B.B.A. (Computer Application) Semester-VI
Formerly known as B.C.A.

RECENT TRENDS IN IT

GAUTAM BAPAT

NIRALI PRAKASHAN
ABHYUDAYA PRAKASHAN

A Book Of

RECENT TRENDS IN IT

B.B.A. (Computer Application) Semester - VI
Formerly known as B.C.A.

As Per Revised Syllabus
Effective from June 2015

Gautam Bapat
MCA (Sci.), PGDBM (Mkt.), MMS (Mkt.)
Asst. Professor, Computer Science & Applications
MITSOM College
Pune

Price ₹ 180.00

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N3474

RECENT TRENDS IN IT **ISBN 978-93-5164-853-6**

Third Edition : January 2018

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar
Off J.M. Road, PUNE – 411005
Tel - (020) 25512336/37/39, Fax - (020) 25511379
Email : niralipune@pragationline.com

Polyplate

Printed By :
RACHANA OFFSETS
S. No. 15, Arihant Marg
Sukhsagar Nagar, Katraj
Tel - (022) 2778 2011

➢ DISTRIBUTION CENTRES

PUNE

Nirali Prakashan : 119, Budhwar Peth, Jogeshwari Mandir Lane, Pune 411002, Maharashtra
Tel : (020) 2445 2044, 66022708, Fax : (020) 2445 1538
Email : bookorder@pragationline.com, niralilocal@pragationline.com

Nirali Prakashan : S. No. 28/27, Dhyari, Near Pari Company, Pune 411041
Tel : (020) 24690204 Fax : (020) 24690316
Email : dhyari@pragationline.com, bookorder@pragationline.com

MUMBAI

Nirali Prakashan : 385, S.V.P. Road, Rasdhara Co-op. Hsg. Society Ltd.,
Girgaum, Mumbai 400004, Maharashtra
Tel : (022) 2385 6339 / 2386 9976, Fax : (022) 2386 9976
Email : niralimumbai@pragationline.com

➢ DISTRIBUTION BRANCHES

JALGAON

Nirali Prakashan : 34, V. V. Golani Market, Navi Peth, Jalgaon 425001,
Maharashtra, Tel : (0257) 222 0395, Mob : 94234 91860

KOLHAPUR

Nirali Prakashan : New Mahadvar Road, Kedar Plaza, 1st Floor Opp. IDBI Bank
Kolhapur 416 012, Maharashtra. Mob : 9850046155

NAGPUR

Pratibha Book Distributors : Above Maratha Mandir, Shop No. 3, First Floor,
Rani Jhanshi Square, Sitabuldi, Nagpur 440012, Maharashtra
Tel : (0712) 254 7129

DELHI

Nirali Prakashan : 4593/21, Basement, Aggarwal Lane 15, Ansari Road, Daryaganj
Near Times of India Building, New Delhi 110002 Mob : 08505972553

BENGALURU

Pragati Book House : House No. 1, Sanjeevappa Lane, Avenue Road Cross,
Opp. Rice Church, Bengaluru – 560002.
Tel : (080) 64513344, 64513355,Mob : 9880582331, 9845021552
Email:bharatsavla@yahoo.com

CHENNAI

Pragati Books : 9/1, Montieth Road, Behind Taas Mahal, Egmore,
Chennai 600008 Tamil Nadu, Tel : (044) 6518 3535,
Mob : 94440 01782 / 98450 21552 / 98805 82331,
Email : bharatsavla@yahoo.com

niralipune@pragationline.com | www.pragationline.com

Also find us on www.facebook.com/niralibooks

Preface ...

I take this opportunity to present this book entitled as **"Recent Trends in IT"** to the students of Sixth Semester of B.B.A. (Computer Application). The object of this book is to present the subject matter in a most concise and simple manner. The book is written strictly according to the Revised Syllabus.

The book has its own unique features. It brings out the subject in a very simple and lucid manner for easy and comprehensive understanding of the basic concepts, its intricacies, procedures and practices. This book will help the readers to have a broader view on E-commerce. The language used in this book is easy and will help students to improve their vocabulary of Technical terms and understand the matter in a better and happier way.

I sincerely thank Shri. Dineshbhai Furia and Shri. Jignesh Furia of Nirali Prakashan, for the confidence reposed in me and giving me this opportunity to reach out to the students of management studies.

I thank Mrs. Aabha Athavale, Mrs. Anita Panajkar for their important inputs time to time and Mr. Akbar Shaikh and Ms. Chaitali Takale, who painstakingly attended to all the details to make this book appear good.

I have given my best inputs for this book. Any suggestions towards the improvement of this book and sincere comments are most welcome on niralipune@pragationline.com.

Author

Syllabus ...

1. **Software Process and Project Metrics, Analysis Concepts and Principles** **[06]**

   Measures, Metric indicators, Metric in process and the project domains, Software measurement, Metrics for software quality, Software quality assurance, Requirement analysis, Communication techniques, Analysis principles, Software prototyping, Case Study.

2. **Distributed Databases** **[08]**

   Standalone v/s Distributed databases, Replication, Fragmentation, Client/Server architecture, Types of distributed databases.

   Object - Relational Databases

   Abstract Data types, Nested Tables, Varying Arrays, Large Objects, Naming Conventions for Objects, Case Study.

3. **Data Warehouse** **[08]**

   What is Data Warehouse? A Multidimensional Data Model, Data Warehouse Architecture, Data Warehouse Implementation, Data Cube Technology, From Data Warehousing to Data Mining, Data Mining, Functionalities, Data Cleaning, Data Integration and Transformation, Data Reduction.

4. **Network Security** **[14]**

   Cryptography, Introduction to Cryptography, Substitution Ciphers, Transposition Ciphers, One-Time Pads, Two Fundamental Cryptographic Principles; Symmetric Key Algorithms' DES - The Data Encryption Standards, AES - The Advances Encryption Standard; Public Key algorithms; RSA - Other Public Key algorithms, Digital Signature, Symmetric - Key Signature, Public Key Signature, Message Digests.

5. **Computing and Informatics** **[08]**

   Introduction to Computing and Informatics, Types of Computing Cloud, Green, Soft, Mobile, Case Study.

Contents ...

Chapter 1...

Software Process and Principles

Contents ...

1.1 Introduction

Software process and product metrics are quantitative measures that enable software people to gain insight into the efficiency of the software process and the projects that are conducted using the process as a framework. Basic quality and productivity data are collected. These data are then analyzed, compared against past averages, and assessed to determine whether quality and productivity improvements have occurred. Metrics are also used to pinpoint problem areas so that remedies can be developed and the software process can be improved.

1.2 Measures [Oct. 16]

Measure: A standard or unit of measurement; the extent, dimensions, capacity, etc., of anything, especially as determined by a standard; an act or process of measuring; a result of measurement.

Measurement: Measurement is the act or process of measuring. A figure, extent, or amount obtained by measuring. Also a result, such as a figure expressing the extent or value that is obtained by measuring.

An example of measure: five centimeters. The centimeter is the standard, and five identifies how many multiples or fractions of the standard are being appraised. With the centimeter, someone measuring something in the United States is going to get the same measure as someone in Europe.

Let's relate this to software, such as lines of code. Currently, there really isn't a universal standard for lines of code. Someone measuring a program's lines of code in one office will probably not get the same count as someone measuring the same program in a different office. Therefore, it is imperative that each organization determine a single standard for what is meant by a line of code and ensure that everyone in the organization understands and uses that standard. Thus, a measure may be universally standard or locally standard, but it needs to be a standard.

Software measurement plays an increasingly important role in Software Engineering. Currently, software metrics are proving to be very effective for building high-quality prediction systems for large database projects for:

- Understanding and improving software development and maintenance projects,
- Assessing and maintaining system quality by highlighting problematic areas,
- Determining the best ways to help practitioners and researchers in their work, etc. Furthermore, software metrics are important tools to help assess and institutionalize Software Process Improvement in software-intensive organizations.

In software development and software testing, most commonly used measures are:

- Number of Defects found in a system or component.
- Lines of Code (LOC, KLOC).
- Number of Test Cases.

Reasons to use measure in software process:

- To characterize in order to:
  - Gain an understanding of processes, products, resources, and environments.
  - Establish baselines for comparisons with future assessments.
- To evaluate in order to:
  - Determine status with respect to plans.
- To predict in order to:
  - Gain understanding of relationships among processes and products.
  - Build models of these relationships.
- To improve in order to:
  - Identify roadblocks, root causes, inefficiencies, and other opportunities for improving product quality and process performance.

Source lines of code (SLOC), also known as lines of code (LOC), is a software metric used to measure the size of a computer program by counting the number of lines in the text of the program's source code. SLOC is typically used to predict the amount of effort that will be required to develop a program, as well as to estimate programming productivity or maintainability once the software is produced.

There are two major types of SLOC measures: physical SLOC (LOC) and logical SLOC (LLOC). The most common definition of physical SLOC is a count of lines in the text of the program's source code excluding comment lines.[1]

Logical SLOC attempts to measure the number of executable "statements", but their specific definitions are related to specific computer languages.

Consider this small piece of C code as an example to show how to determine SLOC:

```
for (a = 0; a < 100; a++) printf("Hello friends"); /* How many
lines of code is this? */
```

In this example we have:

- 1 Physical Line of Code (LOC)
- 2 Logical Lines of Code (LLOC) (for statement and printf statement)
- 1 comment line

Depending on the programmer and coding standards, the above "line of code" could be written on many separate lines:

```
/* Now how many lines of code is this? */
for (a = 0; a < 100; a++)
{
    printf("Hello friends");
}
```

In this example we have:

- 4 Physical Lines of Code (LOC)
- 2 Logical Lines of Code (LLOC)
- 1 comment line: tools must account for all code and comments regardless of comment placement.

Robert E. Park (while at the Software Engineering Institute) and others developed a framework for defining SLOC values, to enable people to carefully explain and define the SLOC measure used in a project. For example, most software systems reuse code, and determining which (if any) reused code to include is important when reporting a measure.

1.3 Metric Indicators **[April 16, 17, Nov. 17]**

The importance of measurement in software development has been highlighted this past year, largely because of the new Air Force policy on software metrics. At the same time, there seems to be a good deal of confusion on the terminology involved, specifically measure, metric and indicator. It's important to understand differences between these terms.

Metric: A quantitative measure of the degree to which a system, component, or process possesses a given attribute. A calculated or composite indicator based upon two or more measures. A quantified measure of the degree to which a system, component, or process possesses a given attribute.

An example of a metric would be that there were only two user-discovered errors in the first 18 months of operation. This provides more meaningful information than a statement that the delivered system is of top quality.

A metric, in contrast, is a derived value which cannot be measured directly. It is a number derived from one or more measures by a formula Best known metrics in software development and software testing are:

- Number of defects found per KLOC, which serves as an estimation of quality of code.
- Productivity, i.e. Size / Effort.
- Defect Density, i.e. number of defects related to size.

There are two types of metrics, objective and subjective.

Objective metrics can be quantized and are readily available, subjective metrics rely on opinions, gut feelings, personal attitudes etc. An example is CSAT (customer satisfaction), though the former is more reliable, than the later, the reliability of subjective metrics can be improved by having checklists and guidelines.

For example, survey question need to have, probe areas, facet, scale definitions before an option can be chosen.

Effective metrics must be simple, objective, measurable, meaningful and have easily accessible underlying data.

Complexity Metrics: Different types of software metrics can be calculated to find out the complexity of program control flow. One of the most widely used complexity metrics is cyclomatic complexity.

It indicates a total number of possible unique paths from the beginning of a method to exit the method. It is considered crucial since a high value of CC in a method indicates high number of tests required to cover all the paths in the method (In fact, minimum number of tests must be written for the method is equal to CC). Weighted Methods per Class (WMC) is the sum of cyclomatic complexities of all the methods belonging to the class.

Important attributes of a metric:

1. **Simple:** So that errors in computation or interpretation are avoided, and also that the metric can be comprehended.
2. **Objective:** Based on goals and objectives of the company.
3. **Measurable:** Based on things which can be reliably measured, not estimated or guessed.
4. **Meaningful:** So that they can help the managers to understand important aspects of their projects.

5. **Easy to collect:** Metric shall be automated and non-intrusive, i.e. not interfere with other activities of the developers.
6. **Easy to interpret:** So that it is easy to comprehend it, understand the causes which affect it.
7. **Hard to misinterpret:** Even when the metric may be easy to interpret, there may be cases leading to wrong conclusions. Metrics shall be designed with caution so that such cases are avoided.
8. **Valid:** To ensure, whether the metric really measures what it is supposed to; prevent systematic errors.
9. **Reliable:** They must perform their required functions under stated conditions for a specified period of time, providing consistent results; prevent random deviations in measurement.

Indicator: A device or variable that can be set to a prescribed state based on the results of a process or the occurrence of a specified condition.

For example, a flag or semaphore. A metric that provides insight into software development processes and software process improvement activities concerning goal attainment.

As the definition notes, a flag is one example of an indicator. An indicator is something that draws a person's attention to a particular situation. Another example of an indicator is the activation of a smoke detector in your home; it is set to a prescribed state and sounds an alarm if the number of smoke particles in the air exceeds the specified conditions for the state for which the detector is set.

In software terms, an indicator may be a substantial increase in the number of defects found in the most recent release of code.

An indicator is "a thing that indicates the state or level of something", thus it can be simply just a number showing value of a particular measure or metric. A better indicator could be a chart comparing two measures/metrics or projecting how a measure/metric developed during a time period. Also, a semaphore where red means bad and green means good is also a very simple indicator, which can be helpful in particular class of situations.

Thus, indicator is most general of these terms:

- Software process and project metrics are quantitative measures.
- They are a management tool.
- They offer insight into the effectiveness of the software process and the projects that are conducted using the process as a framework.
- Basic quality and productivity data are collected.
- These data are analyzed, compared against past averages, and assessed.
- The goal is to determine whether quality and productivity improvements have occurred.
- The data can also be used to locate problem areas.
- Remedies can then be developed and the software process can be improved.

In software project management, we are primarily concerned with productivity and quality metrics. There are four reasons for measuring software processes, products, and resources (to characterize, to evaluate, to predict, and to improve).

Role of Management in Software Development

The management of software development is heavily dependent on four factors: People, Product, Process and Project.

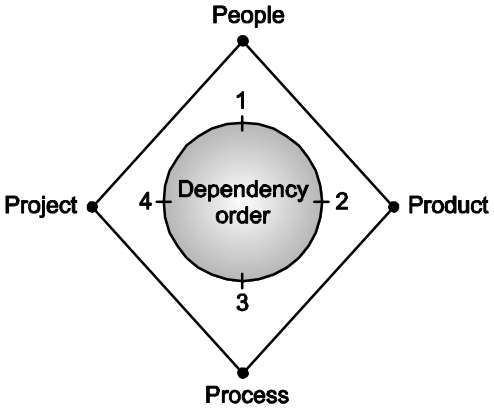

Fig. 1.1: Factors of Management Dependancy

Software development is a people centric activity. Hence, success of the project is depends on the people who are involved in the development.

The People

Software development requires good managers who can understand the psychology of people and provide good leadership. A good manager can not ensure the success of the project, but can increase the probability of success. The areas to be given priority are: proper selection, training, compensation, career development, work culture etc.

Managers face challenges. It requires mental toughness to endure inner pain. We need to plan for the best, be prepared for the worst, expect surprises, but continue to move forward anyway.

Hence, manager selection is most crucial and critical. After having a good manager, project is in safe hands. It is the responsibility of a manager to manage, motivate, encourage, guide and control the people of his/her team.

The Product

We deliver a product to the customer which is a solution to his/her problems. Hence, objectives and scope of work should be defined clearly to understand the requirements. Alternate solutions should be discussed which help the managers to select a "best" approach within constraints imposed by delivery deadlines, budgetary restrictions, personnel availability, technical interfaces etc. Without well defined requirements, it may be impossible to define reasonable estimates of the cost, development time and schedule for the project.

The Process

The process is the way in which we produce software. It provides the framework from which a comprehensive plan for software development can be established. If the process is weak, the end product will undoubtedly suffer. There are many life cycle models and process improvements models. Depending on the type of project, a suitable model is to be selected. Now-a-days CMNI (Capability Maturity Model) has become almost a standard for process framework. The process priority is after people and product, however, it plays very critical role for the success of the project. A small number of framework activities are applicable to all software projects, regardless of their size and complexity. A number of different task sets, tasks, milestones, work products, and quality assurance points, enable the framework activities to be adopted to the characteristics of the project and the requirements of the project team.

A proper planning is required to monitor the status of development and to control the complexity. In order to manage a successful project, we must understand what can go wrong and how to do it right. We should define concrete requirements and freeze these requirements. Changes should not be incorporated to avoid software surprises. Software surprises are always risky and we should minimise them. We should have a planning mechanism to give warning before the occurrence of any surprise.

All four factors (People, Product, Process and Project) are important for the success of the project. Their relative importance helps us to organise development activities in more scientific and professional way.

1.4 Metrics in Process and the Project Domains

Metrics should be collected so that process and product indicators can be confirmed. Process indicators enable a software engineering organization to gain insight into the efficacy of an existing process (i.e., software engineering tasks, work products, and milestones). They enable managers and practitioners to assess what works and what doesn't. Process metrics are collected across all projects and over long periods of time.

Their aim is to provide indicators that lead to long-term software process improvement. Project indicators enable a software project manager to:

1. Assess the status of an ongoing project.
2. Track potential risks.
3. Uncover problem areas before they go "critical".
4. Adjust work flow or tasks.
5. Evaluate the project team's ability to control quality of software work products.

In some cases, the same software metrics can be used to determine project and then process indicators. In fact, measures that are collected by a project team and converted into metrics for use during a project can also be transmitted to those with responsibility for software process improvement. For this reason, many of the same metrics are used in both the process and project domain:

- The aim of process metrics is to provide a set of process indicators that lead to long-term software process improvement.

- The only way to know how/where to improve any process is to:
  - Measure specific attributes of the process.
  - Develop a set of meaningful metrics based on these attributes.
  - Use the metrics to provide indicators that will lead to a strategy for improvement.

Process and Project Metrics

Metrics should be collected so that process and product indicators can be fixed.

Process Metrics **[Oct. 16]**

- Process metrics used to provide indicators that lead to long term process improvement.
- Private process metrics (e.g. defect rates by individual or module) are only known to by the individual or team concerned.
- Public process metrics enable organizations to make strategic changes to improve the software process.
- Metrics should not be used to evaluate the performance of individuals.
- Statistical software process improvement helps an organization to discover where they are strong and where they are week.

Project Metrics

- Project metrics enable project manager to:
  - Assess status of ongoing project.
  - Track potential risks.
  - Uncover problem before they go critical.
  - Adjust work flow or tasks.
  - Evaluate the project team's ability to control quality of software work products.
- A software team can use software project metrics to adapt project workflow and technical activities.
- Project metrics are used to avoid development schedule delays, to reduce potential risks, and to assess product quality on an on-going basis.
- Every project should measure its inputs (resources), outputs (deliverables), and results (effectiveness of deliverables).

Project metrics are used to:

- Minimize the development schedule by making the adjustments necessary to avoid delays and lessen potential problems and risks.
- Assess product quality on an ongoing basis and, when necessary, to modify the technical approach to improve quality.

1.5 Software Measurement

Measurements in the physical world can be categorized in two ways:

- Direct measures (e.g., the length of a pipe) and indirect measures (e.g., the "quality" of pipes produced, measured by counting rejects).

Recent Trends In IT

Publisher : Nirali Prakashan ISBN : 9789351648536 Author : Gautam Bapat