This article is part of the IEEE Reliability Society 2009 Annual Technology Report.

Software Reliability Issues: Concept Map

Goutam Kumar Saha
Email: sahagk@gmail.com

Abstract
This article discusses various key concepts of software reliability issues using a concept map (CMap). The CMap here is meant for at-a-glance meaningful understanding of this important topic. A concept map is a two-dimensional representation of cognitive structures showing the hierarchies and interconnections of concepts. Robust software design techniques including Enhanced Single–Version Programming (ESVP), N-Version Programming (NVP) have been described. The proposed enhanced ***"Single–Version– Based N-Version Programming (SNVP)"*** approach of designing ultra-reliable software is also briefly presented here. This article aims to present an overview of some complex issues concerning software reliability by the graphical representations of software reliability topic.

Keywords: Software Quality, Software Reliability, Single- Version Programming (SVP), Enhanced Single-Version Programming (ESVP), NVP, Enhanced Single–Version–Based N-Version Programming (ESVBNVP or SNVP).

Software Reliability

Software reliability [1,3,5] is defined as the probability of failure-free software operation over a specified period of time in a specified environment. Design perfection is reflected by software reliability. Software reliability problems are common because of the high software – complexity. Software reliability is considered to be an important attribute of software quality. Other attributes of software quality are: software functionality, software usability, software performance, software serviceability, software capability, software install-ability, software maintainability, and software documentation. Software reliability is inversely related to software complexity. Whereas software complexity is directly related to other software quality factors such as, software functionality, capability, etc. Software failures are basically due to software design errors. Software design errors are primarily because of erroneous specifications, immature program coding, insufficient testing, and erroneous usage. Again unexpected failures possibly because of operational or environmental errors are also not very uncommon. Software design faults mostly cause software faults, whereas physical faults of hardware mostly because of hardware faults. Software bug fixing might introduce other failures in software. Better engineering technique like Clean-Room one may be used for increased software reliability. Again software reliability can be enhanced by robust design such as, Enhanced Single-Version Programming (ESVP) approach that employs assertions, application semantic-based assertions, program control checking, replicated instructions, robust data structures, error detection code, register error detection code, processor status word (PSW) error detection or code for recovery and so on. Software quality does not degrade with time but can become unusable after a period of time, because software faults often remain undetected. Software design faults are tolerated or masked off through voting the outputs from multiple versions of software. Software operational faults or transient faults are tolerated by employing software replication. Software design faults cannot be tolerated by replicated software. Again, common mode design faults cannot be tolerated by multi-version software. Software reliability using ESVP is economically cheaper than that of the multiple versions software or N-Version Programming (NVP) [2,4,5].

Overview of the SNVP Approach

In order to design ultra-reliable software, we may think of using ESVP approach for designing each version of software in an NVP. In other words, we introduce ESVP-based robustness while designing each version of software in an N-Version Programming (NVP) based software design. Such software design approach meant for attaining extreme software reliability is a hybrid approach where we design NVP software on employing ESVP. This proposed extreme reliable software design approach of

"Enhanced **S**ingle-**V**ersion–**B**ased **N-V**ersion **P**rogramming" (**ESVBNVP**) or in short, is named as "SNVP" that is, enhanced SV–Based NVP (or **S**ingle-**V**ersion–**B**ased **N-V**ersion **P**rogramming). The proposed ***SNVP*** approach (as shown in figure-1) of ultra-reliable software design aims to have goodness of both the approaches namely, NVP and SVP. The ***SNVP*** software design approach would introduce high robustness in software in order to tolerate software design faults as well as various unforeseen operational faults that might occur during the useful life period of software that needs ultra-reliability. However, this approach needs extra design cost and extra execution time of about 2.14 times (typically for a 3-Version programming where each version is enhanced with assertions and control flow checking fix) the execution time of an application without any fault tolerance fix.

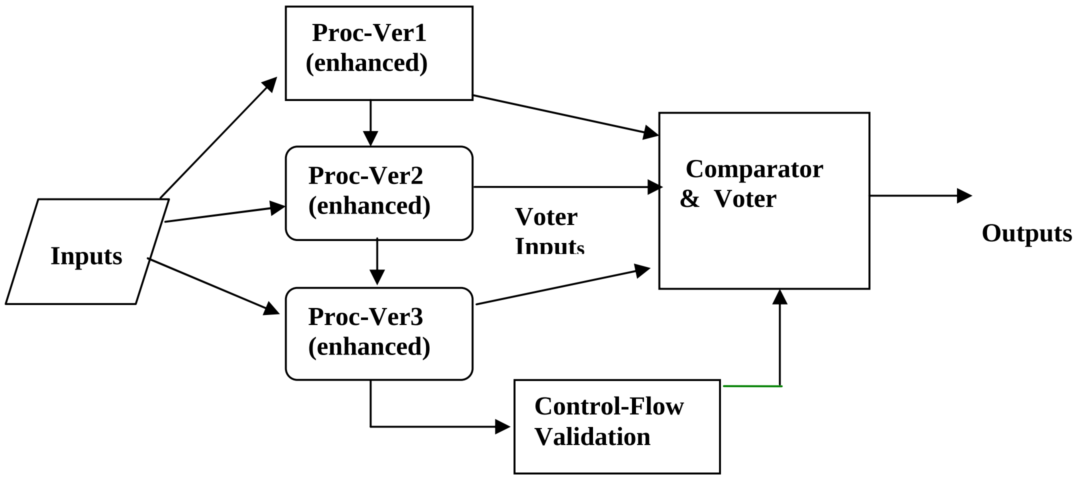

Figure 1. An SNVP Scheme.

Software Reliability Engineering (SRE)

We understand that Software Reliability Engineering (SRE) is nothing but the quantitative study of the operational behavior of software-implemented systems with respect to user expectations on reliability. A software reliability model aims to quantify reliability on considering sufficient amount of software-complexity along with appropriate constraints, assumptions and factors made to suit some situations on using usually a higher order exponential or logarithmic mathematical function to relate the factors with reliability.

Concept Map (CMAP)

Concepts are the generalization of knowledge of ideas conveyed in some forms for example, books, documents, speeches or lectures. Concept is nothing but a perceived regularity in events or objects. Propositions state how concepts are linked together. A Concept Map comprises of concepts and propositions. Concept Maps are the graphical representations of knowledge that are comprised of concepts and the relationships among them. Concept maps are 2-dimensional representations of cognitive structures showing the hierarchies and interconnections of concepts involved in a discipline or a sub-discipline. This is an important tool for developing our both sensing and intuitive skills. Sensing skill is important to focus on already known and new information, whereas intuition skill helps us to construct relationships. It is to organize the information by groups. In a concept map, the nodes (in circles or rectangles) have been used to enclose the key concepts and these nodes have been linked with lines (normally directed downward) and words (e.g., verbs, preposition etc.,) that describe the connection. Professor Joseph Novak developed concept maps that represent organized knowledge. A domain expert has hierarchically structured knowledge. Organized knowledge is comprised of concepts and propositions that are hierarchically structured in cognitive structure to aid creativity that begins with infants. Creativity is must to see interrelationships between various map segments. We need context dependent organized knowledge for effective teaching and effective learning and for answering questions. Creativity only can produce a very high level of meaningful statement. Concept is the highest level of abstraction for the map but it is the lower level of abstraction in the ontology.

CMap Characteristics
(a) A hierarchical concept map contains the most general concept at the top and the most specific one at the bottom,
(b) Cross links are to link different map segments,
(c) Examples are to clarify the meaning of a concept.

In order to construct a concept map we must have familiarity with the general topic as well as an in-depth knowledge and understanding of a specific topic.

CMapping Guidelines
(a) Note the major concepts,
(b) Note more specific concepts for each major concept for grouping related ideas,
(c) Interlink the major ideas,
(d) Write linking words.

CMap Usefulness
Concept maps are very useful as a means for representing the emerging science knowledge and for increasing meaningful learning in sciences in contrast to simply memorizing the text. Representing the expert knowledge of individuals or of teams in research, government, and business and in education becomes easier by a useful concept-mapping tool such as IHMC-KM. It is to stimulate our idea generation and creativity. It is definitely carving out a strong position for brainstorming, complex ideas communication, and formal argument representation. Formalized concept maps are being used in software design or in UML. It is a first step in ontology building. A concept map for software reliability has been developed and presented here (as shown in Figure-2).

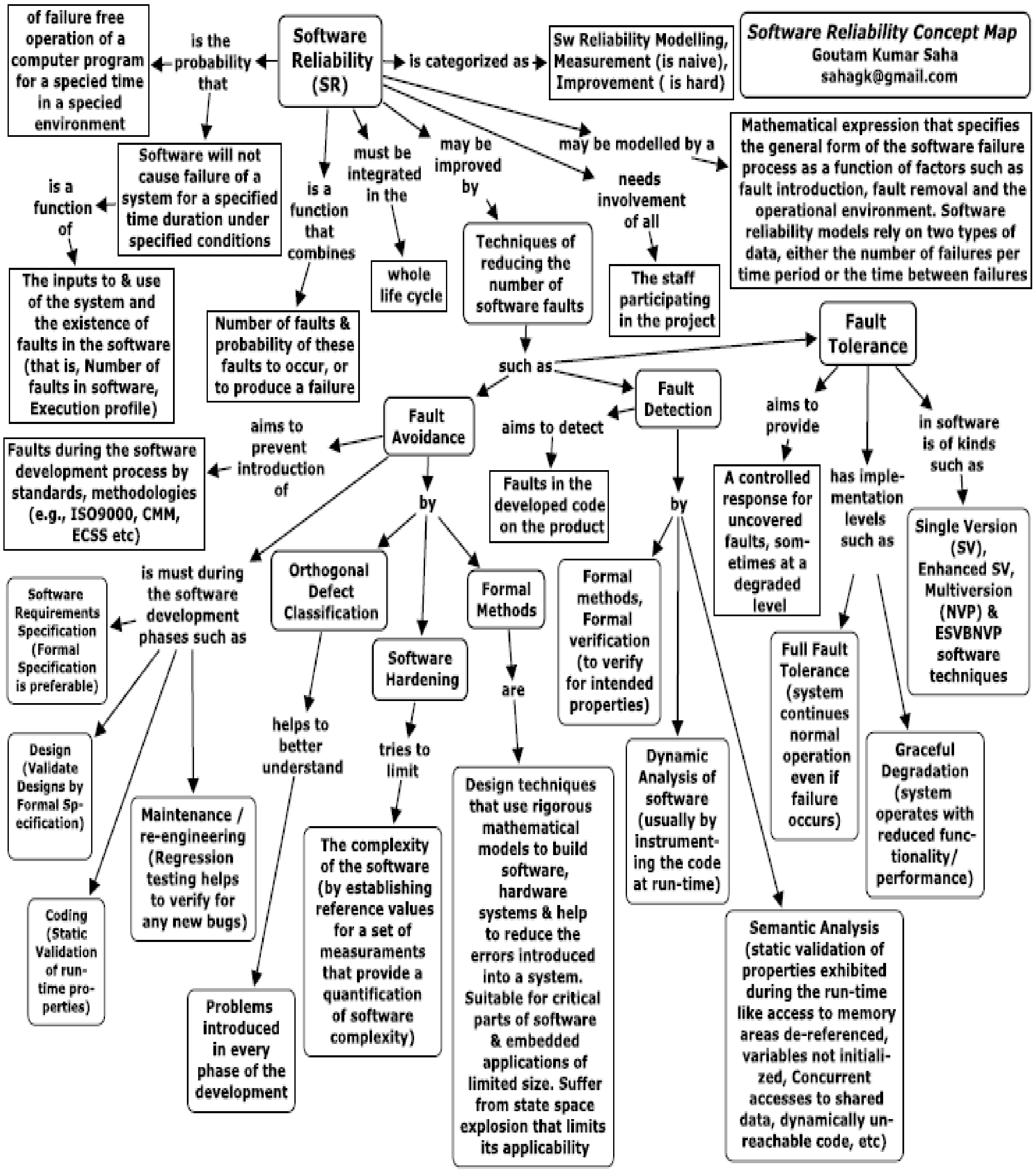
Software Reliability Concept Map
Goutam Kumar Saha
sahagk@gmail.com
Software Reliability (SR)
is the probability that
of failure free operation of a computer program for a speci ed time in a speci ed environment
Software will not cause failure of a system for a specified time duration under specified conditions
is a function of
The inputs to & use of the system and the existence of faults in the software (that is, Number of faults in software, Execution profile)
is categorized as
Sw Reliability Modelling, Measurement (is naive), Improvement (is hard)
is a function that combines
Number of faults & probability of these faults to occur, or to produce a failure
must be integrated in the
whole life cycle
may be improved by
Techniques of reducing the number of software faults
needs involvement of all
The staff participating in the project
may be modelled by a
Mathematical expression that specifies the general form of the software failure process as a function of factors such as fault introduction, fault removal and the operational environment. Software reliability models rely on two types of data, either the number of failures per time period or the time between failures
such as
Fault Tolerance
Fault Detection
Fault Avoidance
aims to prevent introduction of
Faults during the software development process by standards, methodologies (e.g., ISO9000, CMM, ECSS etc)
is must during the software development phases such as
Software Requirements Specification (Formal Specification is preferable)
Design (Validate Designs by Formal Sp-ecification)
Coding (Static Validation of run-time pro-perties)
Maintenance / re-engineering (Regression testing helps to verify for any new bugs)
by
Orthogonal Defect Classification
helps to better understand
Problems introduced in every phase of the development
Software Hardening
tries to limit
The complexity of the software (by establishing reference values for a set of measurements that provide a quantification of software complexity)
Formal Methods
are
Design techniques that use rigorous mathematical models to build software, hardware systems & help to reduce the errors introduced into a system. Suitable for critical parts of software & embedded applications of limited size. Suffer from state space explosion that limits its applicability
aims to detect
Faults in the developed code on the product
by
Formal methods, Formal verification (to verify for intended properties)
Dynamic Analysis of software (usually by instrument-ing the code at run-time)
Semantic Analysis (static validation of properties exhibited during the run-time like access to memory areas de-referenced, variables not initiali-zed, Concurrent accesses to shared data, dynamically un-reachable code, etc)
aims to provide
A controlled response for uncovered faults, som-etimes at a degraded level
has imple-mentation levels such as
Full Fault Tolerance (system continues normal operation even if failure occurs)
Graceful Degradation (system operates with reduced func-tionality/ performance)
in software is of kinds such as
Single Version (SV), Enhanced SV, Multiversion (NVP) & ESVBNVP software techniques

Figure-2. Software Reliability CMap

Conclusion

The article here has presented a brief visual description on various important concepts of software reliability through a concept map. The concept map has been developed to describe important topics and sub-topics of software reliability including their inter-relationships for faster and better cognition and meaningful understanding without using verbose text. Such Concept Map-enabled web-based work is a significant step forward to an effective solution to e-Learning, e-Governance and education technology of the future. The SNVP approach as stated here aims toward developing an high reliability system.

References

[1] Jiantao Pan, “Software Reliability,” 18-849b Dependable Embedded Systems, CMU, 1999.
[2] Goutam Kumar Saha, "Understanding Software Fault Tolerance Using a Concept Map," IEEE Reliability Society Newsletter, Vol. 54, No. 2, June 2008, IEEE Press, USA.
[3] E. Valido-Cabrera, "Software Reliability Methods," Technical Report, Technical University of Madrid, 2006.
[4] Goutam Kumar Saha, "Software based Fault Tolerance: a Survey," ACM Ubiquity, Vol.7, No. 25, pp. 1-15, July 2006, ACM Press, USA.
[5] Goutam Kumar Saha, "Understanding Software Reliability Concepts," IEEE Reliability Society Newsletter, Vol. 54, No. 3, 2008, IEEE Press, USA.