SIG on Formal Methods

NEWS LETTER VOLUME 7 , JULY 2015

SIG on Formal Methods:

Formal Methods (FM) has been in existence since 1940's, when Alan Turing proved the logical analysis of sequential programs using the properties of program states; and Floyd, Hoare and Naur used axiomatic techniques to prove program correctness against the specifications in 1960's.

These initial successes helped inculcate an interest in applying FM to the field of computer science.

Academia has been instrumental in bringing this field to the forefront, through continued research and development. The use of Formal Methods requires an expert skill-set expertise and therefore, its use is limited to those trained in the field.

Mission Statement:

Computer Society of India wants the field of Formal Methods to have a wider audience and more people to benefit from the application of these methods to all spheres of life. There is a need to use effective, correct and reliable approaches to design, develop and qualify complex, high assurance system software with the rigid schedules and budget. For this we need advanced tools, techniques and methods. Industry standards like RTCA DO-178C (Civil Aerospace), ISO 26262 (Automotive), IEC 61513(Nuclear), EN50126 (Railways) have recommended the usage of formal –method based approach to be used in the various phases of engineering process to achieve the required levels of safety and security.

Today there are proven techniques and tools that can be used in specification, design and verification & validation phases to assure correct requirement-capture, implementation, software functionality and security. This helps in developing high assurance software for applications such as cyber-physical systems, net-centric warfare systems, autonomous robots and Next Generation Air Transportation.

- **One day workshop on FM was conducted during April 2013**

Objectives of the Special Interest Group (SIG) are:

- To bring together scientists, academicians active in the field of formal methods and willing to exchange their experience in the industrial usage of formal methods
- To coordinate efforts in the transfer of formal methods technology and knowledge to industry
- To promote research and development for the improvement of formal methods and tools with respect to their usage in industry.
- To bring out practical engineering methods where formal methods will be integrated with current engineering methods

Some of the known applications of formal methods are:

- Formal verification, including theorem proving, model checking, and static analysis
- Techniques and algorithms for scaling formal methods
- Use of formal methods in automated software engineering and testing
- Model-based formal development
- Formal program synthesis
- Formal approaches to fault tolerance
- Use of formal methods in safety cases
- Use of formal methods in human-machine interaction analysis
- Use of formal methods in compiler validation and object code verification

3. Committee Members

1. Ms. Bhanumathi K S, Convener
2. Mr.Chander Mannar
3. Prof.Anirban basu
4. Mr. Suman Kumar
5. Prof. Shyam Sundar
6. Ms. Manju Nanda
7. Ms. J. Jayanthi
8. Ms. Saroja Devi
9. Prof. Meenakshi D'Souza
10. Dr. Yoganand Jeepu
11. Dr. Swatnalatha Rao
12. Dr. Aditya Kanade
13. Dr. A. Indira
14. Mr. Dhinakaran Pillai

Convener:

Bhanumathi K S
“Ganadhakshya” #406, 8 C Main,
H R B R First Block
Kalyan Nagar
Bangalore 560043
Email:bhanushekar@gmail.com
Mobile:+91 95350 92589

Formal Methods in the Requirements Phase of Software Engineering

Compiled by Dr Manju Nanda

Nowadays, the major problems of software engineering are encountered at the high levels of system development, both scientifically and in the industrial practice. The modern software life-cycle models recognize that defects injected in the initial software development phases are the most expensive ones.

Requirement engineering is the most effective phase of software development process. It aims to collect good requirements from stakeholders in the right way. Requirements engineering for software development process is a complex exercise that considers product demands from a vast number of viewpoints, roles, responsibilities, and objectives. The current practice in capturing requirements is done by an ad-hoc and informal manner.

In developing high-quality software understanding and documenting the software requirements is very crucial. Studies have shown that many software defects can be traced to ambiguous, incomplete, and inconsistent requirements specifications and that fixing these defects can be very costly, especially when the defects are detected late in development cycle of the engineering process. A promising approach to construct a precise, complete, and consistent requirements specifications is to represent the requirements in a formal specification language and to check the specification for properties, such as completeness and consistency, with formal analysis techniques.

Formal specifications have properties such as precision and conciseness that make them ideal for the description of requirements. Second, less formal notations allow ambiguities to creep into the specification, making it harder to analyze the requirements and posing errors in the implementation phase of the process.

There exist formal, semi-formal techniques to capture the software requirements. In formal techniques, formal languages help in capturing the requirements completely without any ambiguities. In semi-formal techniques, graphical notations help in capturing requirements visually.

Advantages of formal specifications

Specifications must leave as much Freedom as possible to the implementer

Main advantages of formal specifications:

♦Abstraction: good mechanism to support implementation freedom

♦Precision: still maintain ability to precisely describe what is needed of the system

Sometimes it is indeed possible to refine the formal specification into the final implementation
♦Formal development process

Different levels of rigour
Level 1: Use of concepts and notation from discrete mathematics
Level 2: Use of formalized abstract specification languages with some mechanized support tools
Level 3: Verification of the abstract precise specification
Level 4: Fully formal development process (refinement from abstract specifications)

Classes of Formal Methods:
Model-based approaches (B, VDM, Z)
Algebraic approaches (CASL, Act One, Larch, OBJ)
Transition-based approaches (Statecharts, Promela)
Process algebras (CSP, CCS)
Logic-based approaches (TLA, RTL)
Reactive approaches (Petri nets, SDL, SAO)
ISO standards for VDM and Z

List of requirement specification techniques are listed below :

- Abstract State Machines (ASMs)
- A Computational Logic for Applicative Common Lisp (ACL2)
- ANSI/ISO C Specification Language (ACSL)
- Alloy
- Autonomic System Specification Language (ASSL)
- B-Method
- CADP
- Common Algebraic Specification Language (CASL)
- Java Modeling Language (JML)
- Knowledge Based Software Assistant (KBSA)
- Process calculi
    - CSP**(communicating sequential processes)**
    - LOTOS**(Language Of Temporal Ordering Specification)**
- Actor model
- Esterel
- Lustre
- mCRL2
- Perfect Developer
- Petri nets
- Predicative programming
- RAISE(*Rigorous Approach to Industrial Software Engineering*)
- SPARK Ada

- Spec sharp (Spec#)
- Specification and Description Language
- TLA+
- USL
- SCR (Software Cost Reduction)
- VDM (Vienna Development Method)
    - VDM-SL
    - VDM++
- Z notation
- Rebeca Modeling Language

Model-based approaches

Specify admissible system states (or values) at some arbitrary snapshot, using mathematical entities like sets, relations, first order predicate logic (pre-condition/ post-conditions , invariants)

Example : B, Vienna Development Method, Z

State-based approaches

A system is seen as a set of states and operations that change the state

An invariant is defined as a predicate that must be satisfied in all states

For each operation, a pre-condition is defined, and an operation can take place only if its pre-condition is satisfied by the current state

For each operation, also a post-condition is defined, and this predicate must be true at the modified state that results after the operation has taken place

Algebraic approaches

Specify admissible system behavior in terms of properties (conditional equations) that document the effect of composing functions

Need to state explicitly which properties one wants the system to have, after which any model that satisfies that theory is deemed to be acceptable

Example : CASL, Act One, Larch, OBJ

Transition-based approaches
Characterize admissible system behavior using the required transitions in state machines

E.g. for each input state and triggering event, and/or Boolean expression
Example : Statechart, Promela

Process algebras

A process refers to the behaviour of a system

Typically used to describe distributed and/or parallel systems in an algebraic fashion

Example: Calculus of Communicating Systems (CCS), Concurrent Sequential Processes (CSP)

Logic-based approaches

Expresses system behavior as logic expressions over traces of allowed operation calls

Example: Temporal Logic of Actions (TLA), Real Time Logic (RTL)

Case studies of the formal requirement specification capture

Reactive approaches

Specify system behaviour completely enough so that the specification can run on some Abstract machine as a structured collection of processes

Example: Coloured Petri nets, SDL, SAO

References:

1] Using Formal Methods to Assist in the Requirements Analysis of the Space Shuttle GPS Change Request, Ben L. Di Vito, Larry W. Roberts, http://shemesh.larc.nasa.gov/fm/papers/DiVito-96-cr4752-Shuttle-GPS.pdf

2] Experience using Formal Methods for Capturing Requirements of Web-Based Applications, Abdesselam Redouane, First IEEE International Conference on Cognitive Informatics (ICCI'02), http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1039300

3] A model-driven process for engineering a toolset for a formal method, Paolo Arcaini, Angelo Gargantini, Elvinia Riccobene, and Patrizia Scandurra, SOFTWARE|PRACTICE AND EXPERIENCE, Softw. Pract. Exper.2000;00:1, http://cs.unibg.it/gargantini/research/papers/tbfm_jspe11.pdf

4] An Effective Requirement Engineering Process Model for Software Development and Requirements Management , Dhiren Pandey, U.Suman, A. K. Ramani, International Conference on Advances in Recent Technologies in Communication and Computing,2010, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5656776

5] Experiences Using Lightweight Formal Methods for Requirements Modeling, Steve Easterbrook, Robyn Lutz, Richard Covington, John Kelly, Yoko Ampo, and David Hamilton, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 24, NO. 1, JANUARY 1998, http://www.cs.toronto.edu/~sme/papers/1998/NASA-IVV-97-015.pdf

6] Formal Methods for Specifying, Validating, and Verifying Requirements, Constance L. Heitmeyer, http://www.cs.man.ac.uk/~banach/CSIComm-May07-FM-Papers/020Heitmeyer.pdf

7] Using Formal Methods for Requirements Specification of a Proposed POSIX Standard, Neal R.Reizer, Gregory D. Abowd, B.Craig Meyers, Patrick R.H. Place, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=292395&tag=1

8] http://formalmethods.wikia.com/wiki/VL

9] Formal specification languages, Maurice ter Beek, http://www.liacs.nl/~mtbeek/re-formal.pdf

10] http://sourceforge.net/projects/overture/