Imposing Software Traceability and Configuration Management for Change Tolerance in Software Production

Hussain Saleem[1], S. M. Aqil Burney[2]

[1]Department of Computer Science, UBIT, University of Karachi, Karachi, Pakistan.
[2]College of Computer Science & Information Systems, Institute of Business Management, IoBM, Karachi, Pakistan.

Abstract

The presence of change in any production process, especially in software development, is usually intolerant and highly resistive as it creates confusions among developers and breaks the confidence of customers resulting in non-reliable production. Change is inevitable especially while software is built. There could be several reasons for such confusions likely deployment of the application on multiple variant platforms or having multiple releases, or distinct support tools for simultaneous testing, or developers with deputation over multiple projects at the same time, or following bad or weak practices of undertaking frequent modifications over any certified post-release. Such confusions are indeed needed to be managed in a highly organized manner with recording the traces in the case for the post inspective backpropagation for quick determination of any error causing the software failure. Business ventures are operating in a highly competitive arena globally, where such practices are extremely risky and annoying. Software configuration management, SCM fulfils the objective to avoid such confusions when change is requested, where keeping maximum trace record of development i.e. software traceability highly help to determine the timely error for correction without any hassle. In this paper, we have tried best to give awareness to all software development stakeholders understanding trouble of change, maintaining traceability with a quality focus, where practices of change and configuration management are explored.

Key words:

business; change; configuration; quality; software; traceability;

Acronyms Used:

CA	Computer Associates
CCP	Change Control Process
CM	Configuration Management / Change Management
CMM	Capability Maturity Model
CMM-I	Capability Maturity Model - Integration
CSA	Configuration Status Accounting
Delta - Δ	Delta - Symbol used for small change with increment or decrement
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Standards Organization
JCL	Job Control Language
MobileApps	Mobile Application Software(s)
RCS	Revision Control System
RFC	Request for Change
RT	Requirements Traceability
SCA	Software Configuration Audits
SCAMPI	Standard CMMI Appraisal Method for Process Improvement
SCB	Software Configuration Build(ing)
SCC	Software Configuration Control
SCCB	Software Change Control Board
SCI	Software Configuration Identification
SCM	Software Configuration Management
SDLC	Software Development Life Cycle
Six-Sigma	Six Sigma (Quality Control Standards, with six standard deviations)
SPICE	Software Process Improvement & Capability Enhancement
SQA	Software Quality Assurance
V&V	Verification and Validation
VCS	Version Control System
WebApps	Web Application Software(s)

Manuscript received January 5, 2019
Manuscript revised January 20, 2019

1. Introduction

Software production is one of the profitable and demanding best ventures of business globally since the last five decades. It has a vast future trend as a key market contributor. During the software production process, changes and modifications are requested from various stakeholders being technical or non-technical. It is well said by the Greek Philosopher "Heraclitus" that "Nothing endures but change." (Quote: 544 BC-483 BC). Change creates confusions and it is inevitable while software is built. Software configuration management i.e. SCM fulfils the objective to avoid such confusions when change is requested. According to the survey conducted by Computer Associates (CA) in order to know why confusions are raised generally [1]. It is found that 88% people in software team deploy the application on multiple platforms including 44% deployment on more than 4 platforms. 61% engineers produce and maintain multiple releases over multiple platforms. 52% of SQA engineers acquire support tools for development and testing simultaneously. 46% engineers work on multiple projects at the same time that has multiple lifecycles. 39% people make frequent changes to already released and certified packaged programs. Table-I expresses the data for software engineer's involvement that rise confusion due to change. The same distribution is shown in Fig.1 as a horizontal bar graph. Some of the people follow best practices, some follow fluffy and others follow weak practices while handling software development and testing. These are some of the negligence attempts. Enterprises are operating in a highly competitive arena globally, where such practices are highly risky. Not managing modification or change in an organised and trained way is much riskier which destroy the development efforts. In today's technological revolution, computer hardware and software infrastructures are much complex that covers multiple devices (desktop, laptop, tablet, mobile, wrist watches etc.), platforms, and programming tools (languages), not every business is on network or web-based. Negligence can result in loss of customer confidence, worsening quality, wastage of resources and effort, and ultimately diminishes market competency. Therefore, today every enterprise requires change management support [2] [3] [4].

This paper portrayed ways to overcome disbalance due to change entropy with software configuration management method possessing the future vision of keeping traces for

record software traceability need. With the introduction given in Section-I, software technology and business venture along with software product nature and usage characteristics with the issues related to software quality and trouble of change are discussed in Section-II. Section-III elaborated briefly the concept frame of SCM practice, its activities, focal and functional areas, along with generic change control process. The notion of traceability is discussed in Section-IV with the need and benefit of traceability in Section-V. The paper is concluded in Section-VI. The list of acronyms and abbreviations used in text is also presented at the beginning.

Table 1: Reasons arising confusions during development

Sr.#	Involvement	Reason for Confusion
1.	88%	People in software team deploy the application on multiple platforms.
2.	44%	Deployment on more than 4 platforms.
3.	61%	Engineers produce and maintain multiple releases over multiple platforms.
4.	52%	SQA engineers acquire support tools for development and testing simultaneously.
5.	46%	Engineers work on multiple projects at the same time that has multiple lifecycles.
6.	39%	People make frequent changes to already released and certified packaged programs.

Data extracted from [1]

2. Technological Foundations

2.1 Software Technology and Business Venture

Rapid developments and improvements in software technology have empowered more and more people and enterprises to enter in the software export business around the world [5] [6] [7]. In order to get succeeded by means of getting projects done with quality results and reduced costs, they must be vigilant for finding new ways to adopt methods and to move faster with true vision and wisdom comparing to the other players of the field. There is no exception of policies for Software business establishment and export. Business methods are the same but only thing is that software has a property of volatility in nature.

2.2 Software Product Nature and Usage Characteristics

The software has both characteristics according to its usage: a "product" and a "vehicle to deliver and distribute" the product. It is constructed step-by-step with engineering methodology, but could not be factory-made as a conventional mean. It depreciates but doesn't vanish [8] [9]. Although the industry has followed the trend of constructing component-based delivery but software are thorough custom productions. Millions of developers work together to produce software applications in various technology domains. Several application domains have been categorized in software product line e.g. operating systems, desktop applications, engineering/computing or scientific software, embedded systems software, middleware, legacy software, machine learning software, artificial intelligence

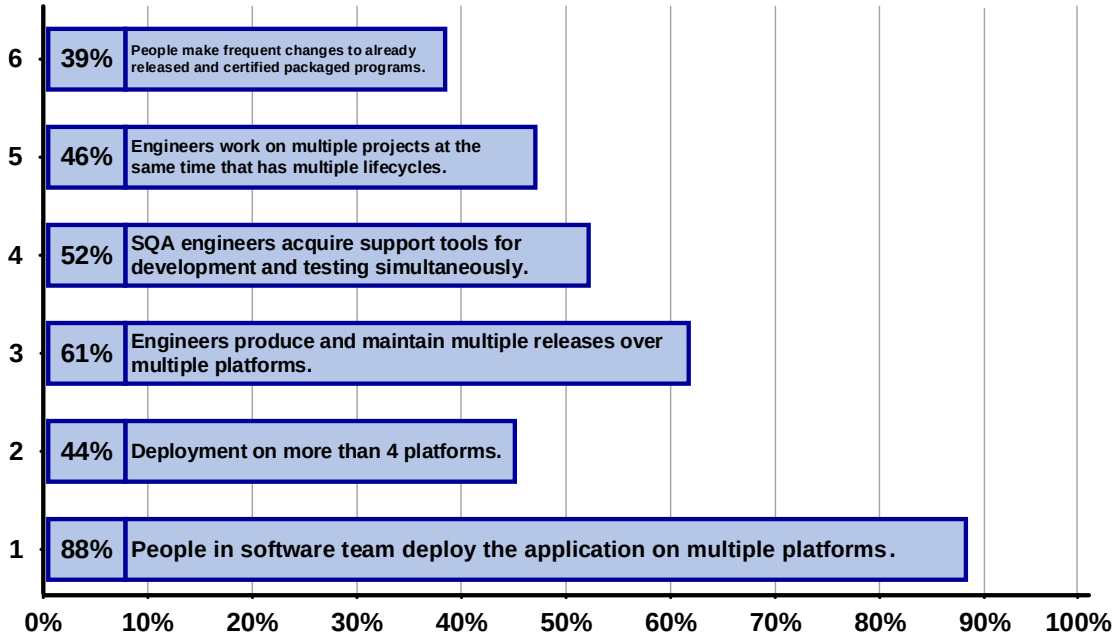

Fig. 1. Bar-Graph describing reasons for confusion due to "Change"

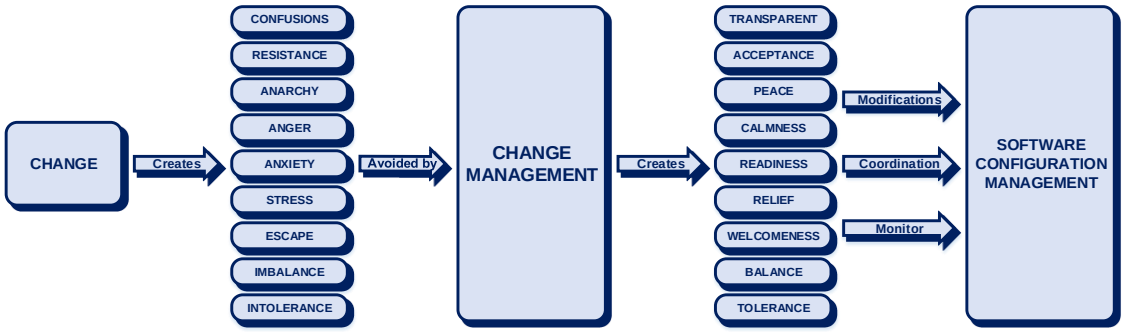

Fig. 2. Confusions avoidance by Change Management and SCM

& expert systems software, product-line and inventory software, gaming & play-store software, and through the recent trend of developing WebApps and MobileApps software [10] [11]. The development ranging from correcting, adapting and enhancing existing applications; but in many others, new systems are proposed to build. It is well said, "Never re-invent the wheel". Hence, software engineers have adopted the environment of Open-world computing, Net-sourcing, Out-sourcing, and Open-source developments to meet new challenges.

It is very difficult to handle old programs i.e. legacy software since full documentation is required to know every whereabouts of them. Aside from above-described categories of software technology, WebApps or MobileApps software has a unique nature other than all. It is network intensive; it requires concurrency and synchronization at the instant grain of time stamp; it bears unpredictable load; it requires high performance for fast access and retrieval; it has availability over 24/7/365 (i.e. 24 hours by a day, or 7 days by a week, or 365 days by a year) which is highly demanded; it is data-sensitive and content driven; it bears continuous evolution; it has current/urgent time for marketing to avoid demand expiry i.e. immediacy; and more overall access security measures [8] [9].

2.3 Software Quality and Trouble of Change

Globalization and open-world computing innovations are viable reasons for increased competition in the software industry [12] [13] [14] [15]. Quality conformance is one of the success factors of a business venture. Software quality is no further become an option. It is an immediate necessity and an un-ignorable user demand [16]. Software users demand high-performance quality by means of speed, access and availability since they heavily invest in purchasing expensive hardware to run their required application. They demand features e.g. quality, reliability, conformance, durability, serviceability, appealing aesthetics, and interoperability. Today software quality remains an issue. Customers blame developers that their chaotic development practices are resulting in low-quality software. Developers blame customers and stakeholders that their demand for illogical delivery dates and a non-stop stream of changes pressurize them to release software before it has been fully tested and validated. Change is one of the most troublesome characteristics of any software project being built. If it is not properly handled, it rises severe confusions, and confusions almost always cause worst quality [8] [9]. Change creates confusions, which are avoided by change management, which are then incorporated in software through software configuration management (SCM) to incorporate hassle-free peacefully project modifications, bring coordination among concerned stakeholders, and monitor the impact of change [17] [18] [19]. Fig.2 shows the confusions avoidance strategy handled by Change Management (CM) and Software Configuration Management (SCM). Software testing and quality assurance certify that acceptable change management process is adopted.

3. Configuration Management Practice

3.1 Concept Frame

Managing project is one of the challenging drives of development from start to finish. Several companion activities like process management, risk management, quality management; run in parallel with this development. Software configuration management (SCM), sometimes referred to as change management or simply configuration management is an umbrella within software project which is executed as a vigilant engineering process to manage changes. SCM provides a control mechanism for the gradual advancement of development of a software system by generating versions, builds, releases and variants of its artefact and their interrelationships [20]. While maintaining, the SCM team is assigned to certify that changes and modifications are incorporated in a controlled manner and records are maintained with fine details of the changes that were requested by customers to be implemented [21] [22]. In other words, the modification and change requests that occur regularly during software development life, and at post development period i.e. after released to customers are properly managed by SCM tool by SCM team which ensure that changes and modifications are incorporated into the software system with orderly control of record maintenance with every detailed stamp of the implemented changes or modifications [8] [9].

3.2 SCM Activities

SCM identifies and records the software traces and system configuration. It controls changes systematically, enforces traceability of the configuration, and maintain integrity thoroughly in SDLC. Components to be controlled via SCM activity comprises: program modules including source code and executable code; system files e.g. data files, build files, execution scripts, and shell scripts; operating system files including compilers, loaders and linkers; system and software specifications containing software architecture specs, requirements specs, design and interface specs, and configuration specs etc.; planning documents e.g. software development plan, configuration management plan, testing plan, quality plan, risk plan etc.; test stuff (test procedures, test scripts, test cases, test data sets, and test results); procedural language descriptions, job control language (JCL); user and system documentation; Third-party integrated tools, debuggers, and other support tools; and development procedures, reports and standards etc. In addition to this; the development environment used to build software is also considered under configuration control [2] [3] [4] [23] [24].

3.3 SCM Focal Areas and their Functions

3.3.1 SCM Six Focal Areas

The SCM process typically consists of six focal areas as mentioned in Fig.3 such as Software Configuration Identification (SCI), Software Configuration Control (SCC), Version Control System (VCS), Software Configuration Building (SCB), Configuration Status Accounting (CSA), and Software Configuration Audits (SCA). All SCIs are deposited to a central repository, where a concrete mechanism of SCM is developed via policies of implementation to ensure data integrity; that provides an integration support platform for other software tools; also provide a method for information sharing among all developers, and implement functions in support of version and change control.

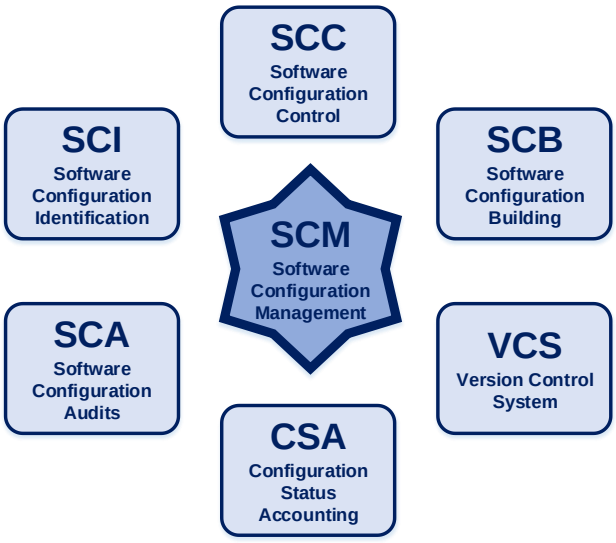

Fig. 3. SCM Focus and Functional Areas

Every artefact or component of software, once developed or reviewed is marked as "Baseline" object. Modifications to a baselined-object become handy in the creation of a new version of that object following pre-defined version nomenclature. The evolution of SCIs over every mark of time become traceable using traceability mechanism of tracker by inspecting the history of revisions and releases which is usually part of the Revision Control System (RCS). The mechanism is well elaborated in Fig.4 [17]. Version control mechanism provides nomenclature with the set of procedures and tools to create release numbers for every variant of SCI.

3.3.2 SCM Functional Areas

Software Configuration Control (SCC) is an integral and procedural activity of SCM that guarantees quality and consistency for any modification to SCI being requested and implemented. The small recognizable modification is usually referred to as "Delta", usually denoted by a small

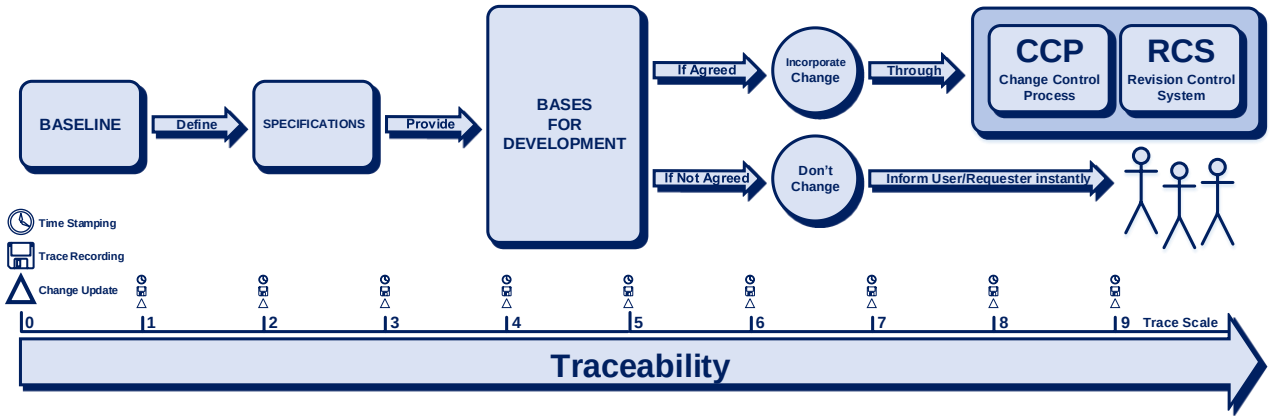

Fig. 4. Traceability from Baseline to Revision Control System (RCS)

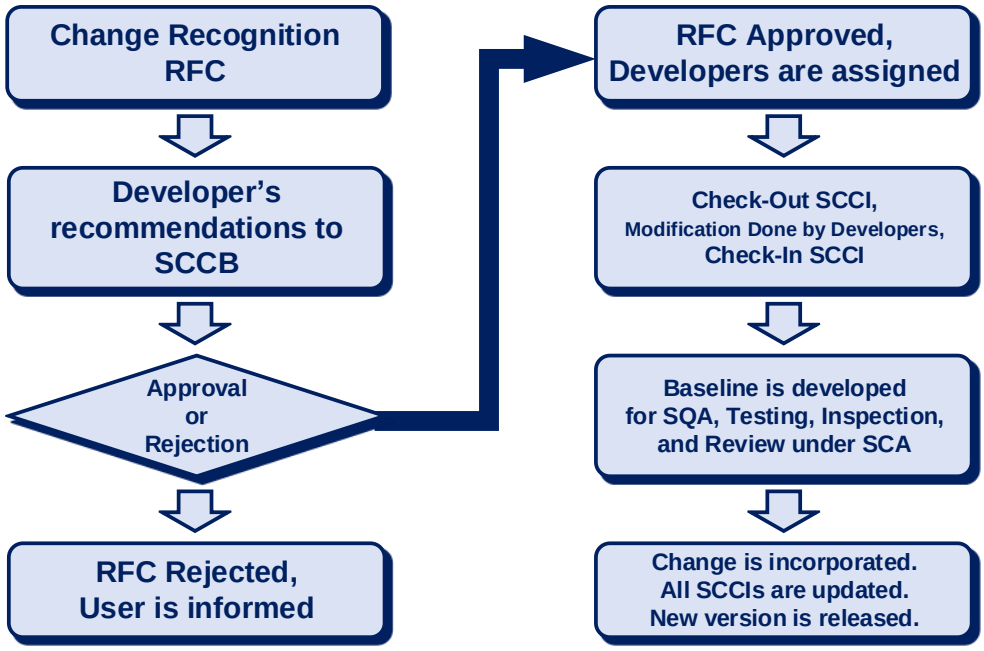

Fig. 5. SCM and Generic Change Control Process - CCP

triangle shape "Δ", where sometimes "Δ_i" with subscript "*i*" used as identifier; "Δ++" for increment or future release; and "Δ--" for decrement or previous release. The SCC process usually begins with a change or modification request forwarded to SCCB (Software Change Control Board) to have a decision to "Approve" or "Reject" the "Request for Change (RFC)", and concludes over a new release with controlled delta update for the SCI submitted to have a modification. The change control process has further elaborated the procedure in the upcoming text. The Software Configuration Audit (SCA) comes under SQA activity that ensures quality which is maintained with conformity. Configuration Status Accounting (CSA) or status reporting share information about each release with the details of the change to those having criticality of impact projection. SCM procedures for WebApps and MobileApps are almost similar as for conventional software except with a need for implementation with special provisions for content management.

3.3.3 Generic Change Control Process

Change is inevitable. And it occurs throughout the life of software development. Once the need for change is recognized, a change request is submitted to SCM by the user in the form of RFC i.e. Request for Change. Fig.5 explains the whole generic change control process (CCP) which is a part of SCC function discussed in previous section. Usually, the developers evaluate the RFCs and generate a report having critical points exploring the positive, negative, and sensitive impacts. This report is presented to SCCB (Software Change Control Board) for the decision of "Approval" or "Rejection". If "Rejected", the user is informed with the reason of denial. If the RFC is "Approved", then a formal procedure is followed. The RFC is queued for modification as suggested by SCCB. A change "Order" is generated. The developers are assigned and they perform modification and submit for Review, Audit, and Inspection; where a "Baseline" for testing is developed by SQA team where testing activities for quality assurance are performed, and changes are promoted for inclusion in next "Release", "Version", or "Revision". SCB re-build appropriate version, and SCA audits the change to all SCCIs. The updates are included in new versions and are released for distribution [8] [9].

4. Notions about Trace Dependency and Traceability

4.1 Traceability or Trace Dependency

In general, "Traceability" is concerned with finding and following footprints (i.e. from, and to whereabouts) of any entity with its size, dimension, sketch, and all maximum relevant information that could be gathered to know the "from", "to", and "current" or "present" stay of any object approaching with forward and backward chaining method; whereas "Trace Dependency" is concerned with determining the meta information e.g. size, dimension, mass, density etc. of determined traces upon which that trace is dependent [25]. Technically, "Traceability" provide linkage between project events i.e. a roadmap of events which help to determine the

best way to proceed. Thus traceability provides better control over changes that occur at any time, wherever in software life for any reason causing confusions [17]. The development artefacts i.e. small pieces, components or information units produced during engineering of software such as specifications, design elements, model descriptions, and source code; are critically interconnected and interrelated as change or modification when requested and later incorporated in any piece, severely affect other elements of a software system with some variation leaving impact. To keep a record of such an association link, the detailed description is indeed needed to be captured. "Traceability" or "Trace dependency" describe such relationships abstractly. This reveals that during the project life cycle, the development artefact are the living entities that require careful maintenance [26]. Fig.6 shows "Traceable" footprints in computers, programs and their inter-associations with users.

According to [27] [28], Traceability defines a relationship and bonding maintained between two or multiple nodes of artefact having successor-predecessor or master-slave channel, which provide a match to confirm that requirements are fulfilled to a certain degree or extent in the software development process. (Extracted from IEEE Std 610.12–1990).

It is usually emphasized to practice establishing trace links among every artefact, but as a matter of fact, exhaustive traceability is not only costly but error-prone as well, while generating highly detailed traceability links among each artefact throughout development and production of software. Furthermore, it put engineers under pressure of burden to maintain and establish trace links rigorously and exhaustively. A lot of tools are developed and are available to integrate traceability infrastructure with the existing development environment for generating and maintaining trace links automatically, but having these tools, the engineers are further burdened to identify semantic traces and ensure the temporal validity of each link. Traceability mechanism is therefore barely or less adopted in software development firms, generally because of complexity and high costs [29].

Fig. 6. Traceable footprints in computers, programs and inter association with users

4.2 Software Configuration Management and Traceability Support

Software configuration management (SCM) with Traceability support are distinct, prominent, and intermingled practices which are principally in-need to support quality assurance throughout software development life cycle (SDLC). Where SCM provide help to manage the evolution, release, and distribution with a proper record of versioning of software artefacts, its documentation, and product line. However, traceability support provides help to manage the knowledge of each and every whereabouts i.e. history of the transit of artefact (from and to), usage place, and record repository information ensuring correctness and consistency developing the process flow of change management. Although the overall method to conduct the SCM process and traceability analysis are inter-related, their implementation is complex and lengthy [30] [31] [32] [33] [34].

5. Need for SCM and Traceability

5.1 The Goal of SCM and Traceability

The main goal of SCM is controlling changes and carefully maintaining traceability, especially in development of high assurance and critical computing i.e. complex and large systems. It avoids anarchy triggered by numerous corrections, replacements, modifications, revisions, and additions incorporated into any of the enormous software projects, servicing its lifetime. Another goal of SCM is to care a "complex software system" to stay in a well-defined state with accurate specifications and verified quality attributes at all times with guaranteed traceable and systematic development process [35]. The main and viable necessity of SCM for a large software project is simply establishing a way to keep track of distinct items that compose the system [31].

The utter necessity of SCM is to design a framework that smoothly serves and organize the lifecycle work-tasks as a "process" to produce quality oriented systems which is quite difficult [2] [3] [4]. Manual, automated, or semi-automated mechanism for generation and validation of trace dependencies are indeed needed throughout the software development lifecycle.

5.2 Benefits of SCM and Traceability

It is usually a general thought that SCM is an activity for simple versioning or control of change. Practically, it is spread beyond the horizon providing service and maintenance to establish quality. Traceability provides a method to process change requested and manage it from project inception to product release ensuring quality performance. SCM is a blessing in disguise for all engineers

Fig. 7. Benefits of SCM and Traceability

developing the quality software. Also, it well serves supplier and customer. Further, it prevents lawlessness and technical anarchy. It avoids shames and embarrassments and strengthens customer satisfaction while eliminates customer dissatisfaction. Traceability mechanism maintains the transparency and consistency between product and relative meta-knowledge about the product [36]. There are many benefits to be gained by an organization that practices SCM. It provides visibility and transparency to the status of the evolving software product. Software developers, testers, project managers, quality assurance personnel, and the customer may benefit from SCM.

There is a large range of some other benefits as well. It reduces maintenance cost by following the notion of "prevention is better than cure". It limits legal liability and records all data and meta-data including meeting minutes, memos, decisions, changes to the technical artefact, and providing a complete paper trail. It allows and provides medium to trace the source of every data and happenings. It organizes activity frameworks that reflect the integrity of the product and the team. It maintains track of changes in artefact over single platform but over pair-programming paradigm or distributed computing environment. It maintains the traceability process from beginning until the end of the project. It ensures that correct "Baseline" is referenced for change with "Delta Δ" increment (Δ+) or decrement (Δ-) to produce new version or variant of software. It ensures compatible, correct, and interpretable configurations of software. It helps to manage software assets. It maintains the conformance of requirements of the customer and provides satisfiability. It provides a stable and reliable development environment integrity to developers that reduce the burden of mistakes and embarrassments. It improves the methodology of the system in a quite disciplined way. It provides a metric approach with meaningful measures of the artefact, code etc. It suggests and follows compliance with standards defined. It provides data to compile reports with various views. It updates every status instantly with the people who want to know. It provides a quick and comfortable environment for auditing. It provides a way to restore versions at certain restore points i.e. baseline ± delta (increment/decrement) for recovery. It provides a communication medium for all concern stakeholders, developers, testers, project managers, and others who want to know about updates of changes. It provides the best assistance in producing error-free quality software. It creates a decision point of when to release the product mentioning the appropriate estimate after all changes to get incorporated [2] [3] [4]. Fig.7 shows some of the benefits of SCM and Traceability.

Besides the above-mentioned benefits, there exists one more thought among large customers. The word "Traceability" is used widely across many consumer industries within Supply Chain Management where visibility of process and presence of a feature of tracking changes accurately is vital. In the software development medium, "traceability" mainly referred to the traceability of requirements throughout application development, till the time ensuring that the distributed software satisfies all requirements, and therefore provide a way to prevent failures [37]. Software product always contains bugs. These bugs need to be fixed ultimately. Traceability prevents the production of failures and hence helps to reduce bugs and delivers the defect-free software. Requirements Traceability (RT) is critically significant assuring successful product development if followed properly. RT is beneficial as it provides a framework to the development team for implementation with the specification that clearly describes the true expectation of SQA team. It plays important role in verification and validation (V&V) process as RT enhances the team motivation by providing the feeling that every member of the team contributing their right effort, doing things right in development (Verification). This maintains quality and it affects customer satisfaction that they got right product as they were expecting before (Validation).

Traceability under SDLC framework ensures the monitoring and control that all requirements are properly captured and frozen. All specifications are moved forward to development and testing benches using V-Process. The review and inspection mechanism become easy for developers and testers. Traceability provides a way to review source-code as per in accordance with acceptance criteria. The coordination, communication, cooperation, and collaboration among members of the team became easy and comfortable. Changes and modifications that occur at any instance of time

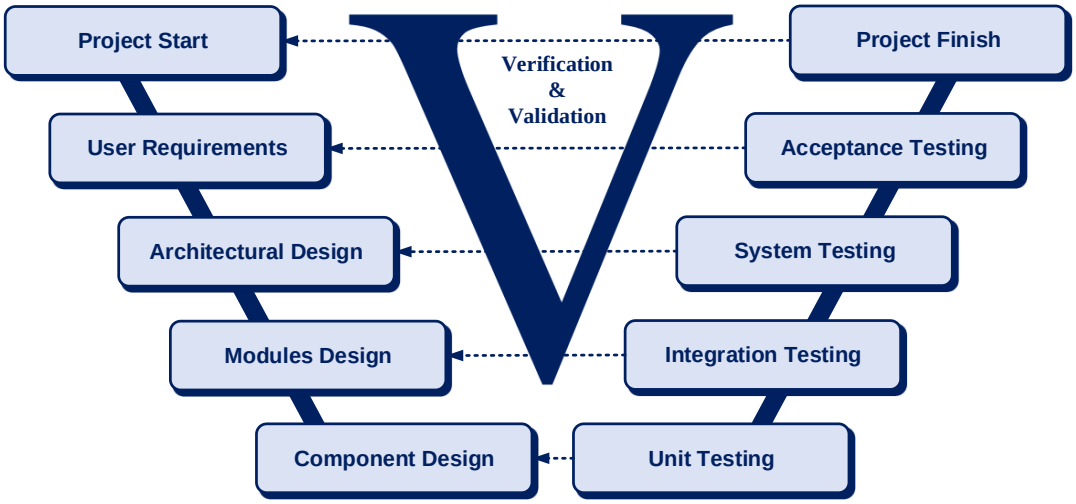

Fig. 8. V-Process Model for Verification & Validation of various project stages

during SDLC are traced. Software becomes testable and ready to be deployed on schedule [37]. Fig.8 shows the V-Process model for verification and validation of various project stages. The quality of project is assured from sign-on till sign-off, i.e. from birth of project till death of project i.e. conception till launch or release. The project finish is validated after assured verification of audit trail generated for every module or stage from project start using trace record. Acceptance testing validates that all user requirements are fulfilled after verification through inspection and audit using trace-tables. System testing validates that all approved architectural designs are implemented after verification through inspection, audit, and review of design specs. Integration testing validates that all modules are properly linked after verification of flow-oriented design specs through inspection and audit. Unit testing validates that all components are properly built as per compliance after verification of component design specs through inspection and audit of design sheets. Thus. "Traceability" as an umbrella of audit, inspection, and review provides evolution of highly assured and reliable systems, especially safety critical systems.

The businesses are also benefitted through traceability implementation. All goals are traced deploying integrity and completeness of requirements without any lapse. SCM with traceability ensures easier impact analysis and reduces uncertainty with improved risk management. It avoids quality and acceptance problems and enables maintenance and product improvement. The product becomes fit for all business needs. Traceability saves time, curtail the considerable cost of debugging, and require reduction of efforts i.e. reducing wastage of time to be spent for over-coming failures, and controls cost deviations. It provides compliance with all industry business standards e.g. CMM, CMM-I, ISO, Six-Sigma, SPICE, SCAMPI etc. [37].

6. Conclusion

Change is actually a disbalance or a source of disbalance appeared in normal situation which require the system to be balanced with rythm. There is a scientific concept of "Entropy" i.e. coming to an equilibrium state from an imbalance state. The abrupt and unexpected change or modification in any medium creates anarchy, anger, anxiety, stress and high tendency to quit or escape from the situation, which is a normal psychological reflective in-tolerant behaviour. In software production setup, following the SCM i.e. software configuration management practices, provide the only and highly organized solution to cope up with the change creating change tolerance throughout the life-cycle if properly understood and managed. Traceability is one of the necessity of SCM process that enable to move frequently throughout in any past or future state as per operating environment following the trace records, time stamping, and change updates in the form of versions and variants. This paper has provided a detailed concept of SCM focus and functions with strategic "Traceability" support for the production of high quality and reliable software suite with having durable usability life. It is highly advisable for developers to follow best practices and standard procedures of software production while learning and utilizing the power of configuration management methodology with formal method model of software traceability. It will surely reduce the time and effort of SQA team over the execution of corrective action to produce the error free software. The reliable and highly assured critical computing systems will therefore reduce the human life hazards caused due to software malfunction resulting high risks.

References

[1] A. Cooper, *Change Management: Quality and Quantity*, Computer Associates, 2003.

[2] J. Keyes, Software Configuration Management, Florida, USA: AUERBACH Publications, CRC Press LLC., 2004.

[3] T. Ursula and B. Lea, "State of the Art of Open Innovation and Design for Sustainability," in *Renewable Energy Policy Efficacy and Sustainability: The role of equity in improving energy policy outcomes*, Singapore, Springer, 2017, pp. 705-717.

[4] F. Schwägerl, *Version Control and Product Lines in Model-Driven Software Engineering*, Germany: Doctoral dissertation, Universität Bayreuth, 2018.

[5] D. Burdick, "Celestica Transforms Competitiveness With C-Commerce," Gartner, Inc., 2000.

[6] C. Debabroto, G. Rajdeep and V. Sambamurthy, "Shaping up for E-Commerce: Institutional Enablers of the Organizational Assimilation of Web Technologies," *MIS Quarterly*, vol. 26, no. 2, pp. 65-89, 2002.

[7] T. L. Friedman, The World is Flat: A Brief History of the Twenty-First Century, Macmillan, 2005.

[8] R. S. Pressman, Software Engineering: A Practitioner's Approach, 7/e, New York: The McGraw-Hill Companies, Inc., 2010.

[9] R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach, 8/e, New York: McGraw-Hill Inc., 2015.

[10] S. Hussain, "Mobile Agents: An Intelligent Multi-Agent System for Mobile Phones," *International Organization for Scientific Research - Journal of Computer Engineering (IOSR-JCE)*, vol. 6, no. 2, pp. 26-34, 2012.

[11] R. Atiya Masood and S. Hussain, "Novel Integrated Sensor based Sleep Apnea Monitoring and Tracking System using Soft Computing and Persuasive Technology for Healthcare Support," in *9th International Conference on Innovative Trends in Management, Information, Technologies, Computing and Engineering (ITMITCE – 2014)*, Istanbul, Turkey, 2014.

[12] K. J. Stewart and S. Gosain, "The Impact of Ideology on Effectiveness in Open Source Software Development Teams," *MIS Quarterly*, vol. 30, no. 2, pp. 291-314, 2006.

[13] B. Fitzgerald, "The Transformation of Open Source Software," *MIS Quarterly*, vol. 30, no. 3, pp. 587-598, 2006.

[14] N. Levina and J. Ross, "From the Vendor's Perspective: Exploring the Value Proposition in Information Technology Outsourcing," *MIS Quarterly*, vol. 27, no. 3, pp. 331-364, 2003.

[15] B. Ives and S. L. Jarvenpaa, "Applications of Global Information Technology: Key Issues for Management," *MIS Quarterly*, vol. 15, no. 1, pp. 33-50, 1991.

[16] S. Hussain, Interviewee, *Software Has Become A Driving Force.* [Interview]. 2004.

[17] S. M. A. Burney and H. Saleem, Software Configuration Management: A Comprehensive Review, Karachi: University of Karachi, 2003, p. 128.

[18] S. Hussain and Z. Faraz Ahmed, "Identification and Realization of Trace Relationships within Requirements," in *International Conference on Software Engineering (ICSE'06)*, Pakistan, 2006.

[19] B. S. M. Aqil, S. Hussain, M. Nadeem and J. Tahseen A., "Traceability Management Framework for Patient Data in Healthcare Environment," in *3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, Chengdu, China, 2010.

[20] W. E. Lewis and G. Veerapillai, Software Testing and Continuous Quality Improvement, 2nd ed., Auerbach Publications, TEAM LinG, 2004.

[21] I. Sommerville, Software Engineering, 9th ed., USA: Pearson Education, Inc., Addison-Wesley, 2011.

[22] I. Sommerville, Software Engineering, 10th ed., USA: Pearson Education, Inc., Addison-Wesley, 2016.

[23] T. Kesse, "Software Configuration Management for Project Leaders," in *Proceedings of Software Technology Conference*, 1997.

[24] T. Kesse and M. Patricia A., "Software Configuration Management for Project Leaders," *Software Quality Professional*, vol. 2, pp. 8-19, 2000.

[25] S. Hussain, "Towards Identification and Recognition of Trace Associations in Software Requirements Traceability," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5(2), pp. 257-263, 2012.

[26] A. Egyed, "A Scenario-Driven Approach to Trace Dependency Analysis," *IEEE Transactions on Software Engineering*, vol. 29, no. 2, pp. 116-132, 2003.

[27] IEEE, "IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990)," Institute of Electrical and Electronics Engineering, Inc., 1990.

[28] P. Parviainen, H. Hulkko, J. Kääriäinen, J. Takalo and M. Tihinen, "Requirements Engineering - Inventory of Technologies," VTT Technical Research Centre of Finland, 2003.

[29] A. Egyed, S. Biffl, M. Heindl and P. Grünbacher, "A Value-Based Approach for Understanding Cost-Benefit Trade-Offs During Automated Software Traceability," in *TEFSE: 3rd International Workshop on Traceability in Emerging Forms of Software Engineering*, Long Beach, California, USA, 2005.

[30] K. Mohan, X. Peng, C. Lan and R. Balasubramaniam, "Improving Change Management in Software Development: Integrating Traceability and Software Configuration Management," *Elsevier: Decision Support Systems*, vol. 45, no. 4, pp. 922-936, 2008.

[31] R. Conradi and B. Westfechtel, "Version Models for Software Configuration Management," *ACM Computing Surveys*, vol. 30, no. 2, 1998.

[32] J. Estublier, D. Leblang, A. v. d. Hoek, R. Conradi, G. Clemm, W. Tichy and D. Wiborg-Weber, "Impact of Software Engineering Research on the Practice of Software Configuration Management," *ACM Transactions on Software Engineering and Methodology*, vol. 14, no. 4, 2005.

[33] C. Gunter, "Abstracting Dependencies between Software Configuration Items," *ACM Transactions on Software Engineering and Methodology*, vol. 9, no. 1, 2000.

[34] B. Aqil, M. Nadeem, J. Tahseen and S. Hussain, "Conceptual Fuzzy Temporal Relational Model (FTRM) for Patient Data," *WSEAS Transactions on Information Science and Applications (Journal)*, vol. 7, no. 5, pp. 725-734, 2010.

[35] W. F. Tichy, Software Configuration Management; Encyclopedia of Computer Science, 4th ed., vol. 4th, A. Ralston, E. D. Reilly and D. Hemmendinger, Eds., Chichester: John Wiley and Sons Ltd., 2003, pp. 1601-1604.

[36] A. E. Lager, "The Evolution of Configuration Management Standards," *Logistics Spectrum*, 2002.

[37] I. Software, "The Benefits of Traceability Within the Application Development Lifecycle," Intland Software Inc. (Offices: Germany, Silicon Valley, Korea), 2017. [Online]. Available: https://intland.com/blog/alm/the-benefits-of-traceability-within-the-application-development-lifecycle/. [Accessed 2017].

Hussain Saleem is presently Assistant Professor at Department of Computer Science, University of Karachi, Pakistan serving since 2002 and was Faculty Member at Sir Syed University of Engineering & Technology, Karachi during 2001-2002. He has completed Ph.D. (Computer Science), MCS (Computer Science), B.S. (Electronics Engineering), and PGD (Statistics). He bears vast experience of more than 21 years of University Teaching, Administration and Research in various dimensions of Computer Science at University of Karachi. He has also professionally worked at Science Labs (Physics & Chemistry) at Aga Khan Ex. Students Association Karachi since 1992-2005, served as Bio-Medical Engineer at Aga Khan University in 1999-2000, remained Project Manager (IT) at Unilever Pakistan in 2000-2001, and Electronic Design Engineer at Pakistan Steel Mills in 1998. Hussain has authored and co-authored more than 40 International Journal publications. He is serving as Scientific & Technical Reviewer of various International Journals and Conferences globally and among top Reviewers from his parent University at Publons (Web of Science/Clarivate Analytics) where reviewed more than 300 Research Papers since 2013. His field of interest is Software Science, System Automation, Hardware Design & Engineering, Data Analysis, Simulation & Modelling, Bio-Medical Science and Healthcare. Hussain is a good speaker and has been invited as a guest in TV programs and disseminated knowledge over the latest technology trends that got high appreciation among young generation. He is a senior and Life member of PEC (Pakistan Engineering Council), IACSIT, URENG, URST, EACEEE, IEEE, ACM and SDIWC.

Dr. S. M. Aqil Burney is presently Professor & HOD leading the Department of Actuarial Science & Risk Management, at College of Computer Science & Information Systems, Institute of Business Management (IoBM), Karachi since 2013. He hold Ph.D. (Mathematics) from Strathclyde University, Glasgow-UK, along with M.Phil., M.Sc. and B.Sc. in Statistics from University of Karachi. Before joining IoBM, Dr. Burney was Meritorious Professor & Chairman at Department of Computer Science, University of Karachi. He bears extensive experience of Academics Management, remained Provost, Registrar of University of Karachi, and Project Director for Development of Department of Computer Science, Institute of Information Technology, and Founding Director of Main Communication Network, at University of Karachi. He has taught for more than 45 years at University of Karachi, and other Degree awarding institutions of Pakistan and abroad. He has published more than 150 research papers, and 07 Books in the field of Computer Science, ICT, Mathematics, and Statistics nationally and internationally. He has supervised more than 15 Ph.D. and 10 M.S./M.Phil. Scholars in Mathematics, Computer Science, and Statistics as an approved HEC Supervisor. Dr. Burney is Chairman (elected) National ICT Committee for Standards PSQCA - Ministry of Science & Technology, Govt. of Pakistan, and member National Computing Education Accreditation Council (NCEAC), Member IEEE (USA), Member ACM (USA), and Former Fellow of Royal Statistical Society (UK) for 30 years. His fields of interests are Algorithmic Analysis & Design of Multivariate Time Series, Stochastic Simulation & Modeling, Software Engineering, Computer Science, Soft Computing, Risk Theory & Insurance, e-Health Management, Data Sciences, Fuzzy and other logic systems. He is the Editor and Member Editorial Board for Pakistan Journal of Engineering & Technology, International Journal of Information Systems, University of Sindh Journal of Information & Communication Technology, and Reviewer of Journal of Neural Computing & Applications.