Software Engineering Practices in Embedded System Design Using Discrete Modeling Techniques

Dr. Manuj Darbari, Hasan Ahmed

***Abstract* - This paper highlights the requirement engineering process for embedded system design in mathematical form. The objective is to formalize the requirement phase with respect to the actual behavior of the processes in real time mode. The paper deals with embedded chip design requirements for Mobile Phones and similar embedded systems.**

***Keywords* – Groups, Rings, Fields.**

I. Introduction

Embedded systems most often need real time programming. Real Time operating systems and their working are generally shown by two methods: 1. Finite state machine 2. Petri-Nets. But theses two modeling methods have shown certain limitations as many sophisticated embedded systems are multiprocessor systems and the processes have short latencies. In order to classify the processes for short latencies we have identified a third methodology named Discrete Modeling Technique.

We have to give human perception and sociology their due weightage as the requirements are gathered from human beings and to bring forth their feelings and anticipations to form a software system, a huge chunk of requirements are to be analyzed and categorized into three sorts of persons - introverts, extroverts and ambiverts.

If we try to express requirements of a software system in mathematical terms, then there can be one of the ways of verifying their consistency and validation. Normally, the requirements are considered in isolation and then enumerated as a list. But, we should not forget that software systems are complex and expose complicated and a large number of ways in which they can be used and hence, a large number of behaviors (expected and unexpected).

Manuscript received March 2, 2010.
Manuj Darbari is with Babu Banarasi Das National Institute of Technology and Management, Lucknow, India (corresponding author) phone: 091-522-2311551 e-mail: manujuma@ rediffmail.com
Hasan Ahmed is working as Sr. Design Engineer in R&D wing of Nokia India Ltd, Bangaluru. (e-mail: hasaninbox@gmail.com)

This also has implications for security requirements (and expectations) from the system in question [1, 3, 4, 5]. Normally, Functional Requirements are first looked into when signing off the requirement phase than Non Functional Requirements (NFRs).

II. The Model

Let us say that a requirement ‘a’ can be met and a software system can be expected to fulfill that. Now, we take another requirement ‘b’ and assign it some expectation from a system. We know how a system would react while obliging to the two requirements individually. But, a third requirement comes into play when the two requirements are to be met concurrently. Two requirements are said to be concurrent when system expectation of the first one is not over and the second one gets triggered. Since this third requirement was not one of the formally specified requirements, it may lead to some unexpected phenomenon (if not perceived/planned for). This can be very dangerous for safety critical systems. We, now, go on to present a mathematical form of meeting the need of validating the requirements (while assuring their completeness) and assuring that the system behaviour lies under the scope of consistency and acceptability.

The concept of Groups, Rings and Fields is well known in algebra [2]. If we translate the requirements of a system (to be built) into mathematical elements, then we will be able to operate on them algebraically. Let us first discuss the just mentioned algebraic concepts. Groups will be discussed first post which Rings and Fields will be discussed.

A group G, denoted by $\{G, \cdot\}$, is a set of elements with a binary operation denoted by ‘$\cdot$’ that associates to each ordered pair (a,b) of elements in G an element $(a \cdot b)$ in G, such that axioms A1 – A4 are followed. Let us go ahead to express these axioms one by one.

1. A1: Closure: If a and b belong to G, then $a \cdot b$ is also in G.
2. A2: Associative: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all a, b, c in G.
3. A3: Identity element: There is an element e in G such that $a \cdot e = e \cdot a = a$ for all a in G.
4. A4: Inverse element: For each a in G there is an element a` in G such that $a \cdot a` = a` \cdot a = e$. A group is an Abelian group if it satisfies A5 also in addition to A1-A4.
5. A5: Commutative: $a \cdot b = b \cdot a$ for all a, b in G.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

We define exponentiation within a group as repeated application of the group operator, so that a^3 = a · a · a. We also define a^0 = e (which is an identity element) and a^n = (a`)^n. A group G is cyclic if every element of G is a power of a^k (k is an integer) of a fixed element a of G. A cyclic group is always Abelian and may be of finite/infinite elements.

A ring R, denoted by {R, +, X} is a set of elements with two binary operation called addition and multiplication, such that for all a, b and c in R the following axioms are obeyed apart from A1-A5.

6. M1: Closure under multiplication: If a and b belong to R, then 'ab' is also in R.
7. M2: Associativity of multiplication: a(bc) = (ab)c for a, b and c in R.
8. M3: Distributive laws: a(b + c) = ab + ac for all a, b and c in R. Similarly, (a + b)c = ac + bc for all a, b and c in R.

In essence, a ring is a set wherein we can do addition (and hence subtraction) and multiplication without leaving the set.

A ring is said to be commutative if it satisfies M4 also.

9. M4: Commutative of multiplication: ab = ba for all a, b in R.

A field F, denoted by {F, +, X} is a set of elements with two binary operations called addition and multiplication such that for all a, b and c in F, A1-A5 and M1-M7 are obeyed where M5-M7 are stated below.

10. M5: Multiplicative Identity: There is an element 1 in R such that a1 = 1a = a for all a in R.
11. M6: No zero divisors: If a and b are in R and ab = 0, then either a = 0 or b = 0.
12. M7: Multiplicative inverse: For each a in F, except 0, there is an element a^(-1) in F such that a(a^(-1)) = (a^(-1))a = 1.

In essence, a field is a set in which we can do addition, subtraction, multiplication and division (a/b = a(b^(-1))) without leaving the set. The figure 1 here represents Group, Ring and Field in set relationship. Having described the algebraic concepts, we can associate the elements of such sets as requirements of a software system and the results of operations on set elements as system behaviour. The result of an operation of addition of two requirements can be described as results of requirements following each other with no overlap in time. The result of an operation of multiplication of requirements can be called as a result of meeting the two requirements simultaneously in terms of system behaviour (More explicably, we can call a multiplied by b as requirement a is yet being met by the system and b gets triggered). Similarly, other operations can be thought of.

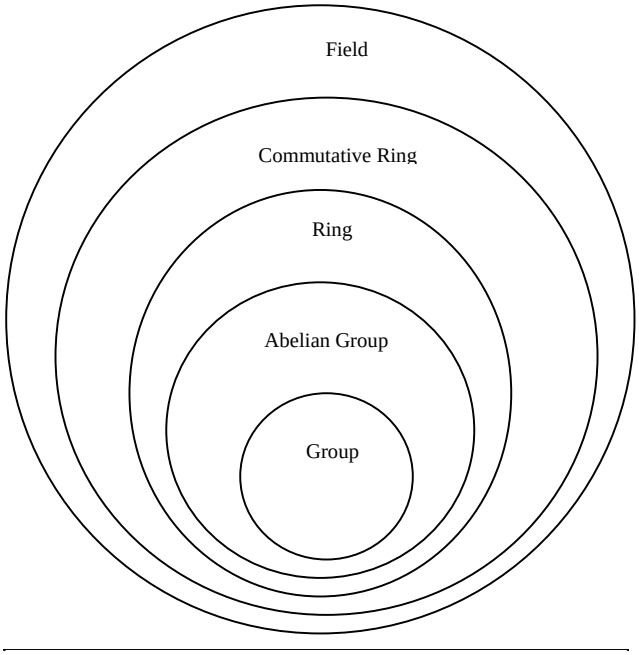

Figure 1: Basic categorization of Group, Ring and Field

The behaviour of a system should also be expressible as lying under groups, rings or fields under a set of requirements. This will help us in identifying the systems in various levels of risk proneness. Closure of meeting any two requirements under any operation can be whether such an operation can be met while remaining in a set of defined operations (or expected system behaviors)(A1). Associativity can hold when a third requirement is to be met while two other requirements are already in progress under some defined operation and the result is same (in terms of system behaviour) even if we change the order of triggering requirements for the same operation (A2). Identity element in requirements can be an operation which does not affect/disturb an already ongoing operation (for a triggered requirement) when coupled with every other requirement (e.g. LCD light gets switched on whenever we touch keypad for any triggering any function in a mobile phone (A3). Inverse element in requirements can be triggering a requirement which when applied to any particular requirement(s) under a pre-defined operation would switch the system back to standby state (i.e. ability to abort a given operation without the system getting hanged or malfunctioned)(A4). Commutativity can be defined as the same result (in terms of system behaviour) for meeting the two requirements under a defined operation irrespective of the order of triggering those two requirements (A5). Similarly, other axioms can be defined. But such definitions would depend on the context of the system and requirements.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

The figure 2 depicts the case of addition of three requirements A, B and C. Herein, the states W, X , Y and Z can be the same state or different ones. For example, W=X=Y=Z if the system is expected to get back to the same state (e.g. standby state) after meeting requirement A or B or C.

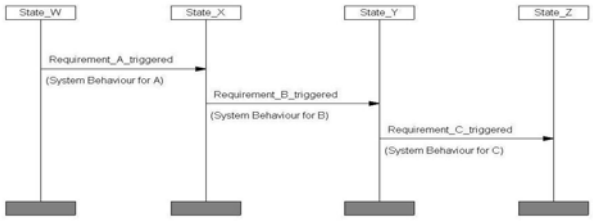

Figure 2: Addition of A, B and C

The figure 3 depicts the case wherein a second requirement is triggered while the first one is yet to complete. Here again, the states W, X and Y can be the same state or different ones. The result (i.e. system behaviour) of this use case can be thought of as multiplication of A and B. Also, please mind the transitory state T when B gets triggered. There should be enough system resources to, at least, register and later process (if not start processing with immediate effect) the triggered requirement B.

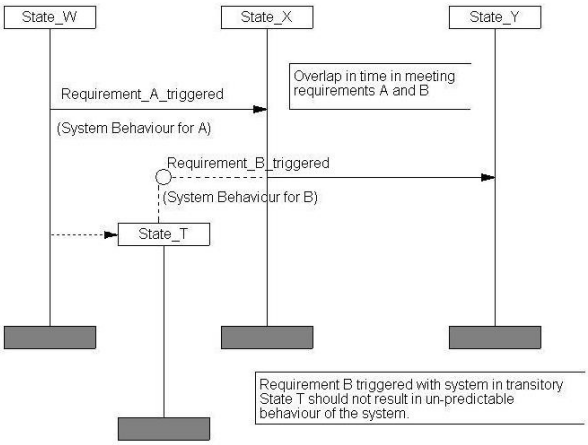

Figure 3: Multiplication of A and B

The figure 4 depicts the case of the system behaviour for the algebraic expression: A· B+C. Here too, the states W, X, Y and Z can be the same state or different ones.

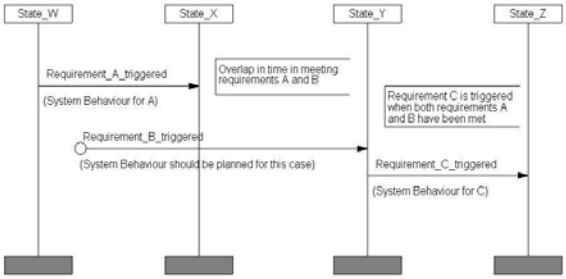

Figure 4: Expression A· B+C

III. Conclusion and Future scope

We have to keep in mind that the set of FRs (Functional requirements) is not mutually exclusive from NFRs (Non Functional requirements). If we can describe the requirements and system behavior (at some level of abstraction), we can use specification languages (based on logic) for automated reasoning and analysis. This would help us in better predictability of system's behavior. Exemplifying, deontic logic we can describe permissions (and obligations) while temporal and linear logic can well formalize timing information and use of resources, respectively. This approach can be well extended beyond embedded systems to networks and even social networking sites (where in a lot ways data given by the user and collected by the site can be subjected to distributed usage control) [6, 7, 8]. If used for distributed usage control, the concept can lend a mathematical modeling means to privacy, data control and usage [9, 10].

However, a lot has to be answered and formalized in this approach. This would depend upon the context of system, the extent to which any requirement can be aborted after being triggered (like in case of safety critical systems), whether a big requirement can be broken into a number of individual requirements (while not forgetting their dependence or independence) and to what extent, etc. The scheduling of various resources for requirements (or sub-requirements) and the determination whether a requirement (or a sub-requirement) is mutually exclusive is also possible using this approach. However, one major challenge would be to define the operations and system behaviour in logical (mathematical) terms.

IV. Acknowledgment

We are thankful to Mr. Allan Frederiksen, the Head of GERAN Radio Software at Nokia R&D Bangalore, for his co-operation in letting us bring the paper in the current form. We are also obliged to Nokia R&D GERAN team at Bangalore for useful comments and discussions with respect to the employability of this paper's concepts for Nokia phones.

References

[1] G. Hogben, "Security Issues and Recommendations for Online Social Networks," ENISA, Tech. Rep., October 2007.

[2] W. Stallings, Cryptography and network security: principles and practice. Prentice Hall, 4 edition, November 2005.

[3] Simone Fisher Hubner. "IT-Security and Privacy: Design and Use of Privacy-Enhancing Security Mechanisms." Springer, 1 edition, June 15 2001.

[4] Russel, D, Gangemi, G.T., "Computer Security Basics", O'Reilly, 1991.

[5] RFC 2828, Internet Security Group. http://www.ietf.org/rfc/rfc2828.txt.

[6] A. Acquisti and R. Gross, "Imagined Communities: Awareness, Information Sharing, and Privacy on the Facebook," in Privacy Enhancing Technologies Workshop (PET), Robinson College, Cambridge, United Kingdom, June 2006.

[7] L. Edwards and I. Brown, "Data Control and Social Networking: Irreconcilable Ideas?" Harboring data: Information security, law and the corporation, A. Matwyshyn, ed., Stanford University Press, 2009.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

[8] S. Weiss, “The Need for a Paradigm Shift in Addressing Privacy Risks in Social Networking Applications,” The Future of Identity in the Information Society, 17th June 2008.

[9] M. Hilty, D. Basin, and A. Pretschner, “Distributed Usage Control”, 2006.

[10] M. Hilty, D. Basin, and A. Pretschner. “On obligations.” In Proc. ESORICS, pages 98-117, 2005.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)