - t0in
      atest3(l) = a(5)+dftodr-t0in
    at0in(l) = t0in - dftodr
c
c........determine inlet static temperature from isentropic relations
    tsrat = gasisn(emach1, 2, gam)
    tsin = tsrat*t0in
C?  TSIN -- TEMPERATURE_ABSOLUTE
    atsin(l) = tsin - dftodr
c
c........determine inlet density, velocity, viscosity, Reynolds number
    rhoin = [illegible]
    uin = emach1*sqrt(gam*rgas*tsin)
    arhoui(l) = rhoin*uin
    auin(l) = uin
    visin = visref*(tsin/tvisrf)**vispwr
C
C?  uin -- VELOCITY<SCALAR>
    recx1 = rhoin*uin*chordx/visin
```

◆ Quantity/Math: DENSITY
⌄ Location: Expression
⌄ Units/Dim/NonDim: slugs/ft3, length**-3 mass**1 , ONE
⌄ Axis: SCALAR

Deduced from equation:
DENSITY = PRESSURE / WORK_PUM

Expertise: GASDYNAMICS

File: flow_inlet.f   Undefined: 14   Errors: 2   Ambiguous: 5   Not Understood: 18

'WORK'
That fundamental physical quantity, work.

'WORK_PUM'
That fundamental physical quantity, work, but per unit mass.

'WORK_TOTAL'
Work plus kinetic energy.

Figure 1: GUI display for the semantic analysis program. The top area displays the user's code; the middle region explains selected text; the bottom area is a dictionary/lexicon.