AGILE SOFTWARE DEVELOPMENT: EVOLUTION OF CASE TOOL IN SOFTWARE ENGINEERING

Prof. Sudhindra K. Madi

Department of ISE, Gogte Institute of Technology, Belgaum, India

ABSTRACT: The software has grown up in diverse fields as a giant for collaboration of all activities in a miraculous way. We can visualize a whole world in finger tips is not just an assumption, but amazing fact. A glance at different phases of development till date is contextual to realize the truth. Way back in 1940's, extending the hardware functionalities using an assembly language itself was a fashion of devising new software. A step ahead, in the year1964 IBM computers designed a first of its kind software for its 360 series systems having installation facility and external to the system, which was devised by combining scientific and business domain features of that era. Obviously this was believed to be an inspiration for further inventions and development in the field. Subsequently new tools and techniques came out with flying colors making 'Software' as a distinguished industry of present day.

KEYWORDS: Software Engineering, NATO, ICSE, Software Architecture, Architectural Styles, SDLC, Knowledge Management, Computer Aided Software Engineering CASE, Agile Software Development Models.

INTRODUCTION

The Software became popular among commercial domain because of its suitability and effective use. Slowly the creative individuals of mutual interest with sound knowledge communicated and formed committees or Study Groups to foresee scope of software in different fields of engineering. The conferences served the purpose for sharing their ideas or thoughts. In the year 1967, one of such Committee with members of different countries initiated in assessing the field of computer science, discussed, planned promising scientific projects. Even though having ideal thoughts, there was a lack of unifying theme which was vital for the Study Group. Dr. Friedrich Ludwig Bauer (a German computer scientist and professor emeritus at the Technical University of Munich) member of committee said, "The whole trouble comes from the software for which there is no clean fabrication process defined yet". It was found true by other members also, otherwise the question for everyone was '*software* is unique for variety of hardware products of civil, mechanical, electrical and computer engineering domains, but which engineering domain it belongs to?'

The first conference[3] sponsored by 'North Atlantic Treaty Organization (NATO)' Science Committee Garmisch, Germany, 7th to 11th October 1968, the Chairman: Professor Dr. F. L. Bauer and Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms. In that conference at a resort hotel in Garmisch, Germany scientists, engineers and mathematicians gathered to talk about the relation of software and hardware in a computer system, software production design, implementation or distribution & services. As a chairman of the said conference along with the participants discussed software crisis and associated problems that they had witnessed in the preparation of large military systems. Management and communications related to software size, performance and cost improvements were other major issues discussed. Out of 50 or more participants' significant contribution were from A notable computer scientists, C.A.R. Hoare England and Edsger Dijkstra Netherlands who took the first steps to develop theories of large programming. The second Conference[4] was sponsored by the NATO Science Committee, Rome, Italy, 27-31 Oct. 1969, discussed specification and implementation languages, coding, testing and simulations of large programs for quality, performance and accuracy suitable to the end user requirements were highlighted.

Another mile stone in the history was 'Institute of Electronics and Electrical Engineers' in 1975 started its journal "IEEE Transactions on Software Engineering". By September 11-12 of same year first conference[5] named as 'National Conference on Software Engineering' sponsored by National Bureau of Standards and IEEE Computer Society held at Washington DC, SA. Right from 2nd conference conducted every year with the name 'International Conference on

Software Engineering (ICSE)' sponsored jointly by Association of Computing Machinery (ACM) and IEEE in different countries like USA, Japan, Singapore, Italy, Germany, India & China. The 36th ICSE was conducted here at Hyderabad, India in the year 2014.

ACCUMULATING THOUGHTS FOR ENGINEERING A SOFTWARE

Based on the discussions and guidelines put by ICSE creative authors scientists published articles, books on software engineering. Among them was "Software Engineering Economics" by Barry W Boehm In 1981, where many managerial issues including the time and cost estimation techniques of software development were analyzed. Software trends[20] in that context were multi-user systems, databases, and structured programming. Due to its feasibility of operations, adaptability in using the software geared up to a complex process[9]. The most significant twist in software development[7] by 1990's was the internet and distributed systems to get an implementation matched to the specification. Tools and methods addressing major issues provided solutions to software development, which formulated models in software engineering too.

Using Formal languages[6] to automate specification and analysis process, whereby various specification defects can be identified and eliminated was a only possibility. But formal specification languages were limited in 1992 were immature. A small example is, by taking safety-critical systems into account, Department of Defense (DoD) cross checked syntax and semantics accuracy, with or without compiler implementation, in systems using ADA and various languages. ADA was successful due to its strict compilation strategies. Hence later Powerful functional specification languages like Z, RSML, State-charts, and PVS developed. They can be engineered and applied widely on practical projects.

Beyond those days arrived domain specific and model based special-purpose languages[10], the combination of formal languages and application generators. They were having more powerful and useful features for the simulation or direct execution with existing data-set to see the results graphically on the spot. Familiar examples of such type are Simulink, SCADE for aerospace software engineering, network simulators-NS. Later steps sought quality of developed software using assessment standards like DO-178B[12], IEEE STD-1228-1994, also static analysis, inspection and reviews made the software matching industry features.

Majority of aforesaid developments were the basis of software engineering field to come out with refining theme and relevant, updated definition "An engineering discipline that is concerned with all aspects of software production for medical, industrial, military, communications, aerospace, business, scientific, and general computing applications by the establishment and use of sound engineering principles in order to economically obtain the software that is having trust worthy performance or having a good efficiency at concerned tasks in real time implementations". Accordingly the major properties or attributes of software engineering can be short listed as follows,

- Requirements engineering: The documentation, analysis, specification, and testing methods requirements.
- Embedded software: computer software for control machines or devices typically specialized for the particular hardware having time and memory constraints.
- Software evolution: The process of developing software initially, then repeatedly updating it for various reasons.
- Site Reliability Engineering: Management of large scale distributed systems used in an operations environment.
- Software design: The process of defining the architecture, components, interfaces, and other attributes.
- Software construction: Creation of software in combination with coding, verification, unit testing, integration testing, and debugging.
- Software engineering process: The definition, implementation, assessment, measurement, management, change, and improvement of the software life cycle process itself.
- Software engineering tools and methods: The software tools that assist the quality software development life cycle processes (CASE-Computer-aided software engineering) and the methods with a goal to make all activities systematic and successful.
- Software engineering management: The application of management activities—planning, coordinating, measuring, monitoring, controlling, and reporting—to ensure that the development and maintenance of software is systematic, disciplined, and quantified.
- Software testing: An empirical, technical investigation information about the quality of the product or service to stakeholders.
- Software configuration management: Identification of distinct points in time of system for the purpose of systematic control over changes to maintain integrity and traceability throughout the system life cycle.
- Software maintenance: The cost-effective support to software activities in .total.
- Software durability: The solution ability of serviceability of software and to meet user's needs for a relatively long time.

- Software trustworthiness: a guarantee or complete assurance that it will execute its required functions under all probable situations without fail.
- Software dependability: The solution ability of serviceability of software that can defensibly be trusted and justified.
- Software usability: Usability is a degree of software to perform a specific task as prescribed.

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

The software development process right from requirement collections phase to maintenance and its service upgrades is a Development Life Cycle [18][19], where it keeps applications abreast of the technology in the interest of end user. The Fig.1 shows different phases of software development. Even at present as per industry quality norms software developers having best experience and practice biased on trends in the field, follow this criteria. Aforesaid phases of software life-cycle apply to every new project in developing, extending or replacing the services of the existing software system.

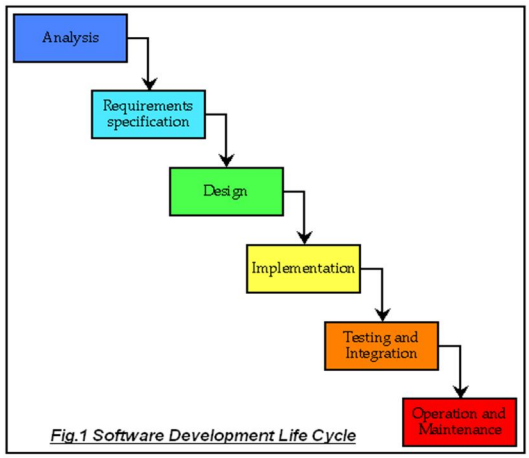

Fig. 1. Software Development Life Cycle

In present technology trends[13], it has become a practice to foresee and check an alternative to cope up with frequent changes in technology, which is one of the software life-cycle principles. This is a part of business promotional activity also, helping developers to earn profits apart from making their products popular in the industry. This has lead Indian IT Industries growth rate[15][16] to at least (3) fold in last decade. Fig.2a and 2b depict the fact.

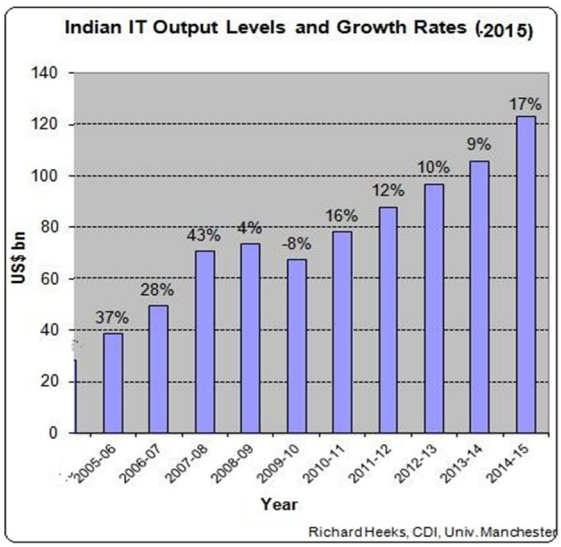

Fig.2a. Indian IT Industry Growth Rate

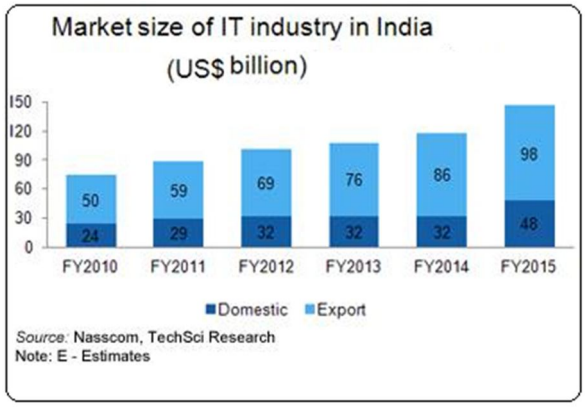

Fig.2b. Indian IT Market size

SOFTWARE ARCHITECTURE AS A TEMPLATE FOR SOFTWARE DESIGN

Software architectures originated to serve guidelines for software design process[17]. Alternatively we can assume it as the plan or blue print of software project [11] which includes requirements engineering. The Software to be designed needs requirement to be specified (SRS document) describing how the system works[18]. Resultant document is Software design document SDD.

Once verified and validated design, performing risk analysis and relative measures gets into detailed design. Then questions arising are, which is the model? What are parameters? where to start? Here 'Software Architecture 'comes into picture as it has huge collection of styles to approach and implementation patterns [12]to follow, giving solution to software development right from its initial phase, A prototype of a desired system is created using specific pattern. This is known as system modeling, a significant part of software development. As referred earlier different architectural styles[21] available are, domain specific Independent process, data flow, data centered/repository, call & return, virtual machines.

SOFTWARE DEVELOPMENT MODELS

Software categorized on the basis of system functionalities, system development and area of use like system software, Programming languages (compilers) and Application software. Keeping operating systems and Programming languages aside, Application software further classified into generic and customized application software with examples as follows,

a. Generic Software products- Internet browsers, word processors, file converters, graphic designers, common office automation products etc., available for purchase and the free wares on open source.
b. Customized software product- CAD, Financial Accounting and alike any industry specific activity maintenance or automation software purchased product having a license with installation keys, terms and conditions of use.

Software developers, studying the stake holder's requirements, set up goals for new project development using best implementation practices, accurate methods or models of software engineering. Fig.3 shows the development history of such methods or models.

In this scenario methodologies mainly categorized into 3 ways.

- Iterative Software Development Process (ISDP),
- Object-Oriented Methodology (OOM)
- Unified Modeling Language (UML-Industry wide accepted object-oriented model at present)

Next step is to select one of the suitable software development model [18][19] enlisted to initiate the process.

- Waterfall & Iterative waterfall model
- Iterative & Small Iterative model
- Spiral model
- Prototyping model

Every model has unique features, attributes and formalities related to implementation. A research on impact of software engineering [11], classifying the methods, models based on suitability of both ends i.e. developer and end-user is vital. A good influencing factor to consider was 'Knowledge Management & Engineering' discussed in next section. Here is a glance at individual model for its advantage and disadvantages with brief note.

Waterfall model: The advantages are, conventional, linear and sequential in operation. Project divided into sequential phases, some overlap and splash back acceptable between phases. Emphasis on planning, time schedules, target dates, budgets and implementation of an entire system at one time. Tight control over the life of the project via extensive written documentation maintained. Every phase reviews and approval/signoff by the user and information technology management before beginning of the next phase. Each stage has well defined deliverable simple to use and understand mechanism. The biggest disadvantage is. you cannot go back even a single step for alteration ,modification else everything goes worse if the design phase has gone wrong. This is its greatest advantage in implementing system in systematic sequential phase.

Iterative waterfall model: The advantages are, much better model for the software process, giving feedback to proceeding stages. It can be used for projects, where requirements not understood well in advance. Disadvantages are not easy to manage, no clear milestones in the development process. No stage is really finished.

Spiral model: The advantages are focus on risk assessment and minimizing project risk by breaking a project into smaller segments and providing more ease-of-change. Each cycle involves a progression through the same sequence of steps. Each trip around the spiral traverses four basic quadrants: (1) determine objectives, alternatives, and constraints of the iteration (2) evaluate alternatives, Identify and resolve risks (3) develop and verify deliverables from the iteration, finally (4) plan the next iteration. Disadvantages are requirement risk identification, its projection, risk assessment and risk management, which is not an easy task. Cost and time estimations are not easy and lower end. Not suitable for smaller projects

Prototyping Model: The advantages are, early view of the prototype to users gives an idea of what the final system looks like. This encourages active participation of users and producer. Cost effective (Development costs reduced). Increases system development speed assists to identify any problems with the efficacy of earlier design, requirements analysis and coding activities. Disadvantages are project management difficulties causing systems to be left unfinished, premature implementation, inadequacy for overall organization needs. Due to lack of flexibility not suitable for large applications.

Task	Period	Particulars	Method / Model
Mastering machine	(1956–1967)	Hardware dependent high level languages Online. Code and fix	Batch Sequential Interactive
Mastering process	(1968–1982)	Crisis. Development process software engineering Ensure correctness. Models inapplicability in big problems	Process Formal
Mastering complexity	(1983–1992)	Personal computer. Expanding data and functional convergence Reusing new programming approaach	Structured Object oriented
Mastering communications	(1993–2001)	Internet. Client/server complex projects Integrated methods quality	Industrial Distributed
Mastering productivity	(2002–2010)	Conceptual level expansion Customer productivity Customer involvement	Abstraction Agile
Mastering market	(2011.– 2015..)	Outsourcing services Orchestrating services Market demands. Downloads	Service Mobility

Fig.3. History Software Engineering Methods/Models

KNOWLEDGE MANAGEMENT BEGETS RESEARCH

In project management very important topic is - how to manage the acquired knowledge? Few software developers have mostly failed here due to frequent changing, challenging environment as acquired knowledge gets obsolete very fast. Knowledge Based Engineering (KBE)[25] in software engineering researches seek better and fast ways to develop and evaluate software based on both short-term and long-term benefits apart from the quality, cost, and competitive features of the software products.

Whatever may be the model chosen based on strategy, the knowledge [26] used behind devising the system is cumulative for enhancement and development. This instigated developers to find an alternative to the 'Traditional method'. In fact this was root cause for the development of 'Agile method'. Hence, alternative names to both methods are 'Predictive' and 'Adaptive' accordingly. A predictive team can report exactly what features and tasks are planned for the entire length of the process development depends on advance requirement collection. But, in case of adaptive team features to consider are interim software changing, testing, new requirements collection and all possible factors giving quick result.

Here researchers faced the questions about best methods of software development, analyzing, project design, evaluation and generalizing product class. Solution to the majority of issues was 'Agile Method' or 'Agile software development method'. This is a set of principles for software development in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement and encourages rapid and flexible response to change. 'Agile Method' doesn't contain any specific methods and is unique.

Traditional method focus on analyzing and requirement collection and result delivery, where documents are vital. This is not the case with adaptive type as any of the customer's representatives witnessed the process, knows the operation and queries if any, that's all. Otherwise empirical documentations enough for overall software development. For example one of the differences between agile method and waterfall model is the approach to quality and testing. In the waterfall model, there is always a separate testing phase after a build phase; however, in agile development the testing is done

concurrently. Hence to devise dynamic, light weight software 'Agile Method', having object oriented, parallel or concurrent processing, work dividing, keep watching, monitoring ad hoc code attribute altering properties is used.
In this scenario, to make time consuming traditional software development method as an agile method, use of most popular CASE tools come into picture. In last half decade we have seen many software varieties bundled different purposes, for example web development, information processing, infotainment and complete office automation. Elaborating software development next section describes the importance of such tools. Being fast adopting, dynamically changing, small in operating CASE tools are (can be) definitely categorized as 'Agile Software Development' model or CASE is an agile model for software development.
Popular agile software development methods and/or process frameworks include (but are not limited to):

- Adaptive software development (ASD)
- Agile modeling
- Agile Unified Process (AUP)
- Business analyst designer method (BADM)[39]
- Crystal Clear Methods
- Disciplined agile delivery
- Dynamic systems development method (DSDM)
- Extreme programming (XP)
- Feature-driven development (FDD)
- Lean software development
- Kanban (development)
- Scrum
- Scrumban

COMPUTER AIDED SOFTWARE ENGINEERING

Background

During the 1970s when computer companies were beginning to borrow ideas from the hardware manufacturing process and apply them to software development, Even though it was not assumed to be a professional method, but realized as the computer-assisted method or tool for organizing and controlling the new software development. In the context a software company 'Nastec Corporation' of Southfield, Michigan in 1982, who was first to coin the term 'Computer Aided Software Engineering' CASE, inspired by the CAD (Computer Aided Design tool for machinery equipment plans) devised the software tool integrated graphics and text editing features named as 'GraphiText'. This was first software to use hyperlinks to cross-reference text strings in the documents.
The concept of CASE tools[24] further reached its peak in the early 1990s with major industry giant IBM, by the proposal of alliance with different software vendors, constructed the Software repository of tools for application development. This strategy configured using IBM AD/Cycle [7], DB2 and OS/2 on the mainframe. IBM tie-up few more software vendors for marketing their product. Ultimately this helped to achieve complete life-cycle for certain software products. Within a decade the decline of mainframe and concerned, those CASE tools died off.

CASE tools: categories and Support

CASE tools encompass methodologies together with monitoring all processes of software development in a systematic way Majority of CASE tools right from its origin supported the concepts of structured programming approaches of software engineering[19] 'Object Oriented Methods (OOM)'[20] and 'Visual Programming Tools' like the Object Management Group OMG's Unified Modeling Language (UML)[20] designed to provide a standard way to visualize the design of a system and widely accepted as the industry standard for object-oriented modeling at present.
The underlying idea of CASE technology[8] is to support, combine the tools with the structured development methodologies within SDLC, which are categorized as,

1. Business and Analysis: Graphical modeling tools. E.g., E/R modeling, object modeling, etc.
2. Development: Design and construction phases of the life-cycle. Debugging environments. E.g., GNU Debugger.
3. Verification and validation: Analyze code and specifications for correctness, performance, etc.
4. Configuration management: Control the check-in and check-out of repository objects and files. E.g., SCCS, CMS.
5. Metrics and measurement: Analyze code for complexity, modularity (e.g., no go-to's), performance, etc.
6. Project management: Manage project plans, task assignments, scheduling.
7. Documentation: A tool for generating the project documents.

8. Reverse Engineering: a tool to trace back the problems in any phase of software life cycle for a solution.

For any specific tasks listed above, the CASE[24] can be one among 3 distinguished tool types. Those are Upper CASE or front-end, Lower CASE or back-end and Integrated CASE tools. The functions associated with each type described as follows,
Upper CASE front-end Tools focus on concept-level of products providing support for the early stages in the systems development, supporting business and analysis modeling. They also support traditional diagrammatic languages such as ER diagrams, Data flow diagram DFD, Structure charts, Decision Trees, Decision tables, etc.,
Lower CASE back-end tools provide support for the later stages in the life cycle such as code generation and testing, component integration, maintenance, reverse engineering and relevant supporting development activities. Both Upper and Lower Case tools share equal responsibility in entire software life-cycle.
Integrated CASE tools support both, Upper and Lower CASE tools.

CASE Software : Groupings tools for software life-cycle

CASE Software devised with automated features as elaborated in 'part-b' above, can be classified into 3 major groups [8]. . Those are tools, Workbenches and Environments briefed as follows,

- Tools – Support specific tasks in the software life-cycle. Like business and Analysis modeling. Graphical modeling tools. Example: ER modeling, object modeling, etc.
- Workbenches – Take care of specific part of the software life-cycle by combining two or more tools. A best example of workbench is Microsoft's Visual Basic which is programming environment incorporated with tools like GUI builder, smart code editor, debugger etc,.
- Environments - combine two or more either tools or workbenches and support the complete software life-cycle. A best example is fourth generation language 4GL environments where first environments to provide deep integration of multiple tools focusing specific types of applications like user-interface (UI) driven applications that performed standard transactions with a relational database.(Informix and Focus).

CASE : benefits and few risk factors: Industry accepted major benefits of CASE[27]

Making the product that meets real-world requirements having customer as part of the process development.

1. Due to an organized approach of product development with testing and redesign cost of servicing reduces significantly in its lifetime.
2. Support the development and business community through its Automated Diagram associated with other popular features,
   - Checks for syntactic correctness, Data dictionary support
   - Checks for consistency and completeness, Navigation to linked diagrams
   - Layering approach, Requirements traceability
   - Automatic report generation
   - System simulation
   - Performance analysis
3. Improved quality product results in showcasing developers' image, providing a competitive edge in the software industry
4. Enhance reuse of models or models' components reduce development time..
5. As-a-whole this improves productivity.

Some of the risk factors are,

1. Inadequate standardization. Organizations usually have to accommodate and adopt methodologies and tools as per their specific requirements.
2. Unrealistic expectations. The promoter of CASE technology especially vendors marketing expensive tool-sets giving a hope that solves all of the problems.
3. As with any new technology, CASE requires time to train the people in how to make use of the tool.
4. CASE provides significant new capabilities. Without the proper process guidance cause Inadequate process controls, leading to restriction on tools capabilities.
5. For the changing environments, platform independent CASE tools are in need by the industry.

CONCLUSION

Software industry is volatile. Every now and then challenges arise to face new technology trends. Most of the universe activities are digitized and accessed through internet continuously. Industry needs new software based on the individuals or customers interest with formalities. Traditional software development methods are almost obsolete. For any automating objective new approaches in software development with quick coding and speed implementing methods are dominant in the industry. In this scenario, to achieve perfect implementation for the requirements, a suitable environment with set of tools is vital. The CASE tools suffice this need to a maximum extent. Obviously the most of us concerned about tools' standards, compatibility, dependency, ease of use, efficiency and the cost.

Based on Research and developments in Software Engineering, every new CASE tool enters software industry should have a distinct competitive features, good priorities over one another in selection at affordable price among its main stream. A document or manual specifying customizable features and its compatibility is preferred. Using cheap and partly compatible CASE tools have to be relinquished by the concerned authorities of the industry.

REFERENCES

[1] ACM/IEEE web site;(2016); The history of International Conference on Software Engineering ICSE; 1 ;1
[2] Phillip Sainter; Oldham Keith; Larkin Andrew; (2001); Achieving benefits from knowledge-based engineering (KBE) systems in the longer and short term - Knowledge Engineering & Management Centre, Coventry University, West Midlands, UK.; 1
[3] Bender; Richard; (1992); Software Considerations in Airborne Systems and Equipment Certification the software quality in safety-critical systems; RTCA/DO-178B ; 82
[4] Bruege B; Duboit A; (2004); Object Oriented Software Engineering Using UML, Patterns, and Java; Prentice Hall
[5] Buschmann F; Meunier R; Rohnert H; Sommerlad P; M. Stal; (1996); Pattern-Oriented Software Architecture. A System of Patterns; John Wiley & Sons Ltd., Chichester UK
[6] Czarneski K.; (2011); Software Engineering Lecture Notes; ECE Publications Waterloo; 355
[7] Forte G.; McCully K.; (1991); CASE Outlook: Guide to Products and Services; CASE Consulting Group, Lake Oswego
[8] Forrester consultant; (2015); Existing and potential impact of technology; American independent technology and market research company
[9] Fuggetta Alfonso; (1993); A classification of CASE technology; IEEE Computer Society; 26; 12
[10] Gartner research pages; (2015); Technology and trends in software engineering ; Gartner Inc.; Gartner CIO survey
[11] Editor; (2014); Analysis of technology trends, global geospatial industry, research and academia of upcoming technologies; Geospatial World a popular publication; 1
[12] Heninger K. L.; (1980); Specifying software requirements for complex systems : New techniques and their application: IEEE Transactions on Software Engineering Journal; SE-6, 1; 2 - 13
[13] IMF Editorial; (2016); International Monetary Fund World Economic Outlook - ranked top 10 countries based on GDP of 2014- India is at 3rd rank;1 ; 1
[14] IBEF wesite; (2015); India Brand Equity Foundation (IBEF) Department of Commerce, Ministry of Commerce and Industry, Government of India. Facilitates knowledge of Indian products and services; 1
[15] Isabel M; Palma del ; Águila,José ; Túnez Samuel; (2014); Milestones in Software Engineering and Knowledge Engineering History: A Comparative Review; The Scientific World Journal; 3; 10
[16] Jeffrey A;.Hoffer Joey; George F; Valacich Joseph S; (2002); Modern Systems Analysis and Design, Automated Tools for Systems Development; Prentice-Hall, Inc; 3
[17] Johnston Stuart; (1990); AD/Cycle Application Development Cycle-strategy and architecture; Info World IBM Systems Architecture; 12, 28; 13
[18] Knight John C; (2003); Advances in software technology-since 1992 a modest proposal for their incorporation into certification; Department of Computer Science University of Virginia; 1
[19] Mahoney M S ; (2004) ; Finding a history for software engineering; IEEE Annals of the History of Computing; 26 No.1; 8–19
[20] Osterweil Leon J.; Carlo Ghezzi; Jeff Kramer; Wolf Alexander L.; (2008); Determining the Impact of Software Engineering Research on Practice; IEEE Computer Society;.41, 3; 39-49
[21] Pressman Roger S; (2014); Software Engineering: A Practitioner's Approach; -McGraw-Hill (Singapore); 7e
[22] Perrotta Paolo; (2012) ; A short history of Software Engineering; A talk from Barcelona Ruby Conference (Baruco); Oct 4, 2012
[23] Randell B; Naur P; (1969); Report of a Software Engineering conference; sponsored by the NATO Science Committee Scientific Affairs Division, Garmisch, Germany held 7-11 Oct. 1968; 231

[24] Randell B; Buxton J.N; (1970); Software Engineering Techniques; report of conference sponsored by the NATO Science Committee Scientific Affairs Division, Rome, Italy held 27-31 Oct. 1969; 164
[25] Shaw M.; Garlan D.; (1996); Software Architecture: Perspectives on an Emerging Discipline;. Prentice Hall Englewood Cliffs, NJ, 5
[26] Sommerville Ian; (1995); Software Engineering; Addison-Wesley Publishing Co.; 5
[27] Teichroew Daniel; Hershey III; E A; (1977); PSL/PSA a computer-aided technique for structured documentation and analysis of information processing systems; IEEE Tr. Software Engineering.; Se-3,1; 41-48