SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCSA1306 DESIGN THINKING

SCSA1306 DESIGN THINKING

COURSE OBJECTIVES

- To understand the principles of Design Thinking, a creative solution-based approach to problem solving.
- To understand about Agile methodology as a practice to promote continuous iteration of development and testing throughout SDLC.
- To understand the basics about development cycles, IT Operations & faster innovation.
- To understand the practice of design thinking for Strategic Innovation
- To understand DevOPs the advanced process of software engineering for faster problem resolution & team collaboration.

UNIT 1 INTRODUCTION TO DESIGN THINKING 9 Hrs.

Introduction to Design Thinking – Importance of Design Thinking – History of Design Thinking- Design Thinking Framework - Design Thinking Methods - Empathise –Define – Ideate – Prototype – Test- Software Development Methodology – Waterfall model – V –model -Customer Example.

UNIT 2 INTRODUCTION TO AGILE9 Hrs.

History of Agile – Agile principles – Agile Vs Waterfall – Agile Methodology Overview- Agile frameworks – Extreme programming - Rational Unified Process (RUP) - Test Driven Development (TDD) – Feature Drive Development (FDD)- Scrum - Kanban Methodology – Agile and Devops.

UNIT 3 AGILE SOFTWARE DEVELOPMENT9 Hrs.

Software Development- using Extreme Programming – Roles & Rules - Software Development using Scrum Framework – Scrum team – Sprints – Sprints planning – Metrics – Scrum tools - Case Studies.

Unit 4 DESIGN THINKING FOR STRATEGIC INNOVATION 9 Hrs.

Innovation Management-Changing Management Paradigms-Design Thinking related to Science and art-Design Thinking in Business-Linking Design Thinking Solution to Business Challenges

UNIT 5 DEVOPS 9 Hrs.

Introduction to DevOps – DevOpsvs Agile – DevOps Principles and Life Cycle – Introduction to CI / CD &DevOps Tools– Version Control – Build Automation – Configuration Management – Containerization – Continuous Deployment – Continuous Integration – Continuous Testing –Continuous Monitoring.

Max.45 Hours

COURSE OUTCOMES

On completion of the course the student will be able to

CO1:Apply design thinking concepts to give solution for the problems identified

CO2:Implement Agile software methodology for faster development of quality software

CO3:Describe how to improve collaboration between development andoperations.

CO4: Design innovative products

CO5:Implement Automated Installations and Deployments

CO6:Unresolve different transformations of a product or a service through brainstorming and incremental approach, etc.

TEXT /REFERENCE BOOKS

1. MaurícioVianna, YsmarVianna, Brenda Lucena and Beatriz Russo," Design thinking : Business innovation", MJV Technologies and innovation press, 2011.
2. Design Thinking: Integrating Innovation, Customer Experience, and Brand Valueby Thomas Lockwood (Editor) Published February 16th 2010 by Allworth Press.
3. KalloriVikram, —Introduction to DevOps, 1 st Edition, KalloriVikram Publication, 2016.
4. Jaokim Verona, —Practical DevOps, 2 nd Edition, Packt. Publication, 2018.
5. Stephen Fleming, Pravin, —DevOps Handbook: Introduction of DevOps Resource Management—,1st Edition, Createspace Independent Pub. , 2010.
6. Len Bass, Ingo Weber, Liming Zhu, G., —DevOps: A Software Architect's Perspective, 1st Edition, Addison-Wesley Professional, 2015.
7. Alistair Cockburn, "Agile Software Development", 2nd ed, Pearson Education, 2007.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – I – Design Thinking – SCSA1306

UNIT 1 INTRODUCTION TO DESIGN THINKING

Introduction to Design Thinking – Importance of Design Thinking – History of Design Thinking- Design Thinking Framework - Design Thinking Methods - Empathise –Define – Ideate – Prototype – Test- Software Development Methodology – Waterfall model – V – model -Customer Example.

1.1 Introduction to Design Thinking

- Design thinking is a methodology that designers use to brainstorm and solve complex problems related to designing and design engineering.
- It is also beneficial for designers to find innovative, desirable and never-thought-before solutions for customers and clients.

Brainstorm: to try to solve a problem by talking with other people : to discuss a problem and suggest solutions, Figure 1.1.

Examples:

- Mind Mapping.
- Brainwriting.
- SWOT Analysis.
- Role Storming.
- Step Ladder Brainstorming.
- Design Charrette.

Fig 1.1 Rules for Brainstorming

Mind Mapping

A mind map is an image that contains any sort of graphical element to express an idea, Figure 1.2.

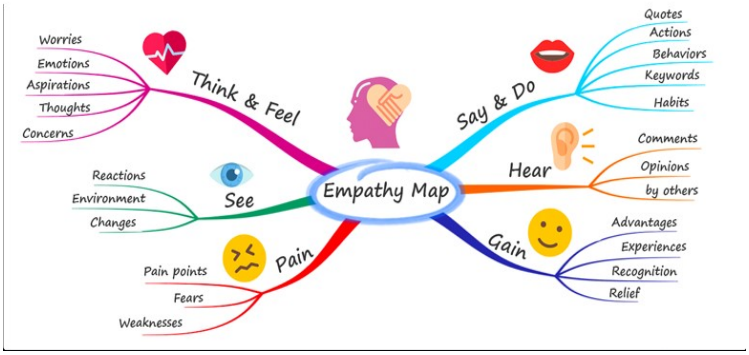


Fig 1.2 Mind Mapping

Brain writing

6-3-5 Six people sit in a table to write 3 ideas in 5 minutes. To write down their ideas about a particular question or problem on sheets, Figure 1.3

Fig 1.3 Brain writing

SWOT ANALYSIS

Fig 1.4 SWOT analysis

SWOT stands for Strengths, Weaknesses, Opportunities, and Threats, and so a SWOT Analysis is a technique for assessing these four aspects of your business, Figure 1.4 & 1.5.

Strengths **What do you do well?** **What unique resources can you drae on?** **What do others see as your strength?**	**Weaknesses** **What could you improve?** **Where do you have fewer resources than others?** **What are others likely to see as weakness?**
Opportunities What opportunities are open to you? What trends could you take advantage of? How can you turn your strengths into opportunities?	Threats What threats could harm you? What is your competition doing? What threats do your weakness es expose to you?

Fig 1.5 SWOT Analysis

Role storming

Jane and Bob are sales managers who rarely interact personally with clients. In Rolestorming, they interact as Client and Sales Person.

Jane asks for detailed information about the product before she makes a purchase, and the sales person realizes he has almost no details available.

Fig 1.6 Role Stroming

Step Ladder Brainstorming

Developed in 1992, this style of **brainstorming** encourages every member in the team to contribute individually before being influenced by everyone else. Once the topic is shared, everyone leaves the room except two members of the team. These two members will then discuss the topic and their ideas.

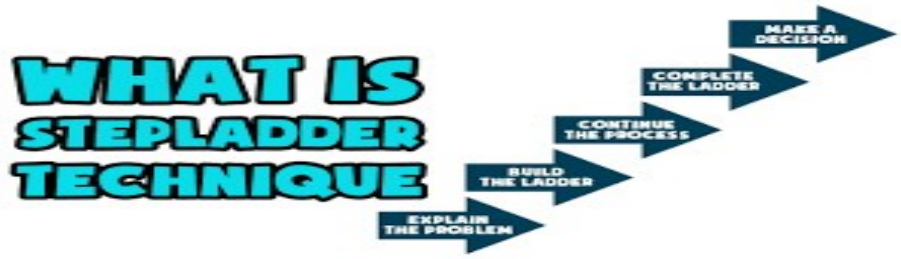


Fig 1.7 **Stepladder** Technique

Design Charrette

A Design Charrette is a type of participatory planning process that assembles an interdisciplinary team - typically consisting of planners, citizens, city officials, architects, landscape architects, transportation engineers, parks and recreation officials, and other stakeholders - to create a design and implementation plan for a specific area.

1.2 Importance of Design Thinking

Design thinking is a blend of logic, powerful imagination, systematic reasoning and intuition to bring to the table the ideas that promise to solve the problems of the clients with desirable outcomes. It helps to bring creativity with business insights, Figure 1.8-1.11.

Fig 1.8 Importance of Design Thinking

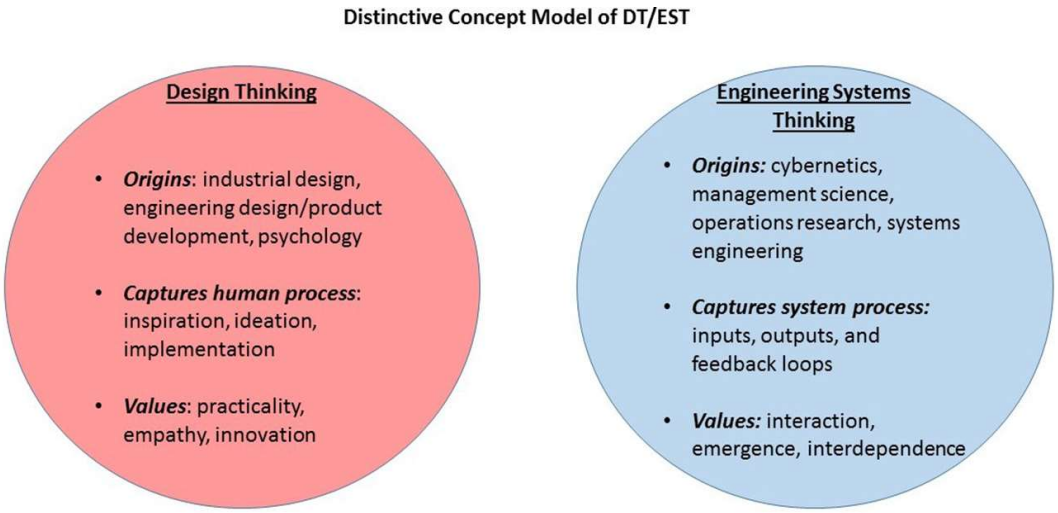


Fig 1.9 Distinctive model of DT/EST

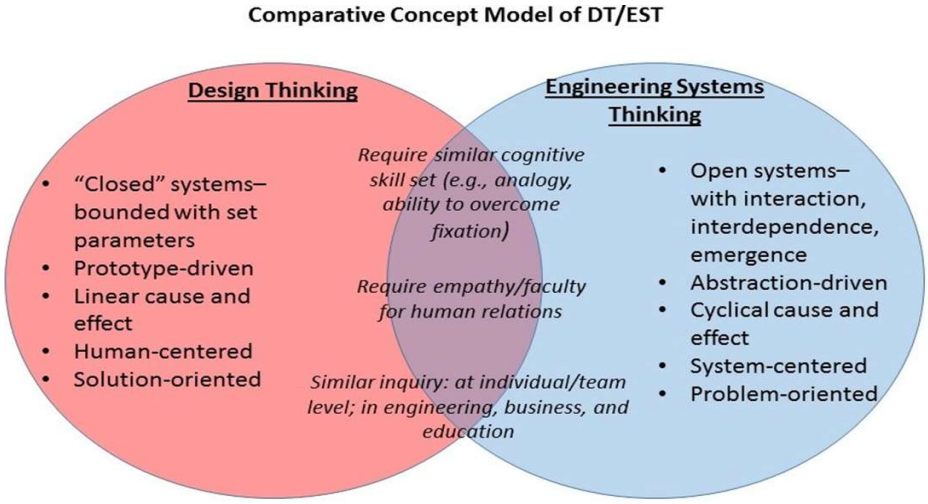


Fig 1.10 Comparative Concept model of DT/EST

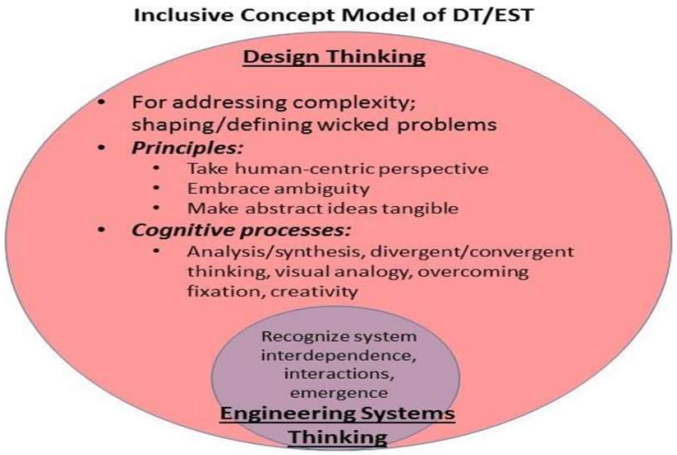


Fig 1.11 Inclusive Concept Model of DT/EST

Wicked Problem

Wicked problems are **problems** with many interdependent factors making them seem impossible to solve.

Because the factors are often incomplete, in flux, and difficult to define, solving **wicked problems** requires a deep understanding of the stakeholders involved, and an innovative approach provided by **design thinking**.

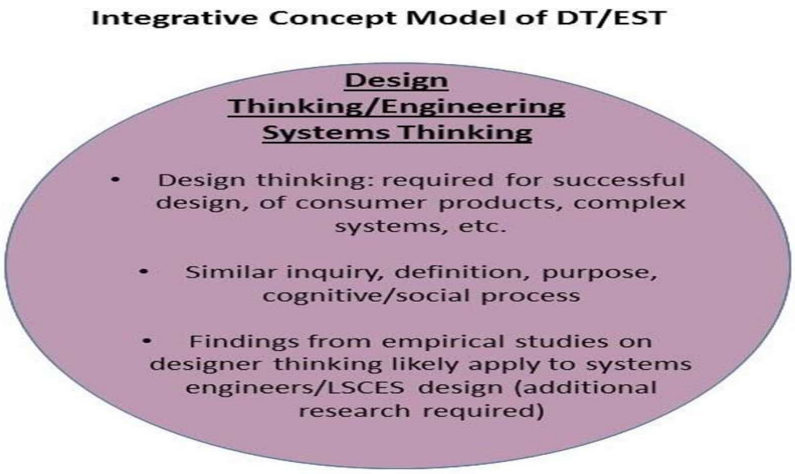


Fig 1.12 Integrative Concept Model of DT/EST

1.3 History of Design Thinking

Introduction to Design Thinking

Design thinking is a methodology that designers use to brainstorm and solve complex problems related to designing and design engineering. It is also beneficial for designers to find innovative, desirable and never-thought-before solutions for customers and clients.

Design thinking is used extensively in the area of healthcare and wellness, agriculture, food security, education, financial services, and environmental sustainability, to name a few.Design thinking has helped in the digital space, contributed to the development of physical products, spurred social innovation projects and much more.

The iterative design process helps the designers to involve clients and customers in meaningful ways. It is not just a strategy to come up with feasible solutions to a problem, but also a method to think of unimaginable solutions and then trying to make them not just feasible, but also viable.

Design thinking is a blend of logic, powerful imagination, systematic reasoning and intuition to bring to the table the ideas that promise to solve the problems of the clients with desirable outcomes. It helps to bring creativity with business insights.

Origin of Design Thinking

It is a methodology of design that originated in Stanford University and is today considered to be one of the most sought after skills in the industry. The concept of design thinking began only with a few domains under consideration, but is now found to be applicable to a myriad of disciplines, ranging from medicine and aeronautics to management, operations, and human resource planning.

The teaching and acquisition of design thinking skills has assumed so much importance that it is now being taught at some of the leading universities of the world, as well as the leading global corporate houses across the globe.

Infosys Ltd., India's second largest IT-based company providing business consulting, information technology and software engineering services, has also made design thinking a mandatory skill to be acquired by each of its employee.

Stanford University in the United States and the University of Potsdam in Germany have also promoted design thinking, citing it as one of the most useful skills for professionals.

Application across Professions

In the wake of such support and encouragement for design thinking by big entities, it is easy to understand the significance and influence that design thinking will assume in the near future for all sorts of professions. Design thinking is a methodology for finding simplicity in complexity, improving quality of experience with the designed products and serving the needs of customers by addressing the target problem faced by them. Design thinking is at the core of the development of efficient and effective strategies for organizational change.

Design thinking is a five-step process, where each step focuses on a specific goal. Each of the steps is independent of the next step but is borne out of the previous step. Design thinkers are expected not to think of the following steps when working on one step.

For example, it is not recommended to think of solutions, when the problem is being defined. The problem definition must be written in detail without missing any point, even if it makes finding a solution difficult. In this tutorial, we will understand the importance of design thinking, its impact of strategy development and we will then explore each of the steps of design thinking.

For a better understanding of the project, its requirements, and schedule we can use the methodology called Design Thinking.

It helps in the discovery phase of the product. That means **How** and **What** will make a great product. This creative approach is very important in creating things with systematic design.

Design Thinking is also known as Out of the Box thinking. Designers are trying to find more and more ways for solving the most common problems. Design Thinking is a way to

develop the products by the complete study of how customers interact with products and the conditions in which they implement.

Design thinking helps us to research the right topic and create exact prototype according to the customer's needs. It helps to go in-depth for better understanding and powerful research.

It discovers new ways to upgrade the product with its design. Design Thinking is an iterative process. It is non-linear too. It means that the previous results are continuously examined and reviewed by the development team for better understanding. It also helps to add an alternative solution that replaces the previous model or design.

Features of Design Thinking

Such problems require multidimensional solutions. Design thinking helps in this regard. It not only assists a professional to come up with a solution, but it also helps the organization to gain a competitive edge over its rivals. Following are the benefits conferred by design thinking. These are incidentally also the distinguishing features of design thinking, Figure 1.12.

- Finding simplicity in complexities.
- Having a beautiful and aesthetically appealing product.
- Improving clients' and end user's quality of experience.
- Creating innovative, feasible, and viable solutions to real world problems.
- Addressing the actual requirements of the end users.

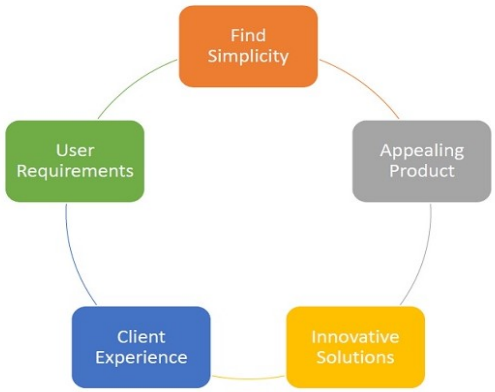


Fig 1.12 Features of Design Thinking

Most of the challenges in the world do not get solved because people trying to address those problems focus too much on the problem statement. At other times, the problem statement is overlooked and there is too much stress to find a solution.

Design thinking helps to gain a balance between the problem statement and the solution developed. A design-oriented mindset is not problem focused, but solution focused

and action oriented. It has to involve both analysis and imagination. Design thinking is the way of resolving issues and dissolving problematic situations by the help of design.

Strategy of Innovation

Design thinking is also considered to be a strategy for innovation. It leads to dramatic improvements in innovation. This is why design thinking forms the core of effective strategy development and seamless organizational change. Anything that involves human interaction, from products, services, processes etc., can be improved through design thinking. It all depends on the designer's way to create, manage, lead, and innovate.

Use of Design Thinking

The basic principle of design thinking is that innovation can be disciplined. Innovation is not an elusive entity that only a few genius people can experience. It is, rather, a practice that can be systematically approached by a set of practical and meticulous tools, methodologies, and frameworks.

Design thinking helps you learn the following.

- How to optimize the ability to innovate?
- How to develop a variety of concepts, products, services, processes, etc. for endusers?
- How to leverage the diverse ideas of innovation?
- How to convert useful data, individual insights and vague ideas into feasible reality?
- How to connect with the customers and end-users by targeting their actual requirements?
- How to use the different tools used by designers in their profession for solving your customers' problems?

Fig 1.13 Design Thinking Help

Design thinking helps, Figure 1.13 people of every profession to arrive at solutions in a planned, organized, and systematic manner. The step-by-step process helps to create solutions with both the problem statement and the required solution in mind.

Definition

Design thinking is a non-linear, iterative process that teams use to understand users, challenge assumptions, redefine problems and create innovative solutions to prototype and test.

Involving five phases, Figure 1.14—

- Empathize
- Define
- Ideate
- Prototype and
- Test

—it is most useful to tackle problems that are ill-defined or unknown.

Fig 1.14 Phases of Design Thinking

Empathize: This phase is basically the Information Gathering phase. Business-related information gathered by searching and understanding the customer's views. It is done by interviews, group discussions, and most of the observations. Along with this the questions related to What, How, Why take into consideration.

Define: In this phase, we focus on the collection and classification of the information from the empathize phase. The information gets categorized according to ideal customers, their problems, the solution to their problems and needs, and fears of users that we have to consider.

Ideate: In this phase, we give an optimized and real-time solution to the problems. No irrespective and illogical thinking accepted. These solutions are raised by Sketching and Prototyping.

Prototype: In the prototyping phase, the basic implementation of the design thinking solution is used to verify the solution in real life. During prototyping it finally takes our idea in real life. The prototype must be less expensive and the very first version of the ideal solution.

Test: After the above phases finally, it is time to verify the product in real life. Customers are able to use it and give feedback for their personal experience. Also, the designer can ask questions on how to improve such products for better usage.

Design Thinking - Applications

Design Thinking is Helpful in Many Areas:

It is used in project management; it is used to define the scope and architecture of the project.

It is used for business management. It used to focus on the features which have more value in the actual world.

It helps to allocate the goal so that we can go towards the exact direction with more clear views. In this way, it is helpful in the development field.

For most of the team works, It allows us to work in a more effective manner and according to users' requirements.

Design thinking finds its application across a variety of professions. From sports, education and research to business, management and design, design thinking is widely used by professionals around the globe.

Design thinking is halfway between analytical thinking and intuitive thinking. Analytical thinking involves purely deductive reasoning and inductive logical reasoning that utilize quantitative methodologies to come to conclusions. However, intuitive thinking refers to knowing something without any kind of reasoning.

These are two extreme kinds of thinking. Design thinking makes use of both the extremes in an optimum manner. The intuitive thinking helps in invention for the future, whereas analytical thinking to create something creative in the present, which is replicable. The willingness to use these futuristic solutions is what is called abductive logic.

Business

Design thinking helps in businesses by optimizing the process of product creation, marketing, and renewal of contracts. All these processes require a companywide focus on the customer and hence, design thinking helps in these processes immensely. Design thinking helps the design thinkers to develop deep empathy for their customers and to create solutions that match their needs exactly. The solutions are not delivered just for the sake of technology.

Information Technology

The IT industry makes a lot of products that require trials and proof of concepts. The industry needs to empathize with its users and not simply deploy technologies. IT is not only about technology or products, but also its processes. The developers, analysts, consultants, and managers have to brainstorm on possible ideas for solving the problems of the clients. This is where design thinking helps a lot.

Education

The education sector can make the best use of design thinking by taking feedback from students on their requirements, goals and challenges they are facing in the classroom. By

working on their feedback, the instructors can come up with solutions to address their challenges.

For example, Michael Schurr, a 2nd grade instructor from New York, realized that his students would be more comfortable with bulletin boards lowered. He also found the idea of creating comfortable semi-private space for working students as it provided them space to study. As a result, his students became more engaged and felt free to move.

Healthcare

Design thinking helps in healthcare as well. The expenditure on healthcare by the government and the cost of healthcare facilities is growing by the day. Experts worldwide are concerned about how to bring quality healthcare to people at low cost.

Venice Family Clinic in Venice, California has come up with innovative solutions to the challenge of opening a low-cost children's clinic to serve the low-income families. Problems of finance, transportation, and language barriers had to be solved. And all this had to be done at low cost for the poor kids. Fostering good health along with profits was a challenge, as it does not sound sustainable. Using design thinking, the inefficiencies in the system and the perennial crises were addressed.

Analysis + Synthesis = Design Thinking

Analysis and synthesis, thus, form the two fundamental tasks Figure 1.15 to be done in design thinking. Design thinking process starts with reductionism, where the problem statement is broken down into smaller fragments. Each fragment is brainstormed over by the team of thinkers, and the different smaller solutions are then put together to form a coherent final solution. Let us take a look at an example.

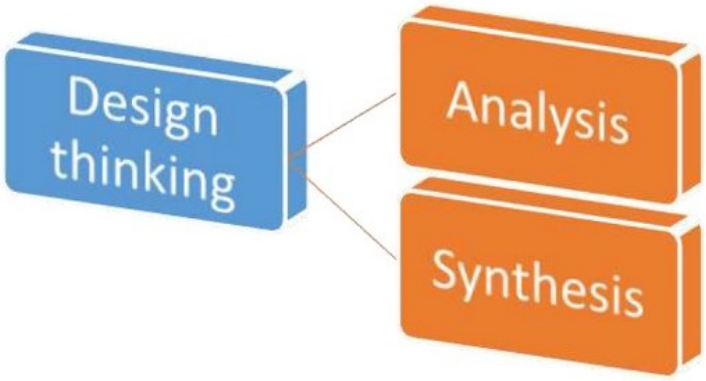


Fig 1.15 Design Thinking Fundamental Task

Analysis

Analysis is derived from the Greek word 'analusis', which translates into 'breaking up' in English. Analysis is older than the times of great philosophers like Aristotle and Plato. As discussed in the previous section, analysis is the process of breaking down a big single entity

into multiple fragments. It is a deduction where a bigger concept is broken down to smaller ones. This breaking down into smaller fragments is necessary for improved understanding.

So, how does analysis help in design thinking? During analysis, design thinkers are required to break down the problem statement into smaller parts and study each one of them separately. The different smaller components of the problem statement are to be solved one-by-one, if possible. Then, solutions are thought for each of the small problems. Brainstorming is done over each of the solutions.

Later, a feasibility check is done to include the feasible and viable solutions. The solutions that don't stand firm on the grounds of feasibility and viability are excluded from the set of solutions to be considered.

Design thinkers are, then, encouraged to connect with the diverse ideas and examine the way each idea was composed. This process of breaking down the bigger problem statement at hand into multiple smaller problem statements and examining each as a separate entity is called analysis.

Reductionism

The underlying assumption in analysis is reductionism. Reductionism states that the reality around us can be reduced down to invisible parts. The embodiment of this principle is found in basic axioms of analytic geometry, which says "the whole is equal to the sum of its parts". However, understanding of a system cannot be developed by analysis alone. Hence, synthesis is required following analysis.

Synthesis

Synthesis refers to the process of combining the fragmented parts into an aggregated whole. It is an activity that is done at the end of the scientific or creative inquiry. This process leads to creation of a coherent bigger entity, which is something new and fresh. How does synthesis come into picture in design thinking?

Once the design thinkers have excluded the non-feasible and non-viable solutions and have zeroed-in on the set of feasible and viable solutions, it is time for the thinkers to put together their solutions.

Out of 10 available solutions, around 2-3 solutions may need to be excluded since they may not fit into the larger picture, i.e. the actual solution. This is where synthesis helps.

The design thinkers start from a big entity called the problem statement and then end up with another bigger entity, i.e. the solution. The solution is completely different from the problem statement. During synthesis, it is ensured that the different ideas are in sync with each other and do not lead to conflicts.

Design Thinking - Divergent

Design thinking involves two types of thinking, viz. convergent thinking and divergent thinking, Figure 1.16. One needs to think of many solutions to a common problem statement and then arrive at the correct and the best solution.

Divergent thinking is the process of devising more than one solution for a problem statement. It refers to the thought process of generating creative solutions. The main features of divergent thinking are −

- It is a free flowing chain of ideas.
- It happens in a non-linear manner, i.e. it does not follow any particular sequence of thinking. Moreover, multiple ideas can emerge at the same time, rather than one idea coming up only after the other has occurred.
- Non-linearity also means that multiple solutions are thought of and explored at the same time. This happens in a very short amount of time and unexpected connections are developed between the ideas.

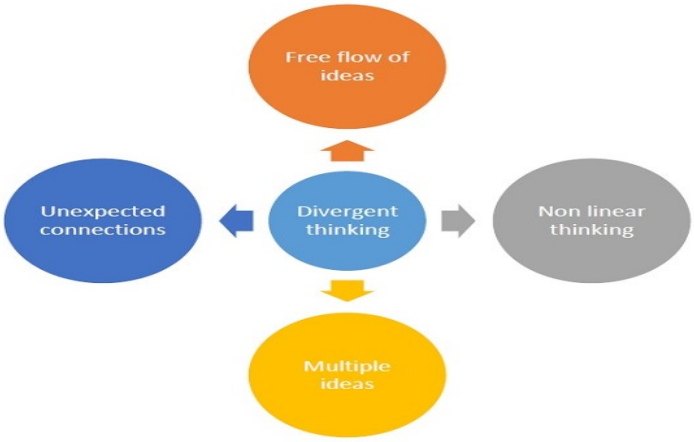


Fig 1.16 Divergent Thinking

A stimulus is provided to the design thinker and that prompts creative elaboration of ideas by the thinker.

Divergent thinking is supposed to enhance creativity of thinkers. The term 'Divergent Thinking' was first coined by J. P. Guilford in 1956. The Free Association Theory of Creativity says that concepts are connected inside our brains as semantic networks. Psychologists have claimed that the difference in creativity levels of people is dependent on the type of semantic networks of concepts inside the human mind. Following are the two types of connections −

- Flat
- Steep

The design thinkers with flat networks are those with numerous loose conceptual connections. They are more creative. The people with steep networks are more logical, because of the linear associations between the nodes. Because divergent thinking proceeds in a non-linear fashion, a person with flat associative network will be more successful in divergent thinking.

Before getting into the exercise of design thinking, a person has to find out what type of thinker the person is. If a person can think of diverse solutions, without any pre-determined set of solutions, then the person is a divergent thinker. Let's take a look at an exercise on divergent thinking.

Case Study

Problem Statement − The process of knowledge transfer is a huge problem for the organization. Let's call our organization 'DT'. DT wants to eliminate the overhead of shelling out extra money and investing time for transferring knowledge to its new employees. The problem statement at hand is "Knowledge transfer adds to the cost of the company". Let's think of ways to eliminate or at least, reduce the cost to the company.

Solution − Following can be some of the possible and even not-so-possible solutions.

- DT can eliminate the process of knowledge transfer.
- DT can conduct classroom sessions for knowledge transfer, where a large number of new employees can be seated and just one instructor can deliver sessions to many employees at once. This will reduce the cost as the number of paid instructors required will be less.
- DT can come up with a document for knowledge transfer and can mail it to every new employee. The employees can go through the document and hence, can selfhelp for knowledge transfer.
- DT can ask the employees to search for material online to gain knowledge of new tools and processes, which are currently in use in the industry.
- DT can hire only those employees who have adequate knowhow of tools and techniques that DT works on. This will eliminate the need of knowledge transfer.

Design Thinking - Convergent

Convergent thinking is exactly opposite of what divergent thinking is. The term 'Convergent Thinking' was coined by Joy Paul Guilford in 1956. The concept of convergent thinking requires the design thinker to go through all the possible solutions thought during divergent thinking and come up with a correct solution. This convergence on a single solution or a mix of limited number of solutions is the essence of convergence thinking.

Convergent thinking is the type of thinking in which a thinker is generally supposed to come up with a single well-established best-possible solution to a problem. This step delivers the best and a concrete solution to a problem statement, taking into account all the factors and requirements specified in the problem statement.

Convergent thinking requires speed, accuracy, efficiency, logical reasoning, and techniques. A thinker is supposed to recognize the patterns, reapply a few techniques, and accumulate and organize the stored information.

Aspects of Convergent Thinking

The principle aspect of convergent thinking is that it should help us arrive at a singlebest answer without any room for ambiguity. The ideas thought of in the process of divergent thinking are either considered to be possible or impossible in convergent thinking phase.

Another important aspect of convergent thinking is that judgment is an important part of this process. Divergent thinking requires thinkers to suspend judgment. Convergent thinking encourages thinkers to apply the power of judgment.

Let's look at the exercise of divergent thinking and start applying convergent thinking on it

We got the following ideas in the divergent thinking exercise.

- Elimination of knowledge transfer program.
- Having a single instructor for knowledge transfer program in a classroom session.
- Preparing a document for knowledge transfer program.
- Making it mandatory for employees to search for knowledge resources online.
- Hiring only those employees who are experienced enough and who don't need knowledge transfer.

Now, looking at the five ideas, it can be easily said that option 1 is not feasible. Every employee does not have an idea of a company's tools and techniques and hence, cannot be expected to survive without knowledge transfer.

For the same reason, option 5 is also not acceptable. The best practices of a company are seldom known to new employees and taking an assumption about an employee's knowledge level is a huge mistake. It is considered to be a good HR practice to have knowledge transfer session for new employees.

If we go by option 4, we are not assured of the pace at which learning will happen for the new employees. Each employee can take variable amount of time to grasp the concepts. The time taken to search materials online and read them is an overhead in itself and it cannot be monitored.

Hence, the two better options that remain are option 2 and option 3. However, one cannot correctly estimate the effectiveness of a document for knowledge transfer. It is similar to reading materials online. Hence, the best option available is to have an instructor teaching employees in a classroom program.

Although, the employees won't get personal attention at times, yet by maintaining a fine balance between the strength of the batch and the length of class, this can be the best option to reduce cost and overhead. The reduction in the number of instructors will lead to less

expenditure for DT and at the same time, the effectiveness of a paid instructor will remain, making the process of knowledge transfer as effective as before.

This is how convergent thinking comes into picture.

Design Thinking - Attributes

The Principles of Design Thinking

According to Christoph Meinel and Larry Leifer, there are four principles to design thinking, Figure 1.17.

- The Human Rule − This rule states that all kinds of design activity are ultimately social in nature.
- The Ambiguity Rule − This rule requires all design thinkers to preserve ambiguity in the process design thinking.
- The Re-design Rule − The re-design rule states that all designs are basically examples of re-design.
- The Tangibility Rule − The tangibility rule states that making ideas tangible always facilitates communication between design thinkers.

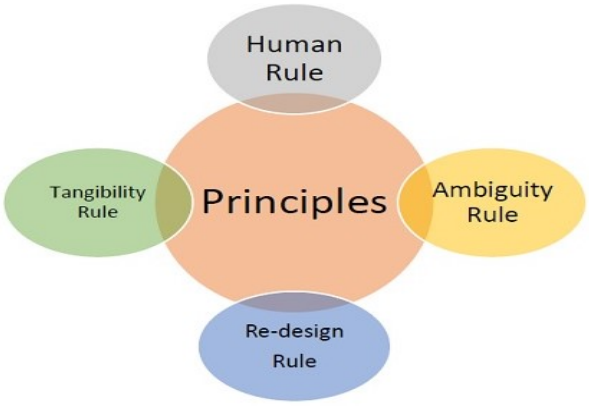


Fig 1.17 Design Thinking Principles

The Challenges

The next attribute is called as the ‘wicked problems’. These are the challenges that are faced by the design thinkers. Design thinking helps the designers in almost all professions to tackle these wicked problems. These challenges are supposed to be ill-defined or tricky.

Horst Rittel was the first person to refer to such problems with the word ‘wicked problems’. In the case of ill-defined problems, the problem statement and the solution are both unknown at the beginning of the design thinking exercise. In well-defined problems, at least the problem statement is clear and the solution is available through technical knowledge.

In wicked problems, the design thinker may have a general idea of the problem, but significant amount of time and effort goes into requirement analysis. Requirement gathering, problem definition, and problem shaping are major parts of this aspect of design thinking.

The Aha-Moment

Once the design thinker has spent considerable amount of time in finding a solution, there occurs a moment when the thinker suddenly finds his way clear of all obstructions. This is the moment when the solution or a bright idea strikes the thinker's mind. The aha-moment is the time when the results of convergent thinking and divergent thinking, analysis, problem definition and shaping, requirements analysis and the nature of the problem all come together and the best resolution is captured.

At the aha-moment, the process of design thinking begins to appear clear, which actually appears hazy and unidirectional before the moment. The focus on the solution grows clear after this moment and the final product or the final solution is constructed hereafter.

Design Methods

Every design discipline makes use of a set of specific techniques, rules, and ways of doing things. These are called design methods. The methods include tasks like interviewing, creating user profiles, searching for other available solutions in the world, creating mind maps, creating prototypes for solving a problem and asking for answers to questions like five whys.

The 'five whys' is an iterative interrogative technique, which is used to explore the causeand-effect relationships underlying a particular problem. The technique helps to determine the root cause of any problem by repeating the question 'Why?' Each question forms the basis of the next question. This technique is developed by Sakichi Toyoda. This helps to find the root cause of many problems faced by designers. Five whys technique is used for root cause analysis.

1.4 The Five-step Process of Design Thinking

The design thinking process or method has five steps in all to be followed, Figure 1.18. The process starts with empathizing with the problem of the customer or the end-user. The process then moves to ideate on solutions using divergent thinking. The prototype is developed after convergent thinking and then the design thinkers resort to testing the prototype. We will learn more about each of these steps in the subsequent chapters of this tutorial.

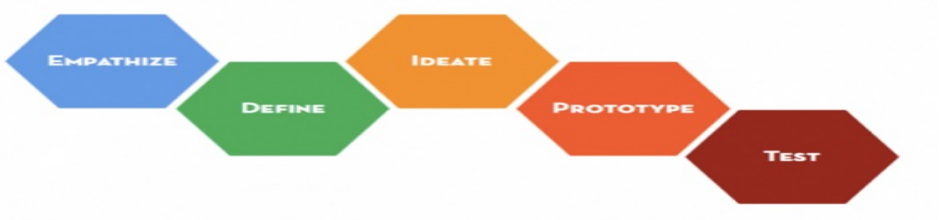


Fig 1.18 Steps involved in Design Thinking Process

1.5 Design Thinking - Empathize Stage

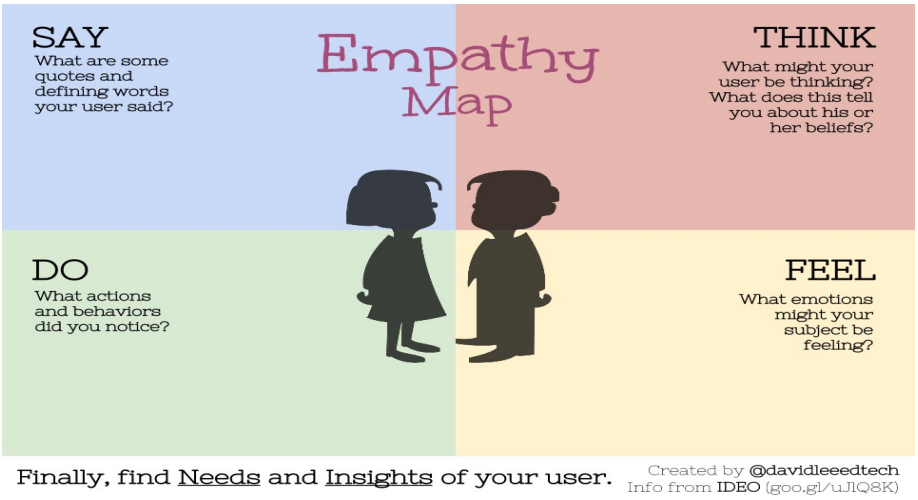


Fig 1.19 Empathy Map

This step involves putting oneself into the shoes of the customer or the end-user of our solution. We need to understand the problems faced by the customer and we, as design thinkers, need to empathize with the customer. This step is carried out in the form of requirement gathering, which involves interviews and sometimes, even field visits. This step involves the process of analysis, Figure 1.19. There are a few points to be considered while interviewing the customer.

- The interviewer must brainstorm for the questions beforehand and must be fully prepared for the interview.
- The questions being asked must be open questions. No such question should be asked for which the interviewee can answer only in Yes or No. Such binary questions must be avoided.
- The interviewer must have plenty of 'why' questions. Here, the 'five whys' method can help.
- The themes of the questions must not be intermingled. The themes must be arranged properly and questions pertaining to a particular theme must be asked together.
- The questions must be refined thoroughly so that no trace of ambiguity is left in them.

Let's take a deeper look at this section using the example of DT's problem statement. To fully understand DT's problem, we need to engage in an interview with DT employees, those who are working and even those who are leaving. It is important for us, as design thinkers,

to observe, engage, and listen to the interviewee. To create meaningful innovations, we need to understand the needs of the customer and know how it feels. Following can be few of the questions that can be asked to the employees.

Regarding Motivation to Work

- What motivates you to come at the workplace?
- What is the thing that drains you off energy at workplace?
- Is the factor for demoralization related to company policies or your peers?

Regarding Leaving the Company

- What are your aspirations?
- How is the other company fulfilling your aspirations?
- Is your decision related to something other than workplace motivation?

Regarding Time of Leaving

- How does this time suit your decision to leave the organization?
- Does your decision has anything to do with appraisal? If yes, how?

The following questions must be asked to DT's management.

Regarding Employee Attrition

- Has any pattern been observed between the employees leaving the organization and their appraisal ratings?
- What are the issues that the employees have complained about in the past regarding their workplace?

Regarding Knowledge Transfer Mechanism

- What does a knowledge transfer program constitute of?
- How much money goes as expense of knowledge transfer program?
- What is the current methodology of knowledge transfer program and how effective is it?
- How can the budget allocated to knowledge transfer program be increased or decreased?
- What are the indispensable resources and pre-requisites for a knowledge transfer program?

Once these questions have been answered, we can proceed to the further steps with more clarity. This way, a design thinker will be able to cover all the necessary details related to the

problem, gather all the requirements and think of the solutions with an exhaustive set of facts and information in hand. This will help in converging at a solution that takes into consideration the answers of all the questions.

1.6 Design Thinking - Define Stage

The first step towards defining a problem is to find who the user is, what is his/her/their needs and then develop insights from the answers. Think of 'How might we?' questions. For example, 'how might we motivate the employees in DT?', 'How might we address the concern on the connection between appraisal ratings and attrition?', 'How might we reduce the cost of knowledge transfer program without compromising its quality and the mandatory pre-requisite resources?' and many other questions along the similar lines.

1.7 Design Thinking - Ideate Stage

The third component of design thinking process is the most interesting and perhaps, the most rigorous as well. In this section, called Ideate, a design thinker is supposed to bring to the table as many ideas as possible, Figure 1.20. While brainstorming for ideas, it is not checked whether the idea is possible, feasible, and viable or not. The only task of thinkers is to think of as many ideas as possible for them.

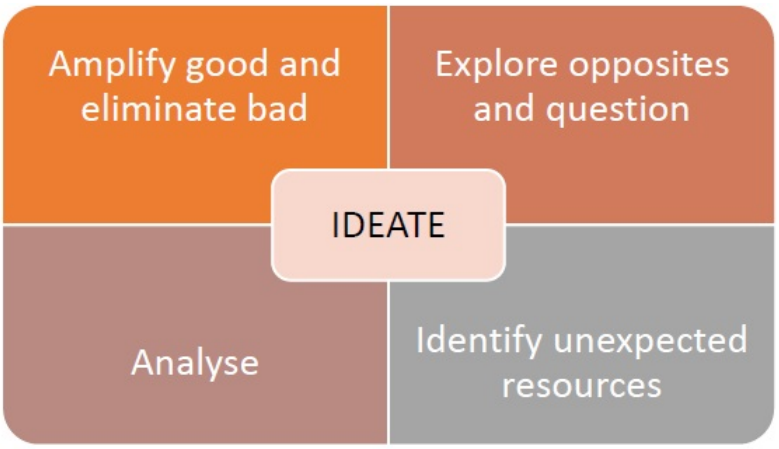


Fig 1.20 Design Thinking Ideate Stage

In this process, design thinkers also resort to the use of boards, sticky notes, sketching, chart papers, mind maps, etc. We will take a look at mind maps later in this section. Design thinkers also build on the ideas of other design thinkers. All solutions suggested by design thinkers are brought to the table and thought over. There are rules for brainstorming. They are as follows.

Rules for Brainstorming

- Only one conversation is allowed at a time. No other person must intervene when an idea is being given.

- Focus must be on the quantity and not on quality. In this step, the group must have large number of ideas with them.
- Think out of the blue. Wild ideas must be encouraged even if they invoke plain humor or seem impossible.
- The group leader must defer judgment. The fellow thinkers also need to suspend judgment. Judgmental attitude leads to an obstruction for the thinkers.
- Visualization is important. The design thinkers must create a visual picture of the problem statement and then try to see a visual image of their ideas as well.
- Build on each other's ideas. Support other ideas and build on them through group discussions and healthy debates.

Following is one of the techniques to brainstorm for ideas.

Mind Maps

Mind map is a diagram that helps to observe and study information in a visual manner. Mind map is created around a single problem statement and all the ideas to solve the problem are written around it. The problem statement usually is written at the center of a blank page as a hub and branches shoot out in all directions representing the solutions.

The ideas can be represented as text, images, trees, and even smaller mind maps. The entire map looks like a top view of a tree, with the problem statement as the trunk and the solutions as branches. It is also known by the name of spider diagram.

However, mind map is not a mere haphazard diagram. It is a well-structured organized diagram meant to aid the thinking process and to streamline the analysis and synthesis process. The guidelines to create a mind map are as follows.

Guidelines to Create Mind Maps

- Begin with the problem statement at the center of a blank white page.
- Use images, different colors, symbols, caricatures, abbreviations and codes to depict your ideas. Text can be boring, but different depictions can add an altogether different charm to your mind map.
- Keywords must replace long statements. The mind map must give a hint to the design thinker about an idea quickly. Reading a long statement is waste of time.
- Each and every word written in the mind map Figure 1.20 must be connected to the central hub by some or other line or set of lines.
- Use multiple colors for visual stimulation.
- Use radial hierarchy and make use of emphasis, italics, and underlines to stress on a point.

Fig 1.21 Mind Map

Ideate process can also be done with the help of sketches, screens, and storyboards. There are teams in corporate organizations which have large whiteboards and they paste their ideas on it using sticky notes. Different categories of ideas are represented in sticky notes of different colors and this helps in segregation of ideas, Figure 1.22.

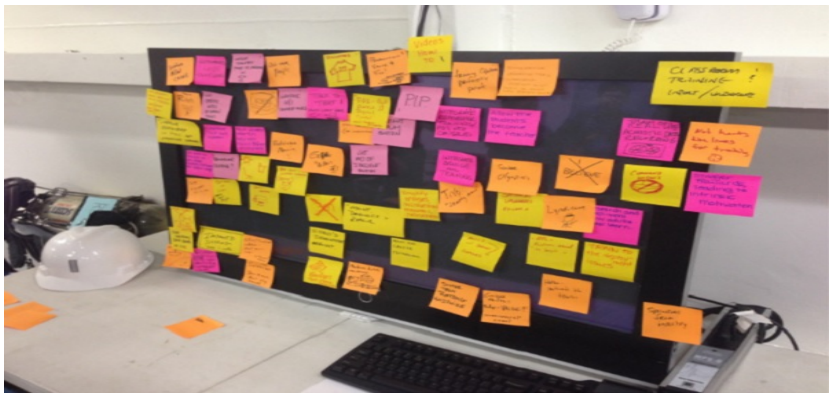


Fig 1.22 Whiteboards for segregation of ideas

The main idea behind the ideate step in design thinking process is to generate ideas and try to segregate them into categories. This helps in brainstorming without judgment, helps in bringing all the ideas to the table and helps proceed to the next step called 'Prototyping', where the ideas are checked for their feasibility and value.

Let's try to ideate the DT problem.

Let's bring out all the ideas. Some of the ideas can be as follows.

- Have a different mechanism for appraisal of employees.
- Organize events that feature team-building activities. This will help boost the morale of employees and will make them work in a team in a better fashion.
- Discard the appraisal system.
- Paste motivational posters in cubicles and pantry area.
- Call a motivational speaker and conduct a session.

- Encourage fellow employees to take the onus of motivating other employees.
- Introduce a bond period for the employees so that they don't leave soon.
- Eliminate knowledge transfer program.
- Ask only for expert employees to join the organization.
- Ask employees to fend for themselves for knowledge transfer.
- Conduct large classroom sessions with a huge audience listening to one instructor.
- Make an online document for knowledge transfer program.
- Make video tutorials.
- Have online instructor teaching across geographies.

And the list goes on endlessly….

You can come up with even more and better ideas. There is no limit to the generation of ideas. Let's represent these ideas using a mind map, Figure 1.23.

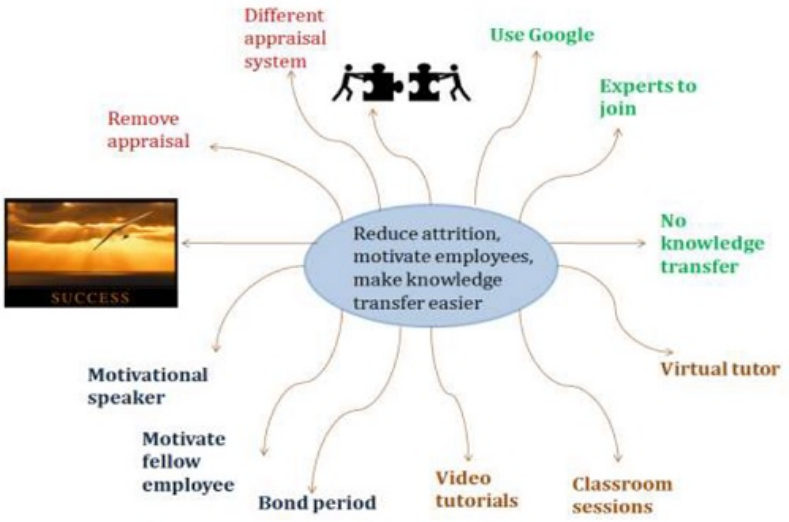


Fig 1.23 Mind map

This is when we can draw analogy with similar situations. Take for example the case of schools. The knowledge transfer program is nothing much different than schools teaching the students. How does a school manage to keep the students motivated towards studies? How does a school teach the kids?

If we draw the analogy, we would understand that in schools, a single teacher teaches around 30-40 kids in a classroom. To keep them focused on studies, exams are conducted periodically. Using digital technology, smart classrooms teach kids using videos, presentations, and audio aids.

The same model can be replicated in DT company as well. We can have a single instructor teaching the new employees with the help of videos and presentations. A proctored exam will help to assess the learning levels of the new employees.

1.8 Design Thinking - Prototype Stage

This step deals with building the ideas and checking for their feasibility to arrive at the final solution. This is the step in which three things are mainly taken care of.

- Creation of experience
- Getting feedback
- Iteration

The step of prototyping is the one in which the end user comes into picture. The end user is actively involved in this component of design thinking. All the feedback is taken from the customer, and based on the criticisms, suggestions, and appreciations received, the design thinkers create a better solution after iterating the process of design thinking's first three steps, viz. Empathize, Define, and Ideate.

Prototyping requires thinkers to create tangible products, which can be small-scale models of the exact solution.

Primary Guidelines for Prototyping

- Take the first step and start to build the prototype. Don't procrastinate.
- Don't waste too much of time on building a single prototype.
- The prototypes must be built with the end user in mind.
- The prototype must not be a mere piece of trash; it must create an experience for the user.
- Think of open questions that the user can shoot towards you when he experiences the prototype.

The prototype is meant solely for the end user. There is no value in the prototype in case the user does not feel comfortable and satisfied with it. Once the prototype has been developed, the next steps are as follows.

- Take the end user through the prototype and let him/her experience it completely.
- Throughout the experience, make the user speak about his moment-by-moment experience. This will help you, as a design thinker, to capture the minute details of the experience.
- Try to actively observe and enthusiastically engage with the user during the experience.

- Once the experience is over, follow up with the user who had the experience with a set of questions. It will be better if the set of questions is not impromptu and is prepared beforehand.

Let's have a look at the DT example.

Knowledge transfer program cannot be eliminated as it is not wise to assume that all new employees will possess adequate knowhow of the technologies in the industry beforehand. It is considered to be a good HR practice to provide a knowledge transfer program to each new employee. Even if we question this, we can find that the assumption that applicants for a job will already possess all the knowledge can fire back at us.

Moreover, asking the employees to motivate other fellow employees can be unsustainable as there will be too much of reliance of employees for managing this issue. There will be no regulation over what employees might say in the name of motivation, and hence, employees can even end up inciting others to leave the company.

The best option for knowledge transfer program, at present, is to have a classroom session where many people can study at once. This will reduce the cost and streamline the knowledge transfer program making it effective as well. Moreover, team-building activities can add to the budget of the company if done outside the premises.

However, small activities outside the working hours inside the company itself can help in team-building amongst employees. This bond can help to make them stay together as a team and stay longer in the company. Motivational posters and timely appreciation can also help.

Final Prototype

So, our prototype looks like this. We can renovate a small section of the company's premise, for example, a small section in the ground floor of a building of the company, which will have motivational posters pasted on walls. A set of team-building activities willbe conducted for a week and feedback will be taken from the employees on how they felt about it. We need to understand if they felt happy to have such an activity inside DT.

In the meantime, a single instructor led classroom session can be organized for a week for all new joinees and feedback can be taken on their level of satisfaction over the session. An exam will check their learning levels as well.

Along similar lines, many other prototypes can be created for testing.

1.9 Design Thinking - Test Stage

This phase is also called as 'Execute'. This is the phase where the final solution is tested on a full scale basis. The idea that seems the best according to the feedback of the customers and end users in the prototype phase will be executed. In this step, the design thinkers are supposed to be collaborative and agile.

Testing will help to understand what actually works and what does not. This step can be the most rewarding, if the prototypes succeed to give positive results, or can be the most annoying, if the prototype fails. After testing, the entire process of design thinking may have to be repeated. If the end user approves the solution, then the process of design thinking stops here.

Iterate Phase

If the end-user is not satisfied with the results, the design thinker will need to frame a new problem definition by incorporating the insights from the last Test phase and will have to again empathize in a better way with the end user. Ideate process will be repeated, followed by prototyping and another round of Testing. If Test phase fails to give positive results again, another round of iteration will have to be done, Figure 1.24. This way, the process of design thinking can stretch infinitely as well.

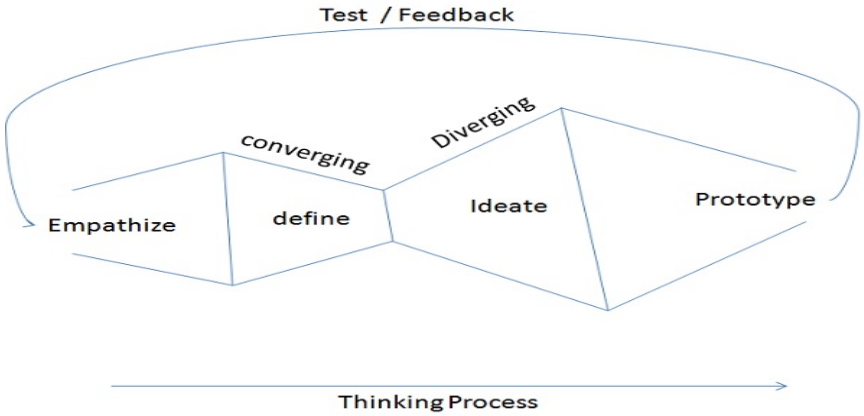


Fig 1.24 Design Thinking Test Phase

Let's take a look at the DT problem.

Suppose the prototype has given us positive results for the small scale model. We can then replicate the model on a larger scale inside the entire company building. We can perhaps take it to all the buildings as well. Motivational posters will be pasted over the walls and team-building activities will be conducted. Moreover, classroom session aided with digital technology will be instrumental in driving our prototype forward.

There can be cases where some problems may arise. For instance, an employee who works at the client location outside the company's premises may feel left out since he/she cannot participate in the activities that happens inside the company premises. Such people can also ask for similar activities in the offices of DT's client, which may not be possible since the client may not grant permission.

However, classroom sessions may work for tutoring on some technologies. For instance, a software tool can be taught to the new employees of DT via huge classroom session, but operating machines requires each employee to learn the techniques under careful personal supervision. This model won't find place in the areas were the operations deal with operating large machines. For getting hands-on experience, the employees will need to have the

instructor giving them individual attention. For this, either a large number of instructors are required or the duration of knowledge transfer program has to be increased, which will lead to increase in costs.

The design thinkers will need to draft a new problem definition and will have to brainstorm for ideas to solve the new issue and to have a uniform solution implemented across the company.

1.10 Software Development Methodology

Agile project management framework

Scrum is a framework that helps teams work together. Often thought of as an agile project management framework, scrum describes a set of meetings, tools, and roles that work in concert to help teams structure and manage their work.

Agile vs. Design Thinking

Software development is an area that is driving innovation for many different aspects of our lives. Agile methodologies and scrum processes have changed how teams work together and have enabled faster decisions, quicker releases, and early prototyping and testing. This has been necessary to operate in a market where

1) Resources are expensive and

2) Conditions are changing rapidly.

If you compare agile methodologies with Design Thinking, you might think that the latter one is already present in software development processes.

Regarding parallels between these two, Lindberg names “user-centricity”, “iterative learning and development processes” and “extensive team communication” to underscore strong similarities.

However, he also sees crucial differences between the two: The focal point of design thinking, according to Lindberg, is to put divergent options on the table and prototype. Iterative prototyping is at the core of design thinking. Agile, on the other hands, focuses on the question on what to do next.

How to use Design Thinking for software development projects?

With obvious differences being present between Agile and Design Thinking, Lindberg also introduces new solutions on how to actually introduce Design Thinking into software development.

Important for his approach are three key aspects that he sees as the foundation of Design Thinking. They are:

Exploring the problem space: What problem does the user have? What is the broader context? What is the collaborative understanding of the situation?

Exploring the solution space: What are creative solutions to the problem?

and the iterative alignment of both spaces: Iteration is what Agile and Design Thinking have in common, but iteration is used to a far greater extent in Design Thinking.

How can teams work on integrating these approaches into their workflows?

Lindberg et al. see a solution in introducing Design Principles as a precursor to the actual agile development process. He writes:

“a company pursuing an agile approach can start with design thinking-inspired concept development before beginning the actual agile development process.”

However, he sees one important conflict between Design Thinking and agile dev processes that is illustrated in different understandings of the role of a “prototype”:

“Design thinking prototypes have the main purpose of supporting learning about the underlying product concept. In contrast, software prototypes are generally made of the same material as the final product, and are – in the case of agile – continuously iterated into the final product.”

The prototypes of Design Thinking are intended as learning possibilities: the idea is to find a solution without running into the wrong direction for a long time, but the prototype isn’t necessarily related to the final product.

1.11 Design Thinking in a Waterfall World

The Waterfall model Figure 1.25 is the most straightforward design thinking methodology in which the outputs of each phase are the inputs to the following stage. It is best suited for small, simple, and well-defined problems.

Waterfall is expensive, time consuming and unforgiving of errors.

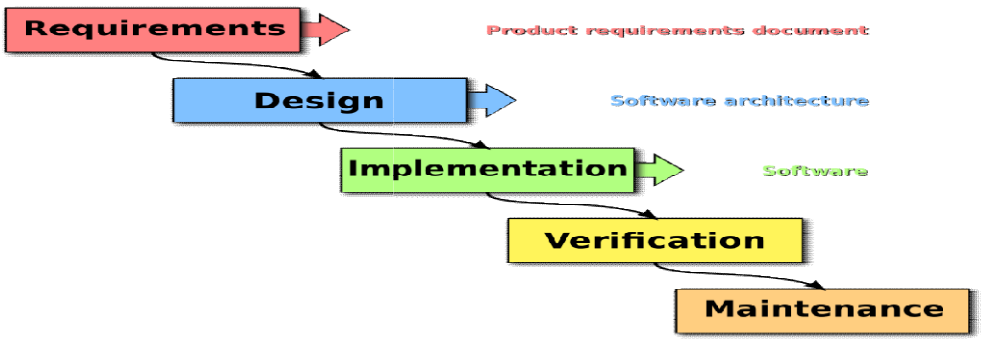


Fig 1.25 Waterfall model

Design thinking, on the other hand may be equally as expensive especially for organizations with legacy burdens switching over to an agile way of work, has picking up errors early built right into its DNA and offers opportunities to succeed quickly or fail fast.

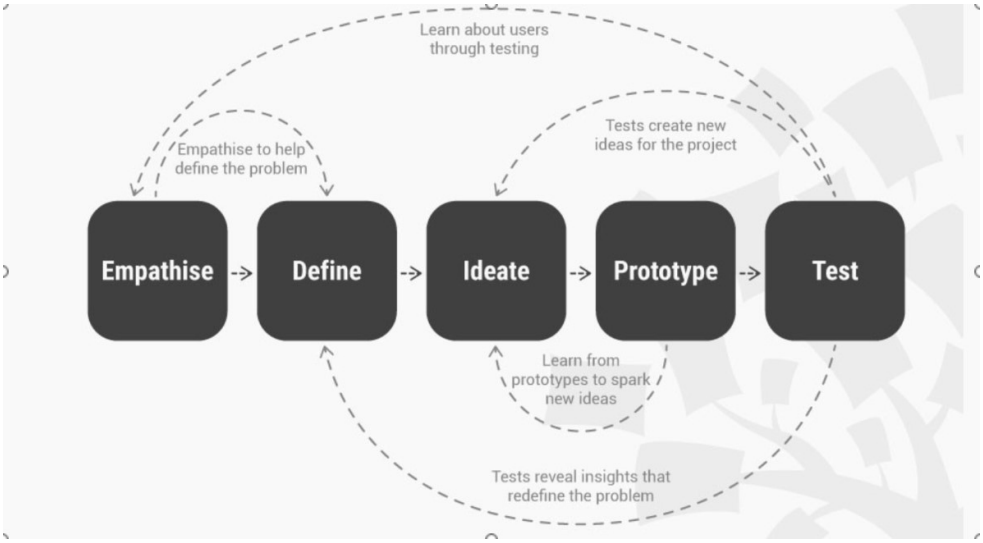


Fig 1.26 Design Thinking Techniques

A variation of the plan-do-check management methodology – the decision is easy to reach if one strips down the elements of the methodology into its bare bones fundamentals –design thinking allows the business or the delivery team to involve the customer for whom a product or service is being designed early on in the delivery effort. Design thinking offers up an opportunity to build a product truly needed by the customer and not one that was assumed the customer needs.

An important point of departure is that agility is a mindset, a mindset written into an organisations' DNA and not only those of its project delivery teams. Agility does not rest solely on the tools and techniques that an organisation or its teams chooses to work with, rather it relies heavily on the collective agreement between teams and individuals; the capacity to work quickly, to experiment all the time, to learn fast and to be honest with itself all the time.

That said, with an agile mindset, design thinking techniques can be applied within each phase of the waterfall methodology.

For example, in the requirements phase of a waterfall project, the elements of design thinking's empathise, define and ideate can readily be adopted. A bold team may and should also prototype and test their assumptions and some of the features of their product/service with real customers. Done well, the requirements phase of a waterfall project doesn't then just end with a signed off requirements specification, but also the knowledge of what the customer truly wants backed with empirical data. And potentially the design and

implementation phases have a ton of knowledge of the market to go on with – in essence most of the work in these phases of waterfall delivery would have been completed during the define stage as done as, Figure 1.27.

Waterfall Phase	Design Thinking Steps
Requirements – identify and document in acceptable language all the features that a new product/service or an existing product/service now requires to meet business' objectives	**Empathise** – ask and find out what the customer truly needs.
	Define – agree the customers' needs with the customers and determine the subset of these that your business or project that is most strategic for the project to delivery
	Ideate – work with actual customers to explore pain points, fixes for theses, and boundaries for these fixes. May help clarify the true needs of the customer and help separate these from their wants.
	Prototype – create illustrations which can vary between drawings on paper and an interactive mock-up of the product or service
	Test – observe the customers interact with the prototype. Question all assumptions and discard all that are determined to hold no relevance for the

Fig 1.27 Waterfall model Requirement Phase in Design Thinking steps

The actual waterfall phases of design and implementation then just becomes opportunities for refinements of the work done in the requirement phase using design thinking, and for honoring the legacy requirements of phase gates.

Waterfall Phase	Design Thinking Steps
Design – identify and document in acceptable language all the features that a new product/service or an existing product/service now requires to meet business' objectives	**Empathize** – understand customer pain points and overall objectives to delight
	Define – design approaches to take in helping meet the known customer needs
	Ideate – play with multiple ideas of the solution(s)
	Prototype – select with some of the users some of the design ideas that may work
	Test – test the ideas prototyped with actual customers to determine which to concretise and implement

Fig 1.28 Waterfall model Design Phase in Design Thinking steps

The refinement efforts in each of the design and implementation phases could also leverage design thinking. For example, the design phase of waterfall could leverage prototyping and test multiple user experience approaches – building on the work started during requirements.

And in the implementation phase will build based on the output from the design phase which has determined with a high degree of accuracy what the customer truly wants.

The test phase of the waterfall delivery effort, may end up being systems' tests (regression and systems integration tests) and confirmatory usability tests – as actual usability tests would have been completed upfront during requirements and design phases where the team had brought in the real customers to empathise with, define and ideate with, prototype for and tested with.

One can then safely conclude that waterfall can co-exist with design thinking (and really most other agile techniques) if the organisation's mindset focused on agility and not processes for the sake of processes.

V Model

The V-model is used to link every stage of the product through verification and validation. The V-Model is designed as a methodology of systems development, which makes the developing systems understandable. In systems engineering, it is used to define a uniform procedure for developing a product or project. Rather than moving down linearly (like the waterfall model), the process moves upwards after the coding phase to form a typical V shape, Figure 1.29.

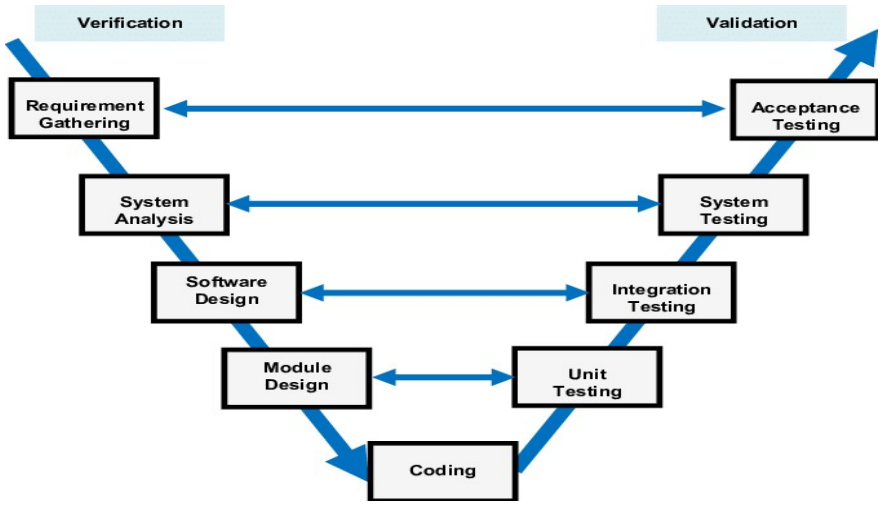


Fig 1.29 V-Model

TEXT /REFERENCE BOOKS

8. MaurícioVianna, YsmarVianna, Brenda Lucena and Beatriz Russo," Design thinking : Business innovation", MJV Technologies and innovation press, 2011.
9. Design Thinking: Integrating Innovation, Customer Experience, and Brand Valueby Thomas Lockwood (Editor) Published February 16th 2010 by Allworth Press.
10. KalloriVikram, —Introduction to DevOps, 1 st Edition, KalloriVikram Publication, 2016.
11. Jaokim Verona, —Practical DevOps, 2 nd Edition, Packt. Publication, 2018.
12. Stephen Fleming, Pravin, —DevOps Handbook: Introduction of DevOps Resource Management—,1st Edition, Createspace Independent Pub. , 2010.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – II – Design Thinking – SCSA1306

UNIT 2

INTRODUCTION TO AGILE

History of Agile – Agile principles – Agile Vs Waterfall – Agile Methodology Overview- Agile frameworks – Extreme programming - Rational Unified Process (RUP) - Test Driven Development (TDD) – Feature Drive Development (FDD)- Scrum - Kanban Methodology – Agile and Devops.

2.1 History of Agile

What is Agile?

Agile is the ability to create and respond to change. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment.

The authors of the Agile Manifesto chose "Agile" as the label for this whole idea because that word represented the adaptiveness and response to change which was so important to their approach.

It's really about thinking through how you can understand what's going on in the environment that you're in today, identify what uncertainty you're facing, and figure out how you can adapt to that as you go along.

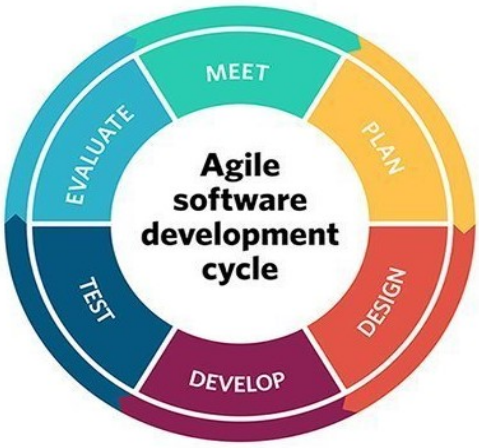


Fig 2.1 Agile software development cycle

Agile Manifesto

Manifesto for Agile Software Development, Figure 2.1

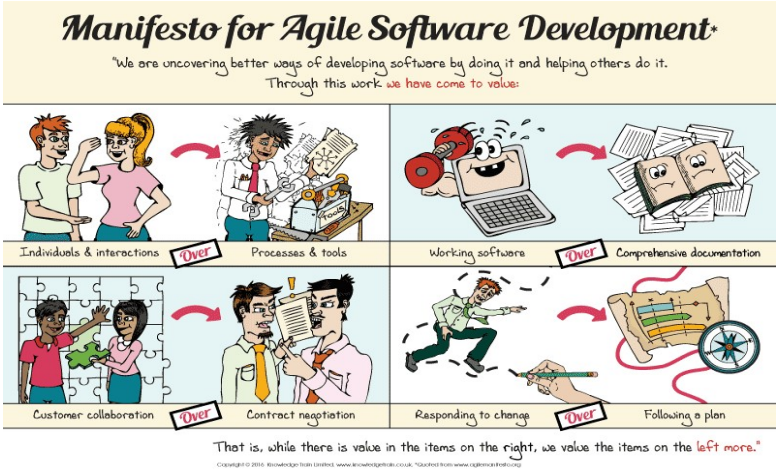


Fig 2.1 Agile Manifesto

Agile Principles

The 12 Agile Principles described in Figure 2.2

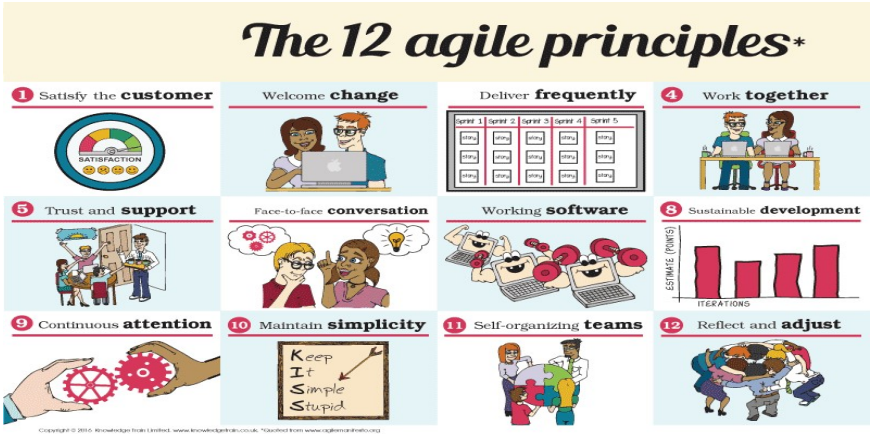


Fig 2.2 Agile Principles

What is Agile Software Development?

Agile software development is more than frameworks such as Scrum, Extreme Programming, or Feature-Driven Development (FDD).

Agile software development is more than practices such as pair programming, test-driven development, stand-ups, planning sessions, and sprints.

Agile software development is an umbrella term for a set of frameworks and practices based on the values and principles expressed in the Manifesto for Agile Software Development and the 12 Principles behind it. When you approach software development in a particular manner, it's generally good to live by these values and principles and use them to help figure out the right things to do given your particular context.

One thing that separates Agile from other approaches to software development is the focus on the people doing the work and how they work together. Solutions evolve through

collaboration between self-organizing cross-functional teams utilizing the appropriate practices for their context.

There's a big focus in the Agile software development community on collaboration and the self-organizing team.

That doesn't mean that there aren't managers. It means that teams have the ability to figure out how they're going to approach things on their own.

It means that those teams are cross-functional. Those teams don't have to have specific roles involved so much as that when you get the team together, you make sure that you have all the right skill sets on the team.

There still is a place for managers. Managers make sure team members have, or obtain, the right skill sets. Managers provide the environment that allows the team to be successful. Managers mostly step back and let their team figure out how they are going to deliver products, but they step in when the teams try but are unable to resolve issues.

When most teams and organizations start doing Agile development, they focus on the practices that help with collaboration and organizing the work, which is great. However, another key set of practices that are not as frequently followed but should be are specific technical practices that directly deal with developing software in a way that helps your team deal with uncertainty. Those technical practices are essential and something you shouldn't overlook

2.3 Agile Methodology Overview

What is Agile Methodology?

The Agile Methodology is an advanced approach to develop software with **flexibility and speed**. The methodology has introduced new concepts of iterative and incremental development methods to ensure foolproof and accelerated delivery.

Traditional methods like Waterfall used to dominate the industry, but not anymore. The reason for **Waterfall methods** to start fading is the absence of no backward movement.

There is no room for adding new features, and the maintenance is also very challenging.
Mike Cohn, the Contributor of the Scrum Software Development Method, said:

The worst thing you can deliver in a two-year project is what the customer wanted on day one.

The Agile methodology of software development, Figure 2.4 allows backward tracking and enables the development team to build a broader feature set in time-boxed cycles.
It follows a typical development process from requirement gatherings to design, development, testing, deployment, and maintenance. However, the strategy in the Agile methodology changes at each stage.

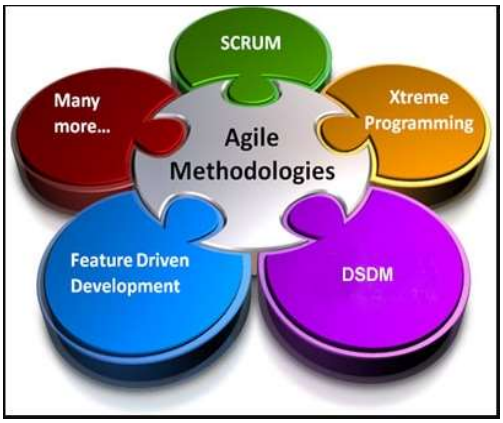
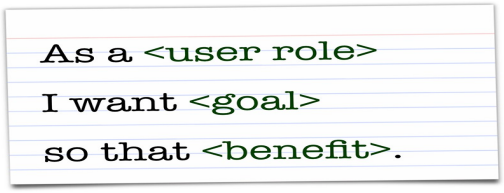


Fig 2.4 Agile Methodologies

User Story



2.2 Agile vs. Waterfall Comparison

Many development companies are moving towards Agile methodology and moving away from the predictive, Waterfall methodology when developing software

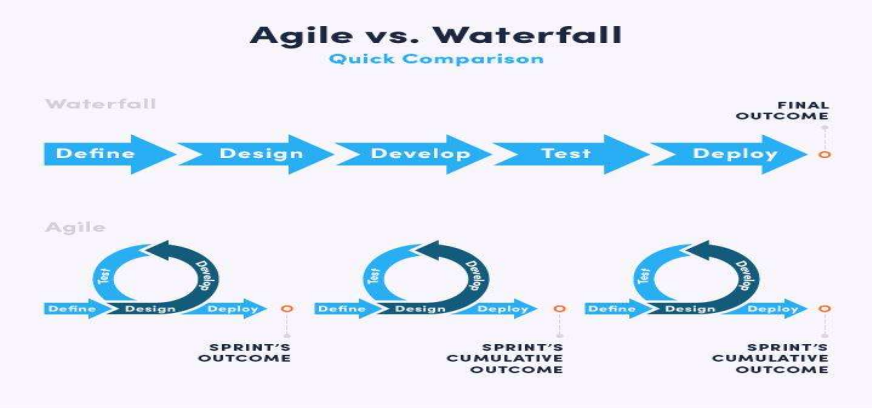


Fig 2.5 Agile Vs Waterfall

Roles & Responsibilities in Agile Software

The Agile software development lifecycle (SDLC) is developed with a clear goal of rapid delivery of the software through an incremental and iterative process designed to adapt and improve software quality from the client's perspective. Agile teams are comprised of the following key roles and responsibilities:

Fig 2.6 Agile Team Roles

Product Owner

The Product Owner represents the Stakeholder of the project. He/she understands the project from the Stakeholder's perspective. The crucial role of a Product Owner is to set the direction for product development or project progress.

Key Responsibilities for a Product Owner:

- Product backlog management
- Release management
- Stakeholder management

Team Lead or Scrum Master

The Team Lead or the Scrum Master is the main bridge between the team members and the Product Owner. The Team Lead ? Scrum Master takes instructions from the Product Owner and supports the project's progress between individual team members.

Key Responsibilities for Team Lead/Scrum Master

- Arranging and managing the daily Scrum and Sprint initiatives
- Coordinating between team members about changing requirements
- Motivating team members for delivering results
- Managing tasks such as conducting meetings, facilitating collaboration, and eliminating hurdles affecting project progress

- Keeping team members protected from external interferences and distractions

This role also plays extra responsibilities, such as:

Implementing changes

- Coordinating between stakeholders to find necessary resources
- Helping Product Owners optimize the backlog planning for maximum performance

More about the role and responsibilities of the Scrum Master, you can read in my article *Why do you need a Scrum Master? A report from the battlefield.*

Development Team Members

Team members within the Development Team comprise experienced individuals who are directly or indirectly linked to product development. The team is a combination of different roles responsible for taking an idea or requirements and transforming it into a tangible product for end-users.

Key Roles of Team Members

1. Product designer
2. Programmer
3. Tester
4. UX/UI specialist

Key Responsibility for Development Team

- Delivering the work throughout the Sprints according to the Product Owner's requirements,
- Taking items from Product Backlog to the Sprint Backlog and committing to deliver the value during the Sprint,
- Self-organizing in order to fulfil their tasks.

Stakeholders

The Stakeholder may not be directly involved in the development process, but they play key roles, and their decision can impact the Scrum team's work. A stakeholder can be:

- The end-user of the product (Client)
- Business Executives (Employees of the company for whom the Software is under development)
- Production Support staff (Testing Staff)

- Investors
- External Auditors (Appointed by the Client)
- Scrum Team Members from Associated Projects and Teams

Additional Roles for Larger Scrum Projects

If there is a larger product, it needs some additional roles. Here're the details of these additional roles.

- Technical domain experts with the knowledge of technology and a wide variety of stakeholder requirements or expectations.
- An independent testing and audit team may join the Scrum team members and work throughout the product development lifecycle.
- An Integrator may be required among large teams that work on independent but closely coordinated subsystems for a project.
- An Architect Owner may be required for architectural envisioning, planning, and decision making.

2.4 Agile Frameworks

Sprint

a set period of time during which specific tasks must be completed, Figure 2.3.

"team members discuss issues with each other at the end of every sprint"

Fig 2.7 Sprint

Scrum is a framework that helps teams work together. Often thought of as an agile project management framework, **scrum** describes a set of meetings, tools, and roles that work in concert to help teams structure and manage their work.

2.5 Extreme Programming (XP)

Definition

Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.

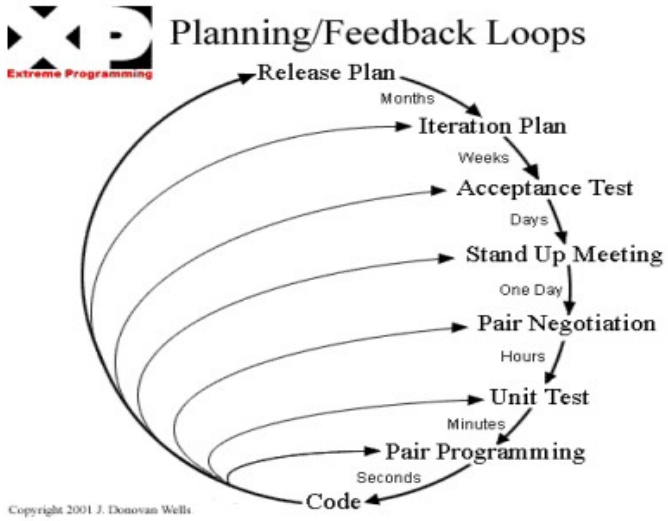


Fig 2.8 Extreme Programming

When Applicable

The general characteristics where XP is appropriate were described by Don Wells on www.extremeprogramming.org:

- Dynamically changing software requirements
- Risks caused by fixed time projects using new technology
- Small, co-located extended development team
- The technology you are using allows for automated unit and functional tests

Due to XP's specificity when it comes to it's full set of software engineering practices, there are several situations where you may not want to fully practice XP. The post When is XP Not Appropriate on the C2 Wiki is probably a good place to start to find examples where you may not want to use XP.

While you can't use the entire XP framework in many situations, that shouldn't stop you from using as many of the practices as possible given your context.

Values

The five values of XP are communication, simplicity, feedback, courage, and respect and are described in more detail below.

Communication

Software development is inherently a team sport that relies on communication to transfer knowledge from one team member to everyone else on the team. XP stresses the importance of the appropriate kind of communication – face to face discussion with the aid of a white board or other drawing mechanism.

Simplicity

Simplicity means "what is the simplest thing that will work?" The purpose of this is to avoid waste and do only absolutely necessary things such as keep the design of the system as simple as possible so that it is easier to maintain, support, and revise. Simplicity also means address only the requirements that you know about; don't try to predict the future.

Feedback

Through constant feedback about their previous efforts, teams can identify areas for improvement and revise their practices. Feedback also supports simple design. Your team builds something, gathers feedback on your design and implementation, and then adjust your product going forward.

Courage

Kent Beck defined courage as "effective action in the face of fear" (Extreme Programming Explained P. 20). This definition shows a preference for action based on other principles so that the results aren't harmful to the team. You need courage to raise organizational issues that reduce your team's effectiveness. You need courage to stop doing something that doesn't work and try something else. You need courage to accept and act on feedback, even when it's difficult to accept.

Respect

The members of your team need to respect each other in order to communicate with each other, provide and accept feedback that honors your relationship, and to work together to identify simple designs and solutions.

Practices

The core of XP is the interconnected set of software development practices listed below. While it is possible to do these practices in isolation, many teams have found some practices reinforce the others and should be done in conjunction to fully eliminate the risks you often face in software development.

The XP Practices have changed a bit since they were initially introduced.The original twelve practices are listed below. If you would like more information about how these practices were originally described, you can visit http://ronjeffries.com/xprog/what-is-extreme-programming/.

- The Planning Game
- Small Releases
- Metaphor
- Simple Design
- Testing
- Refactoring
- Pair Programming
- Collective Ownership
- Continuous Integration
- 40-hour week
- On-site Customer
- Coding Standard

Below are the descriptions of the practices as described in the second edition of Extreme Programming Explained Embrace Change. These descriptions include refinements based on experiences of many who practice extreme programming and reflect a more practical set of practices.

Sit Together

Since communication is one of the five values of XP, and most people agree that face to face conversation is the best form of communication, have your team sit together in the same space without barriers to communication, such as cubicle walls.

Whole Team

A cross functional group of people with the necessary roles for a product form a single team. This means people with a need as well as all the people who play some part in satisfying that need all work together on a daily basis to accomplish a specific outcome.

Informative Workspace

Set up your team space to facilitate face to face communication, allow people to have some privacy when they need it, and make the work of the team transparent to each other and to interested parties outside the team. Utilize Information Radiators to actively communicate up-to-date information.

Energized Work

You are most effective at software development and all knowledge work when you are focused and free from distractions.

Energized work means taking steps to make sure you are able physically and mentally to get into a focused state. This means do not overwork yourself (or let others overwork you). It also means stay healthy, and show respect to your teammates to keep them healthy.

Pair Programming

Pair Programming means all production software is developed by two people sitting at the same machine. The idea behind this practice is that two brains and four eyes are better than one brain and two eyes. You effectively get a continuous code review and quicker response to nagging problems that may stop one person dead in their tracks.
Teams that have used pair programming have found that it improves quality and does not actually take twice as long because they are able to work through problems quicker and they stay more focused on the task at hand, thereby creating less code to accomplish the same thing.

Stories

Describe what the product should do in terms meaningful to customers and users. These stories are intended to be short descriptions of things users want to be able to do with the product that can be used for planning and serve as reminders for more detailed conversations when the team gets around to realizing that particular story.

Weekly Cycle

The Weekly Cycle is synonymous to an iteration. In the case of XP, the team meets on the first day of the week to reflect on progress to date, the customer picks the stories they would like delivered in that week, and the team determines how they will approach those stories. The goal by the end of the week is to have running tested features that realize the selected stories.
The intent behind the time boxed delivery period is to produce something to show to the customer for feedback.

Quarterly Cycle

The Quarterly Cycle is synonymous to a release. The purpose is to keep the detailed work of each weekly cycle in context of the overall project. The customer lays out the overall plan for the team in terms of features desired within a particular quarter, which provides the team with a view of the forest while they are in the trees, and it also helps the customer to work with other stakeholders who may need some idea of when features will be available.

Remember when planning a quarterly cycle the information about any particular story is at a relatively high level, the order of story delivery within a Quarterly Cycle can change and the stories included in the Quarterly Cycle may change. If you are able to revisit the plan on a weekly basis following each weekly cycle, you can keep everyone informed as soon as those changes become apparent to keep surprises to a minimum.

Slack

The idea behind slack in XP terms is to add some low priority tasks or stories in your weekly and quarterly cycles that can be dropped if the team gets behind on more important tasks or stories. Put another way, account for the inherent variability in estimates to make sure you leave yourself a good chance of meeting your forecasts.

Ten-Minute Build

The goal with the Ten-Minute Build is to automatically build the whole system and run all of the tests in ten minutes. The founders of XP suggested a 10 minute time frame because if a team has a build that takes longer than that, it is less likely to be run on a frequent basis, thus introducing longer time between errors.
This practice encourages your team to automate your build process so that you are more likely to do it on a regular basis and to use that automated build process to run all of your tests.

This practice supports the practice of Continuous Integration and is supported by the practice of Test First Development.

Continuous Integration

Continuous Integration is a practice where code changes are immediately tested when they are added to a larger code base. The benefit of this practice is you can catch and fix integration issues sooner.
Most teams dread the code integration step because of the inherent discovery of conflicts and issues that result. Most teams take the approach "If it hurts, avoid it as long as possible".

Practitioners of XP suggest "if it hurts, do it more often".

The reasoning behind that approach is that if you experience problems every time you integrate code, and it takes a while to find where the problems are, perhaps you should integrate more often so that if there are problems, they are much easier to find because there are fewer changes incorporated into the build.

This practice requires some extra discipline and is highly dependent on Ten Minute Build and Test First Development.

Test-First Programming

Instead of following the normal path of:

develop code -> write tests -> run tests

The practice of Test-First Programming follows the path of:
Write failing automated test -> Run failing test -> develop code to make test pass -> run test -> repeat

As with Continuous Integration, Test-First Programming reduces the feedback cycle for developers to identify and resolve issues, thereby decreasing the number of bugs that get introduced into production.

Incremental Design

The practice of Incremental Design suggests that you do a little bit of work up front to understand the proper breadth-wise perspective of the system design, and then dive into the details of a particular aspect of that design when you deliver specific features. This approach

reduces the cost of changes and allows you to make design decisions when necessary based on the most current information available.
The practice of Refactoring was originally listed among the 12 core, but was incorporated into the practice of Incremental Design. Refactoring is an excellent practice to use to keep the design simple, and one of the most recommended uses of refactoring is to remove duplication of processes.

Roles

Although Extreme Programming specifies particular practices for your team to follow, it does not really establish specific roles for the people on your team.

Depending on which source you read, there is either no guidance, or there is a description of how roles typically found in more traditional projects behave on Extreme Programming projects. Here are four most common roles associated with Extreme Programming:

The Customer

The Customer role is responsible for making all of the business decisions regarding the project including:

- What should the system do (What features are included and what do they accomplish)?
- How do we know when the system is done (what are our acceptance criteria)?
- How much do we have to spend (what is the available funding, what is the business case)?
- What should we do next (in what order do we deliver these features)?

The XP Customer is expected to be actively engaged on the project and ideally becomes part of the team.

The XP Customer is assumed to be a single person, however experience has shown that one person cannot adequately provide all of the business related information about a project. Your team needs to make sure that you get a complete picture of the business perspective, but have some means of dealing with conflicts in that information so that you can get clear direction.

The Developer

Because XP does not have much need for role definition, everyone on the team (with the exception of the customer and a couple of secondary roles listed below) is labeled a developer. Developers are responsible for realizing the stories identified by the Customer. Because different projects require a different mix of skills, and because the XP method relies on a cross functional team providing the appropriate mix of skills, the creators of XP felt no need for further role definition.

The Tracker

Some teams may have a tracker as part of their team. This is often one of the developers who spends part of their time each week filling this extra role. The main purpose of this role is to keep track of relevant metrics that the team feels necessary to track their progress and to identify areas for improvement. Key metrics that your team may track include velocity, reasons for changes to velocity, amount of overtime worked, and passing and failing tests.

This is not a required role for your team, and is generally only established if your team determines a true need for keeping track of several metrics.

The Coach

If your team is just getting started applying XP, you may find it helpful to include a Coach on your team. This is usually an outside consultant or someone from elsewhere in your organization who has used XP before and is included in your team to help mentor the other team members on the XP Practices and to help your team maintain your self discipline.

The main value of the coach is that they have gone through it before and can help your team avoid mistakes that most new teams make.

Lifecycle

To describe XP in terms of a lifecycle it is probably most appropriate to revisit the concept of the **Weekly Cycle and Quarterly Cycle**.

First, start off by describing the desired results of the project by having customers define a set of stories. As these stories are being created, the team estimates the size of each story. This size estimate, along with relative benefit as estimated by the customer can provide an indication of relative value which the customer can use to determine priority of the stories.

If the team identifies some stories that they are unable to estimate because they don't understand all of the technical considerations involved, they can introduce a spike to do some focused research on that particular story or a common aspect of multiple stories. Spikes are short, time-boxed time frames set aside for the purposes of doing research on a particular aspect of the project. Spikes can occur before regular iterations start or alongside ongoing iterations.

Next, the entire team gets together to create a release plan that everyone feels is reasonable. This release plan is a first pass at what stories will be delivered in a particular quarter, or release. The stories delivered should be based on what value they provide and considerations about how various stories support each other.

Then the team launches into a series of weekly cycles. At the beginning of each weekly cycle, the team (including the customer) gets together to decide which stories will be realized during that week. The team then breaks those stories into tasks to be completed within that week. At the end of the week, the team and customer review progress to date and the customer can decide whether the project should continue, or if sufficient value has been delivered.

Origins

XP was first used on the Chrysler Comprehensive Compensation (C3) program which was initiated in the mid 90's and switched to an XP project when Kent Beck was brought on to the project to improve the performance of the system. He wound up adding a couple of other folks, including Ron Jeffries to the team and changing the way the team approached development. This project helped to bring the XP methodology into focus and the several books written by people who were on the project helped spread knowledge about and adaptation of this approach.

Primary Contributions

XP's primary contribution to the software development world is an interdependent collection of engineering practices that teams can use to be more effective and produce higher quality code. Many teams adopting agile start by using a different framework and when they identify the need for more disciplined engineering practices they adopt several if not all of the engineering practices espoused by XP.

An additional, and equally important, contribution of XP is the focus on practice excellence. The method prescribes a small number of absolutely essential practices and encourages teams to perform those practices as good as they possibly can, almost to the extreme. This is where the name comes from. Not because the practices themselves are necessarily radical (although some consider some of them pretty far out) rather that teams continuously focus so intently on continuously improving their ability to perform those few practices.

2.6 Rational Unified Process (RUP)

Rational Unified Process (RUP) is a software development process for object-oriented models. It is also known as the Unified Process Model. It is created by Rational corporation and is designed and documented using UML (Unified Modeling Language). This process is included in IBM Rational Method Composer (RMC) product. IBM (International Business Machine Corporation) allows us to customize, design, and personalize the unified process.
RUP is proposed by Ivar Jacobson, Grady Bootch, and James Rambaugh. Some characteristics of RUP include use-case driven, Iterative (repetition of the process), and Incremental (increase in value) by nature, delivered online using web technology, can be customized or tailored in modular and electronic form, etc. RUP ,Figure 2.9 reduces unexpected development costs and prevents wastage of resources.

1. **Inception –**

- Communication and planning are main.
- Identifies Scope of the project using use-case model allowing managers to estimate costs and time required.
- Customers requirements are identified and then it becomes easy to make a plan of the project.
- Project plan, Project goal, risks, use-case model, Project description, are made.
- Project is checked against the milestone criteria and if it couldn't pass these criteria then project can be either cancelled or redesigned.

2. **Elaboration –**

- Planning and modeling are main.
- Detailed evaluation, development plan is carried out and diminish the risks.
- Revise or redefine use-case model (approx. 80%), business case, risks.
- Again, checked against milestone criteria and if it couldn't pass these criteria then again project can be cancelled or redesigned.
- Executable architecture baseline

3. **Construction –**

- Project is developed and completed.
- System or source code is created and then testing is done.
- Coding takes place.

4. **Transition –**

- Final project is released to public.
- Transit the project from development into production.
- Update project documentation.
- Beta testing is conducted.
- Defects are removed from project based on feedback from public.

5. **Production –**

- Final phase of the model.
- Project is maintained and updated accordingly.

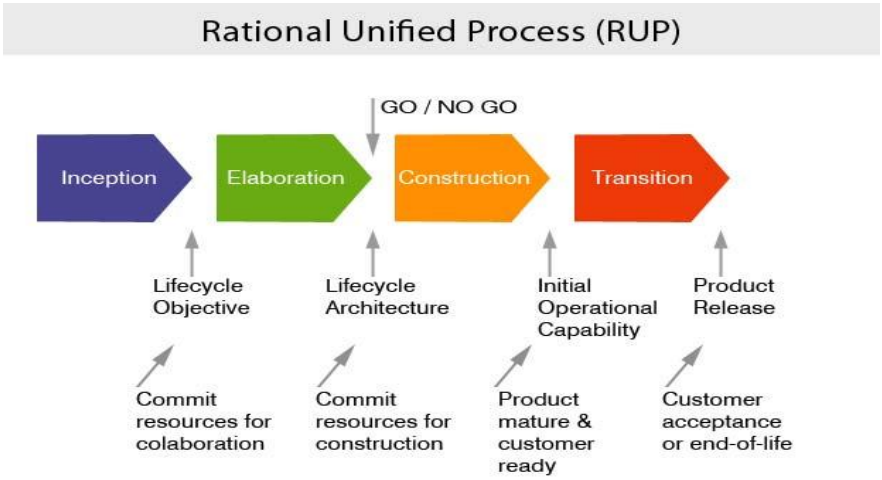


Fig 2.9 Rational Unified Process

2.7 Test Driven Development (TDD)

In layman's terms, Test Driven Development (TDD), Figure 2.10 is a software development practice that focuses on creating unit test cases before developing the actual code. It is an iterative approach that combines programming, the creation of unit tests, and refactoring.

The TDD approach derives its roots from the Agile manifesto principles and Extreme programming. As the name suggests, the test process drives software development. Moreover, it's a structuring practice that enables developers and testers to obtain optimized code that proves to be resilient in the long term.

In TDD, developers start creating small test cases for every feature based on their initial understanding. The primary intention of this technique is to modify or write new code only if the tests fail. This prevents duplication of test scripts.

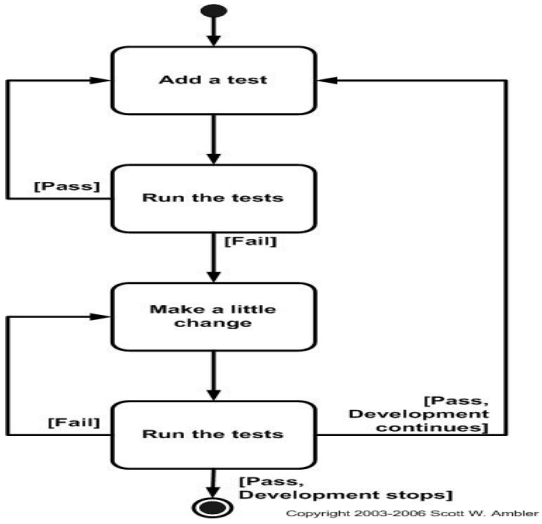


Fig 2.10 Test Driven Development

Three phases of Test Driven Development

1. **Create precise tests:** Developers need to create precise unit tests to verify the functionality of specific features. They must ensure that the test compiles so that it can execute. In most cases, the test is bound to fail. This is a meaningful failure as developers are creating compact tests based on their assumptions of how the feature will behave.
2. **Correcting the Code:** Once a test fails, developers need to make the minimal changes required to correct the code so that it can run successfully when re-executed.
3. **Refactor the Code:** Once the test runs successfully, check for redundancy or any possible code optimizations to enhance overall performance. Ensure that refactoring does not affect the external behavior of the program.

Benefits of Test Driven Development (TDD)

1. Fosters the creation of optimized code.
2. Helps developers better analyze and understand client requirements and request clarity when they are not adequately defined.

3. The addition and testing of new functionalities become much easier in the latter stages of development.
4. Test coverage under TDD is much higher compared to the conventional development models. This is because the TDD focuses on creating tests for each functionality right from the beginning.
5. Enhances the productivity of the developer and leads to the development of a codebase that is flexible and easy to maintain.

Frameworks for Test Driven Development

Based on unique programming languages, there are multiple frameworks that support test driven development. Listed below are a few popular ones.

1. **csUnit and NUnit** – Both are open source unit testing frameworks for .NET projects.
2. **PyUnit and DocTest**: Popular Unit testing framework for Python.
3. **Junit:** Widely used unit testing tool for Java
4. **TestNG**: Another popular Java testing framework. This framework overcomes the limitations of Junit.
5. **Rspec**: A testing framework for Ruby projects

2.8 Feature Driven Development (FDD)

Feature Driven Development (FDD) is an agile framework that, as its name suggests, organizes software development around making progress on features. Features in the FDD context, though, are not necessarily product features in the commonly understood sense. They are, rather, more akin to user stories in Scrum. In other words, "complete the login process" might be considered a feature in the Feature Driven Development (FDD) methodology.

An Agile methodology for developing software, Feature-Driven Development (FDD) is customer-centric, iterative, and incremental, with the goal of delivering tangible software results often and efficiently. FDD in Agile encourages status reporting at all levels, which helps to track progress and results.

FDD allows teams to update the project regularly and identify errors quickly. Plus, clients can be provided with information and substantial results at any time. FDD is a favorite method among development teams because it helps reduce two known morale-killers in the development world: Confusion and rework./p>

First applied in 1997 during a project for a Singapore bank, FDD was developed and refined by Jeff De Luca, Peter Coad and others. The original project took 15 months with 50 people, and it worked; it was followed by a second, 18-month long, 250-person project./p>

Since then, it's become a pragmatic approach ideal for long-term, complex projects looking for a simple but comprehensive methodology. While Scrum and new variations of Agile

are more widely recognized methods (especially outside of software development), FDD can be a good option for software development teams looking for a structured, focused Agile methodology that can be scaled across the product organization and will deliver clear outcomes.

History of Feature Driven Development

The first real-world application of the Feature Driven Development methodology was on a 50-person software-development project for a Singapore-based financial institution, and the first public discussion of the methodology was in the 1999 book Java Modeling in Color with UML.

FDD was designed to follow a five-step development process, built largely around discrete "feature" projects. That project lifecycle looks like this:

1. Develop an overall model
2. Build a features list
3. Plan by feature
4. Design by feature
5. Build by feature

The framework Figure 2.11 has since gained widespread use particularly in larger organizations, and today there is a thriving Feature Driven Development community with its own website.

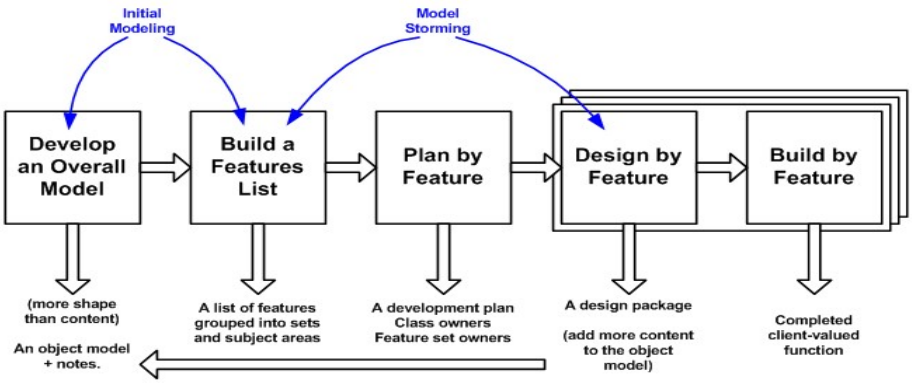


Fig2.11 Feature Driven Development

An FDD project starts by performing the first three steps in the equivalent of the DAD's Inception phase or XP's "iteration 0", the goal being to identify the scope of the effort, the initial architecture, and the initial high-level plan. Construction efforts occur in two-week (or less) iterations, similar to XP or DAD teams, with the team iteratively working through all five steps as needed. As with other agile software development processes, systems are delivered incrementally by FDD teams.

There are six primary roles on an FDD project: Project Manager, Chief Architect, Development Manager, Chief Programmer, Class Owner, and Domain Expert. An individual will take on one or more roles on a project as you would expect. The concept of a class owner is where FDD differs from XP. XP includes a practice called Collective Ownership the idea of which is that any developer can update any artifact, including source code, as required. FDD takes a different approach in that it assigns classes to individual developers, so if a feature requires changes to several classes then the owners of those classes must work together as a feature team to implement it. Just like programming pairs will model storm to think something through before they code it, so will feature teams.

FDD also defines a collection of supporting roles, including:

- Domain Manager
- Release Manager
- Language Guru
- Build Engineer
- Toolsmith
- System Administrator
- Tester
- Deployer
- Technical Writer

FDD's five steps are supported by several practices. The first is domain object modeling, the creation of a high-level class diagram and supporting artifacts that describes the problem domain. Developing by feature and individual class ownership are also good practices, as is having developers work together in feature teams. Inspections are an important aspect of FDD. FDD also insists on regular builds, similar to XP, and configuration management. Finally, FDD promotes a best practice called reporting/visibility of results, similar to XP and AM's philosophy of open and honest communication.

How would Agile Modeling (AM) be applied on an FDD project?

The principles and practices can be clearly applied to FDD's two modeling-oriented steps - develop an overall model and design by feature. The only apparent mismatch between the two processes is FDD's practice of class ownership and AM's practice of collective ownership, but I would argue that this isn't the case. FDD's practice pertains to coding but does not to modeling, on a FDD project people work together in teams to model, along the lines of AM's model with others practice, and therefore several people will be working on your shared collection of modeling artifacts.

How is FDD Different from Scrum?

FDD is related to Scrum, but as its name implies, it's a feature-focused method (as opposed to a delivery-focused method). Features are a foundational piece of FDD; they're to FDD what user stories are to Scrum: Small functions that are, most importantly, client-valued.

"During FDD, a feature should be delivered every 2-10 days – which differs from Scrum, in which sprints typically last two, but sometimes four, weeks."

FDD values documentation more than other methods (Scrum and XP included), which also creates differences in the roles of meetings. In Scrum, teams typically meet on a daily basis; in FDD, teams rely on documentation to communicate important information, and thus don't usually meet as frequently.

Another key difference is the end user; in FDD, the actual user is viewed as the end user, whereas in Scrum it's typically the Product Owner who is seen as the end user.

How Does FDD Work?

Typically used in large-scale development projects, five basic activities exist during FDD:

- Develop overall model
- Build feature list
- Plan by feature
- Design by feature
- Build by feature

An overall model shape is formed during the first two steps, while the final three are repeated for each feature. The majority (roughly 75%) of effort during FDD will be spent on the fourth and fifth steps – Design by Feature and Build by Feature.

Teams using all Agile methodologies operate with the primary goal of quickly and effectively satisfying the needs of their customers; FDD is no exception.

However, the difference is that once a goal has been identified, teams following FDD organize their activities by features, rather than by project milestones or other indicators of progress.

Characteristics of FDD

- **Short iterative:** FDD lifecycle works in simple and short iterations to efficiently finish the work on time and gives good pace for large projects.

- **Customer focused:** This agile practice is totally based on inspection of each feature by client and then pushed to main build code.

- **Structured and feature focused:** Initial activities in lifecycle builds the domain model and features list in the beginning of timeline and more than 70% of efforts are given to last 2 activities.

- **Frequent releases:** Feature-driven development provides continuous releases of features in the software and retaining continuous success of the project.

Strengths and Weakness of Feature Driven Development

FDD's strengths include:

- Simple five-step process allows for more rapid development
- Allows larger teams to move products forward with continuous success
- Leverages pre-defined development standards, so teams are able to move quickly

FDD's weaknesses include:

- Does not work efficiently for smaller projects
- Less written documentation, which can lead to confusion
- Highly dependent on lead developers or programmers

Stages of Feature-Driven Development

Stage 0: Gather Data

As with all Agile methodologies, the first step in FDD is to gain an accurate understanding of content and context of the project, and to develop a clear, shared understanding of the target audience and their needs. During this time, teams should aim to learn everything they can about the why, the what, and the for whom about the project they're about to begin (the next few steps will help clarify the how). This data-gathering can be thought of as stage 0, but one that cannot be skipped. To compare product development with writing a research paper, this is the research and thesis development step.

Once teams have a clear understanding of their goals, the targeted audience and their current (and potentially, future) needs, the first named stage in FDD can begin: Developing an Overall Model.

Develop an overall model

Continuing the research paper metaphor, this stage is when the outline is drafted. Using the "thesis" (aka primary goal) as a guide, the team will develop detailed domain models, which will then be merged into one overall model that acts as a rough outline of the system. As it develops and as the team learns, details will be added.

Build a features list

Use the information assembled in the first step to create a list of the required features. Remember, a feature is a client-valued output. Make a list of features (that can be completed in two weeks' time), and keep in mind that these features should be purposes or smaller goals, rather than tasks.

Plan by Feature

Enter: Tasks. Analyze the complexity of each feature and plan tasks that are related for team members to accomplish. During the planning stage, all members of the team should take part in the evaluation of features with the perspective of each development stage in mind. Then, use the assessment of complexity to determine the order in which each feature will implemented, as well as the team members that will be assigned to each feature set.

This stage should also identify class owners, individual developers who are assigned to classes. Because every class of the developing feature belongs to a specific developer, someone is responsible for the conceptual principles of that class, and should changes be required to multiple classes, then collaboration is necessary between the owners of each to implement them.

And while class owners are important to FDD, so are feature teams. In feature teams, specific roles are defined, and a variety of viewpoints are encouraged. This ensures that design decisions consider multiple thoughts and perspectives.

“Kanban provides a shared, single source of truth that enables teams to not only manage support tickets more effectively, but also add infinitely more value to the product organization.”

Design by Feature

A chief programmer will determine the feature that will be designed and build. He or she will also determine the class owners and feature teams involved, while defining the feature priorities. Part of the group might be working on technical design, while others work on framework. By the end of the design stage, a design review is completed by the whole team before moving forward.

Build by Feature

This step implements all the necessary items that will support the design. Here, user interfaces are built, as are components detailed in the technical design, and a feature prototype is created. The unit is tested, inspected and approved, then the completed feature can be promoted to the main build. Any feature that requires longer time than two weeks to design and build is further broken into features until it meets the two-week rule.

2.9 Agile Development Models: Scrum & Kanban

There are many methods that Agile teams use to develop software. But the most prominent methods are two:

1. Scrum Methodology
2. Kanban Methodology

Scrum Methodology

Agile Scrum Methodology is the **most prominent and simplest method**. According to the State of Agile Report, a whopping 56% of teams use the Scrum methodology. It is a framework for developing and sustaining complex products. Scrum definition and the idea behind is contained in the Scrum Guide. In my article Scrum Guide 2020 Update. Key Changes and Their Impact on Development Teams you can find the latest changes in the Scrum Guide which are important to the development process, Figure 2.12.

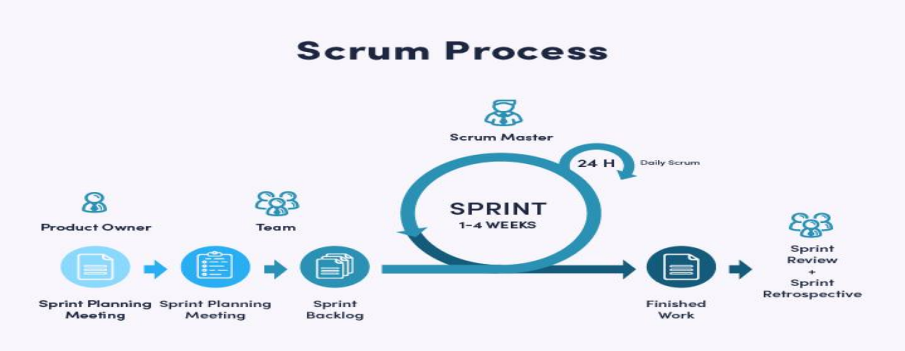


Fig 2.12 Scrum Process

It works on these bases:

- The Scrum method revolves around the responsibilities of the Scrum Master, who supports the Product Owner to identify and prioritize goals. The details of the goals and the set-of-features are then added to what's called a "Product Backlog."
- The cross-functional team reviews the backlog and decides the Sprints to deliver "potentially shippable increments" of software. A Sprint is up to 4 weeks.
- There are daily standup meetings for clear communication between the team members to stay up-to-date with development progress.
- After every Sprint ends, the team analyzes the backlog and gathers for a demo meeting to present the functionality to the Product Owner and his stakeholders.
- The last step of the Agile Scrum method is a retrospective meeting. In this meeting, the team discusses the parts of their process which need improvements.

Scrum Standup

- 15 minutes, same time every day
- 3 questions:
  - What did I accomplish yesterday?
  - What do I plan to accomplish today?
  - What are my impediments?
- Not a status meeting
- If further discussion is needed, takes place immediately after stand-up
- Distributed team challenges

Fig 2.13 scrum Stand up

Kanban Methodology

Kanban is also a popular method among development teams. This method focuses more on continuous delivery where things are kept simple, so the team doesn't get overstressed, Figure 2.14.

Fig 2.14 Kanban Methodology

It works this way:

- The teams **visualize** their workflow with the help of Kanban boards. These Kanban boards use cards and columns to help teams make their work visible, show it to others, and complete the job. The team can also use software like Jira or Atlassian. The software offers a "board" style interface where teams can see their "to-do" or "in progress" tasks.
- The best part about the Kanban approach is that the team knows **how much they can or will do** in a sprint, and nothing will be added more. This helps balance the workflow, and teams stay motivated without being burnout.
- In the Kanban approach, the team **knows exactly what will come next**, so they have half the preparation of the tasks in advance. This helps keep the priorities fulfilled.

Comparison between scrum and kanban depicts below Figure 2.15

Scrum	Kanban
Team involved in a specific iteration	Optional involvement
Uses speed (velocity) as a measure for improving processes	Uses deadline / lead time as a measure for improving processes
Prescribed estimations	Optional estimations
One sprint backlog belongs to a team	Kanban-board may be shared
Involves using at least three roles (Product Owner / Scrum Master / Scrum Team)	Does not use roles
The Scrum-board is reset between sprints	The Kanban-board does not change, it is persistent
For each sprint, priorities are established on the sprint backlog	Establishing priorities is optional

Fig 2.15 Comparison between scrum and kanban

2.10 Agile and Devops

- DevOps is the change in IT culture.
- It focuses on rapid IT service delivery through the adoption of agile and lean practices in the context of a system-oriented approach according to Gartner.
- An amalgamation of two words, 'development' and 'operations' it aims at combining software development and software operations.
- It breaks the barrier between development and operation teams.
- The collaborative work between them leads to the benefit of combined skill.

What Is DevOps?

- The primary foundation of DevOps is a method that at its core, focusses on collaborating and standardizing the communication of IT professionals. This allows for products to be distributed in the most timeous manner possible.
- At the heart of the DevOps culture is the partnership between Development and Operations teams. The collaboration of these teams enables code deployment and production to take place in a more rapid and automated fashion. It's important to remember that previously these teams operated in silos from the initial to completion stages.
- In an organization, DevOps should not be seen as a title but rather as a responsibility that they undertake, which allows for a smoother flow of processes.

Three ways define the principles of DevOps. These are:

- Principles of flow
- Principles of feedback
- Principles of continuous learning
- The above three principles focus on how to fast track how work can flow during every stage of the process. It looks at the regularity and speed of feedback that is consistent throughout the cycle of development. There needs to be continuous learning and analysis as an entrenched part of an organization.
- The mindset of DevOps requires the inclusion of a shared culture and flexibility. It is centered around IT and software development.
- The performance of software delivery plays an important role because it affects the mission of the organization, profit margins, and customer satisfaction. Organizational culture is an important aspect because this ultimately affects performance.

TEXT /REFERENCE BOOKS

1. MaurícioVianna, YsmarVianna, Brenda Lucena and Beatriz Russo," Design thinking : Business innovation", MJV Technologies and innovation press, 2011.
2. Design Thinking: Integrating Innovation, Customer Experience, and Brand Valueby Thomas Lockwood (Editor) Published February 16th 2010 by Allworth Press.
3. KalloriVikram, —Introduction to DevOps, 1 st Edition, KalloriVikram Publication, 2016.
4. Jaokim Verona, —Practical DevOps, 2 nd Edition, Packt. Publication, 2018.
5. Stephen Fleming, Pravin, —DevOps Handbook: Introduction of DevOps Resource Management—,1st Edition, Createspace Independent Pub. , 2010.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – III – Design Thinking – SCSA1306

UNIT 3
AGILE SOFTWARE DEVELOPMENT

Software Development- using Extreme Programming – Roles & Rules - Software Development using Scrum Framework – Scrum team – Sprints – Sprints planning – Metrics – Scrum tools - Case Studies.

3.1 Extreme Programming (XP)

Extreme Programming (XP), an Agile software development framework, is specifically designed for improving the quality of the software, the work process for the development team and increased customer satisfaction.

It is a method devised for a smoother and efficient software development life cycle (SDLC) for your projects, and it was first implemented on a project on March 6, 1996.

Why Extreme Programming (XP)?

Extreme Programming works towards providing iterative and recurrent software releases throughout the project; instead of everything together after a single, long project development lifecycle.

These short iterative cycles help both team members and customers to assess and review the project's progress throughout its development.

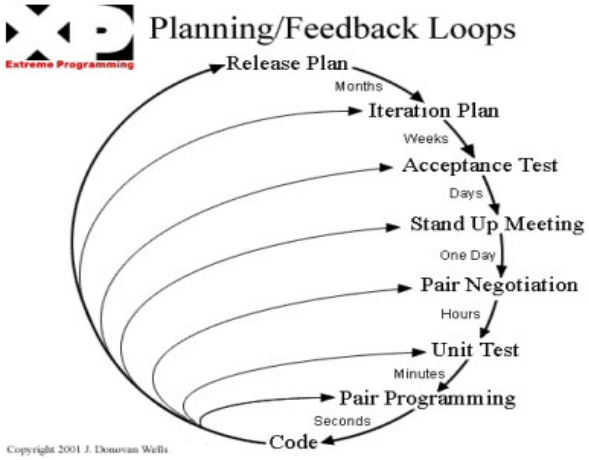


Fig 3.1 Extreme Programming

When Applicable

The general characteristics where XP is appropriate were described by Don Wells on www.extremeprogramming.org:

- Dynamically changing software requirements

- Risks caused by fixed time projects using new technology
- Small, co-located extended development team
- The technology you are using allows for automated unit and functional tests

Due to XP's specificity when it comes to it's full set of software engineering practices, there are several situations where you may not want to fully practice XP.

What is Extreme Programming (XP) made of?

The Values.

XP incorporates the following 5 values:

- **Communication**: Software Development projects or projects in any industry rely heavily on communication. XP focuses on effective communication between the team and the customer.
- **Simplicity**: XP looks for the simplest ways to get things done. This means doing what is essential thereby reducing waste, address only the known issues and keeping the design simple for effective creation and maintenance.
- **Feedback**: Feedback plays an important role in project improvement. XP encourages instantaneous feedback. This helps the team identify room for improvement and revise practices.
- **Respect**: The team must respect each other both personally and professionally, to achieve goals.
- **Courage**: XP endorses courage at all levels. This can include speaking up against what does not work and anything that affects the project's effectiveness, or accept feedback and improve methodologies.

The Practices

Fig 3.2 **Pair Programming**

The core of XP is an interconnected set of software development practices. While it is possible to implement these practices in isolation, many teams have found that some practices reinforce the others and should be done in conjunction. This can enable fully eliminating the risks you often face in software development.

The original twelve practices for XP comprise:

- **The Planning Game**- determines the scope and timing of the next release.
- **Small Releases**- puts a simple system into production quickly.
- **Metaphor**- guides all development through a simple shared story of the overall system.
- **Simple Design**- the system should be designed as simply as possible at any moment.
- **Testing**- is constantly undertaken for development to continue.
- **Refactoring**- restructures the system without changing its behaviour.
- **Pair Programming**- all production code is written by two programmers at one machine, Figure 3.2.
- **Collective Ownership**- anyone can change any code anywhere at any time
- **Continuous Integration**- integrates and builds the system many times a day.
- **40-hour week**- work no more than 40 hours a week as a rule.
- **On-site Customer**- must be available full-time.
- **Coding Standard**- emphasises communication throughout the code.

Over the years, teams have found that some practices reinforce the others. To eliminate risks, these should be unified. The following descriptions include some of the refinements based on various teams' experiences:

Whole Team: Teams should comprise cross-functional groups of people with different skills. In this way, they can complement each other to accomplish a specific outcome.

Sit Together: Most people agree that face to face conversations are the best form of communication. Teams should sit together without barriers to communication e.g. cubicle walls.

Informative Workspace: Teams should be arranged to sit in a way to make the team's work transparent to each other and the affiliated people outside the team.

Energized Work: This means making sure that a person is mentally and physically healthy to focus on work. This also implies there should be no over-work and respect teams to support their mental and physical health as well.

Pair Programming: The idea behind this practice is that 2 brains are better than one. Pair Programming refers to software production through 2 people sitting at the same machine. By

this, there is a continuous work review and problems receive a faster response. This method has been shown to improve quality and stay more focused.

Stories: Stories define the features that the product should have that would be meaningful to customers and users. These stories are used for planning and also serve as reminders for further conversations.

Weekly Cycle: The first day of every week, the team meets to reflect on the progress to date. The stories that should be delivered in the week are selected by the customer. The team determines how to approach those stories. The goal behind this is to achieve a running, verifiable feature by the end of the week. The fixed period allows for the production of a feature that can be shown to the customer for feedback.

Quarterly Cycle: The purpose of the quarterly cycle is to check the detailed work of each weekly cycle in the context of the overall project. The customer provides the overall plan for the team within a particular quarter. This not only gives the team a view of the project but also helps the customer to work with other stakeholders involved.

Slack: This implies adding a few, low priority tasks or stories in the weekly and quarterly cycles. If the team is lagging on more important tasks, these can be dropped. Else, these will also be completed, increasing the chances of meeting the estimated schedules.

Ten-Minute Build: The entire system and all of the tests should be run within 10 minutes. If the time exceeds this limit, multiple reruns will cost larger periods between errors. This practice encourages automation of the build process, making it feasible regularly, to run all of your tests.

Continuous Integration: This practice encourages immediate testing of new code to the existing larger codebase. This helps catch and fix integration issues sooner. This practice requires discipline and depends on the practices of Ten Minute Build and Test First Development.

Test-First Programming: Instead of following the regular way i.e.,

Develop Code -> Write Tests -> Run Tests

The practice of Test-First Programming takes the path of:

Write Failing Automated Test -> Run Failing Test -> Develop Code to Make Test Pass -> Run Test -> Repeat

This practice, too, reduces the feedback cycle for issue identification and resolution. This results in a reduction in the number of bugs that get introduced into production.

Incremental Design: This practice portrays doing a certain amount of work upfront to understand the breadth-wise perspective of the system design. After that, work further on the details of a particular aspect of the design when specific features are delivered. This approach reduces the cost of changes and allows you to make design decisions when necessary based on the most current information available.

3.2 Roles and Rules

XP incorporated particular practices for your team to follow and does not establish specific roles for the team members. However, according to the requirement, **the 4 most common roles are**:

The Customer: The XP Customer is expected to actively participate in the project. The Customer makes all of the business decisions regarding the project such as:

- What should the system do? This refers to the features that are included and what they accomplish
- When is the system done? This implies the acceptance criteria
- How much should be spent? Which means the budget for the project, and
- What should be done next? The order ins which the features are delivered.

The Developer: Developers realize the stories identified by the Customer, which means deliver a project with decided features.

The Tracker: The tracker is an optional role and depends if the team requires one. This is carried out by one of the developers for keeping track of relevant agile metrics, and it is essential for progress evaluation and identification of key areas for improvement. This is important for progress tracking and identification of key areas for improvement. Some of these metrics may include the amount of time worked, amount of overtime, the passing and failing tests, velocity, and reasons for variations to velocity.

The Coach: This role is helpful particularly if the team is just starting up. The Coach can be an outside consultant who has used XP before and can help mentor the team on the XP Practices as well as self-discipline. Employing the Coach helps avoid potential mistakes that new teams may make, expediting the project.

Advantages of Extreme Programming

- Extreme programming allows the software developers to focus on coding and not worry about the unproductive activities related to the project
- The most important benefit of extreme programming is that it allows the software companies to reduce the expenditure of resources like money and time on useless activities when they can be spent on activities like project realization and other brainstorming sessions
- Extreme programming also reduces the risks of project failure or coding malfunction, ensuring that the client will get their desired product in the end
- Extreme programming is an amazing methodology that doesn't require the code to be complex and hard for everyone to understand and that shows in the code of the

developers that use this methodology because whenever someone else takes over their position, they can understand the code very easily

- One of the best things about XP is that everything is transparent and Infront of everyone which helps keep everyone and everything accountable
- Constant feedback is also an incredible feature of extreme programming which allows the developers to code fearlessly and without the fear of judgment because they can always fix the minor mistakes, they make through the help of the feedback they receive
- Regular testing of all of the elements of the software, bug detection for all of the code, and the use of customer validation tests ensure that the client gets a working prototype or the actual working software in less time than normal
- Extreme Programming also helps companies in satisfying their customer and retaining their business for a longer time
- In extreme programming methodology, everyone is an equal member of the herd and everyone must share the burden as their peers, which means that from the requirement to code, developers will work side by side so that no one feels unappreciated or forgotten.

XP strategy(Rule)

- The Management Strategy
- The Development Strategy
- The Design Strategy

The Management Strategy

The management strategy that emerges from evaluation/ use of Accepted responsibility, Quality work , Incremented change, Local adoption, Travel light and Honest measurement principles will guide us towards decentralised decision making and leave managers with Planning game practice, using metrics as the basic XP management tool.

The Development strategy

The Development strategy is a radically transformed format of the traditional view of the development process. Its motto is that in XP all activities are centred around programming, i.e. “everything you do in XP looks like you are doing programming”.

You start with Iteration planning (from the Planning strategy) and include practices such as Continuos integration of written code, Collective (code) ownership and Pair programming, which ties the development process together.

Design strategy

All four XP values work towards the XP Design strategy, which requires the simplest design that runs the current test suite. You also use principles such as Small initial investment, Assume simplicity, Incremental change and Travel light, whch will result in gradual design change, no extra design results, working on the simplest design we can imagine and removing

all unnecessary functionality XP strategies are put into practice through a lifecycle of an "ideal" XP project.

It consists of a short initial development phase followed by a long-term simultaneous production support and refinement.

Exploration: prepare production, practice writing user stories, estimate and experiment with technology and programming tasks.

Planning: run the planning game practice; agree the smallest set of stories to be completed. Iterations to First Release: produce a set of functional test cases that should run at the end of iterations.

Productionizing: you need one-week iterations, certify that the software is ready for production; implement a new test and tune the system.

Maintenance: simultaneously produce new functionality and keep existing system running, refactor if necessary or migrate to a new technology; experiment with new architectural ideas.

Death: XP project ceases to exist; new functionality do not have to be added or can not be delivered.

3.3 The Extreme Programming (XP) Lifecycle

The XP lifecycle can be explained concerning the Weekly Cycle and Quarterly Cycle.

To begin with, the customer defines the set of stories. The team estimates the size of each story, which along with relative benefit as estimated by the customer, indicates the relative value used to prioritize the stories.

In case, some stories cannot be estimated by the team due to unclear technical considerations involved, they can introduce a Spike. Spikes are referred to as short, time frames for research and may occur before regular iterations start or along with ongoing iterations.

Next comes the release plan: The release plan covers the stories that will be delivered in a particular quarter or release.

At this point, the weekly cycles begin. The start of each weekly cycle involves the team and the customer meeting up to decide the set of stories to be realized that week. Those stories are then broken into tasks to be completed within that week.

The weekends with a review of the progress to date between the team and the customer. This leads to the decision if the project should continue or if sufficient value has been delivered.

The XP values are distilled into concrete principles, which determine XP practices.

The XP basic principles are

Rapid feedback: any learning on how best to design, implement and test the system must be fed back in seconds/minutes, rather than monthdyears

Assume simplicity: treat every problem as though it could be solved the simplest way possible

Incremental change: solve any problem with a series of small changes that make a difference.

Embracing change: preserve most options while solving the most pressing problem.

Quality work: you should enjoy your work:

This is how you produce good software.

XP activities

The four basic XP activities are:

Coding: is XP basic activity: whether you draw diagrams that generate code or you type at the browser, you are coding. Source codes should be used for everything: to communicate solutions, describe algorithms, express tests ... etc.

Testing: is important: automated tests often test hnctionality, and non-hnctional requirements cannot be avoided (e.g. adherence of code to standards) are also important. Unit tests are written by the programmers, in order to prove that programs work the way they are expected to. Functional tests are written by customers to convince themselves that a system as a whole works as it is expected to.

Listening: XP develops rules that encourage structured communication and discourages communication that does not help: it is not simply enough to say "everybody should listen to each other".

Design: is part of the daily business of all programmers in XP in the midst of their coding. It is based on the concept "that a change of one part of the system does not always require a change in another part of the system!" In good design every piece of logic in the system has only one home, it puts logic near the data it operates on and allows the extension of the system with changes in only one place.

3.4 Software Development using Scrum Framework

Three essential elements are

Product Goal: A goal that describes the future state of a product. The product can be a solution, service, or product such as software product, hardware product, or firmware product such as payment wallet, a vacuum cleaner, a movie, or interior design.
Product Backlog: It consists of the work that helps in meeting the product goal. The Product Backlog is an emergent, ordered list of what is needed to improve the product. It is the single source of work undertaken by the Scrum Team. Product Backlog items that can be done by

the Scrum Team within one Sprint are deemed ready for selection in a Sprint Planning event. The product backlog should be transparent to allow stakeholders to make decisions.

Scrum Team: The scrum team consists of three roles, a product owner, a developer, and a scrum master. A product owner maximizes the value of work done by a Scrum team, developers, or development team to develop a quality product. Scrum Master is accountable for the scrum team's effectiveness. They all work in a timebox sprint to produce an increment of done work as per the definition of done.

Four acceptable practices are

Product backlog refinement session: Organize a few sessions to refine product backlog to keep it ready for 2–3 sprints. It also helps make decisions about design, architecture, decompose features in small stories, and prepare acceptance criteria. The team can come up with estimates to support the product owner in ordering product backlog items.

Definition of Done: A checklist to make increment transparent. Definition of Done may consist of all the work that team is planning to do to produce software such as UI design, writing services, performing integration testing, meeting acceptance criteria, and keeping it in a releasable state. Definition of done also consists of work that demonstrates commitment towards quality and development practices like TDD, BDD, DevOps, Refactoring and code quality, etc.

Team Agreement: The team is self-managed in Scrum, and self-management needs some rules to resolve internal conflicts, stay focused, and be committed towards the work. Scrum values become a great input to draft team agreement using courage, commitment, focus, respect, and openness. While preparing the team agreement, the team also learn about each other and expected behavior in self-management.

Sprint Duration: Sprint should not be more than a month, and better to keep it consistent. What should be the ideal duration? One week, two weeks, and four weeks? It depends on uncertainty or ambiguity in the requirement, complexity in doing the work. Better to go with a shorter cycle if there are high ambiguity and complexity to minimize the risk.

3.5 Scrum Team

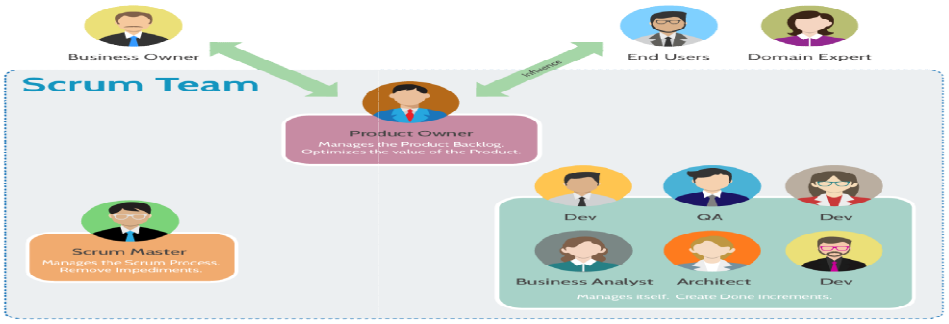


Fig 3.3 Scrum Team

A Scrum Team, Figure 3.3 is **a collection of individuals (typically between five and nine members) working together to deliver the required product increments**. The Scrum framework encourages a high level of communication among team members, so that the team can: Follow a common goal. adhere the same norms and rules.

3.6 Sprint Planning

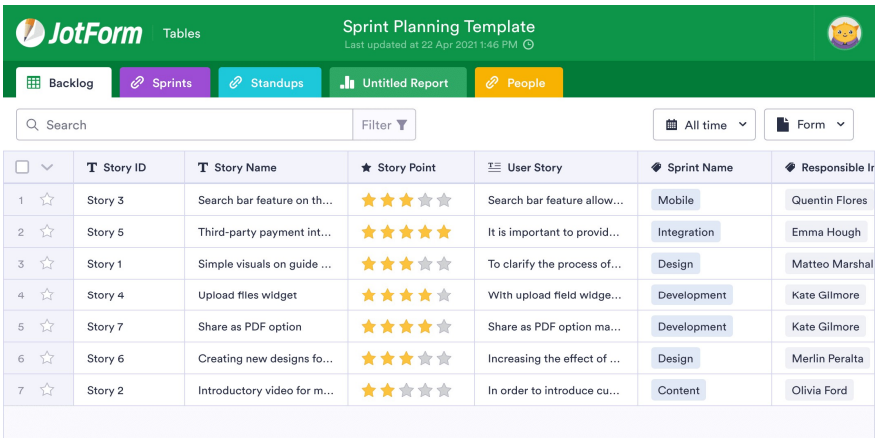


Fig 3.4 Sprint Planning Template

What are the common problems?

PO dictates and deciding what work will be completed

The product backlog is not healthy, prioritized or ready to be discussed

At the end of the sprint, everything is in too much detail and all work is assigned

No one understands what is actually meant by “done”.

The meeting is too long

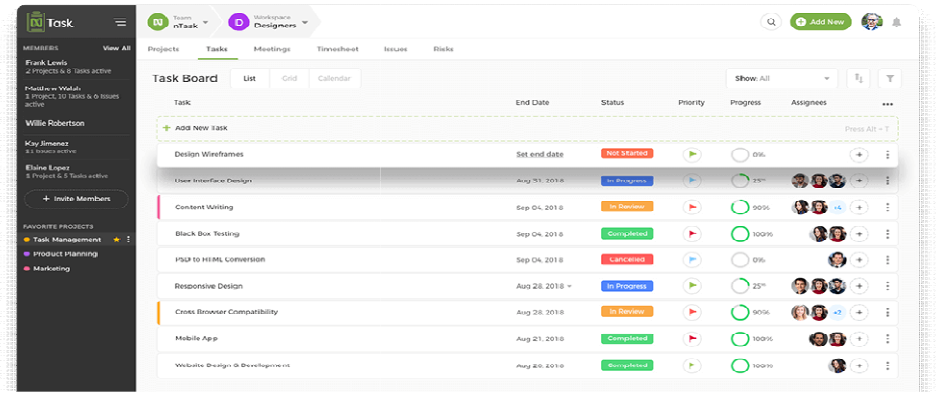
The meeting is not engaging

Some people do not have a voice

Poor environment and the team do not

3.7 Scrum Tools

1. nTask



Starting off the list with one of the most versatile free scrum tools available online, nTask takes the cake for being an all-rounder when it comes to incorporating scrum.

Primarily a project management software, the tool provides a unique set of features that make Scrum implementation painless for you. The only tool that comes with a myriad of features to manage every aspect of your project, and that too without any complicated add-ons.

nTask comes with a powerful meeting management solution that lets you productively execute your daily scrum meetings without any fear of losing productivity along the way.

Let's see how nTask helps in effective implementation of scrum:

What you should look out for:

- Project portfolio: all your projects in one central location, listed according to your preferred view for quick access
- Tasks management: all the associated and independent tasks right in front of you for instant assignment and monitoring of progress
- Daily Scrum management: smart meeting management features, along with advanced meeting participant controls
- Team collaboration: provide timely feedback to your team members through comments within tasks and an updated activity log
- Issue tracking: the timely resolution of issues with elaborate issue management functionality, including issue severity and issue assignment

- Progress reporting: built-in Gantt charts and timesheets for tracking the project progress and deciding the next course of action
- Risk management: creation and prioritization of risks with relevant projects to mitigate their effect on the project
- Team management: independent workspaces for teams working on multiple projects simultaneously, to ensure transparency
- Third-Party Integrations: nTask offers integration with Zapier, Zoom, Google Calendar, Outlook Calendar, and many more.

Pricing

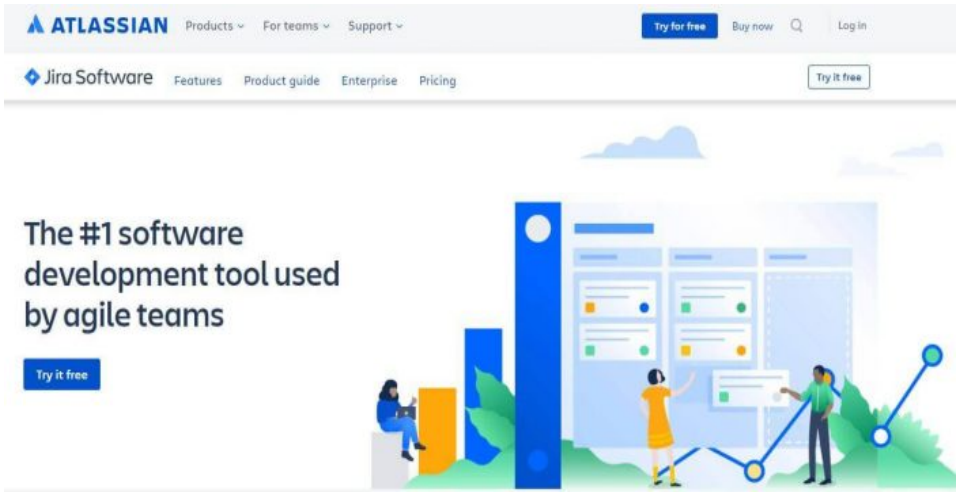
- Free: Unlimited workspaces, timesheet reporting, unlimited tasks, and Scrum meetings
- Pro: Starting at $1 per user/month with unlimited everything

    - Simpler project management for Scrum teams, absolutely free!

      5 workspaces, meeting management, project Gantt charts, bug and issue tracking and a lot more.

    - Get Started for Free

2. Jira



You've likely heard of Jira. It's one of the most widely used agile tools for Scrum, so without a doubt, the second spot in this list goes to Jira.

The tool comes loaded with features that make Scrum implementation seamless for the users, including customizable Scrum boards, custom filters for backlog management, and a number of visual project reports.

Although it comes with a well-balanced functionality, Jira can seem a little overwhelming for the starters. It can take quite some time to get a hang of the software, and it would not be a wise decision to opt for it if you're new to Scrum.

Let's see what key features Jira comes with:

Key Features

- Customizable Scrum boards
- Powerful progress reports
- Backlog management
- User stories mapping
- Bug and issues tracker
- Time tracking
- Customizable dashboard
- Sprints management
- Custom filters
- Real-time reporting
- Numerous third-party application integrations

Pricing

- Up to 10 users: $10 monthly
- 11-100 users: $7 per user/month
- More than 100 users: upon request. For large teams

Jira comes with a free trial of 7 days.

3. Targetprocess

First-time users may feel intimidated by TargetProcess's overly simple user interface. It's not rich in color as Monday or other best scrum tools on this list.

However, TargetProcess is more of an underdog.

We surveyed the market and discovered that the software has been holding up a repute for over 15 years now. Aside from that, TargetProcess comes with friendly user experience. You just feel right at home when using this program as a Scrum tool.

For instance, there is a high degree of customization with TargetProcess. Take the example of numerous templates that are free to use. There are custom cards, graphical reports, and other reporting documents that make up for the meat of Scrum methodology.

The best thing about this tool is support for SAFe methodology. The Scaled Agile Framework is an Agile method that TargetProcess's dev team had an eye on for a long time. This calls for the implementation of the Kanban framework, which is a subcategory of SAFe, value streaming of programs, and team micromanagement.

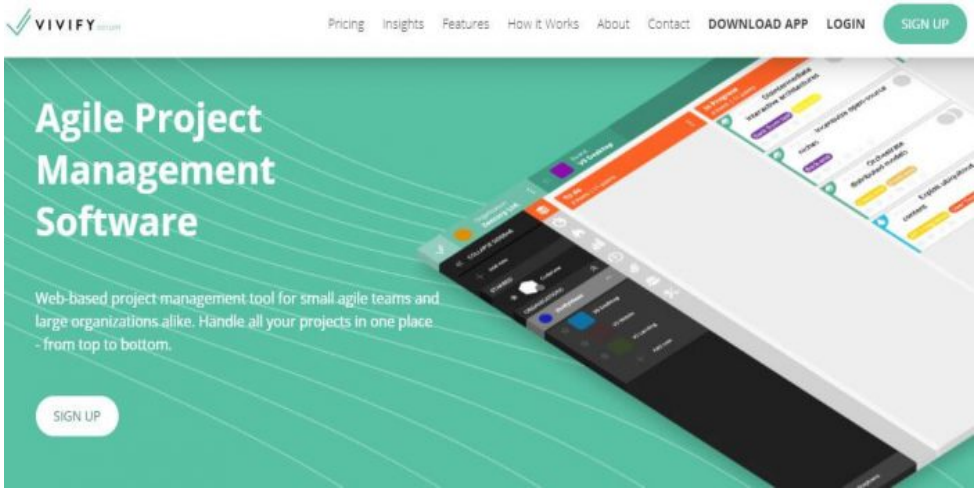
Key Features

- Centralized projects dashboard
- Drag and drop interface
- Project timeline
- Backlog management
- Issues and bugs tracking
- Shareable boards
- Sprint management
- Multiple project reports
- User story mapping
- Numerous third-party application integrations

Pricing

- Team: free for unlimited users and basic support
- Company: $20 per user/month
- Enterprise: upon request. Premium support and other advanced services

4. VivifyScrum



In the 4th place, we have VivifyScrum.

Coming with features tailored to the needs of Scrum and Kanban methodologies, VivifyScrum is a rather easy to use software that comes with a clean and clutter-free interface, making it aesthetically appealing too.

The tool allows users to create virtual organizations and add all the relevant team members. Users can conveniently add projects and boards within an organization and track progress effectively.

VivifyScrum also provides online Scrum training free of cost, in case you want to educate yourself or your team about Scrum.

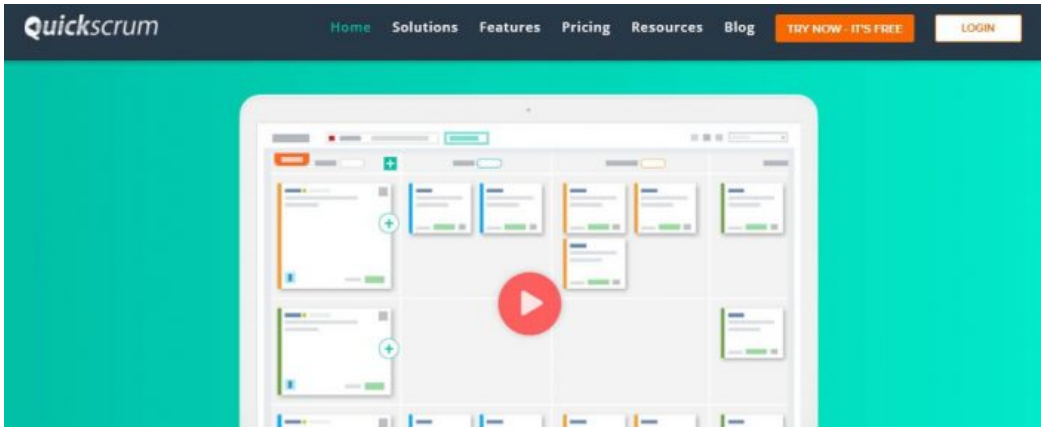
Key Features

- Scrum board
- Product backlog management
- Invoice creation and sharing
- Project calendar
- Customized roles and permissions
- Powerful charts and Scrum metrics
- Time tracking
- Customizable project labels
- Third-party application integrations

Pricing

- Free plan and a premium plan of $8 per user/month

5. QuickScrum



On #5 of our list for the best Scrum tools is QuickScrum.

A web-based tool coming with a simple drag and drop interface for easy backlog management, QuickScrum is one of the most user-friendly tools available online. With an interesting list of clients to bag, this tool can be your next choice for easy Scrum implementation.

While tools like Jira can be complicated for new users, QuickScrum can be easy to use an alternative that's quick to set up too.

Just like VivifyScrum, QuickScrum also provides online Scrum training to individuals and teams.

Key Features

- Scrum board
- Drag and drop functionality
- Comments within work items
- Activity tracking
- Work item statuses
- Customizable filters
- Workflow tracker
- Individual efforts tracker
- Burn up and burn down charts
- Multiple third-party application integrations

Pricing

- 14 days free trial and a paid plan of $3 per user/month

TEXT /REFERENCE BOOKS

1. MaurícioVianna, YsmarVianna, Brenda Lucena and Beatriz Russo," Design thinking : Business innovation", MJV Technologies and innovation press, 2011.
2. Design Thinking: Integrating Innovation, Customer Experience, and Brand Valueby Thomas Lockwood (Editor) Published February 16th 2010 by Allworth Press.
3. KalloriVikram, ―Introduction to DevOps, 1 st Edition, KalloriVikram Publication, 2016.
4. Jaokim Verona, ―Practical DevOps, 2 nd Edition, Packt. Publication, 2018.
5. Stephen Fleming, Pravin, ―DevOps Handbook: Introduction of DevOps Resource Management―,1st Edition, Createspace Independent Pub. , 2010.
6. Len Bass, Ingo Weber, Liming Zhu, G., ―DevOps: A Software Architect's Perspective, 1st Edition, Addison-Wesley Professional, 2015.
7. Alistair Cockburn, "Agile Software Development", 2nd ed, Pearson Education, 2007.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – I V- DESIGN THINKING – SCSA1306

Unit IV

DESIGN THINKING FOR STRATEGIC INNOVATION

Innovation Management-Changing Management Paradigms-Design Thinking related to Science and art-Design Thinking in Business-Linking Design Thinking Solution to Business Challenges

4. Innovation Management

4.1 Overview

Innovationisessentialforbusinesssurvivalinhighlycompetitivemarketswhereit is increasingly difficult to differentiate products and services. Innovation is important for the followingreasons:

- itallowsbusinessestoexpandtheircustomerbasebyrefreshingthemarket with new and improvedproducts
- itisakeycomponentofcompetitiveadvantageandhelpscompaniesstay aheadofcompetitorsbeforerivals'innovationstakemarketshare
- it supports the ability to charge apremium
- it provides incremental revenue and profit and also increases shareholder value.

Businesses that are not growing through new product and service introduction are likely to decline as their existing sales portfolio inevitably matures.

It is not surprising that companies such as Procter&Gamble and General Electric have actively embraced the management of innovation. Their principal goal is to drive growth and then to improve shareholder value.

'Nothingismorecentraltosustaingrowththaninnovationthatleadsanindustry and not only product innovations, but innovative design, innovative marketing, innovative in-store shopping experiences, innovation across the entire business. The companies and brands that lead innovation are the catalysts for growth.'

Definition and concept

'Innovation is generally understood as the introduction of a new thing or method. Innovation is the embodiment, combination or synthesis of knowledge in original, relevant, valued new products, processes or services.' Luecke and Katz, 200

Innovation management involves the process of managing an organization's innovation procedure, starting at the initial stage of ideation, to its final stage of successful implementation. It encompasses the decisions, activities and practices of devising and implementing an innovation strategy.

Fig 4.1 Innovation Management Principles

1. **Realization of value**

   Value, financial or non-financial, is realized from the deployment, adoption and impact of new or changed solutions for interested parties.

2. **Future-focusedleaders**
   Leaders at all levels, driven by curiosity and courage, challenge the status quo by building an inspiring vision and purpose and by continuously engaging people to achieve those aims.

3. **Strategic direction**

   The direction for innovation activities is based on aligned and shared objectives and a relevant ambition level, supported by the necessary people and other resources.

4. **Culture**

   Shared values, beliefs and behaviours, supporting openness to change, risk taking and collaboration enable the coexistence of creativity and effective execution.

5. **Exploiting insights**

   A diverse range of internal and external sources are used to systematically build insightful knowledge, to exploit stated and unstated needs.

6. **Managing uncertainty**

   Uncertainties and risks are evaluated, leveraged and then managed, by learning from systematic experimentation and iterative processes, within a portfolio of opportunities.

7. **Adaptability**

   Changes in the context of the organization are addressed by timely adaptation of structures, processes, competences and value realization models to maximize innovation capabilities.

8. **Systems approach**

Innovation management is based on a systems approach with interrelated and interacting elements and regular performance evaluation and improvements of the system.

4.1.1. **Innovation ManagementApplications:**

Innovation is relevant in any organisation and can be applied in a number of different ways.

Product/service innovation – introducing new goods or services that are new or substantially improved. This could include improvements in functional use, convenience or technical capabilities.

Process innovation – implementing new or significantly improved production or delivery methods.

Business model innovation – changing the way business is done, for example, EasyJet, Dellcomputers and global outsourcing.

Organisational innovation – creating or changing business structures, practices and models.

Marketing innovation – developing alternative marketing techniques to deliver improvements in price, position, packaging, product design or promotion.

Supply chain innovation – improving the way that materials are sourced from suppliers or improving methods of product delivery to customers.

Financial innovation – bringing together basic financial concepts. This might include credit, risk-sharing, ownership or liquidity to produce new financial services, products or ways of managing business operations.

For example, financial innovation adapts to new circumstances and develops new value chains as the compliance and legislative environment evolves. The common link between each of these is an improvement in efficiency, productivity, quality and/or competitive positioning for the organisation.

4.1.2 Key innovation risks include:

Operational

Operational risks include failure to meet specification, costs or launch date. Damage to company reputation and brand is another potential operational risk.

Commercial

Consumer resistance and competition are examples of commercial risk.

Financial

Investment yield may be less than planned. There is also a risk that debt/equity investors become dissatisfied.

4.2 Innovation management

Innovation management is the process of managing innovations, that is, ideas, in organisations through the stages of the innovation cycle.

The innovation cycle describes the activities involved in taking an innovative productorservicetothemarketplace.Inessence,therearetwoaspectstothis:

1. Developing the innovative product orservice.
2. Building the business to market the product orservice.

The Table 4.1belowprovidesanexampleofatypicalinnovationcyclewithactivities at each stage:

Stage	Description	Typical activities
1	Ideas	Identify a market opportunity
2	Resources	Organizepeople,financeandfacilitiesto match the goals of theorganization
3	Investigate	Research the possibilities
4	Patent	Protect the intellectual property
5	Design	Model and test it for users
6	Develop	Improve the technology
7	Make	Start production
8	Sell	Advertise and inform people
9	Service	Communicate with the customers

Innovation Pipeline Figure 4.2 consist several stages. The first stage in the innovation cycle is ideas generation. Ideas will often arise from observation of a current or future problem. They could be inspired by the organization's objectives or by a new market situation that suddenly becomes an opportunity.

Once the opportunity has been recognized, it needs to be evaluated. An important test for an idea is that it matches the goals of the organization and available resources – people, finance and facilities.

If there is alignment with the objectives of the organization, the idea moves to a newstagewhereitcanbeinvestigatedandfurtherdeveloped.Thedevelopment phase may involve further research into the opportunity or the patenting of the concept. Prototypes may well be designed, developed and tested at this stage.

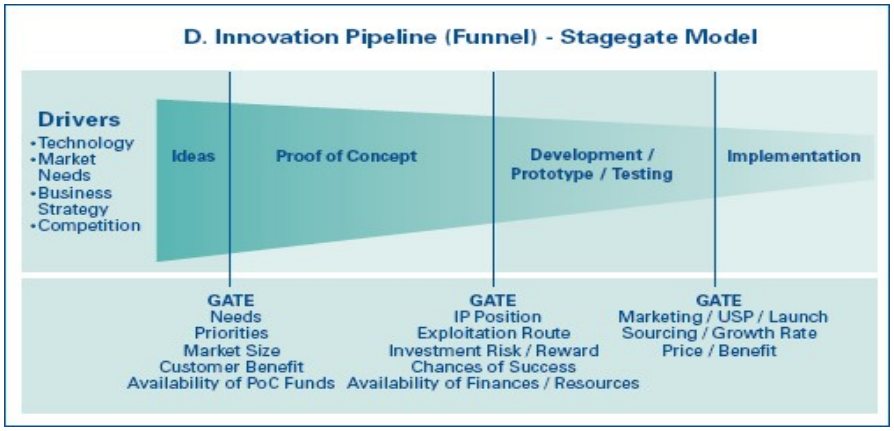


Fig 4.2 Innovation Pipeline

The decision to start selling the innovation is a critical stage. This is when significant resources are often required to support the launch. Sometimes an organization might wait at the end of the development phase for suitable market conditions.

The final stage of the innovation cycle is commercialization, where the innovation is marketed and sold to the customer. The innovation now moves out of the organization's control and into the hands of the users. This is the hardest stage of the innovation cycle for organizations to 'manage'. It is crucial that the organization monitors the innovation's performance so that any short comings are corrected.

Innovative organizations will typically be working on new innovations that will eventually replace older ones. This is important as product life cycles show reduced growth for older products and services. Growth may even begin to decline eventually, therefore impacting an organization's ability to expand.

New incremental innovations or changes to the product allow growth to continue.Companiestypicallygeneratefarmoretechnicalinnovationsthanthey canpossiblyhopetobringtomarketeffectively.Thereisaneedforstructured management and processes to handle innovation from the ideas stage to commercialization.

4.2.1 Innovation funnel

The 'innovation funnel' is a framework for managing innovation:

- itprovidesstructureanddiscipline,andfacilitatestheinnovationprocess
- it can allow faster development of innovations that drive growth
- it defines and tracks innovations according to predetermined criteria
- It provides 'gates' to control innovation resource decisions. This allows the passage of projects more likely to succeed by killing those more likely to fail as early as possible.

Manyorganisationswillhavetheirownversionoftheinnovationfunnel.An example of this is the

'Stagegate'™ model, used by the DTI

4.2.2 Activities are typically categorised into stages

1. **Concept**-Thefocusisongeneratingideasandconceptsandassessingthem against strategic fit, market opportunity, organisational impact and the chancesofsuccess,todecidewhichareworthexploringfurther.

2. **Feasibility**-Projectsareselectedforfurtherdevelopmentbasedonan evaluationofmarketacceptability,theinvestmentrisk/reward,andthe availabilityoftherequiredresources(people,facilitiesandmoney).

3. **Development**-Marketlaunchisdependentonsatisfactoryfeedbackofthe product prototype or service pilot, an evaluation of likely competitor response, and ability to deliver the required supply chain, marketing and pricing/margins.

4. **Implementation**-Executingtheinnovationbusinessplan,monitoringthe launch and performing post launch reviews to understand whether the implementationhasbeensuccessfulandwhat,ifany,changesneedtobe made.

Betweeneachstageareдecisionpoints(or'gates')wheretheideaorprojectis assessedagainstselectedcriteria.Thisistodeterminewhetheritshouldprogress and continue to receive funding and resources. Progression only occurs after satisfyingcertaincriteria,designedtoensurethattheinvestmentisminimizedin theearlystages.Ifnecessary,theprojectisabandonedsoonerratherthanlater.

Itisthisprioritizationprocessthatgivesthefunnelitsshape.Itmeansthatthose innovations that are most likely to succeed are not starved of resources from those that are most likely tofail.

4.2.3 The role of finance

Finance plays a critical role in the innovation process. This requires a delicate balancingactbetweenmanagingriskswithoutallowingthistobluntinnovation.

In essence, it requires finance to:

- Support innovation by providing analytical insight at both the strategic and detailed levels across the whole organisation
- provide an objective viewpoint and inject realism into discussions
- rely on the facts and structured analysis to support decision making
- understand the financial implications of marketing decisions
- ensure that there are clear, measurable gates throughout the project

- focus on sustain ability of innovation(going beyond year one volume)
- Prevent escalation of commitment from clouding judgement. It is human nature ostay committed to a course of action despite receiving negative feedback. It often takes more courage to kill a futile project than it does to sustain it.
- Monitor the success of the innovation post launch and provide feedback on performance for future innovation projects.

4.3 Changing Management Paradigms

A paradigm is a framework of basic assumptions, theories and models that are commonly and strongly accepted and shared within a particular field of activity, at a particular point in time (Mink, 1992; Collins, 1998)

Management paradigm therefore revolves around teamwork, participation, and learning. It also revolves around improved communication, integration, collaboration, and closer interaction and partnering with customers, suppliers and a wider range of stakeholders. Value creation, quality, responsiveness, agility, innovation, integration and teaming are increasingly regarded as useful guiding principles in the evolving new environment.

Value creation	Value added constitutes the basic social responsibility of the enterprise
Quality	Quality as a fundamental requirement influencing competitiveness
Responsiveness	Responsiveness to external environmental changes and customer demands
Agility	Flexibility in communications and operations
Innovation	Fostering new ideas, harnessing people's creativity and enthusiasm
Integration	Integration of a portfolio of technologies for a distinctive competitive advantage
Teaming	Decentralized, multi-functional and multi-disciplinary enterprise teams

4.3.1 Design Thinking related to Science and art:

Design thinking is an art and science to visualize multi-dimensional human reality. This visual representation helps see the relationships between the different dimensions, it gives a more intuitive sense of the whole, and it helps to think about how best to illustrate an idea.

Design thinking is the reverse of Critical thinking.

The two tools – Critical Thinking and design Thinking are nothing but an Interplay of three C's – Complexity, Confusion and Clarity.

Design is a conscious, self-aware activity rooted in science – the science of making things work - but also by its usage it has been elevated to an art form.

As Design Thinking is emerging, it is becoming a major tool for problem solving. From a little bit of what I have dabbled with Design thinking, I have realized that Design talks to the problem and problem talks back. Designing is a reflective conversation with the problem situation. Design is both analytic and synthetic.

Design Thinking in its real sense is experimentation, understanding and creative reframing.

Two elements – Design Attitude and Decision Attitude.

Decision Attitude – is the response which is easy to come up with alternative ways of solutions to consider – which are difficult to choose from.
Design Attitude – it is a difficult response to design a good alternative – but once you have developed a good alternative – the choice and selection of alternative is trivial.
Design thinking and design science are complementary components of an overall design paradigm.

Design concerns with human behaviours, attitudes, values and sensibilities in addition to product characteristics, meanings and styles.

Design Science

Design science adapts the process of design to the scientific method requirements of management science research. In contrast to design science for research, design thinking emphasizes design's ability to deal with human sensitivities, socio-cultural understanding, uncertainty and integrative treatment of ill-defined problems, which are more characteristic of the 'messy' field of management practice and most especially for innovation management.

Design science adapts and supplements the methodical, positivist and rationalist methods, which are used in everyday designing, as a methodology for prescription oriented social science research, the output of which is brought to contexts of social science practice through a growing bank of fully annotated objective knowledge.

4.3.2 Modeling Design Science

Hevner describe the characteristics of good design science research and provide seven guidelines for conducting and evaluating good design science research. These are problem relevance, design evaluation, research contributions, research rigor, design as a search process, communication of research.

Hevner describes a three-cycle view of design science research Figure 4.3The relevance cycle bridges the contextual environment of the research project with the design science activities. The rigor cycle connects the design science activities with the knowledge base of scientific foundations, experience, and expertise that informs the research project. The central design cycle iterates between the core activities of building and evaluating the design artifacts and processes of the research

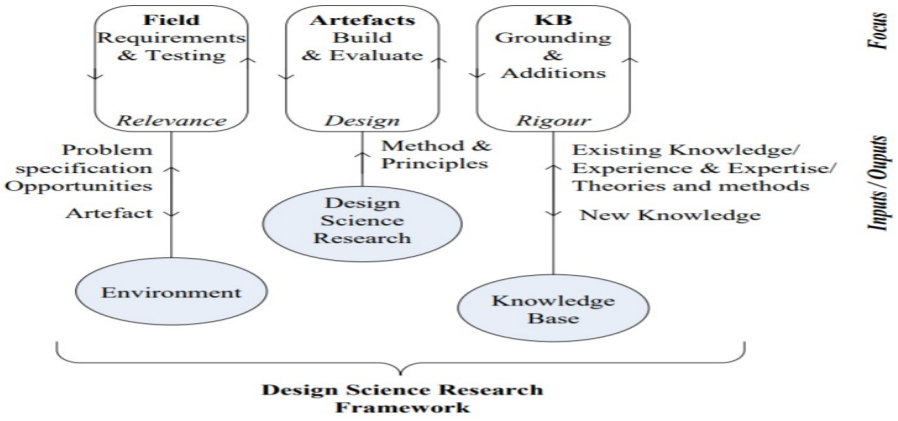


Fig 4.3 Design Science Research Framework

4.3.3 Design Thinking Innovation Framework, incorporating the Three Cycle View of Design Science

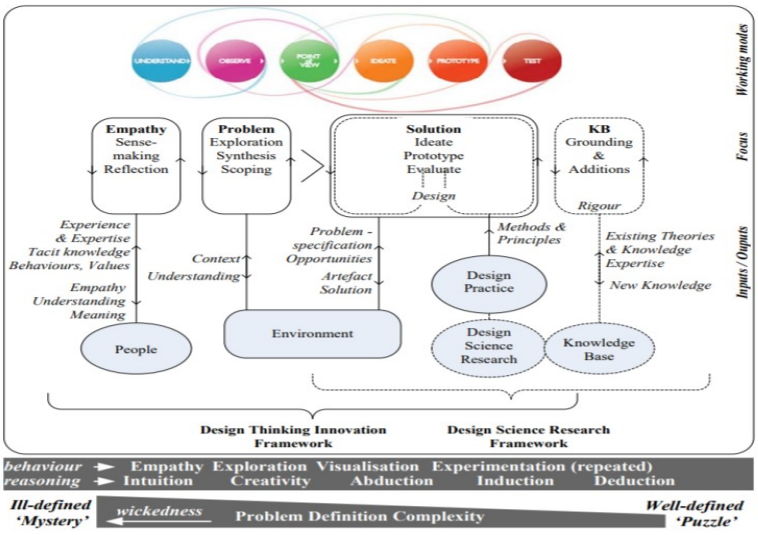


Fig 4.4Design Thinking Innovation Framework, incorporating the Three Cycle View of Design Science

4.3.4Design Thinking in Art

While 'Art Thinking' Figure 4.5focuses on making original solutions without the constraints of 'productivity', Design Thinking can focus on making the 'invented' product better, usable, culturally acceptable and manufacturable!

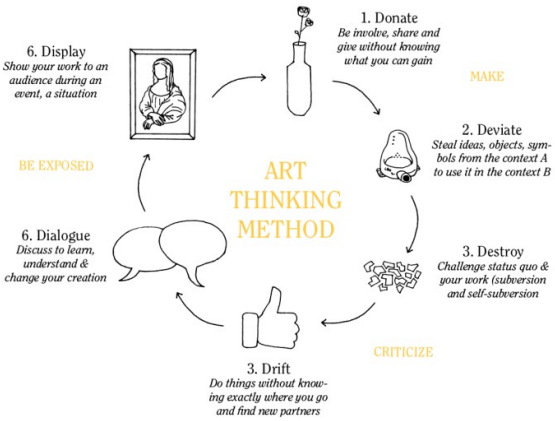


Fig 4.5 Art Thinking Method

Definition

Domain - dominant mindsets and cognitive skills of artists can be labelled “art thinking”

a domain-dominant emphasis in the **following cognitive strategies and mindsets**:

Cognitive Strategies

- Meta cognition

- Use of resource banks
- Prolonged research
- Problem-creation
- Use of constraints and generators
- Conversation with the work
- Delaying closure
- Reflection and evaluation of thematic coherence

Mindsets

- Emotional engagement
- Intuition
- Tolerance of ambiguity.

a set of domain-dominant features of the creative processes of artists ("art thinking") will be proposed that could be infused into the design process and pedagogy to stimulate creativity and self-generative innovation.

4.3.5 ART THINKING: CREATIVE PROCESSES AND MINDSETS OF ARTISTS

The artistic process allows for a different kind of understanding of creativity, one that emphasizes self-generation, meta cognition, and thematic coherence. These attributes could help designers in a contemporary environment in which creativity is seen as essential in developing novel solutions to complex and rapidly evolving environments, conditions and problems.

Domain-Dominant Cognitive Strategies Of Artists

Metacognition refers to the monitoring of one's own cognitive processes and influences while focusing on a specific task. Poor problem-solvers are less efficient at monitoring their own creative processes. While metacognition is a skill that designers need, it is especially acute for artists, as their problems are self-generated and successful solutions are primarily assessed against the artist's conception of the problem. Through metacognitive thinking, the artist has knowledge and control over his or her cognitive processes. He or she must constantly be aware of what is known and unknown while developing a strategy for further inquiry.

Use of resource banks

The artist has a deep understanding of their discipline so that when inspiration or idea strikes, it is recognized and acted upon. This state of awareness functions like a kind of priming device, allowing an artist to be ready to respond when seeking to find, generate, and/or solve a creative problem.

Artists are especially aware of assembling source material as an ongoing process, not just as a means to respond to a creative brief, as a designer might. Experienced design- ers also exhibit high sensitivity to their internal and external environments

Prolonged research

Connected to the practice of using a resource bank is the ongoing, deep immersion in the domain and art making practice that provides artists with a source of creativity. Creativity researchers have emphasized the importance of understanding the domain in which one is operating.

Problem-creating

A key difference in processes between designers, artists, and other domains (such as scientists) is in the problem-finding aspect of creativity. Many creators and researchers have noted that *finding* the right problem (or asking the right question) is far more important than *solving* the problem

Generators and constraints

Following the pre-preparation and preparation stages of the creative process, artists begin to synthesize their research and domain knowledge in the incubation stage. As - which characterize the work of visually talented individuals as they link their impressions into a landscape". Moments of inspiration are often de- scribed in magical, mysterious terms; however, they are actually the product of creating the space to allow the mind to make connections between variousinputs.

Conversation with the work

Developing a dialogue with the pieces, reacting to them, and making incremental changes along the way.

Delaying closure

"Delayed closure", continuing to experiment with solutions longer than non- artists. Those who were willing to delay closure during the art making process moved beyond obvious possibilities and generated more creative artworks

This strategy of delaying closure works in tandem with the rush to problem-solution mentioned earlier. If design students feel that they have satisfied the assignment or the parameters of the brief, they may stop development too early in the process, possibly thwarting the development of a more creative solution. Design educators can emphasize the importance of deadlines, but at the same time allow for solutions to come later in the process. Educators should explicitly communicate this opportunity for students to rework an idea after a stage has been passed if a more creative solution has been identified and developed.

Reflection and thematic coherence

The iterative nature of design thinking highlights that designers are also continually framing and reframing their work.

4.3.6 Domain-Dominant MindsetsOf Artists

Emotional engagement

The artists "emotional preoccupation with self", to create or communicate the experience of emotion. Artists are highly aware of these feelings and use them as source material for their

work. They are also ex- cited and engaged about getting an idea, and this emotional connection, which is mostly positive, continues throughout the project.

Intuition

Connected to the personal and emotional engagement of artists, intuition is a key point of emphasis in art thinking.

Embracing ambiguity

Descriptions of the creative processes of artists and designers often include a tolerance forambiguity which has been described as the most mature stage of ego development. Both designers and artists are comfortable with ambiguity, which may be evident in the sketching process

4.4 Design Thinking in Business:

Design thinking is a process for solving problems by prioritizing the consumer's needs above all else. It relies on observing, with empathy, how people interact with their environments, and employs an iterative, hands-on approach to creating innovative solutions.

The design thinking transformation in business Figure 4.6 - 4.9

- Doing
- Applying
- Value

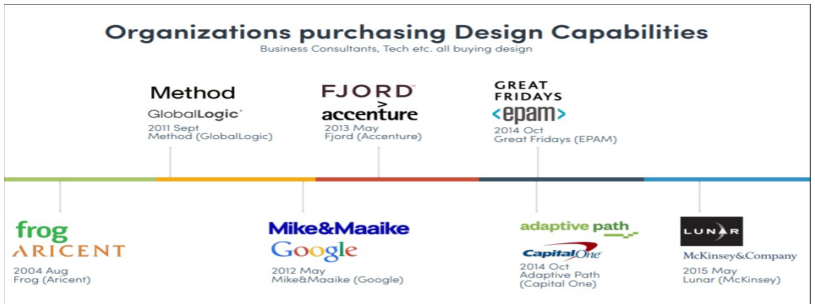


Fig 4.6 Organizations Purchasing Design Capabilities

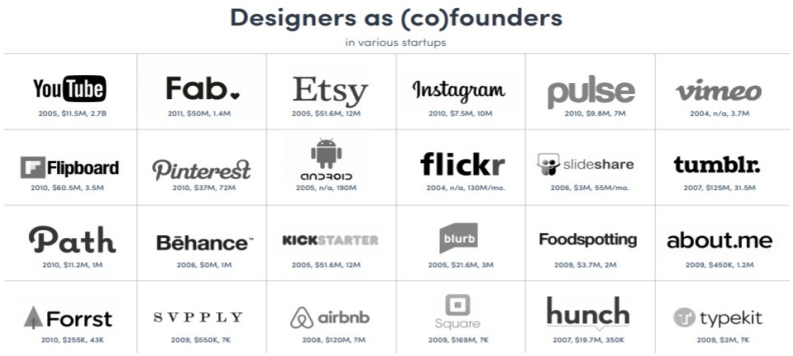


Fig 4.7 Designers as (co) founders

Fig 4.8The corporation incubators

Fig 4.9 Design Thinking in Business

What are the considerations?

Goals What problems are there? what problems could there be?

User Experience What are the user's needs and goals?

Technology Can the technology platform support the idea?

Organization Does the organization have the right resources for this?

Business Process Are there other processes involved in making the idea happen?

Financial How much does it cost, and how much of a return can be expected?

The balancing act of business, people, and technology depict in Figure 4.10

We **inform** business decisions based on our visible knowledge area.

We **influence** business decisions based on our understanding of business.

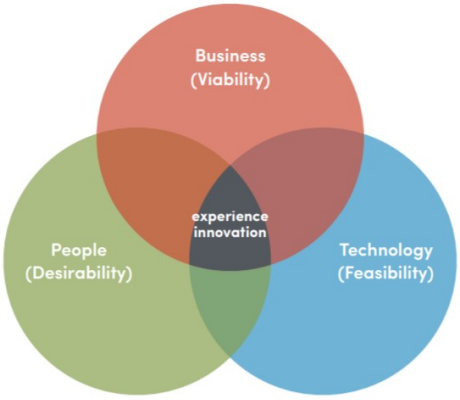


Fig 4.10The balancing act of business, people, and technology

The Designer's questions for business

- How to make sense of ideas from multiple sources?
- How do you factor constraints and capabilities to make the ideas happen?
- How much effort is it needed to make the idea happen?
- How do you prioritize and build on the good ideas?
- How do you take different people's needs into consideration?
- How do you measure the impact of idea?

The Value of Design Thinking in Business

All businesses have a never-ending list of goals, from constantly releasing new products that increase sales by resonating with customers to providing better customer support, Figure 4.11.

When a business decides on a new product, a massive, expensive machine shifts into high gear, especially at large corporations. The costs are enormous. Applying design thinking can help save vast amounts of money right away because it directs attention to the specific solutions people need—immediate cost savings are realized as part of the ROI of design thinking.

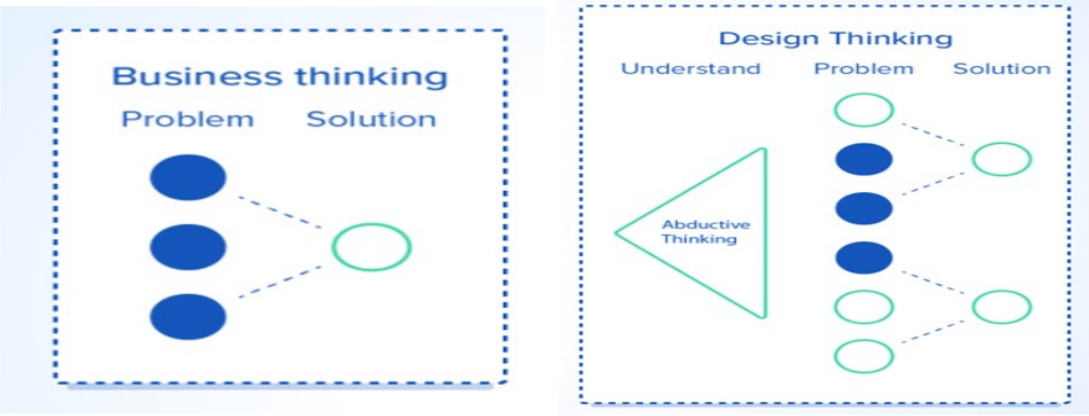


Fig 4.11 Comparison on Business Thinking verses Design Thinking

4.5 Linking Design Thinking Solution to Business Challenges:

In this topic we explore strategic business challenges familiar to most organizations and demonstrate how design thinking approaches can be applied to those challenges. Complex business problems today demand new leaders to manage change effectively, reinvent business models and practices rapidly enough to keep up with the competition, and out innovate them while balancing the management of change with rapid growth.

The adaptability of any organization depends on the effective handling of other key strategic challenges:

- developing adaptable strategies
- avoiding commoditization
- creating sustaining differentiation
- developing innovative culture
- engaging customers and employees
- responding to technological disruptions and
- Balancing short- and long-term strategies.

For each of these business problems we apply a design thinking lens to raise new questions and shape creative ideas. Each subsection is complemented with an activity.

The basic illustrative design thinking implementations in Table 4.2serve as useful examples of how to approach business problems with empathy, creativity, foresight, and consumer-centricity.

They are intended to inspire new pathways that address even the oldest and most wicked challenges. On the following is the list of design thinking solutions that are matched to specific business challenges.

Table 4.2Linking Design Thinking Solutions to Business Challenges

Business Challenges	**Design Thinking Solutions**
Growth	Storytelling
Predictability	Strategic Foresight
Change	Sensing
Relevance	Value Redefinition
Extreme Competition	Experience Design
Standardization	Humanization
Creative Culture	Prototyping
Strategy and Organization	Business Model Design

4.5.1 Business Challenge 01: Growth

Growth is at the forefront of every business leader's mind. Most tend to utilize classic growth strategies to get there: new strategic partnerships, horizontal market expansion, vertical integration, product extension, and franchising. Although all these strategies are well known, how does an organization come to select one over the other as a means for growth? Once decided, how does a company focus its resources? And what are the implications for organizational design?

The process of redefining the boundaries of business and making explicit decisions regarding who it will and will not serve often sparks intense debates around any growth strategy. There are no simple black and white answers to these questions, but design thinking can be used to bring clarity and align its business model to achieve maximum leverage.

Philip Crosby, author of *The Eternally Successful Organization*, says growth is unavoidable "if for no other reason than to accommodate the increased expenses that develop over the years. Inflation also raises the cost of everything, and retaliatory price increases are not always possible. Salaries rise as employees gain seniority. The cost of benefits rises because of their very structure, and it is difficult to take any back, particularly if the enterprise is profitable. Therefore cost eliminations and profit improvement must be conducted on a continuing basis, and the revenues of the organization must continue to increase in order to broaden the base."

Most organizations, however, aspire to grow in order to prosper, not just survive. Growth means different things to different organizations. There are many dimensions a company can select to measure its growth. Although the ultimate goal of most companies is profit, other financial data may be used as indications of growth. Some business leaders use revenue, EBITDA (earnings before interest, taxes, depreciation, and amortization), product line expansion, employees, or other criteria to evaluate organizational growth.

Growth is also the very essence of entrepreneurship, including corporate entrepreneurship. High growth leads to managerial complexity, so the alignment between shareholders, executive leadership, managers, and employees can quickly become choking points. Growth comes with the questions where a company's leadership, culture, systems, management, and business model can hold itself together.

Growth largely depends on increased economic activity, which requires certain preconditions, including consumer confidence and demographic shifts. The 1980s and 1990s were an unprecedented period of economic expansion in the United States as the country was driven by increasing demand for cars, housing, home appliances, and other products. The debt-fueled growth of consumer spending was boosting the gross domestic product (GDP) and creating millions of jobs while corporations were building their production capacity and pushing up their corporate profits. Growth was easy under those circumstances, and business gurus were busy selling any number of "proven" formulas to get it.

These management best practices developed during the boom period between the 1980s and 1990s were widely codified and taught in business schools around the world. They were liberally applied to industry after industry by managers trained in whatever best practice. Today, the danger lies in applying theories and practices based on those outdated models of two or three decades ago. Consider current macroeconomic trends and industry dynamics—growth is a lot harder to come by because we are operating in a very different, much more complex world. And if you're thinking about relying on emerging markets such as China, Mexico, Brazil, and India, be warned that you are entering a very different game with very different rules.

Growth can't continue unchallenged forever; our Earth's resources are finite. Sustainable growth is still a myth: It's quite impossible to decouple economic progress from environmental damage. With all of that, we are looking at a no-growth future. Unless we can reimagine and reinvent new industries, we simply cannot rely on economic growth to power our growth plan.

Growth is also the very essence of entrepreneurship, including corporate entrepreneurship.

Growth Needs a Strategy, and Every Strategy Needs a Story

Most businesses develop specific plans that, over time, will move their business to a level that meets the goals of the executive team, the shareholders, and the investment community. The reason why growth strategies are so vital is they keep things moving. And in business, if you're not going forward, you're going backward.

Growth means creating a clear and compelling vision of the future. Your vision needs to be very clear in terms of what you want from your business. How do we plan to attach an adjacency? How do we become the market leader? How about expanding to multiple geographies? And what's in it for managers and employees? Ultimately, the most meaningful yardstick is one that shows progress with respect to an organization's stated goals, whatever they are. So how do you develop your organization's stated goals? How do you develop the vision of where you want the organization to be in the future?

Design Thinking Approach 01: Storytelling

People who most successfully practice design thinking are curious, imaginative, and filled with wonder. In short, they are people who love stories and people who love to tell stories.

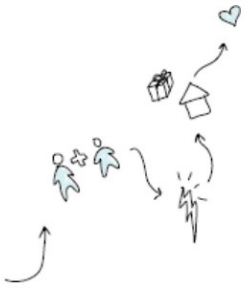
Stories reveal the hopes, dreams, and aspirations of authors, readers, populations, and cultures. They can also reveal the hopes, dreams, and aspirations of large organizations. The story of a company is what truly determines its purpose and value. The right story will always be more substantial than the right strategy because strategies are susceptible to the ingenuity of competitors, disruptive industry shifts, and bad luck while stories are impervious. If you write it, you own it.

Every time a large-scale change effort fails, it's because management fail to connect with mid-level executives and employees in a meaningful way. They circulate memos or PowerPoints. They produce corporate communication booklets. They dust off the pom-poms for the next corporate off-site meeting. Or they throw up a lot of dry data and factual logic that some audience members receive with their dukes up. In short, they fail to tell a good story.

Good storytelling is a technique where a leader is tasked with reframing an organization's past, present, future, problems, needs, desires, and hopes using a narrative built on salient metaphors to help people understand and connect with the company, its values, and its purpose. In a world when being strategic means being logical and fact-based, where do stories belong? They add an emotional dimension to business logic.

"If history were taught in the form of stories, it would never be forgotten."

—Rudyard Kipling



They foster empathy and connectedness. They prioritize information and objectives. They provide a clear beginning, middle, and end. They help us manage crises. And they evoke our hardwired predisposition to process information faster and more holistically when presented to us in the form of a good story.

The narrative structure of a story is a teaching tool that can make complex structures or relationships more easily accessible to an audience. Because the important ideas are set in a metaphor that people can easily understand, both storytellers and listeners can move past arcane details and focus on the problem at hand. The immediacy of the story helps people track the important relationships while empathizing with the subject. This allows for a richer experience and fosters greater insight into the nature of the organization, its place in its environment, and how the choices of its members contribute to its growth and success.

Unlike more traditional commentaries on business growth, design thinking presumes that numbers cannot tell the whole story and that other means of communication are required to define and articulate the future. The design thinking approach believes that stories are uniquely useful in their ability to bring people onto the same page, organize information, and present it in an efficient and accessible manner. Often, many organizations suffer from a divide in the vision of what they actually are. Storytelling is a technique to harmonize the company's vision and translate the key elements of a strategy into a compelling and accessible narrative that connects the past with the present and the future in a cohesive way.

So how do you tell a great story that will inspire senior executives and employees alike?

Make it collaborative

Whether you engage multiple stakeholders in shaping the narrative and its presentation through some form of crowdsourcing or co-creation or you simply gather input from employees at every level through informal conversations, it's important to ensure that elements of what you are about to tell resonate with the audience. No matter how fantastic, every great story has the familiar embedded within its structure.

Make it engaging

The medium can make or break the message. What kind of media, experience, or event is right for your audience? If your answer is PowerPoint, you don't need a good story; you need a good spanking. To really drive home your message, you need to take your audience somewhere new with something new. Consider the simple power of videos, the tangibility of beautiful print, or out-of-office immersions in spaces or places that will inspire people through new experience.

Make it structured

A narrative is employed in storytelling to arrange information in a logical structure that can be followed because we, as humans, are familiar with beginnings, middles, and ends. Because the storyteller and the audience know this structure, they are able to focus on the content of the story. When content flows, a good storyteller can communicate challenges and complex information in a format that is more clear and familiar to the audience. Just remember: The key content is the ending—what a company wants to be, where it wants to go, the future.

Make it performative

A storyteller engages an audience through an oratory recounting of a narrative. An effective storyteller does not simply speak the words but rather brings them to life by leveraging dramatic techniques such as body language, tone, tempo, and timing. Storytellers like Steve

Jobs or the archetype of the trickster capture attention, suspend disbelief, and bring people deep into the emotional dimensions of a strategic narrative through their compelling performance and delivery.

Make it tangible

People like stuff they can see and touch. To help illustrate intentions and what the future might look like, consider how technology demonstrations, prototypes, and other see-able and touchable artifacts can signal the strategic intentions of the organization and articulate how to move, grow, and transform in a particular direction.

Make it fun

Build interactive narratives in the form of games or simulations that enable the audience to encounter stories in a holistic, self-guided, interactive way. These typically encourage the sandbox-like discovery of a growth narrative that is progressive, staged, or a hybrid of both. These serious games allow people to explore and experience the roles, tasks, and relationships within a clearly defined system of rules and understandable context.

Make it real(ish)

Fictiveness refers to how true a story may be. The fictiveness of a story is related to its plausibility, its applicability, and its potential to explain something. Even if it is entirely fictional, the listener can identify if it applies to real scenarios and the circumstances. The perfect fictive ratio for storytelling? Aim for a narrative that includes a vision of the future and possible growth that is just out of reach but that is grounded in a possible and plausible way. Such a story can then inspire and stretch while providing the information needed to guide your team.

Building Empathy and Empowering Transformation

Growth-oriented narratives help elicit emotional inputs and responses from stakeholders and can be used to infuse the organization with optimism or uncover anxieties about the future. This creates stronger internal bonds between stakeholders both within and outside of the organization by using human-centric narratives to foster common purpose. Here, personal feelings, perspectives, assumptions, expectations, and aspirations can be shared, illustrated, and addressed by everyone involved. The resulting dialogue allows organizations to accommodate and unify a large number of diverse perspectives that may or may not initially align. This allows the newly energized organization to become more fluid, agile, and efficient through newfound, narrative-driven cohesion.

A story is a mirror that reflects existing states of culture by illustrating how an organization currently operates and how it can or will have to change in the future. Stories can place an organization's existing culture within new contexts, highlight strengths and weaknesses, establish the purpose surrounding a vision for growth, and illustrate the transformation of roles. This gives stakeholders the chance to rehearse and prepare themselves for change. Furthermore, by hearing and engaging in growth-oriented stories, individual employees can recognize which new skills and capacities they will have to acquire, develop, or improve. They receive a very direct sense of which resources will have to be acquired and developed to pursue new goals and milestones.

THINKING POINTS

There are seven to nine core narrative structures that we typically consume in some form or another on a daily basis. They provide us with the opportunity to experience new contexts and situations. These frameworks allow people to benefit from empathetic role-play and explore the goals, choices, decisions, motivations, actions, and successes and failures in a more intuitive way.

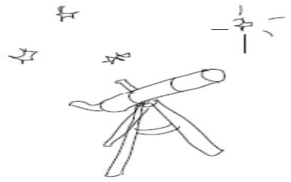
Paths toward organizational transformation and growth can be communicated effectively through the development of characters, personas, artifacts, and future-oriented archetypes that sit within familiar narrative structures. Stakeholders can easily identify, engage, debate, or learn by exploring their choices, actions, and experiences within a variety of contexts and situations.

Narratives tie the past, present, and future of an organization together. One problem faced by organizations is that younger or newer employees lack the long-term view of their older and more experienced coworkers. This often creates a perceptual divide that can fracture the understanding of the company's purpose. Storytelling is an effective way to acculturate new employees. By illustrating the path or journey that the organization has taken, stories help them gain a better understanding of where the organization has come from, what choices and challenges it has faced, and what lies ahead.

4.5.2 Business Challenge 02: Predictability

Business strategy is about finding the balance between two things: predictability and malleability. Malleability is the extent to which the external environment can be influenced and shaped by the actions of companies or industries. Predictability is the extent to which the future of the external environment can be forecast or predicted, which depends on the degree of complexity and speed of change. It usually does not account for any shock or major disruption. It is highly desired and valued in business planning, new product launches, and technology projects, as well as among investors. It builds confidence that managers know what they're doing and will effectively plan for capital and operational investment.

Wondering about the real difference between strategic planning and strategic thinking? Just ask management guru Henry Mintzberg. For him, strategic planning "has always been about analysis, breaking down a goal or set of intentions into steps, formalizing those steps so that they can be implemented . . . and articulating the anticipated consequences or results of each step." In contrast, he writes that strategic thinking is about synthesis. Here, intuition and creativity are the tools that shape a vision of where an organization can or should be going. Essentially, this is strategic foresight, a highly effective way to make sense of possible futures, clarify ambiguity, and create new perspectives that can establish competitive frameworks and opportunities for innovation.



"The best way to predict the future is to create it."

—Peter F. Drucker

Most organizations strive to achieve a value-adding level of predictability by implementing measurable, repeatable, familiar business processes. This type of predictability allows companies to improve efficiency, effectiveness, and productivity while gradually reducing costs. Typically, business leaders identify processes for improvement and predictability by looking for characteristics and parameters that include transportation and supply chain issues. But this is the present, not the future.

The only true way to maintain a useful level of predictability is to actively engage in the shaping of the organization's future. By studying, developing, and visualizing forward-looking scenarios, an organization can equip and prepare itself for tomorrow. Think, for example, how understanding the implications of digital technology, the increasing popularity of torrenting, and shifts in consumer attitudes toward ownership might have given Blockbuster a fighting chance. That fighting chance is strategic foresight.

Design Thinking Approach 02: Strategic Foresight

To face the unknown, businesses must adopt a different approach to predictability. The ability to manage the uncertainties of the future is critical to planning for growth or survival. Because of the rise of the innovation society, new technologies, and a rapidly globalizing economy, business leaders are forced to deal with not only the speed of change but also massive new complexity, uncertainty, and paradox on a global scale. The future will emerge from a constant stream of decisions, strategies, and commitments that have to be made in the present, with as much understanding of risk, possibility, and wisdom as possible.

Most managers appreciate and understand the value of strategic foresight but don't know how to make it tangible enough or integrate it into business strategy. Strategic foresight is not "planning"; it's one of the many inputs for planning. Strategic planning needs to consider a multitude of factors in the present competitive and operational environment and then extrapolate the data into a possible future that is based on a rigorous reading of weak signals. For foresight practitioners, the mission is to imagine what possible futures can be created and provide strategy a vision (or multiple visions) to facilitate a meaningful dialogue and a road map to close any competency gap between today and tomorrow.

"Don't despair: despair suggests you are in total control and know what is coming. You don't—surrender to events with hope."

—Alain de Botton

Strategic foresight is a deliberate and systematic process concerned with establishing well-informed future-oriented perspectives that help guide and inspire innovation, planning, and decision making. It helps us understand, anticipate, and prepare for change by equipping us with the tools and resources needed to ask provocative questions, challenge and test assumptions, rethink goals, and explore meaningful strategic alternatives. It encourages the deliberate and systematic exploration of uncertainties and their potential impact on behaviors and relationships.

Why Does Business Need Strategic Foresight?
To help to prevent or prepare for surprises.

Within dynamic and discontinuous environments, foresight helps organizations better understand the variables influencing the pace, nature, and possible impacts of change. It addresses and confronts the need to continuously orient, reorient, plan, and act within volatile, complex, and uncertain business landscapes so that we're not caught off guard by

change. Identifying and questioning the significance of the variables influencing and driving "change" ultimately help us anticipate and prepare for future conditions.

To help to establish and maintain competitive advantage.

Practicing foresight helps an organization gain a more robust understanding of how competitive dynamics might be changing and where gaps and opportunities might exist. This, in turn, helps enrich an organization's latent innovation potential, resources, and maneuverability. This is often accomplished by strengthening the ability to define and occupy opportunity spaces faster and with greater competence.

To positively influence and support innovation.

Foresight is one of many critical inputs or raw ingredients that help frame, fuel, and manage innovation. Any organization with an 18- to 36-month innovation cycle that is drawing on trend analysis to shape its future might as well throw in the towel. By the time you're ready to launch, your trends are old news. Foresight draws on trends—but it is not about trends today but rather what they might evolve into tomorrow. It can be used at the front end as a guide, as fuel, filter, or catalyst. And it can be used postinnovation as an amplifier or critical future proofing lens.

To empower and engage.

Practicing foresight helps individuals and organizations develop and improve their ability to identify future opportunities and transform them into meaningful outcomes. Developing and practicing these skills also builds the confidence necessary to make decisions within dynamic, uncertain, ambiguous, and less tangible contexts.

"Create your future from your future, not your past."

Design Thinking and Strategic Foresight

Design thinking is about solving "wicked" problems. As the speed of change continues to increase the complexity of doing business, new and more complex systems and even more "wicked" problems will continue to emerge. The race is on between bigger challenges and quicker solutions.

Foresight is an iterative and cumulative learning process that employs the design thinking tool kit, which includes environmental scanning, context mapping, archetype creation, and scenario development. It offers ways to explore, learn, and make sense of the variables and uncertainties shaping possible futures by questioning their relevance, illustrating their qualities and potential, and describing their influence on existing models and relationships.

Exploring the future and its uncertainties should be a daily activity sustained by individuals throughout the organization. Done properly, it should align more closely with planning, strategy, and innovation cycles.

To help organizations win that race there are many foresight tools, processes, and methods that can be employed, most of which begin with weak signals.

What Are Weak Signals?

In the 1970s, Igor Ansoff, an applied mathematician, business manager, and the father of strategic management, noticed that failures in strategic management were causally linked to organizations overlooking vague, anomalous, ambiguous, yet critical information. To rectify that, he developed the weak signal theory.

For him, weak signals represented change or the potential for it. This change can, at first glance, be insignificant (such as the emergence of P2P file sharing) or highly disruptive (such as digital cameras). These signals are not facts or trends. Rather, as signs of new and emerging capabilities that could disrupt or transform existing norms, they represent subtle changes in reality that will manifest in individual or organizational behaviors, needs, desires, or values.

For example, early research investment in an obscure technical domain may be an obvious, strong signal of things to come for experts working within that context who may, even at the very early stages, understand the potential impact of the new capability. For those of us who are outsiders unfamiliar with this particular domain, that signal is still weak.

Beginning with weak signals, there are a number of component parts to the strategic foresight process, discussed next.

Weak Signal Scanning

Signals are driving forces that influence and shape the rules, boundaries, and relationships of future competitive and operational contexts. Ultimately, the objective of scanning is to identify what these driving forces are and build awareness around how they interact and how they might redefine the future.

Weak signal scanning involves searching the environment for information and insights that will help construct the ingredients of near-future contexts. Through desk research on the Internet, scientific and academic journals, media, policy makers, start-ups, competitors, and other sources, an organization can develop an understanding of how new and emerging shifts in technology, behavior, culture, products, services, and other inputs might evolve to shape possible futures.

An evidence-based method of gathering, synthesizing, and socializing new and critical information, this activity can shift the attention of an organization toward new opportunity areas, threats, and potential blind spots by filling in knowledge gaps and developing a more holistic understanding of the world. Key observations and insights generated from scanning can help an organization reevaluate its current behavior in relation to the changing landscape, develop a new dialogue, plan for the future, and compete effectively in the long run.

Traditional approaches to environmental scanning include identifying and analyzing the qualities and characteristics of signals from within the social, technological, economic, environmental, and political contexts (STEEP). As a cumulative process, sustained scanning in these subject areas across a multitude of subcategories can help build a stronger and more competitive knowledge base from which potential futures can be defined, prioritized, mapped, and explored.

Weak Signal Processing

Questioning signals helps decision makers and planners review long-standing assumptions, critique cultural logic, and question strategic rigidity. In this process, stakeholders also become aware of opposing or restraining forces that are barriers to growth. Drivers are found in diverse contexts and identified through different approaches.

Here, signals are used to create an initial analysis and translation of what might be the road ahead. This is essentially a “first cut” at creating order out of the bewildering variety of data that scanning generates. Ultimately, this is an insight organization stage where signals are subject to discussion and analysis based on an identification of structures, patterns, themes, and trends.

Weak Signal Amplification

Determining which signals, insights, and drivers are most critical to include in future contexts and where they fit in relation to one another is key to reducing noise and creating clarity. In this activity, practitioners consider which signals are more relevant to specific business objectives, organizational capacities, or other previously determined goals. Identifying which might have the highest impact on the vision, objectives, and strategies of the organization or its future competitive landscape is the aim.

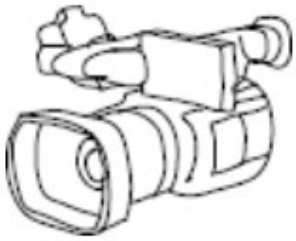
Here, outputs from the previous steps are used to create and expand upon forward-facing views. Scenario, visioning, and normative methods are applied. One should not limit the use

of any specific tools; systems thinking can also be highly effective, or simply drawing a systems map or causal loop diagram forward in time with different assumptions is also a valid technique to examine how different futures may unfold. The question asked at this stage depends on the type of potential futures under consideration and where they are possible, plausible, probable, or preferable.

Context Mapping

Context mapping is a sense-making process that involves establishing and illustrating the conditions and rules that frame future-oriented scenarios. This process seeks to prioritize, organize, amplify, and visualize the significance of the signals collected through scanning. In a sustained or ongoing foresight model, the rules, boundaries, and relationships defined within context maps may evolve over time to match and align with changing market conditions and organizational behaviors.

Scenario Development



Scenarios are storytelling tools that present choices and dramatize the impacts of decisions and strategies. They allow stakeholders to access, experience, debate, and rehearse multiple responses to possible futures. They are also highly effective prototyping tools that mobilize the imagination and place human experience, behaviors, and motivations—both individual and organizational—at the core. As fictive spaces, they allow us to sketch, question, and explore a spectrum of possible futures from many perspectives, while also providing a common artifact that can be shared and responded to.

Scenarios typically emerge from context mapping exercises that have established very clear frameworks, lenses, and systems of rules. By participating in the creation of well-informed scenarios, stakeholders gain a better overall picture of the forces driving change and how futures may evolve and manifest.

Ultimately, this process aims to create tangible outputs that give organizations a range of strategic options for dealing with possible futures. Through written descriptions, visual maps, multimedia presentations, and even experiential events, the goal is to communicate future contexts, potential disruptions, options paths, and the key drivers and rationale that shapes

and surrounds them. Here, building timelines that help organizations imagine pace, impact, change (continuous, rapid disruptive versus sustained and incremental) contexts of adoption, new uses, applications of technology, and the transformation of values, behaviors, and expectations makes preparing for the future feel more present and tangible.

THINKING POINTS

Technology scanning looks beyond popular tech-media and out into the labs, start-ups, universities, and garages of inventors to collect, analyze, and interpret the functional characteristics of emerging technologies. These technologies and the potential they carry may someday transform or disrupt existing models, behaviors, and relationships.

Cross-industry scanning curates market expressions and innovations from adjacent industries to identify new organizational behaviors—research and development (R&D) investments, patent applications, emerging products, services, business models, and experience archetypes—that illustrate how others are creating, defining, or adapting to change.

Scenarios are not an end in themselves. They are a tool to improve the quality of decision making when dealing with many unknowns. The outcome is not one future but multiple futures. The strategic planning approach in most corporations is still heavily biased toward single-point forecasting. In such a context, the executive premise is, "Tell me what the future will look like; then we can make the decision." The existence of this mind-set will not benefit from multipoint forecasting and scenarios and likely cause more confusion and disbelief. They need to be ready to accept that there is no definitive scenario and to review each scenario to determine the optimal setting for each strategy component (for example, What would be the best design or technology investments for scenario A? Scenario B? Scenario C?) and review these scenario-specific settings to determine the most resilient option for each strategy element.

4.5.3 Business Challenge 03: Change

All companies must endure change to survive or grow. Our dynamic world continues to include unexpected events that cause disruption and uncertainty. The dirty little secret of change is that there is no theory for change. Change is the heart of leadership, and leaders must understand its context before designing and implementing any change program. They must anticipate everything possible to avoid and manage resistance to change. Rather than suppressing it, they must encourage an expression of concerns and critiques. Leaders also need to avoid people thinking that change is for change's sake.

Organizations need to plan for change. At a minimum, they should be able to effectively react to problems as they arise. At a maximum, they should know how to anticipate change and capitalize on opportunities that emerge from it. Simply stated, an organization that not only is prepared for but expects change is one that can overcome challenges.

As I mentioned at the beginning of this book, we are living in an age where change is reshaping industries and categories. Whether it's the bursting economic bubbles of the past decade, shifts in regulations, competition from emerging markets, new consumer expectations, or the impact of consumer conversations on the role, value, and legitimacy of

brands, the business landscape today is very different from the one most current chief executive officers (CEOs) first entered.

The research and literature on change indicate that the number one reason for the success or failure of a change initiative hinges on the leadership skills, level of energy, and knowledge of the individuals responsible for leading the change. In light of this, one obvious question arises: What leadership behaviors or competencies are most strongly associated with effectively leading or overseeing change initiatives? You won't get a straight answer for this.

Work environments that nurture the ability to change and encourage employees to develop new and creative ideas will almost always outlast their competitors. Many companies cannot meet the challenges of change because it is easier to stay satisfied with current processes instead of working toward improvement. Striving for change and growth is what keeps companies competitive.

But most companies are designed and built for efficiency and predictability. Few are flexible enough or agile enough to handle the full scope of change as it exists in the real world. Wide-stemming cultural changes regarding how and why people use goods and services are not evaluated correctly through market research. Employees are not empowered to creatively frame solutions to endemic organizational problems. And most companies find it easier to deal with glitches in the system as they arise, rather than be sensitive, intuitive, and creative in the development of preemptive change strategies. This creates a chain reaction of error. For many companies, this resistance to change is the beginning of a slow and continuous decline. Products become obsolete. Brands become irrelevant. Organizations become complacent.

Organizational change ultimately comes down to dealing with three components:

1/ Discrepancy

"We have a strong case for change."

2/ Appropriateness

"We have the right strategy and stakeholders are on board."

3/ Efficacy

"We can handle it and are committed and confident we will survive."

A company's agenda for change might be strategically sound and seemingly able to pave the way forward for years, but visions are susceptible to error and unforeseeable events. To help guard against the chaos of change, a company can use sense making as a means to remain strategically agile and nimble.

Design Thinking Approach 03: Sense Making

The design thinking approach to sense making is both agile and adaptive unto itself. It employs a broad range of "senses" and various techniques to identify, collect, question, and interpret the meaning of increasingly complex situations. As unexpected events continue to cause disruption and threaten prosperity, companies must endure change to survive. A plan is needed—not just as a reaction to change, but also in anticipation of it. It is important to realize that you will need to apply other design thinking tools and techniques to change as well. You will need to recognize where you are today, where you want to be in the future, and what it will take to make that transition. Sense making is a required capability for developing change competency.

Sense making can be a one-time or continuous effort to understand connections and insights in any particular context in order to anticipate their impacts and then act effectively on them. Quite literally about making sense of what's going on, sense making takes an obscure situation that is clouded in uncertainty and complexity and makes it more understandable for decision makers. Here, neither the frame nor the data are locked into place. The frame informs the data, and the data, in turn, inform the frame. Sense making is more than just a process; it's a mind-set that is instrumental in the commitment to understanding, learning, and improvement. Once we have a better idea of what is going on in our world through sense making, we will be more equipped to apply our other capabilities of visioning, imagining, and connecting.

A plan is needed—not just as a reaction to change, but also in anticipation of it.

In business contexts, the design thinking approach to sense making tends to lean toward the qualitative, rather than the quantitative, side of signal processing. Design thinking employs sense-making techniques to understand, question, and confront change so that businesses can actively construct, rather than be passive victims, of the imminent. Seeking out the social, technological, economic, environmental, and political drivers and conditions that shape the behaviors, values, and motivations of people and organizations, sense making inspires teams to pursue new questions rather than lock on to simple, often old answers.

Karl Weick, the organizational psychologist and father of sense making, described the process as "structuring the unknown [by] placing stimuli into some kind of framework [that enables us] to comprehend, understand, explain, attribute, extrapolate and predict."

This activity begins by encouraging and developing a heightened state of awareness and turning up every level of sensitivity. This may come more naturally to some than to others, but it can be nurtured through new relationships and by establishing dialogues with unconventional sources and networks operating beyond the business core.

From Sensing to Sense Making

From the ever-shifting external market conditions that threaten to redefine demand, to the people, processes, values, and beliefs that continue to shape what, why, and how companies work from within, businesses are complex organizations that must remain sensitive and agile to change. Design thinking uses sensing techniques to look for explanations and answers in terms of how people see things and their interrelationships. Sense making suggests that organizational issues—"strategies," "breakpoints," "mental-models," "goals," "aspirations," "core capabilities," "teams," and so on are not things that one can find out in the world or that exist in the organization. Rather, their source is people's way of thinking.

Externally, sensing means understanding consumers and culture, not data-driven markets, so that even the minutest of signals of change can be evaluated before they become sizable shifts. Internally, sensing involves qualitative, people-centered inquiry: understanding the meanings surrounding the rules, practices, relationships, technologies, and contexts that define us as people. Because these are fluid and require some intuitive read on their trajectory, sensing requires being open to and thriving in ambiguity, not seeking hard data point explanations for phenomena. Ultimately, it doesn't solve. It widens the scope of awareness so that the status quo can be questioned and revised.

Sense making is the process by which design thinkers understand experience. It can be used to learn about a sudden shift in markets, value migration between industries, emerging behaviors associated with disruptive technologies, or the reason why a previously successful business model expired and needs a redesign. To do this, the process of analyzing inputs—such as consumer insights, foresight weak signals, business objectives, competitive analysis, and so on—requires moving back and forth from the simple to the complex. Then you go back again to challenge assumptions and identify new knowledge, perspectives, and appreciations for conditions.

Sense making is as much about pattern recognition as it is about anomaly detection. Although discovering outliers through an expansive outlook is useful, sometimes the outcomes reveal trends or trajectories that can be leveraged in the innovation process through one key ingredient: timing. Timing is a critical factor in any innovation pipeline. Through sense making, organizations can get a better sense of the timing required to design and launch a new product or service.

So, how does an organization redesign itself in order to incorporate an internal sense making capability?

1/ Improve the senses to increase agility.

Design thinking promotes a more dedicated effort to adapt and acquire the tools, methods, and resources needed to sense changing behaviors. It also encourages us to remove the filters and dogmas that limit our exposure to, understanding of, and appreciation for the world around us.

2/ Collect the real data.

Sense making requires robust inputs: deep consumer insights, weak signal scanning, competitive analysis, soft and hard leading indicators, and, of course, business performance metrics against which everything else will be measured.

3/ Building sensing capabilities.

Empower individuals or teams in your company to become sensors. Encourage all members of your organization to collect and share signals regularly. Formalize this process and make the content accessible to encourage dialogue and collective sense making.

4/ Cultivate sensing networks.

Encourage cross-functional, cross-market, cross-industry, and cross-cultural connections within professional or social networks outside of your existing department, division, organization, market, and industry context to discover what's going on in adjacent worlds. Learn about the topics of interest, perceived opportunities and threats that are being discussed, and the language and mental models surrounding them.

5/ Leverage social media.

Build a broader automated sensor network through conversational platforms. Identify the critical nodes and monitor them regularly, cross-reference with other sources, and drill into salient topics as they emerge.

THINKING POINTS

You've seen all those photos of design thinkers or creative consultants surrounded by Post-it Notes and probably wondered what they were doing. In many cases, it's sense making. But sense making is about more than using Post-it Notes.

Sense making involves the process of creating mental models or mental maps that serve as memory representations with a salient visual imagery component expressed in terms of concepts, ideas, and knowledge. They are used to explain events, not isolated stimuli, which helps with connecting the dots and making decisions in complex and dynamic environments. Every organization needs to find visual, interactive, and "movable" ways to organize the raw inputs of sense making that, well, make sense to it. Don't get romanced by looking cool. Be effective: Define the best way for your organization to identify, organize, cluster, and prioritize the information on the wall.

Sense making is not a linear exercise, and it is not a process that turns information into insight. Sense making doesn't always have clear starting and ending points. *Visualization* is often used interchangeably with *sense making*, but visualization is not just a shared image with intent; it also implies the proactive use of shaping actions to reduce risk and uncertainty, probing actions to discover system effect implications or opportunities, and modeling actions that include behavioral and cognitive aspects to test and/or transform the environment. Visualization is central to sense making.

4.5.4 Business Challenge 04: Maintaining Relevance

All brands need to establish visibility, purpose, meaning, and credibility to be considered relevant in a category. Through innovation, an organization can elevate itself above its competitors and render them largely irrelevant to consumers. This is quite different from brand preference. Relevance is felt deeper and can create a clear divide between brands.

Innovating in a white space category or subcategory is imperative for market success today. In virtually all categories, from beverages to computers to financial services to air travel, marketing and product improvements rarely affect the sales or profits of a brand because of habitual consumer behavior. Meaningful changes in sales almost always relate to an offering that is created through substantial or transformational innovation.

The expectations of consumers are rising at the same time that many brands are becoming more resourceful and savvy at gaining attention and tailoring their unique selling propositions and reasons to believe to fit the market. But customers are becoming more demanding of companies to stay relevant to their ever-changing lifestyles. Relevance is extremely difficult to maintain long term. Over time, brands must rethink and redefine the value that they bring to consumers. It's getting tougher to lock in on value propositions that will truly satisfy. Value redefinition is a design approach that helps develop a new voice and meaning that will not only resonate with consumers but also sideswipe the competition.

Design Thinking Approach 04: Value Redefinition

Design thinking seeks relevance by promoting harmony with the identities, aspirations, attitudes, beliefs, needs, and desires that shape the ways people perceive and define value. It recognizes that individual perceptions and interpretations of value are constantly in flux and therefore aims to identify the underlying forces influencing this change. It aspires to develop greater empathy among people, brands, and business by observing, engaging with, and listening closely to people. To do this, the focus must be on establishing, framing, and stimulating the right conversations that will power up real insights that help businesses redefine value and maintain their relevance in the face of change.

The design thinking approach to redefining value begins with people, not products. It seeks to locate the functional, emotional, social, and cultural values that already exist within or can be designed into a brand's DNA and align those with the current and emerging values of consumers.

There are many ways to identify and interpret the shifting and relative nature of value. On the brand side of the equation, being honest and authentic is the best policy. Let's face it: Brand teams are so immersed in their daily challenges and their particular brand culture that they fail to see the forest for the trees. Having oversubscribed to the mythology of their brand, few of them are ready, willing, and able to admit what they can't admit: The soda is not refreshing; the detergent is pumping toxins into the water; the mobile phone is clunky; the service sucks; it's not really about convenience; and you're nothing but a me-too innovation organization.

“Price is what you pay. Value is what you get.”

—Warren Buffett

On the consumer side of the equation, rigorous and empathetic human-centric research is the best practice. It’s simple really, and we’ll get to it shortly: You will never truly understand how people define value of products and services through online surveys, consumer panels, and focus groups.

There is a complex and dynamic suite of factors that combine to influence our notion of value. We associate value with the satisfaction or fulfillment of a need. That is, value is associated with a product, service, system, artifact, or relationship that provides a means to a desired end. From another perspective, value is linked to actualization. Something is valuable when it serves a meaningful purpose and provides a benefit that equals or outweighs its cost.

This perception of value is shaped by factors such as personal experience, needs, wants, desires, and expectations. In addition, inherited social cultural norms and the sheer force of fitting in socially influence our notion of values. But many of these inherited norms are more fluid today than ever before, constantly being shaped and reshaped by new information, ideas, relationships, and opinions that challenge definitions of value.

In setting out to manage customer value, you may grapple first with the concept of how customers perceive value and how they are influenced by marketing or preconception or how technology or emerging behavior trends are shaping how value is defined and delivered. Customer value is at the core of any competitive strategy and is often least managed, often resulting in individual marketing, brand, product development, and pricing decisions being made rather than a conscious strategic and design exercise being undertaken.

Here’s a starting point to clarify how customers perceive and define the value of your brand or business:

01/ Identify the functional, social, cultural, and historical reasons that have driven value for your brand, product, or business. How has it been meaningful before, and why? Now, compare that to today and identify the drivers and intensity of those changes.

02/ Determine how your key customers rate you versus competitors on these value drivers. Represent these value drivers through visual mapping to illustrate the differences between key segments.

03/ Define and articulate each of these value drivers in the context of the users.

04/ Identify the rate of change on each of these dimensions and look for signals to confirm which ones are slowing down and which ones are accelerating.

05/ Conduct a workshop to identify opportunities to redefine value ahead of the game on those dimensions that have the largest gap between how they are being served and what their needs are.

06/ Design and conduct a participative design session in which you invite customers to talk about these dimensions to validate your assumptions and allow them to co-create new value combinations.

07/ Analyze the results and conduct a value-mapping workshop to explore how to redefine value to change the competitive landscape. The success of Ikea, Netflix, Zipcar, Nintendo, Amazon.com, Salesforce.com, Zappos, and EasyJet are all classic examples of companies that have been successful in redefining customer value to change the game.

THINKING POINTS

What is our current understanding of customer value? Have the top three attributes valued by customers been validated recently? Which of them are still relevant, and which ones are definitely not? Consider the following nonexhaustive attributes when thinking about customer value.

- **How can you solve my problem quickly?** Value is based on the simplest, cheapest, and most effective ways to solve a customer problem or complete a job. Focus on those three that will help you to secure a strong position.
- **How can you solve my problem the way I want it?** Value is based on a high degree of personalized preferences and is not a one-solution-fits-all scenario.

- Focus on customization and helping customers see that their problem and your solution are somewhat unique and that you are ready to tackle any kind of problem within a certain domain.
- **How can you solve my problem anytime, anyplace?** Value here is defined by your high degree of readiness and accessibility, and even the customer is expecting to pay for the extra cost. Focus on the service, reliability, and experience and define your market based on how availability is important for those segments.
- **How can you solve a problem for me that I don't want to know about?** Value here is defined by how you can eliminate some of your customer's problems with as little involvement as possible from the customer. Focus on how you can help them handle those parts of the tasks that most people would rather not have to deal with or even hear about.
- **How can you solve a problem that I don't even know I have?** Value here is driven by your supreme leadership in any domain. Your customer will believe whatever you communicate and will trust that you are thinking ahead and can anticipate problems throughout a job.

4.5.5 Business Challenge 05: Extreme Competition

So much shareholder value has been destroyed in the past 10 years as a result of mismanagement, poor strategic decision making, and an inability to react to disruptive innovation or extreme competition. Traditional competitive strategy often leads to further commoditization. Look at almost any industry, and you will find companies struggling to differentiate what they have to offer from everything else in the marketplace. Differentiation needs to be relevant and meaningful, so it's hardly surprising that one of the most common complaints from senior executives is: "My product is becoming commoditized, and the pace of commoditization is accelerating. What should I do?"

If business decisions and their tactical approaches were made through purely logical and analytical means, our world would look very different. We would arrive at solutions optimized for efficiency. Competition would be reduced down to highly predictable shifts. And consumers would see all the world's products and services as interchangeable commodities.

But competition in business is not shaped by objectivity. It's shaped by creativity and innovation that accelerates change and drives differentiation. This makes conducting business less like tic-tac-toe and more like chess, where, despite the availability of known moves and patterns of behaviors, the best business leaders deviate from the most common paths and take new avenues toward victory.

Today, navigating those paths is more challenging than ever, thanks, in large part, to overcommoditization. Although some products and brands stand apart from the crowd because of key factors such as craftsmanship, quality, heritage, and long-standing semiotics of value, the majority of products and brands out there are suffering—or unknowingly preparing to suffer—from a lack of differentiation.

When all you have is another hotel brand, energy drink, luxury purse, smartphone, or hybrid car, what do you do? The answer might be experience design. Innovating through experience design offers companies a high degree of differentiation in some of the most ubiquitous product and service categories.

Design Thinking Approach 05: Experience Design

Experience design is a holistic and multidisciplinary approach to creating meaningful contexts of interaction and exchange among users and products, services, systems, and spaces. It considers the sensation of interactions with a product or service on physical and cognitive levels. The boundaries of an experience can be expansive and include the sensorial, symbolic, temporal, and spatial. It can include tangible customer value as well as emotional value. It's not just about interface, usability, or customer service flow. It's about the word: *experience*.

Experience design is an established set of design thinking practices that, when performed properly, can enchant customers and create a sense of loyalty that will keep them keep coming back to you every time. By maximizing each and every dimension of an experience at every stage in, for example, a shopping experience—discovery, interaction, departure, and postdeparture—it considers how customers engage with a product, service, or brand in order to set the stage for something truly different, special, and even magical.

"Focus on the journey, not the destination. Joy is found not in finishing an activity but in doing it."

—Greg Anderson

To set this stage, organizations must consider and evaluate not only how a product or service functions but also how it communicates meaning and intent. Experience design highlights the importance of developing a clear understanding of consumer needs, cultures, expectations, assumptions, and capacities. How an experience speaks to consumers will ultimately determine how one product or service feels different from another.

Design thinkers critically observe and evaluate the various experiences Figure 4.12they encounter throughout their day and reflect on how one may differ from another by asking, What makes a better experience, and why? In fact, most of us do this on some level every day, especially those of us who share our experiences with others in a highly opinionated, social media–driven world.

With increasing competition across multiple categories, the distance between one brand's price, features, and design and another's is becoming shorter and shorter. Intuitively, however, most of us know the difference between Tom's Shoes and Converse, United and Southwest, Target and Walmart, Taco Bell and Chipotle, and Dodge and Mercedes-Benz because we have experienced one or both.

Design thinking seeks to explore the wiggle room between brands like these and transform it into a competitive chasm. When that difference is or might be emotional, design thinkers ask: When does the value of a product exceed its functional value? At what point does emotive value reach a point of diminishing return? What is the value of the emotive premium? Should there be an emotive premium at all? Can we engineer emotive elements into the design without adding cost?

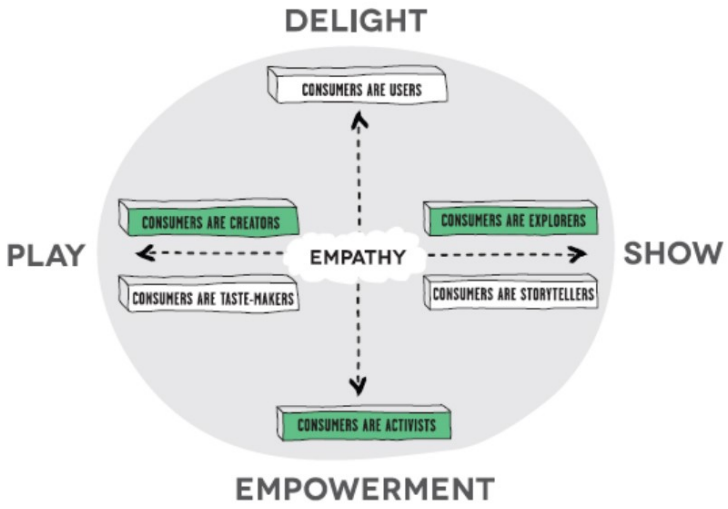


Fig 4.12 Experience Design

The answers to these questions lie buried in the nature of experience. All experiences are functional, social, cultural, and personal. They are important, relevant, and meaningful to people. They have a past, present, and even a future subject to reflection and reflexivity. To communicate and reinforce a value proposition, they need to be thoughtfully facilitated through the purposeful design of brands, products, services, interfaces, and interactions.

How?

Social experiences. Social is about more than Facebook and Pinterest, okay? How can you make your brand more social? Can you connect people? Can you help forge meaningful relationships? Could your stores be redesigned to transform them into more social spaces?

Learning experiences. Can your brand, your stores, or your online property shape learning and mastery, where your customers have access to experiences that make them smarter, faster, stronger, or better?

Meaningful experiences. Beyond social and learning, how else can you give your customers the opportunity to experience greater meaning in their lives through your brand? If your first

instinct is to jump to cause marketing, think again. You need to do more than increase impressions by giving away money.

Geeky experiences. Who doesn't love to geek out on what they love? In the design of stores, for example, consider the immediate expression of experience that walking into an All Saints store with their sewing machines in the window does for apparel geeks or the huge digital displays at Burberry do for fashion fanatics who want to feel the runway.

Understanding the Four Key Dimensions of Experience Design

Because human experience is multidimensional so, too, is the design thinking approach to it. Here are four ways to get started:

1/ Determine the scope of the experience.

Decide on a starting point and an end point. You can expand boundaries, but you need to have them at the beginning. With scope comes the breadth of interactions in store, marketing, digital, and beyond. Where and why do you want to or need to engage customers?

2/ Understand the intensity of experience.

Some experiences are simply a routine way to get something done quickly and effortlessly. Some are complex, intense, and highly emotive. You need to analyze and prioritize your effort based on those parts of the experience that are both highly rational and emotive. That's where the intensity is high, as well as the stakes.

3/ Identify the key experience triggers.

Experiences are triggered by stimuli that elicit recognition. These could include visuals, symbols, photographs, videos, smells, and sound. Experience designers need to consider what combinations of these are the triggers for brand recognition and how to use them to support delivery.

4/ Deepen the customer's engagement to evoke meanings.

Experience design achieves the highest impact when it evokes meanings. Sometimes these are distant from the product or service, but when executed authentically (meaning it's a part of your business culture), they become very close to the brand. At the center of meanings are core values. If you can express them well, customers will subscribe to them.

THINKING POINTS

You may not have a customer experience strategy, but you have a customer experience regardless of whether you create it consciously. Every company provides a customer experience. You may not have total control over that experience, but there are always opportunities to make it emotive and memorable and to make it work for you. So how can you design an emotional experience? The implicit problem is knowing what will work or not work in terms of emotional engagement and economic and operational feasibility. It begins with using customer journey mapping to visually illustrate an individual customer's needs and goals, the series of interactions and information necessary to fulfill those needs, and the resulting emotional states a customer experiences throughout the entire process.

Customer journey mapping succeeds when these exercises are based on ethnographic research and contextual inquiry that allow researchers to experience and perceive the emotions of customers, thereby making it possible for managers to convey more than just anecdotal quotations. The outcome of the exercise shows how customers feel throughout their journey, and customer journey maps invite stakeholders to enter the world of customers and share in their experience. In turn, stakeholders are better able to convey their story to management and frontline employees. This should be the starting point for your experience design.

4.5.6 Business Challenge 06: Standardization

Standardization is a necessary cost driver for every company. It is a means to achieve operational, cost, and performance efficiencies by streamlining activities, leveraging technologies, and maintaining employee workflow to reduce operating costs. But standardizing practices can mean losing the personal touch, reducing the choices customers have, and disconnecting employees. This is the balancing act every company must master: How do you stay lean, mean, and efficient while maintaining the human qualities that will ultimately endear you to people?

To streamline operations and be as profitably productive as possible, every company seeks to better leverage the powers of enterprise technology, design rule-driven workflows, and automate repetitive tasks. It makes sense not to reinvent the wheel every time you need to go for a drive. Regardless of the reasons for pursuing it, however, the race for standardization can come at a serious and often unforeseen expense.

Like companies, many people prefer efficiency to inefficiency. We like reliability. We like consistency. And we like to do things well and quickly. But we also like a degree of feeling in what we do. When a company's primary focus is on making standardization its priority, it can lose sight of the emotional quotient of its brand and alienate consumers. Here, for example, a standardization Figure 4.13 of market research practices can fail to see real human needs, a standardization of product design practices can get sucked into looking no further than today's coolest feature, a standardization of segmentation can miss valuable opportunities to serve the underserved, and an internal mandate for standardization can take some stakeholders' eyes off the real prize of today: a customer-centric culture.

Contrary to what some design-minded authors will tell you, this critique of standardization is not based on a belief that companies that value sameness will be less successful in innovation than companies that value difference. Like people, companies are complex creatures, each with its own history, qualities, and characteristics that, when it comes to innovation, cannot be reduced down to the kind of Top Ten Ways list that proliferates at the lead of your LinkedIn page. Standardization can make internal processes more efficient and effective. It can clearly establish common goals of performance that every employee must meet. It can provide common platforms that make a supply chain run faster and cheaper. And it can be a rallying cry for product departments to make life simpler and more pleasant for consumers through the simplicity of design. Or it can't.

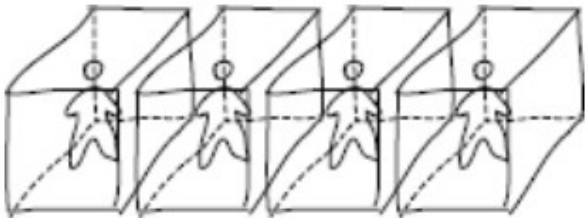


Fig 4.13standardization

One of the most common challenges that clients in industries from consumer packaged goods to technology raise with my teams when we are scoping out a project is that no matter how great an insight or idea might be, it will ultimately be subjected to company standards. In some cases, the standardization of legacy manufacturing poses a big challenge to innovation.

How do you maintain operational efficiencies without sacrificing innovation opportunities and the human lens through which they will ultimately be identified?

Although every good consulting firm worth its billable hours understands that it's going to be a very long, uphill battle to get a company that sells circles to consider squares, clients—particularly in the world of packaged food—are often quick to point out how their company's innovation capabilities are restricted by factors such as size, shape, and ingredients. We all know why the machines were made the way they were made 20 or 30 years ago—efficiency—but that shouldn't stop us from considering a day where squares sell more than circles because consumers need or want them. Here, the standardization of the past can restrict the opportunities of the future.

In other cases, we deploy anthropologists to perform contextual inquiries and they use the simplest, most concise, and most engaging language to communicate the value of an insight that they believe could be leveraged to create real competitive differentiation for a client. But when the client's market research department has such a fixed and highly rigid way of approaching, thinking about, and talking about customers and how it identifies their so-called needs using words such as *target, segment, actionable*, and the worst ever, *reason to believe*, standardization becomes the enemy of innovation. It restricts vision. It dampens empathy. And it ultimately causes companies to miss out on potentially transformative opportunities.

The balancing act of standardization is a difficult one: How do you maintain operational efficiencies without sacrificing innovation opportunities and the human lens through which they will ultimately be identified?

Design Thinking Approach 06: Humanization

Great design refutes standardization in favor of the gentler, more human, and more emotional aspects of customer experience. It's the joy, pathos, laughter, and anxiety found in each can of soda, user interface, and public transit terminal that customers might not be able to openly describe but nonetheless deeply feel. Design thinkers are sensitive to the human touch points that encourage and foster such emotions as profound moments of attachment to a product, service, or brand.

The lack of humanization Figure 4.14 in experiences is not always purposeful but rather naturally occurs as standardization takes hold. A physician who treats scores of injured patients during an emergency must detach from emotion in order to give everyone what they need. Similarly, products, services, and business models naturally detach from the softer side of life to operate efficiently. But ultimately, customers don't care about the operations behind their favorite coffee brand; they care about the exhilaration they feel when they have a cup of coffee in their hands.

Design thinkers remind businesses that they are ultimately responding to human values, beliefs, and needs. They understand that efficiency and standardization will always have a place in business processes but recognize that it's the human touch points that resonate most in real-life customer experience to give products, services, and brands true value and meaning.

Fig 4.14 Humanization

Humanization From and Within Culture

Great design is only as good as its ability to connect and forge relationships with the people who ultimately use it. This means that the first task of great design is to understand the cultural touch points to humanization that make such relationships possible.

Culture is the shared values, beliefs, and performances that define a group of people and their behaviors. Without a thorough understanding of culture, businesses cannot understand the human emotions that either make or break customer experiences. As a valuable tool in design thinking, culture is a springboard for unpacking opportunities for humanization in design.

Understanding culture means unpacking all the social meanings (and emotions) that define a particular customer's experience. For example, drinking coffee can involve the emotions associated with enjoying that wistful first cup on a foggy Saturday morning, recovering from

a college all-night study party, or kicking back for an after-dinner "release" after overindulging at a favorite restaurant. It is a unit of cultural capital for the barista, the coffee connoisseur, and the housewife who wants her special treat when the kids take their nap. So many emotions and contexts define the coffee experience. Design thinkers unpack each coffee context in search of humanization opportunities. From this, a more holistic picture of drinking coffee in the real world emerges—defining what coffee means to consumers in their everyday lives. More culturally relevant designs ensue, providing brands meaningful customer interactions that stand the test of time.

Humanization doesn't just come from culture; it is also produced from within cultures. Designers, like the businesses they work for, are people who impart social values and beliefs on the things they produce. Design thinkers seek to understand the cultures not only of others but also of themselves, recognizing that their own emotions, practices, and belief systems inform what, how, and why they do what they do. To balance this, design thinkers encourage a dialogue between everyone in the product or service ecosystem by consulting with the company, their partners, brand representatives, and end customers.

"The essence of being human is that one does not seek perfection."

—George Orwell

Humanizing Through Language

Consider the language that most businesspeople use today: *target, driving performance, leveraging outcomes, actionable*, and other taglines that make managers feel like they're in some kind of special ops group. Here, the language of power, pragmatism, and predictability creates the illusion that businesses are machines. They're not. Businesses are organizational cultures filled with human beings who define and attach value to what they do every day. And, oh yeah, they are also organizational cultures that should be fulfilling human needs.

If all businesses are human enterprises that produce things made for human beings, it's time to start humanizing the business narrative. Design thinking seeks to reinsert human-centered qualities that can introduce new meaning. This means using real talk about the personal histories, dreams, and desires that define each worker, team, business unit, and office to produce human narratives of company culture that resonate worldwide. Remember: Each worker isn't a cog in the wheel. The suit that goes to work also goes home. It's worn by a person with a meaningful life—and that's a valuable brand story of not just what a company does but who a company is: its people.

THINKING POINTS

One route to greater humanization is reassessing how your organization does research on consumers and talks about or represents them. On the research front, consider hiring people who are specialists in human culture: sociologists, anthropologists, and other social scientists who specialize in understanding us without putting us in focus group facilities and looking at tracking studies. On the talking and representation front, remember that language structures all cultures; organizations that talk about *targets* and *segments* distance themselves from people with every utterance.

Humanization can be leveraged by usability, human factors, customer experience design, and brand storytelling. All customers, employees, and their friends are part of your story network and can all be part of your humanization design; if you let them bring their humanity to your brand or organization, they'll also bring your brand into their networks and their lives. Brands that have been humanized attract and sustain communities of real live people and make customers more forgiving when organizations make mistakes.

4.5.7 Business Challenge 07: Creative Culture

Big or small, many companies fall into the trap of trying to innovate the "right" way. By spending so much time, energy, and resources on getting it right and avoiding failure, not enough time, energy, and resources are dedicated to playful, experimental creation.

Innovative cultures are creative cultures. Few of us would disagree that those companies that innovate well have some degree of creativity in their DNA. But having a creative culture is not about a workplace that's filled with funky chairs, table tennis games, a video game zone, or Post-it Notes. It's about recognizing, even celebrating, uncertainty and ambiguity from the get-go and applying your passion and skills—whether they live in the right or left side of the brain—to come up with ideas, find solutions, and solve problems in new and strategic ways.

Creativity cannot be taught. Some companies run occasional creative workshops with designers; hold creative events featuring artists, architects, or chefs; and encourage creative projects on company time hoping this will promote it among employees so that they come up with more innovative solutions in their daily work. Others companies, such as Pixar, whose Pixar University runs more than 100 courses on everything from improv to cooking to self-defense, nurture the creativity of employees as an ongoing, in-house initiative that supports a macro organizational attitude. For those who do it well, however, the creative capital of teams and companies is earned only through experiences of trial, error, and occasional failure. Here, the name of the game is experimentation, and when done well, it nourishes a culture of curiosity and excitement.

Because most companies are run by the numbers and tend to be very risk-averse, the attitude toward creativity in those whose bottom line is not connected to a creative output (such as animation) tends to be very narrow when it comes to the softer, fuzzier, warmer side of innovation. In fact, many such companies are actually fearful of creativity, thinking it's a plague against efficiency. They're wrong.

If an organization's culture is not engineered or nurtured to truly be innovative and to take risks, it will surely fall victim to competitors that are more tolerant of or driven by creative experimentation. In a business age fueled by how well new ideas do in the face of intense competition and overstandardization, creativity and design thinking need to be the centerpiece of any company.

In 1995's *Defying the Crowd: Cultivating Creativity in a Culture of Conformity*, American psychologists Robert Sternberg and Todd Lubart argue that "People are not born creative; rather, creativity can be developed." Putting forward their investment theory of creativity, which describes an influence of the environment on the judgment of creativity, they suggest that the relationship between people and the environment is like that between investors and a

stock market. Here, creative people are like good investors who buy low and sell high. Innovative ideas might not be considered by the market to be creative at first, but these people try their best to change that judgment in order to sell high. Once the market has been convinced of the creative value of the idea or product, these people move on to the next idea that will, not surprisingly, be unappreciated at first. Why are these people so creative and so tenacious? According to Sternberg and Lubart, they are able to draw on six resources that are critical to cultivating creativity: intelligence, knowledge, thinking styles, personality, motivation, and the environment.

These are the raw ingredients for building a creative culture. Design thinking offers many ways to help a company develop such a culture, but the one that stands out the most—that encourages trial, error, experimentation, and creative play—is rapid prototyping.

Design Thinking Approach 07: Rapid Prototyping

Rapid prototypingis an iterative learning process that acquires and expresses increasingly complex information of higher fidelity over time through repetitive and cumulative cycles of build, test, see, and refine. Figure 4.15 Prototypes are traditionally used in design and engineering environments, but when they become integral to other organizational cultures, they serve to develop a creative mind-set that is able to effectively focus on imagining, socializing, and testing any idea, including work processes, team structures, business models, and, of course, products and services. Because every output of design thinking begins as a prototype, think of them as part of a culture and not specifically as a tool.

Most businesspeople are familiar with the “fail fast, fail cheap, and fail early” concept. Design thinking suggests another dimension to this rule: “Learn fast, learn cheap, and learn early.” Prototyping is the way to open up that dimension; it’s a relatively low-cost, hands-on activity that helps bring people on to the same conceptual page, uncover new knowledge, and identify and mitigate design and development risks early on. This is done to avoid downstream costs while also building up critical assets for the internal and external communication and socialization of ideas. As part of the hands-on approach, prototyping typically seeks to involve and engage multiple stakeholders and so-called end users as participants at every stage of iteration, from paper to final production.

Fig 4.15Rapid Prototyping

“You may never know what results come of your action, but if you do nothing there will be no result.”

—Mahatma Gandhi

Prototyping is a currency for creative dialogue. It embraces play behaviors as a critical component of imagination and ideation. By creating multiple feedback loops through employee-to-employee iterations and employee-to-user testing and refinement of illustrations, models, or product and service mock-ups, it is a way to produce new knowledge and make ambiguous concepts more tangible. The concept is quite simple: It’s much easier for people to articulate what they want by playing with prototypes than by enumerating business requirements in Excel or PowerPoint. Given that almost every organization claims it wants to design a more agile process for faster product development, it’s shocking how few of them utilize rapid prototyping as a way to accelerate learning and speed up feedback loops that can assess potential implications and generate shared mental models of the business problem.

The Benefits of Prototyping in Business Design
01/ Increase team ownership.

Rapid prototypes can ensure that all those leading or affected by changes to a company are involved in understanding and perhaps even shaping the future in a context that comes to life in a more tangible way.

02/ See the possibilities.

Rapid prototypes expose potential solutions, obstacles, and unanticipated outcomes of an idea. By bringing everyone on to the same page to discuss and develop solutions, they create a shared understanding of a current vision or a future state.

03/ Deepen insight.

Rapid prototypes let you see the big picture without losing sight of the details. By utilizing them in context with both stakeholders and potential users, teams are able to better identify the kind of usability issues, workarounds, gaps, brand connections, and opportunity extensions that drive development and refinement.

04/ Promote cross-functional collaboration.

Prototyping facilitates social bonds among employees who may not have the chance to interact and empathize with one another. The experiences, artifacts, and stories that emerge from prototyping become a part of that group’s shared narrative, which can be passed down and transmitted to others within the organization. In addition, the bonds that form among employees engaged in co-creative activities increase relationships by reducing assumptions, stigmas, and other barriers to communication.

05/ Improve visibility and predictability.

Rapid prototypes provide a blueprint for change that can be as detailed as needed, leading to a clear future state with concrete milestones and transparent progress. Whether it’s about designing a product, a service, or a new way to work within a company, this results in increased predictability.

Building, Fostering, and Embedding Creative Confidence

Prototyping plays an important role in developing a more engaged and creative organizational culture by encouraging people to bring their ideas to life in a more tangible way. This makes it easier to share ideas with others and to leverage feedback to make improvements upon them. Encouraging and enabling aspects of prototyping across an organization can help foster a more creative culture that is more comfortable, confident, and capable in its ability to propose, explore, and learn about innovative solutions to problems and opportunities.

It is important for leaders to support a culture of prototyping with the right resources for creating, for example, dirty spaces—workshops or labs for making stuff with simple tools and basic materials such as paper, markers, foam, clay, cardboard, tape, and even computer-aided design (CAD) software or 3-D printers. There's no need to invest a lot of money; just demonstrate a commitment to prototyping and a willingness to encourage and support it in a tangible way. In addition, training sessions that introduce specific techniques and processes can be adopted to help improve existing skill sets and the quality of the prototypes.

THINKING POINTS

Remember the excitement of show-and-tell when you were a kid? Or the thrill of story time? Prototyping can bring that magic back and facilitate the kind of dialogue and learning among employees that is sometimes lost. By promoting prototyping events where employees can showcase and share their work and ideas, companies will learn to work better and move faster to take advantage of new and disruptive opportunities.

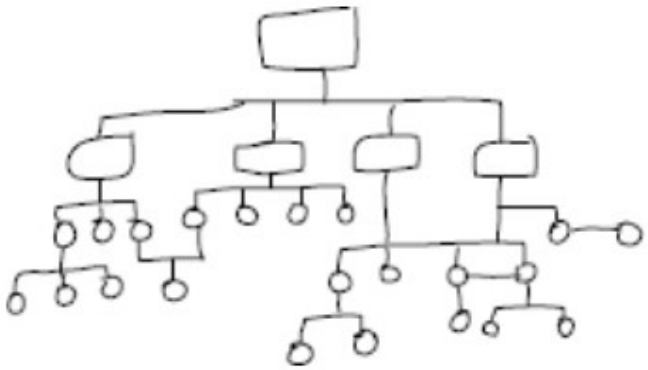
Everyone can participate in rapid prototyping. You don't need to be creative. You need to be willing to try—sketch, draw, cartoon, map, make, and show. However rough and raw the prototype might be, every employee can get an idea down on paper to share and develop an idea.

Business Challenge 08: Strategy and Organization

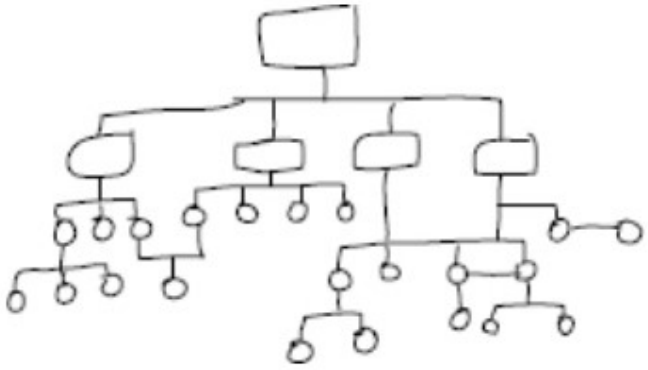
Not every company needs to make radical changes to its core business, but when the case for change is desperately obvious, it's desperately obvious. Consider the cases of Kodak, Sony, Palm, and GM: In each case, a business model redesign of structures, systems, cultures, and capabilities was driven by strategic innovation in order to play a new game.

In cases like these, such transformations may include strategic decisions to move upmarket, move downmarket, or shift from products to services and solutions. When the pace of commoditization continues to accelerate or the industry's boundaries open up for new entrants, this type of transformation in critical. This is where design thinking comes into play.

New disruptive business designs often leverage new and emerging technologies. Through creative and innovative approaches, many organizations make large shifts toward capital-light business models through a variety of means, including the adoption of open source technologies, the development of strategic alliances, or a reconfiguration of their role in the value chains. Designing a capital-light or capital-efficient business model is in the best interests of shareholders and can potentially add real value and a remedy to stagnation.



Companies need to find a future that they can play a key role in to create value.



Increased disruption and the volatility of many business environments can make strategic planning a demoralizing exercise among executives who realize they must attempt to shape a future that is never truly known or under their control. Tasked with managing today's business activities and cash flow, most get trapped in calendars not designed to enable them to be flexible enough to deal with tomorrow's changes in the marketplace.

Leaders and executives are often expected to have visionary capabilities, able to see into the future and deliver a coherent strategy for their company. This is a dangerous approach to strategic planning because what happens if and when that leader or executive is wrong? Knowing this can and does happen rather frequently, and whether the process occurs inside the private mind of the CEO and executive team or with the help of consultants, most leaders hedge their bets by relying on specialized planning departments run with elaborate data systems that do little more than further embed old assumptions or organizational dogmas.

Because imagination and creativity have little or no place in the formal planning process, operational improvements usually take center stage because the benefits of them are easily quantifiable and can be benchmarked. As time goes by, however, many companies forget that there is a limit to the extent to which continual operations improvements can sustain growth. Instead of all these small and ongoing tweaks, companies need to find a future that they can play a key role in to create value. To do that requires a degree of corporate imagination in business model design.

4.5.8 Design Thinking Approach 08: Business Model Design

There are plenty of myths around business model innovation, Figure 4.16 and it is often used interchangeably with business strategy and business design. There are two reasons for its rise in popularity: (1) the proliferation of new technologies that gain consumer acceptance but a company fails to figure out a way to make money from them and (2) the intensification of interindustry competition as well as rapid emergence of disruptive challengers. In strategy, the highest form of value creation comes from business model design, and a robust business model can raise barriers and allow companies to sustain their competitive advantages for decades. Think Ikea, Apple, Zara, Amazon, Costco, and ING. A good business model is also hard to replicate because it requires interlocking components.

Some business models are more powerful than others. Their robustness depends on how value creation is matched with value capture. To innovate a company's business model, one must first understand how the previous model was defined and then examine what options and paths exist for improvement in the future. The aim is to create models that create customer stickiness, loyalty, or barriers to entry for competitors. The process is transformative, a complete redesign of core offerings to capture value in a new way.

"You never change something by fighting the existing reality. To change something, build a new model that makes the existing model obsolete."

—Buckminster Fuller

Everyone knows that a successful business model should unlock latent value from all of a company's assets. Even though so much has been written on the subject, however, the term *business model* is poorly understood. The fuzziness and confusion stem from the fact that when different authors write about business models, they are not necessarily referring to the same thing. Some are referring to parts of a revenue model (a subscription or license model); some, to specific processes (a distribution model); and others, to business strategy concepts (elements and relationships between value chain players). But only something that refers to the totality of a business's activities—not just one part—can be called a business model.

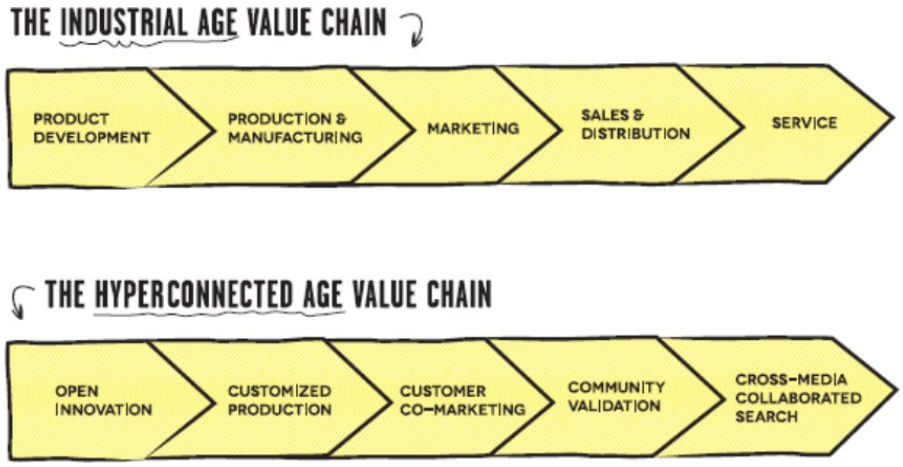


Fig 4.16Business model innovation

The Functions and Over complexity of Business Models

The functions of a business model are many. It should articulate a strong value proposition for some set of customers. It should define the structure of the value chain and how the company's role in it delivers value to those customers. It should describe the position of the company in the value network that links suppliers and customers, including the identification of potential collaborators and competitors. And it should design a value flow that illustrates where revenue will come from and how it can be optimized over time.

These days, there is a lot of talk about how some business models are overly complex or difficult to manage from a risk and operating perspective. Strategic competitive advantage is created by the combination of strategy, culture, systems, brands, products and services, and the people in the organization. This in turn creates a complex ecosystem that extends beyond the boundaries of the boardroom and indeed the business units. Often, value resides in these complex ecosystems and makes it difficult for competitors or new entrants to replicate.

Many large organizations are so complex that it is hard to fully understand how they operate and how they make money. It often seems as if they are running on autopilot. In fact, they are being run by big enterprise software systems, which interact with other specialized systems to form the organization's central nervous system.

But there is inevitably a problem that arises when the software gets too big and too many systems are running. Like robots out of control, they can behave in unintended ways and trigger unpredictable systems behaviors. Whether software is to blame for Toyota's recent crisis or not, its failure did raise a critical question: Is software today so complex that it can't be controlled? Also, what toll will the exposure of inherent bugs exact when millions of lines of codes turn into tens of billions?

This question goes beyond software. Today, everything is too complex. Some systems, like some businesses, are too complex to be allowed to fail. Policy makers worried about the next banking or other crisis argue that this complexity should be avoided or eliminated through regulatory changes. In a sense, the idea is that dangerously complex systems can't be relied on to operate smoothly or even at all. Bigness is the implicit culprit.

Recently, there has been talk about the collapse of complex business models. From an investor's point of view there is a philosophical argument about whether it is better to invest in companies whose business models are easy to understand versus those that are overly complex. But sometimes complexity can be an advantage, often a very big one. Companies that are successful in leveraging advantageous complexity and eliminating disruptive complexity in their business strategy can enjoy a sustainable competitive advantage. GE is a good example of a company that leverages complexity to its advantage. In fact, research shows that there is a direct correlation between the successful management of complex business operations, overall company performance, and return on capital.

What's the Difference Between Business Model and Business Strategy?

Organizations need to understand the difference between how business models and strategies affect decision making and performance. One question today is whether or not a business model and business strategy are one and the same. Business models were popularized when there were many start-ups that had little or no idea of how to create value or make money. As

such, we can see why a start-up can be seen as a company with a strategy but no idea of how to make money.

One case in point is Twitter. For four years they've have a great strategy but no real business model. Three years into it, Twitter hired one person whose job it is to develop a business model, but there's no real expectation of seeing major revenue anytime soon.

Any successful business model should operate under basic strategic principles, including the concepts of fit and differen-tiation. This is an old idea made popular by Harvard Business School professor Michael Porter in the 1980s. It's about lining up all of your business operations behind a central logic of value creation, and it is the most rigorous and practical strategy concept ever developed. The fit and differentiation concepts are also unlike business model design or business model innovation, which operate fuzzily around crafting value propositions, business activities, and a revenue model.

Business strategy and business model innovation diverge when we are dealing with radical business model innovation. Strategy is competing within today's defined market space and industry boundaries and perfecting a set of business activities that differentiate you from your competitors. Business model design is part of strategy innovation and is essentially about designing business activities around tomorrow's opportunities. Its main mission is to answer the following questions.

Business Model Versus Business Strategy

It is generally acceptable to use the terms *business strategy* and *business model* interchangeably, as long as we understand the subtle distinctions. Both are used to refer to all the components and activities that guide managerial decisions. There are four key distinctions between the two:

01/ Both describe the business system and activities that show how each of the pieces of a business fit together. However, strategy mainly deals with competition and sustaining differentiation, whereas business models focus on the interplay of different elements, relationships, and how value is being created, captured, and transferred. In short, the core focus is how you make money.

02/ Business strategy includes the organizational capabilities that have a direct impact on operational decisions and the organizational culture or on how a strategy is to be executed. A business model is more about how a business functions in the value chain or networks. Business model execution and resource allocation decisions are mostly neglected.

03/ A business model has a narrow focus on how a company can make money by focusing on pricing strategy, strategic partnerships, and channel choice. All of these decisions affect the top line and have as their goal the optimization of revenue or sometimes the survival of an early phase of a business. On the other hand, business strategy deals with much broader and deeper issues, including capital structure, cost structure, and asset utilization.

04/ Business models are subject to changes in technology and competitive dynamics and are therefore constantly subject to change themselves. On the other hand, once a strategy is set, it's all about the execution.

Business Model Design Framework

Forget what your company knows it can already do. Instead, begin this process by asking what value is being created today and how it aligns with established or emerging customer needs. What is the exchange of value between everyone in the ecosystem? How does the current system answer customer needs and create customer surplus while generating profit? This purposeful linking of interdependent activities performed by the company in collaboration with suppliers, partners, and customers is the essence of business model design.

A business model design exercise that puts the essential building blocks and their relationships in the value chain together makes it easier for managers to make decisions based on a better understanding of value flow and how a company makes money. They can adapt and fine-tune the business model once it is working. Explicating, capturing, and visualizing a business model will improve planning and provide a common understanding among key stakeholders. From a design thinking perspective, business model innovation leans toward exploring new approaches to the softer and less tangible qualities such as value proposition, organizational culture, values, and core competencies. Rather than building fixed assets or long-term infrastructure, there are mountains of opportunities to be discovered through more resourceful and creative tactics. Design thinking approaches to business model innovation may examine and seek to exploit the latent and potential value existing within an organization by identifying how its core competencies, resources, and assets (including intellectual property) can be leveraged in new and innovative ways. Design thinking may seek to interpret the qualities and characteristics of this latent potential, question what part of the value chain an existing model sits within, and explore ways in which the organization's latent potential or untapped assets and resources can be realigned into a new model.

There has been little knowledge or practice development to date devoted to this important undertaking. It is as much a creative exercise as an analysis exercise. Here we provide a conceptual tool kit that enables readers to design their future business models and that also helps managers analyze and rethink their current designs so that they can be fine-tuned for the future. It needs to begin with identification of emerging behavior that will affect how value is being created today and identification of what the drivers of chance are. What is the value exchange among key parties involved, including everyone in the activity system such as the purchasers and consumers, which sometimes are different parties? How does the current system respond to customers' needs and create customer surplus while generating a profit for the company?

That objective is reflected in the customer value proposition that is linked to an economic logic that requires a host of activities that include the engagement of human, physical, and/or capital resources of any party to the business model to serve a specific purpose toward the fulfillment of the overall objective.

The purposeful design of interdependent activities performed by the company itself or in collaboration with suppliers, partners, and/or customers—is the essence of the business model design. It is basically a design thinking undertaking; experimenting and mapping links between different activities will provide insights into the processes that will enable the evolution or radical redesign of a company's activity system as its competitive environment changes or the business opportunity justifies it. The purposeful design through visual modeling and activities mapping within and across firm boundaries is the context of the business model design.

The company's revenue model plays an important role in the overall business model design. It is linked to a pricing strategy for specific products or services. The revenue model complements a business model design, and the two cannot be separated, just as a pricing strategy complements a marketing strategy. They are closely linked but yet distinctive. A business model is geared toward total value creation for all parties involved. It lays the foundations for the focal company's value capture by codifying the overall size of the pie, or the total value created in transactions, which can be considered the upper limit of the firm's value capture potential.

The business model design Figure 4.16 determines the company's bargaining power based on its strategic assets, access, and capabilities, as well as the intensity of the competition: The greater the total value created and the greater the focal firm's bargaining power, the greater the amount of value that the focal company can appropriate. How much of the total value the firm actually captures, however, depends on its revenue model and if there is any lock-in mechanism for the business model, such as barriers to switch. Business model design for an innovative product or service could have very different implications; it may not be about increasing the share or the size of the pie but about creating a new pie that has not existed previously but that can be scaled very quickly. The adoption curve of that new business can impact the business model as the business scales.

When designing a new business model, one precondition is to re examine assumptions regarding what to consider core capabilities to the company and the business boundaries. Strategic discussions including Who are we? What is unique about us? What businesses are we in? How do we differentiate? How do we create and capture value? and What is our operating model? Many value chain activities are based on transactions only, and when design thinking is applied in business model design, we need to be creative and look at what existing and new activities should be considered for higher value. High value activities are the activities that either generate revenue directly or indirectly enable you to generate revenue. Unfortunately, many of the activities that companies perform are not high value but taking the same level of energy and resources to maintain operations.

Fig 4.16 Business Model Design Framework

Starting with the Concept Metaphor

To use design thinking for business model innovation, we need to start with the right conceptual metaphors to aid in the exploration of an organization's latent potential in relation to new and innovative business models. In the traditional world of business education this exercise would start from the value chain and apply the concept of fit to decide in what part of the value chain the company should choose to participate. But in design thinking, we start from the emerging customer needs and use that to map customer value clusters. Based on those cluster scenarios, we examine what value systems are not being fulfilled today. Each of those then provides relevant differentiation and sustaining market power. With that ideal state in mind, we can look at how existing capabilities can deliver against those activities.

In the process of designing the value delivery system, Figure 4.17 businesses also need to seek to identify which role—or position in the chain they currently occupy or would like to occupy—can potentially be eliminated or outsourced to a vendor to lessen capital investment, especially in times of scarce resources. This can often reduce organizational burdens while simultaneously opening paths toward innovation.

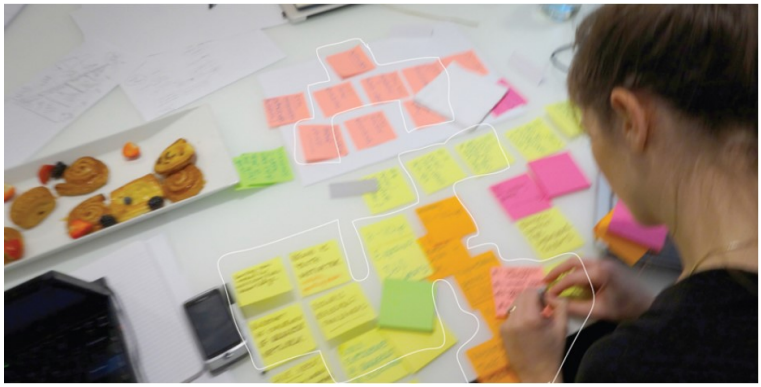


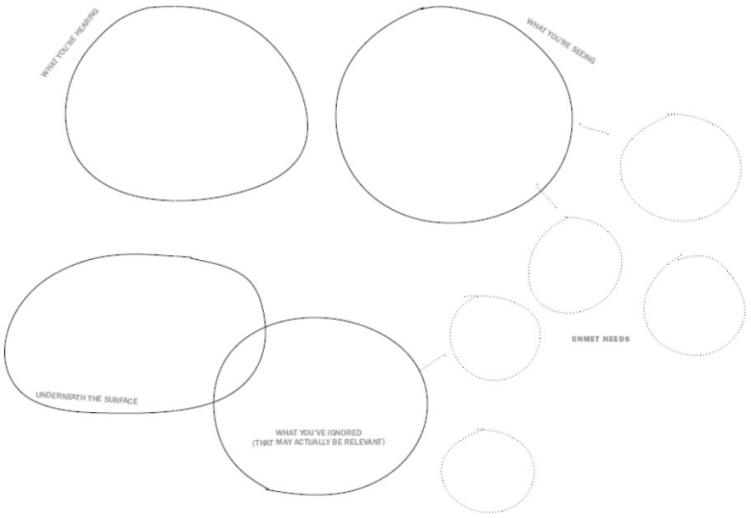
Fig 4.17 Concept Metaphor

Applied Design Thinking for Business Model Design

This is where you can apply real design thinking to real subjects. Get your pencils out and your thinking caps on.

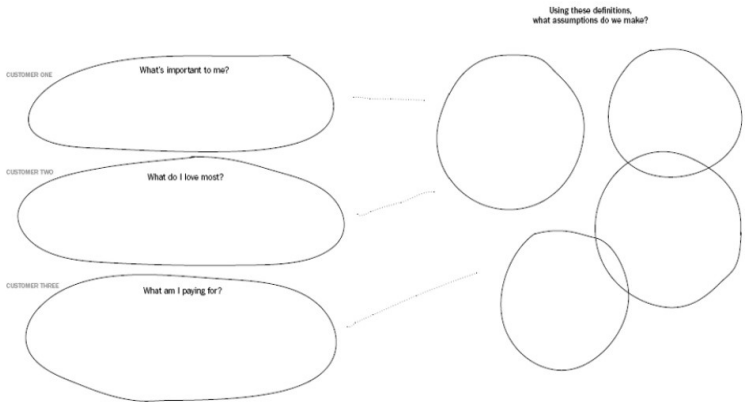
01 What are the emerging behaviors that can potentially affect the value creating activities system today?

02 What are the current unmet needs of customers in a particular category, and how will each of these be affected by these emerging behaviors?

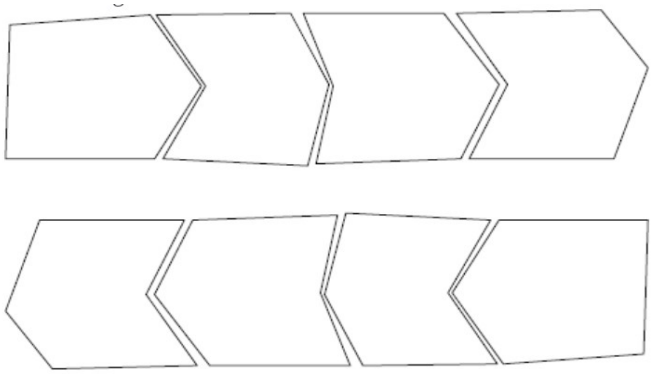


03 How does each segment define value, and are those definitions aligned with what the company thinks?

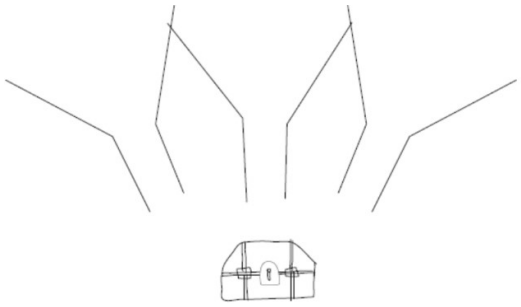
04 Identify the key gaps and explore the reasons behind what's causing the differences.



05 How is value being created and distributed in the current state?



06 Is the business model based around a single stream of revenue or multiple streams? What is the logic behind these decisions?

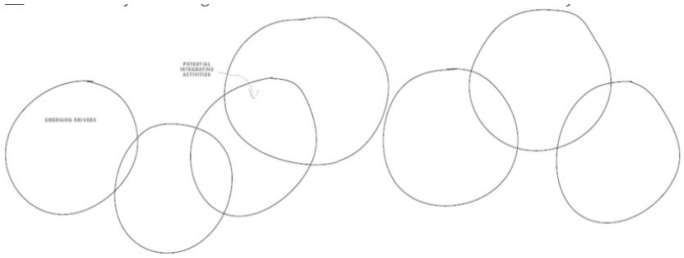


07 What are the emerging drivers for change in customer value systems? What is the potential for integrating activities?

08 What are the unique value creation activities in the system?

09 How is value migrating within the system today?

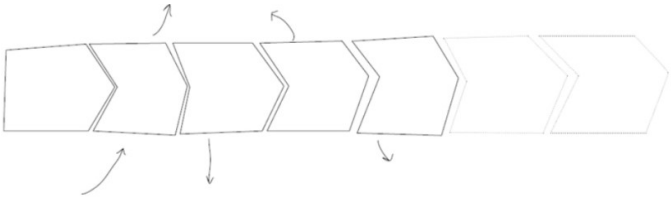
10 How do they all fit together to form the most attractive business system?



11 How does this value creation and delivery system compare to the current one?

12 Outline the distinctive benefits of the old and the new business models.

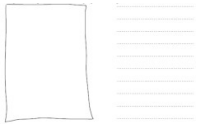
13 How would those benefits change over the short and medium term, and what causes the change?



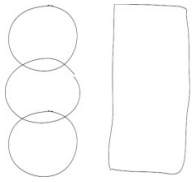
14 What are the other creative ways to combine capabilities and assets to deliver to these value systems?

15 What are the underlying revenue model options (for example, selling, licensing, subscriptions, advertising, sponsorship, transaction)?

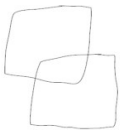
16 What are the other payer options? Can this product be subsidized? Are there bundling opportunities?



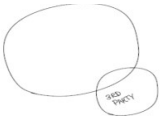
17 What are the underlying purchase behaviors affecting the revenue model decision?



18 Does our revenue model link to our customers' value or their success?



19 Does the business model rely heavily on third-party partnership to develop and provide a revenue stream?



20 At what point we need to explore changing a business model because of a change in revenue source?

21 How will the value migration evolve over the next three to five years? What is the estimated life span of this business model, and at what point does the current business model become invalid?

THINKING POINTS

The most sustainable competitive advantage lies in a robust business model design. Although on a surface level there are only limited business model options, companies competing using different business models can have very different outcomes. Analysis of business models cannot be mechanical and needs to be considered on a systems level, which often becomes complex. Network effects are an exogenous feature of technologies, and a winner-take-all end result is common among platform-based competition. Design thinking can be applied to visualize and identify the relationships and long term implications of different business models and their strengths and weaknesses. When they are performed as a simple spreadsheet exercise there is a false sense of certainty about the future. Spreadsheets won't provide users the ability to see the entire supply and demand relationship and it is the reason why so many new business models fail, even though they look robust on spreadsheets and in PowerPoint.

A business model redesign often requires transforming organizational structures, processes, and culture in addition to capabilities. This is often a missing consideration or an afterthought, a common cause of less successful business model design implementation. There is a strong link between business models and organizational form and decision-making structure. Most unique business models require the support of a unique organizational design.

References

Amabile,T.etal.*Assessingtheworkenvironmentforcreativity.*Academyof ManagementJournal,October1996,Volume39,Issue5,pp1154-1184

Website: Australian Academy of Technology, Sciences and Engineering
http://digbig.com/4wxmw

DTi,2006.*60minuteguidetoinnovation:turningideasintoprofit.*Norwich:The StationeryOffice

Luecke,R.andKatz,R.(2003).*Managingcreativityandinnovation.*Boston,MA: Harvard Business SchoolPress

Articles

Full text available from Business Source Corporate
www.cimaglobal.com/mycima

Adams,R.,Bessant,J.andPhelps,R.*Innovationmanagementmeasures:a review.*InternationalJournalofManagementReviews,March2006,Volume8, Issue 1, pp21-47

Genus, A. and Coles, A. *Firm strategies for risk management innovation*. InternationalJournalofInnovationManagement,June2006,Volume10,Issue2, pp113-126

Hamel, G. *Management innovation*. Leadership Excellence, January 2007, Volume 24, Issue 1, p.5

Krishnan, V. and Loch, C. *Introduction to the special issue: management of productinnovation*.ProductionandOperationsManagement,Fall2005,Volume 14, Issue 3, pp269-271

Anderson, J., Donnellan, B. and Hevner, A. (2011). Exploring the Relationship between Design Science Research and Innovation: A Case Study of Innovation at Chevron, European Design Science Symposium

Lindberg, T., Gumienny, R., Jobst, B. and Meinel, C. (2010). Is there a need for a Design Thinking Process?, Proceedings of the 8th Design Thinking Research Symposium (DTRS8).

Linking Design Thinking Solutions to Business Challenges https://www.oreilly.com › view › xhtml › Chapter04

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – V- DESIGN THINKING – SCSA1306

Unit V

DEVOPS

Introduction to DevOps – DevOps vs Agile – DevOps Principles and Life Cycle – Introduction to CI / CD & DevOps Tools–Version Control – Build Automation – Configuration Management – Containerization – Continuous Deployment – Continuous Integration – Continuous Testing –Continuous Monitoring.

5.1 Introduction to DevOps:

DevOps is the acronym given to the **combination of Development and Operations**. It refers to a collaborative approach to make the Application Development team and the IT Operations team of an organization to seamlessly work with better communication.

What is DevOps?

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes, Figure 5.1.

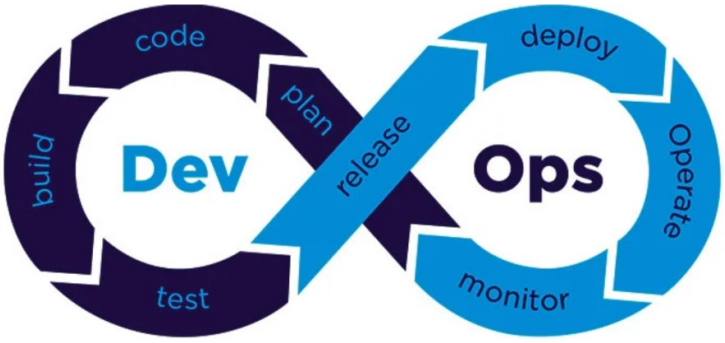


Fig 5.1 DevOps Architecture

DevOps vs Agile:

DevOps promotes a fully automated continuous integration and deployment pipeline to enable frequent releases, while Agile provides the ability to rapidly adapt to the changing requirements and better collaboration between different smaller teams.

5.2 Difference Between Agile and DevOps

Parameter	Agile	DevOps
What is it?	Agile refers to an iterative approach which focuses on collaboration, customer feedback, and small, rapid releases.	DevOps is considered a practice of bringing development and operations teams together.
Purpose	Agile helps to manage complex	DevOps central concept is to

Parameter	Agile	DevOps
	projects.	manage end-to-end engineering processes.
Task	Agile process focusses on constant changes.	DevOps focuses on constant testing and delivery.
Implementation	Agile method can be implemented within a range of tactical frameworks like a sprint, safe and scrum.	The primary goal of DevOps is to focus on collaboration, so it doesn't have any commonly accepted framework.
Team skill set	Agile development emphasizes training all team members to have a wide variety of similar and equal skills.	DevOps divides and spreads the skill set between the development and operation teams.
Team size	Small Team is at the core of Agile. As smaller is the team, the fewer people on it, the faster they can move.	Relatively larger team size as it involves all the stack holders.
Duration	Agile development is managed in units of "sprints." This time is much less than a month for each sprint.	DevOps strives for deadlines and benchmarks with major releases. The ideal goal is to deliver code to production DAILY or every few hours.
Feedback	Feedback is given by the customer.	Feedback comes from the internal team.
Target Areas	Software Development	End-to-end business solution and fast delivery.
Shift-Left Principles	Leverage shift-left	Leverage both shifts left and right.
Emphasis	Agile emphasizes on software development methodology for developing software. When the software is developed and released, the agile team will not care what happens to it.	DevOps is all about taking software which is ready for release and deploying it in a reliable and secure manner.
Cross-functional	Any team member should be able to do what's required for the progress of the project. Also, when each team member can perform every job, it increases understanding and bonding between them.	In DevOps, development teams and operational teams are separate. So, communication is quite complex.
Communication	Scrum is most common methods of implementing Agile software	DevOps communications involve specs and design

Parameter	Agile	DevOps
	development. Daily scrum meeting is carried out.	documents. It's essential for the operational team to fully understand the software release and its hardware/network implications for adequately running the deployment process.
Documentation	Agile method is to give priority to the working system over complete documentation. It is ideal when you're flexible and responsive. However, it can hurt when you're trying to turn things over to another team for deployment.	In the DevOps, process documentation is foremost because it will send the software to the operational team for deployment. Automation minimizes the impact of insufficient documentation. However, in the development of complex software, it's difficult to transfer all the knowledge required.
Automation	Agile doesn't emphasize on automation. Though it helps.	Automation is the primary goal of DevOps. It works on the principle to maximize efficiency when deploying software.
Goal	It addresses the gap between customer need and development & testing teams.	It addresses the gap between development + testing and Ops.
Focus	It focuses on functional and non-function readiness.	It focuses more on operational and business readiness.
Importance	Developing software is inherent to Agile.	Developing, testing and implementation all are equally important.
Speed vs. Risk	Teams using Agile support rapid change, and a robust application structure.	In the DevOps method, the teams must make sure that the changes which are made to the architecture never develop a risk to the entire project.
Quality	Agile produces better applications suites with the desired requirements. It can easily adapt according to the changes made on time, during the project life.	DevOps, along with automation and early bug removal, contributes to creating better quality. Developers need to follow Coding and Architectural best practices to maintain quality standards.

Parameter	Agile	DevOps
Tools used	JIRA, Bugzilla, Kanboard are some popular Agile tools.	Puppet, Chef, TeamCity OpenStack, AWS are popular DevOps tools.
Challenges	The agile method needs teams to be more productive which is difficult to match every time.	DevOps process needs to development, testing and production environments to streamline work.
Advantage	Agile offers shorter development cycle and improved defect detection.	DevOps supports Agile's release cycle.

5.3 DevOps Principles and Life Cycle

DevOps Principles

- Customer-Centric Action
- Create with the End in Mind
- End-To-End Responsibility
- Cross-Functional Autonomous Teams
- Continuous Improvement
- Automate Everything You Can

Customer-Centric Action:

Customer-centric (also known as client-centric) is a business strategy that's based on putting your customer first and at the core of your business in order to provide a positive experience and build long-term relationships

Create with the End in Mind:

Begin with the End in Mind means to **begin each day, task**, or project with a clear vision of your desired direction and destination, and then continue by flexing your proactive muscles to make things happen.

End-To-End Responsibility:

End-to-***end responsibility*** means that the team holds itself accountable for the quality and quantity of services it provides to its customers

Cross-Functional Autonomous Teams:

Team members have to be empowered and supported in having the ability to take on many roles within the team to ensure that value is delivered to the customer with minimal bottlenecks and roadblocks. Rapid delivery to the customer can be achieved.

Work is to be managed as a product rather than a project so that teams are autonomous in the decision making tasks required to successfully deliver and maintain their product. Work is never done, it continues to get better and better until end of life.

Continuous Improvement:

Continuous improvement, sometimes called continual improvement, is the ongoing improvement of products, services or processes through incremental and breakthrough improvements. These efforts can seek "incremental" improvement over time or "breakthrough" improvement all at once.

Automate Everything You Can:

What does it mean to "automate everything"? It means you have recognised a need to do things better, cheaper, faster, with more quality, security, accuracy and more importantly, to free up people from mundane tasks that are demotivating, laborious, costly and potentially high risk to your employees, business and customers.

DevOps Life cycle

DevOps characterizes an agile connection between operations and development. It is a cycle polished by the development group and operational specialists together from the starting to the last phase of the item. The DevOps lifecycle Figure 5.2 incorporates **seven stages and tools Figure 5.3** as given underneath:

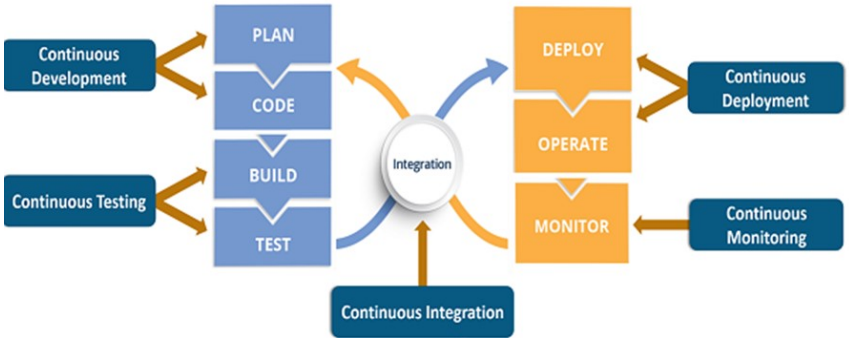


Fig 5.2 DevOps Life Cycle

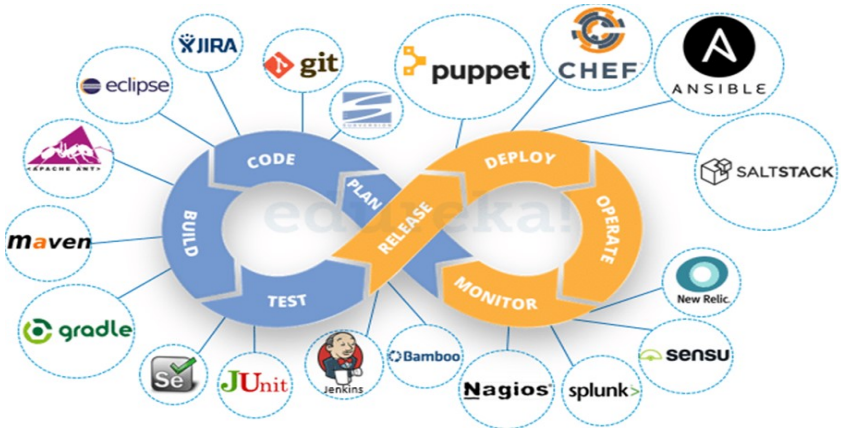


Fig 5.3 DevOps Tools

1. **Continuous Development**

This stage incorporates the orchestrating and coding of the item. The vision of the endeavor is picked during the orchestrating stage. Also, the designers start building up the code for the application. There are no DevOps apparatuses needed for arranging; however, there are a few devices for keeping up the code.

2. **Continuous Integration**

This stage is the core of the whole DevOps lifecycle. It is a product development practice in which the engineers need to submit changes to the source code all the more often. This might be on a day by day or week after week premise. At that point, each submission is fabricated, which permits the early location of issues if they are available. Construction regulation isn't just elaborate aggregation. Yet, it incorporates unit testing, combination testing, code survey, and bundling.

The code supporting new usefulness is continuously incorporated with the current code. Thus, there is a continuous development of programming. The refreshed code should be coordinated continuously and easily with the frameworks to reflect changes to the end-clients, Figure 5.4.

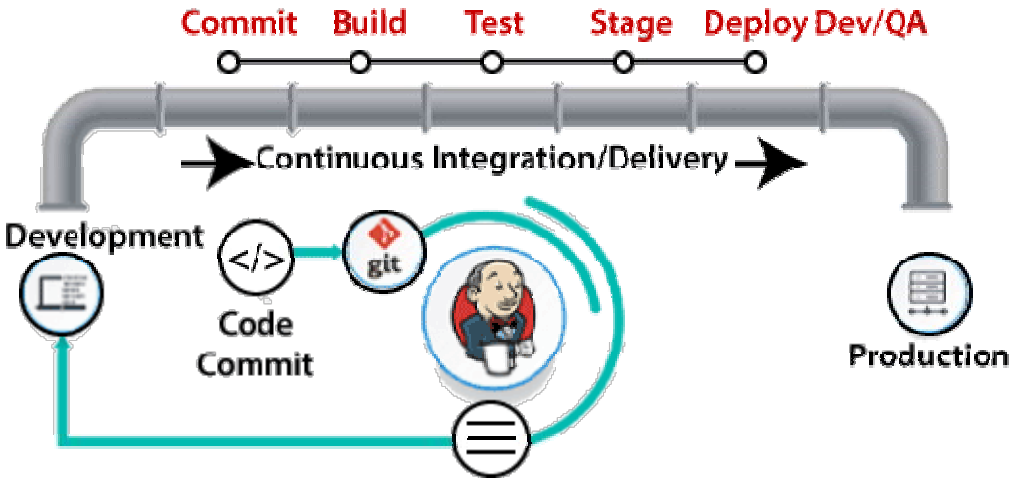


Fig 5.4 Continuous Integration

Jenkins is a well-known apparatus utilized in this stage. At whatever point there is an adjustment in the Git archive, at that point, Jenkins gets the refreshed code and readies a form of that code, which is an executable record as war or container. At that point, this fabricate was sent to the test worker or the creative worker.

This stage, where the created programming is continuously testing for bugs. For consistent testing, mechanization testing devices, for example, TestNG, JUnit, Selenium, and so forth, are utilized. These devices permit QAs to test numerous code-bases completely corresponding to guarantee that there is no blemish in the usefulness. In this stage, Docker Containers can be utilized for reenacting the test climate.

Selenium does the robotization testing, and TestNG creates the reports. This whole testing stage can robotize with the assistance of a Continuous Integration instrument called Jenkins.

Computerization testing spares a great deal of time and exertion to execute the tests instead of physically. Aside from that, reportage is a major addition. The assignment of assessing the experiments that fizzled in a test suite gets less difficult. Additionally, we can plan the execution of the experiments at predefined times. After testing, the code is continuously coordinated with the current code.

3. Continuous Testing

This phase, where the developed software is continuously testing for bugs. For constant testing, automation testing tools such as TestNG, JUnit, Selenium, etc are used. These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there is no flaw in the functionality. In this phase, Docker Containers can be used for simulating the test environment.

Fig 5.5 Continuous Testing

4. Continuous Monitoring

Monitoring is a stage that includes all the operational components of the whole DevOps measure. Significant data about the product's utilization is recorded and deliberately handled to discover drifts and recognize trouble spots. Normally, the monitoring is incorporated inside the operational capacities of the product application.

It might happen as documentation records or perhaps produces enormous information about the application boundaries in a continuous use position. The framework blunders, for example, workers not reachable, low memory, and so on, are settled in this stage. It keeps up the security and accessibility of the administration.

5. Continuous Feedback

The application development is reliably improved by investigating the outcomes from the operations of the product. This is done by putting the basic period of steady feedback between the operations and the development of the following adaptation of the current programming application.

The congruence is the basic factor in DevOps. It eliminates the superfluous advances needed to take a product application from development, utilizing it to discover its issues and deliver a superior variant afterward. It slaughters the productivity that might be conceivable with the application and decreases the quantity of intrigued clients.

6. Continuous Deployment

In this stage, the code is conveyed to the creation of workers. Likewise, it is basic to guarantee that the code is effectively utilized on all the workers.

The new code is conveyed continuously, and the design of the board instruments assume a fundamental function in executing assignments frequently and rapidly. Here are some

mainstream instruments utilized in this stage, for example, Chef, Puppet, Ansible, and SaltStack.

Containerization devices are likewise assuming a fundamental function in the deployment stage, Figure 5.6. Transient and Docker are well-known instruments that are utilized for this reason. These apparatuses help to deliver consistency across development, organizing, testing, and creation climate. They additionally help in scaling up and downsizing examples delicately.

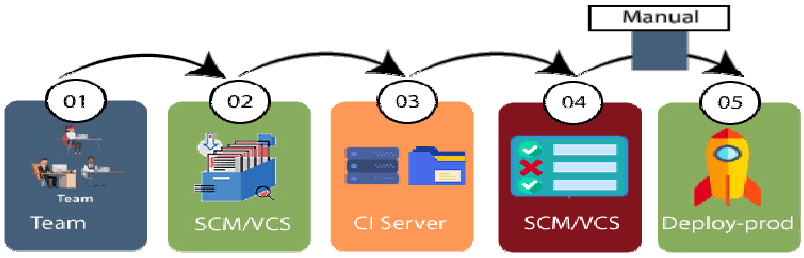


Fig 5.6 Containerization

Containerization instruments help keep up consistency over the conditions where the application is tried, created, and conveyed. There is no way of blunders or disappointment in the creation climate as they bundle and repeat similar conditions and bundles utilized in the testing, development, and organizing climate. It makes the application simple to run on various PCs.

7. **Continuous Operations**

All DevOps operations depend on the congruity with complete robotization of the delivery cycle and permit the association to quicken the general opportunity to advertise continuingly. It is obvious from the conversation that progression is the basic factor in the DevOps in eliminating steps that frequently divert the development, take it longer to identify issues, and produce a superior rendition of the item following a while. With DevOps, we can make any product item more proficient and increment the general include of intrigued clients regarding your item.

5.4 Introduction to CI / CD & DevOps Tools:

Continuous Integration and Delivery (CI/CD) Figure 5.7 are tools that automate the process of software development. In an organization, there is a need for teams to synchronize their work without breaking the code, and we often refer to this as the pipeline of CI/CD.

CI/CD is one of the best practices to integrate workflow between development teams and IT operations. It serves as an agile approach that focuses on meeting business requirements, quality code, and security while the implementation and deployment process is automated.

Continuous Integration (CI) is a programming practice requiring developers to incorporate code changes into a shared repository. Changes committed to the repository are verified by an automated construction, ensuring that bugs are spotted early before deployment.

Fig 5.7 Continuous Integration and Delivery

DevOps tools:

- Container Management tools - Docker, Kubernetes
- Application Performance Monitoring tools - Dynatrace, Prometheus
- Deployment & Server Monitoring tools - Splunk, Datadog
- Configuration Management tools - Chef, Puppet
- CI / Deployment Automation tools - Jenkins, IBM UrbanCode
- Test Automation tools - Test.ai, Selenium
- Artifact Management tools - Sonatype NEXUS, CloudRepo
- Codeless Test Automation tools - AccelQ, Testim.io

5.5 Version Control:

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.

Version control allows you to manage changes to files over time and store these modifications in a database.

Tools: Github, Subversion, and Mercurial

This includes version control software, version control systems, or version control tools. Version control is a component of software configuration management. It's sometimes referred to as VCS programming.

How Do Version Control Systems Work?

Version control systems allow multiple developers, designers, and team members to work together on the same project. Also known as VCS, these systems are critical to ensure everyone has access to the latest code. As development gets more complex, there's a bigger need to manage multiple versions of entire products.

What Are the Different Version Control Systems? (Examples)

There are several types of version control systems that teams use today. Some are centralized. Some are distributed.

Here are a few of the most popular types of VCS:

- Helix Core (Perforce)
- Git
- SVN
- ClearCase

- Mercurial
- TFS

What Is Git Version Control?

Git version control is open source. It's distributed free version control software. In fact, Git version control is one of the most popular options. It's open source, so anyone can use it.

But there are some drawbacks to using Git. Git lacks security. It's impossible to scale. And it's very difficult to get a single source of truth.

That's why teams who choose Git version control often add other tools on top. This includes Helix TeamHub for Git hosting and code reviews. Or Helix4Git, a Git server inside a Perforce server that can bring projects from third parties into your pipeline.

Top Version Control Benefits

There are many benefits to version control. The biggest benefits of the right VCS include:

A single source of truth across teams, code, and assets.

Traceability for every change ever made.

Speed for development teams who don't want to waste time looking for a file.

Features of Version Control Software

Each version control option comes with different features.

1. Concurrent Development

Projects are getting more complex. There’s a growing need to manage multiple versions of code and files — and entire products.

This means multiple developers and designers can work on the same set of files — without worrying that they are duplicating effort or overwriting other team members’ work.

Let’s say you're managing an IOT deployment for high-end internet-connected security cameras. Over the product lifecycle, you may use ten different types of cameras, each with a different chip. As a result, each will have different software.

Using the right version control software means you can maintain multiple versions of your code to manage the specific functionality of each camera's chip and operating system.

So, when you need to deploy a critical security patch to prevent bad guys from hijacking those cameras, it’s easy. You’ll instantly see which code is impacted, make the changes, and deploy a fix.

2. Automation

Higher quality and increased productivity are top priorities for today’s development teams. And your team can reach your goals by automating tasks, such as testing and deployment.

In software development, Continuous Integration (CI) with automated builds and code reviews are a standard operating procedure.

For hardware development, such as semiconductors, automation can include:

Testing with field programmable gate arrays (FPGAs).

Integration with simulation verification and synthesis systems.

Such FPGAs or verification and testing systems must themselves be controlled and versioned, along with potentially very large data files. It is vital that even small changes are tracked and managed. This means your version control software is the center of the IP universe. The right one can handle millions of automated transactions per day with millions of files.

3. Team Collaboration

Companies operate where the talent lives. This means you might have design and development centers in Minneapolis, Seattle, Toronto, and Shanghai. And when deadlines loom, you might need to add engineers in Paris and possibly even in Taipei.

You need a way to provide global access to your team members at all your facilities. Having a single source of truth — with appropriately secured identity and access management — is critical for your success.

With the right VCS, each team member is working on the latest version. And that makes it easier to collaborate.

4. Tracked Changes — Who, What, When, Why

Every development team needs visibility into changes. Tracking who, what, when, and why changes are made is important.

Version control software captures this detailed information and maintains this history forever. So, everyone gets access to who is working on what — and the changes that are made

This is especially important if you have governance, risk, and compliance (GRC) or regulatory needs. Audit log history is especially key in the automotive, aerospace, medical device, and semiconductor industries.

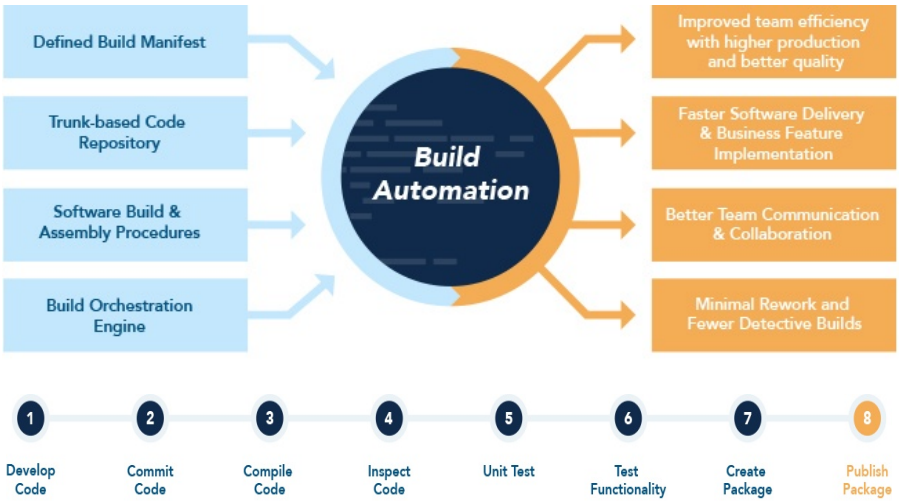
5. High Availability/Disaster Recovery

The most expensive asset you have is your product development team. You can't have them idled because they've lost access to the code.

With the right version control software, you can have a replica of your enterprise repository – – your single source of truth — operating in another location. If something happens, you can immediately switch over to a replica of the master for uninterrupted availability.

5.6 Build Automation

Build Automation, Figure 5.8 is the process of scripting and automating the retrieval of software code from a repository, compiling it into a binary artifact, executing automated functional tests, and publishing it into a shared and centralized repository.



5.8 Build Automation

Key Metrics

- **Number of Features / User Stories per Build -** Indicates the number of changes being implemented and maps to business value being created.
- **Average Build Time -** Indicates the average time to perform a build.
- **Percentage of Failed Builds -** impacts the overall team output due to rework.
- **Change Implementation Lead Time** - affects the number of releases per a given period and overall product roadmap planning.
- **Frequency of Builds** - indicates the overall output and activity of the project.

What Is the Role of Continuous Integration in the Automated Build Process?

Build automation enables Continuous Integration (CI). So, the role of CI in the automated build process is that CI uses build automation to verify check-ins and enable teams to detect issues early.

Because of its relationship with CI, build automation also makes Continuous Testing (CT) and Continuous Delivery (CD) — possible.

Benefits of Build Automation

There are **five main benefits** of build automation.

1. **Increases Productivity**

Build automation ensures fast feedback. This means your developers increase productivity. They'll spend less time dealing with tools and processes — and more time delivering value.

2. **Accelerates Delivery**

Build automation helps you accelerate delivery. That's because it eliminates redundant tasks and ensures you find issues faster, so you can release faster.

3. **Improves Quality**

Build automation helps your team move faster. That means you'll be able to find issues faster and resolve them to improve the overall quality of your product — and avoid bad builds.

4. **Maintains a Complete History**

Build automation maintains a complete history of files and changes. That means you'll be able to track issues back to their source.

5. **Saves Time and Money**

Build automation saves time and money. That's because build automation sets you up for CI/CD, increases productivity, accelerates delivery, and improves quality.

How to Automate the Build Process

- Write the code.
- Commit code to a shared, centralized repository — such as Perforce Helix Core.
- Scan the code using tools such as static analysis.
- Start a code review.
- Compile code and files.
- Run automated testing.
- Notify contributors to resolve issues.

Automated Build Tools

These tools help you automate the process of building, testing, and deploying code.

Example: Jenkins

Jenkins is a popular build runner. Many teams automate their CI/CD pipeline with Jenkins.

5.7 Configuration Management

Configuration management is a systems engineering process for establishing consistency of a product's attributes throughout its life. In the technology world, configuration management is an IT management process that tracks individual configuration items of an IT system.

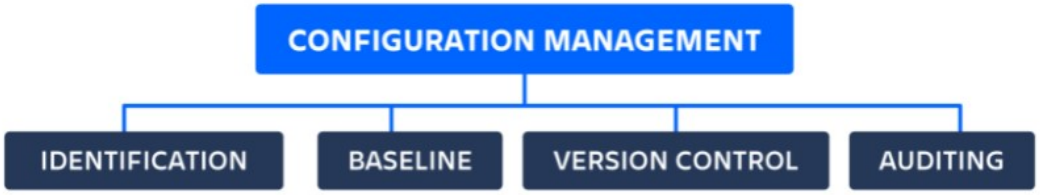


Fig 5.7 Configuration Management

Components: Configuration Management in DevOps

Configuration management takes on the **primary responsibility for three broad categories required for DevOps transformation**:

- Identification

- Control and
- Audit processes.

Identification:

The process of finding and cataloging system-wide configuration needs.

Control:

During configuration control, we see the importance of change management at work. It's highly likely that configuration needs will change over time, and configuration control allows this to happen in a controlled way as to not destabilize integrations and existing infrastructure.

Audit:

Like most audit processes, a configuration audit is a review of the existing systems to ensure that it stands up to compliance regulation and validations.

There are **primary components** that go into the comprehensive configuration management required for DevOps, Figure 5.8:

- Artifact repository
- Source code repository
- Configuration management data architecture

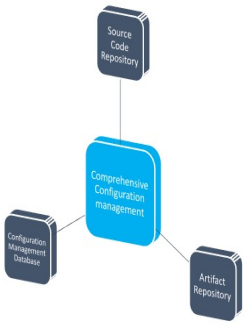


Fig 5.8 comprehensive configuration management

Artifact Repository

An artifact repository is meant to store machine files. This can include binaries, test data, and libraries. Effectively, it's a database for files that people don't generally use. In DevOps, artifacts, like binaries, are a natural result of continuous integration. DevOps developers are always pushing out builds which, in turn, create artifact files that need to be stored, but not necessarily accessed.

The artifact repo comes with two logical partitions, Figure 5.8 for storing and managing binaries:

1. Snapshot
2. Release

Fig 5.9 Snapshot

Source Code Repository

Conversely, the source code repository is a database of source code which developers use. This database serves as a container for all the working code. Source code aside, it stores a number of useful components including various scripts and configuration files.

While some developers store binaries in this same repository, that's not a best practice. In DevOps, due to the sheer number of builds and off-shoot binaries, it's recommended that an artifact repository is developed for the purpose of storing binaries and other artifacts.

It's not hard to determine what goes into the source code repository. A quick litmus test is to ask yourself, "are the files human-readable?"

If yes, there's a good chance they belong in the source code repository as opposed to anywhere else. There are two types of source code repositories: centralized version control system (CVCS) and distributed version control system (DVCS).

In a CVCS, the source code lives in a centralized place, where it can be retrieved and stored. However, in DVCS, the code exists across multiple terminals useful in the development process. It's faster and more reliable. Most often, DVCS is the chosen source code repository of today's DevOps professionals.

Configuration Management Data Architecture

The idea of having data architecture dedicated to configuration management is a principle of ITIL service management framework. A configuration management database or (CMDB) is a relational database that spans across multiple systems and applications related to configuration management, including services, servers, applications, and databases to name a few.

CMDB is helpful for change management, as it allows users to audit the relationships between integrated systems before configuration changes are made. It's also a useful tool for provisioning as you can glean all identifying information for objects like servers. A CMBD is an essential tool when it comes to incident management, too, as it helps teams escalate issues to resolution.

Outcomes of Properly Managed Configurations

When a system is properly configured and managed, you can expect certain outcomes. Among these outcomes are delivering infrastructure-as-a-code and configuration-as-a-code. Below, we will look at the role each outcome has in configuration management:

Infrastructure-as-a-Code

Infrastructure-as-a-code (IaaC) in simplest terms is a code or script the automates the environment necessary for development, without manually completing all the steps necessary to build the environment. When we use the word 'environment' in this way, we are referring to the set up of all computing resources required to create the infrastructure to perform DevOps actions. This could be servers, networks of configurations and other resources.

Configuration-as-a-Code

As the name suggests, configuration-as-a-code (CaaC) is a string of code or script that standardizes configurations within a given resource, like a server or network. These configurations are applied during the deployment phase to ensure the configuration of the infrastructure makes sense for the application.

Benefits of IaaC and CaaC

Fans of continuous integration should be pretty familiar with the benefits of IaaC and CaaC, but for those new to DevOps, you'll want to know what to expect. These are some of the benefits of the two key outcomes defined in this section:

- Automation of the infrastructure environment provides standardization
- Setups are free of human error
- Collaboration is enhanced between operations and development
- Keeps configurations from drifting
- Makes infrastructure more flexible, ready to scale
- Each step is consistent across all resources
- Version control is a given

With these benefits applied to an organization, efficiency and greater agility are a natural result. DevOps is practically synonymous with configuration management, IaaC, and CaaC.

Configuration Management Database

The Configuration Management Database (CMDB) comes from the Information Technology Infrastructure Library (ITIL) service management framework. The CMDB is a repository of various infrastructure devices, applications, databases, and services.

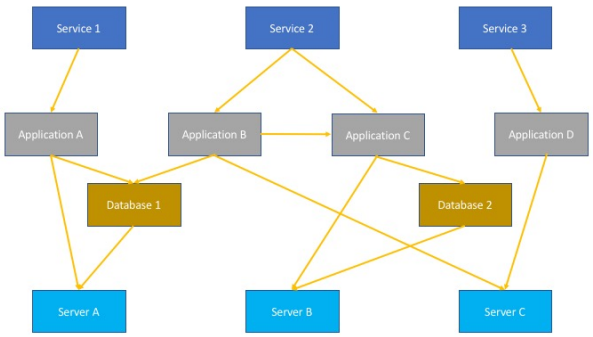


Fig 5.10 Configuration Management Database

In the Figure 5.10 above, services, applications, databases, and servers are represented by the different colored boxes. In the illustration, Service 1 depends on Application A, as seen by the arrow relationship. Application A leverages on Database 1. Both the Application A and Database 1 reside on Server A.

CMDB for Change Management

The CMDB is particularly useful when you are trying to make a change to any of the applications, databases or servers.

Let's say you want to make a change to Application B. To make the change, you must first do an impact assessment. CMDB helps in performing impact assessments. In the illustration, suppose changes are done to Application B. The impact assessment will read that any changes done to Application B will impact Application C, as the data is flowing through it.

Today, software development seldom happens in isolation. The software to be developed is either an improvement over existing code or is getting plugged into an enterprise network of applications. Therefore, it is critical that the impacts are assessed to the tee and CMDB is a significant help in this area.

CMDB for Provisioning Environments

Another application of CMDB in DevOps is in environment provisioning. Today we can spin up environments on the go with tools like Ansible in our scripts. When you key in the exact configuration of the production server, the environment provisioning tools create a prod-like server in a snap of a finger.

But how is it that you are going to obtain the complete configuration of a server? The most straightforward way is to refer to a CMDB.

Let's say Server C is a production server that needs to be replicated. In the CMDB, the server C entry will provide the complete configuration of the server, which is an immense help in scripting provisioning scripts such as Playbooks (compatible with Ansible.)

CMDB for Incident Management

The CMDB also has other benefits such as supporting the incident management teams during incident resolutions. The CMDB readily provides the architecture of applications and infrastructure, which is used to troubleshoot and identify the cause of the issue.

Configuration Management Tools
It is common for configuration management tools Figure 5.11 to include automation too. Popular tools are:

- Red Hat Ansible
- Chef
- Puppet

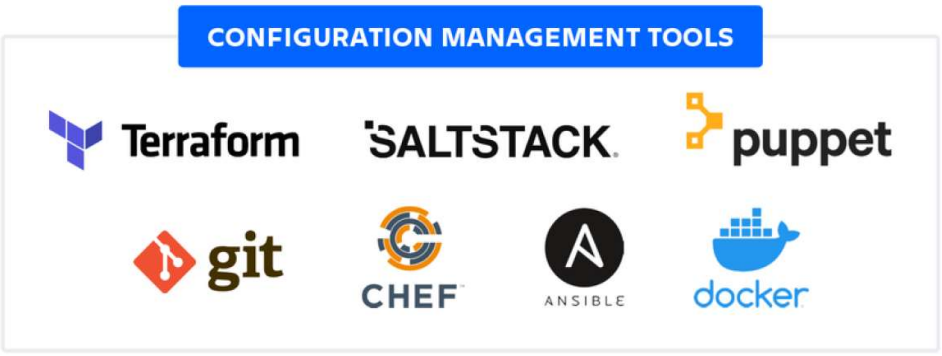


Fig 5.11 Configuration Management Tools

Git

Git is the industry-leading version control system to track code changes. Adding configuration management data alongside code in a Git repository provides a holistic version control view of an entire project. Git is a foundational tool in higher-level configuration management. The following list of other configuration management tools is designed to be stored in a Git repository and leverage Git version control tracking.

Docker

Docker introduced containerization that is an advanced form of configuration management -- like a configuration lockdown. Docker is based on configuration files called Dockerfiles, which contain a list of commands that are evaluated to reconstruct the expected snapshot of operating system state. Docker creates containers from these Dockerfiles that are snapshots of a preconfigured application. Dockerfiles are committed to a Git repository for version tracking and need additional configuration management to deploy them to infrastructure.

Terraform

Terraform is an open source configuration management platform by HasiCorp. Terraform uses IaC to provision and manage clusters, cloud infrastructure, or services. Terraform supports Amazon Web Services (AWS), Microsoft Azure, and other cloud platforms. Each cloud platform has its own representation and interface for common infrastructure components like servers, databases, and queues. Terraform built an abstraction layer of configuration tools for cloud platforms that enable teams to write files that are reproducible definitions of their infrastructure.

Ansible, Salt Stack, Chef, Puppet

Ansible, Salt Stack, and Chef are IT automation frameworks. These frameworks automate many traditional system administrators' processes. Each framework uses a series of configuration data files -- usually YAML or XML -- that are evaluated by an executable.

The configuration data files specify a sequence of actions to take to configure a system. The actions are then run by the executable. The executable differs in language between the systems -- Ansible and Salt Stack are Python based and Chef is Ruby. This workflow is similar to running ad-hoc shell scripts but offers a more structured and refined experience through the respective platforms ecosystems. These tools are what will bring enable the automation needed to achieve CI/CD.

5.8 Containerization

Containerization is the **process of packaging software code**, its required dependencies, configurations, and other detail to be easily deployed in the same or another computing environment. In simpler terms, containerization is the encapsulation of an application and its required environment.

List of Containerization DevOps Tools

1. Marathon:

Marathon, an Apache Meso framework that was designed solely to manage containers can make your life pretty easy. In comparison to the other prevailing orchestration solutions such as Kubernetes, Docker Swarm, Marathon will allow and ensure that you will be able to scale your container infrastructure by just automating most of the management and also the monitoring tasks.

Over the days, Mesos Marathon has been evolving into a very sophisticated and feature-rich tool. It becomes even more difficult to bring the better of Apache Mesos Marathon into the limelight all by itself.

Following are some of the advantages of using Marathon, let us now take a look at each and every one of them:

Advantages of Marathon:

Apache Mesos Marathon ensures super ultra-high availability. Marathon software will let you run multiple schedules at the same point in time as if one goes down, the system will still keep ticking showing that there is nothing that can stop it. Docker Swarm and Kubernetes also promise high availability, but Marathon takes this to the next level.

Marathon has multiple CLI clients that can be separately installed along with Marathon. These CLI clients give you loads of options for managing or scripting the tool in a varied number of complex ways.

It is very easy to run locally for development purposes as against Kubernetes

Application health checks provide you with all the information in detail that you would require of your instance – like the performance monitoring and stuff.

2. Fleet:

CoreOS Container Linux is said to be the topping the charts in the space of Container Operating Systems – which are by default designed to be managed and to be run at humongous scales with the least possible or minimal operational overhead.

Applications with the Container Linux run inside these containers and provide a developer-friendly set of tools for the software deployments. Container Linux runs on nearly almost all the possible combinations of platforms, be it physical, virtual, public, or private cloud spaces.

CoreOS also do provide the fleet functionality based on the fleets cluster manager daemons which do control the CoreOS's separate system instances at the cluster level itself.

Following are some of the advantages of using Fleet, let us now take a look at each and every one of them:

Advantages of Fleet:

CoreOS claims that the configuration values are distributed within the cluster for applications to read them and these values can be programmatically changed, smart applications can reconfigure these automatically. The basis on this point, you will never have to run Chef on all the machines to change just a single configuration value.

CoreOS provides you with the highest possible availability at a relatively lower price

CoreOS lets you maintain different versions of software on machines, and upgrading these machines is done without any downtime at all

CoreOS goes a step further than Docker by replicating the Cluster and Network setting between the Development and Production environments as well, whereas Docker just ensures that these environments are similar but not to the level that CoreOS does this for us.

Developer machines can be brought UP and running within seconds, as there would not be a requirement to install all the required software from the scratch one after the other

The cost of replicating software like Heroku can be drastically brought down.

3. Swarm:

Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host. A swarm contains more than one Docker host that run in the Swarm mode, it acts as a manager to manage the membership and workers which would then run the Swarm services. A Swarm cluster consists of Docker Engine deployed on multiple nodes. Manager nodes perform orchestration and cluster management. Worker nodes receive and execute tasks from the manager nodes.

Following are some of the advantages of using Swarm, let us now take a look at each and every one of them:

Advantages of Swarm:

Starting with Docker Engine 1.12, Docker Swarm is completely available alongside Docker.

If you use the latest Docker, then the Swarm setup is already done for you in the latest versions.

Docker Swarm is easily integrated with Docker, it hooks directly into the Docker API and then is compatible with all of the Docker's tools.

4. Docker Hub:

The Docker Hub can be very easily defined as a Cloud repository in which Docker users and partners create, test, store, and also distribute Docker container images. Through the use of Docker Hub, a user can very easily access public, open-source image repositories and at the same time – use the same space to create their own private repositories as well.

Following are some of the advantages of using Docker Hub, let us now take a look at each and every one of them:

Advantages of Docker Hub:

Forms the central repository for all the public and private images created by the users

Provides central access to all the available Public docker images

Users can safely create their own private docker images and save them under the same central repository, the docker hub

Watch this video on "Top 10 Highest Paying IT Jobs in 2021" and know how to get into these job roles.

<iframe width="560" height="315" src="https://www.youtube.com/embed/G-vSRFhkeeU" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>

5. Packer:

Packer is free and open-source software that finds its usage to create identical machine images or containers for various platforms from a singly available source configuration. Having pre-baked machine images is very advantageous because to create them scratch is a very tedious task.

There were not as many tools earlier that could perform this task, and even if there exists such software or a tool – then there would have been a huge learning curve that gets associated with it.

As a result of that, earlier to Packer – the creation of machine images was always a threat to the agility of the operations team and also weren't used despite their massive benefits. Packer with its invent has been able to replace these for quite a long, as Packer is very easy to use and also automates the process of creation of any kind of machine image.

Packer encourages modern configuration machines using frameworks like Chef or Puppet to install and also to configure this software that you are planning to Packer-made images. To be very precise, Packer brings the concept of pre-baked images into the modern age, therefore, encouraging untapped potential and also encourage newer and newer opportunities.

Having said all this, let us take a look into the advantages that it has in store for us:

Following are some of the advantages of using Packer, let us now take a look at each and every one of them:

Advantages of Packer:

Packer ensure that the infrastructure deployment process is done at a super-fast pace

Packer ensures multi-provider portability, meaning it creates identical images for various other platforms that it supports. Using this, a Production setup can run on an AWS, Staging / QA might run on something like OpenStack, and maybe the development on Desktop Virtualization solutions.

Packer enforces improved stability as it installs, configures all the software at the time the image is built

A machine that is built by Packer can very quickly be launched and tested (smoke tested) to verify that the things are all good and appear to be working.

6. Kubernetes:

Kubernetes was built by Google based on its experience of running Containers in various Production environments. With a combination of great software engineers working on the project plus the fact that Google was behind the evolvement of Kubernetes, it is one of the best-suited tools that run some of the largest software services by scale.

This combination ensured that this rock-solid platform can take any scaling needs of an Organization head-on. Kubernetes is an open-source system for deploying, scaling, and also to manage containerized applications. Kubernetes brings both the software design and also software operations together as one single operation by design.

Kubernetes enables the deployment of cloud-native applications anywhere and also manages these deployments exactly, the same way as you like from everywhere. With the Containers, it is very easy to ramp up the application instances to match the spikes in the demand whenever observed.

These containers do obtain these resources from the core host OS, they are considered much lighter weight than those of the traditional Virtual machines. By this, also ensures that the underlying server infrastructure is highly efficiently made use of.

Following are some of the advantages of using Kubernetes, let us now take a look at each and every one of them:

Advantages of Kubernetes

Kubernetes proves high scalability, easier to use container management, and at the same time helps to reduces delay in communication.

Building micro-services and adding lifetime replications based on the need is a super easy task with Kubernetes. If the Project demands many more of these and makes changes also, there is not much effort that is needed.

Kubernetes manages the balancing load on all the participating nodes via the load balancer and keeps the Master away from being overloaded with all the tasks at once.

7. Nomad

Nomad is a Cluster Manager and also a Scheduler that is designed for Micro-services and also to handle batch workloads. It is also distributed, highly available and at the same time scales to thousands of nodes or clusters that can span amongst multiple data centers and regions.

It does provide a common workflow that helps deploy applications across the infrastructure. Developers or any other individuals for that case can provide declarative job specifications to define the way or manner that the applications must be deployed and resources must be allocated.

Nomad accepts requests for executing such jobs, and also finds all the resources that need to run these jobs as well. The scheduling algorithm that is used by Nomad, it ensures that all the constraints that it needs are satisfied and packs applications on the host to help optimize resource utilization.

It additionally supports virtualized, containerized, and also standalone applications that run on major operating systems. Nomad is also finding its application in the production environments as well.

Following are some of the advantages of using Nomad, let us now take a look at each and every one of them:

Advantages of Nomad:

Nomad uses bin-packing to optimize application placement onto servers to maximize resource utilization, increase density, and help reduce costs.

In addition to providing its support to Linux, Windows, and Mac environments it extends its support towards containerized, virtualized, and standalone applications as well.

Simplified operations via Nomad make it safe to handle upgrade jobs, automatically handles machine failures, and also provide a single workflow for application deployments.

Nomad has the ability to span across many public and private clouds, to treat all infrastructure as a pool of resources that were used as expendables.

Nomad is a single binary that schedules applications and services on Linux, Windows, and Mac. It is an open-source scheduler that uses a declarative job file for scheduling virtualized, containerized, and standalone applications.

8. OpenVZ:

OpenVZ can be described as a Container-based Virtualization solution for Linux environments. It does it by creating multiple secure and isolated Linux servers termed as the Virtual Private Servers (VPS) on a single physical machine. Each of these containers (VPS) performs or executes instructions as if they are run on a standalone server.

The only way that OpenVZ containers differ from the traditional Virtual machines is that they run on the same OS Kernel as of the host itself but in turn, allows multiple Linux variants in

individual Containers, and because of this running of these containers is done with very less overhead. With the same, it also provides greater efficiency and manageability than the traditional old Virtualization technologies.

Following are some of the advantages of using OpenVZ, let us now take a look at each and every one of them:

Advantages of OpenVZ:

Since that OpenVZ uses a single Linux Kernel implementation, it has the utmost possibility to scale well. It can scale up to thousands of CPUs and TBs of RAM.

Very low Virtualization overhead again because of the single Linux Kernel implementation.

Live migration of Virtual Private Servers (VPS) from one physical host to another without even shutting them down during the process.

Resource management is done in a very efficient manner with OpenVZ and alongside that resource isolation, performance and security are its other core attributes.

IPsec is very much supported inside these containers since the Kernel version v2.6.32

Container hardware remains independent as OpenVZ restricts container access to physical devices.

9. Solaris Containers:

The very first thing that might hit your mind is the very name of the tool as Solaris and Containers are two words from two different extremes, but let me clarify that it is very much possible in this decade.

Over the past few years, the discussion over Containers usually happened with Docker, CoreOS, and LXD on Linux (to some extent over Windows and Mac OSX too) but with Solaris (Oracle's UNIX like OS) have had containers for quite a long time now. Though with the confusion that the name creates, Solaris Containers are pretty hardly identical to those of Docker, CoreOS containers.

These do similar things as virtualizing software inside isolated environments curtailing the overhead of having a hypervisor or a VMware instance. Though the world might be considering Docker and the like for their Linux environments Solaris Containers are also interesting enough to gain knowledge all about.

There is a plan to bring Docker to Solaris Containers as confirmed by Oracle – which only means that Solaris Containers can be seen more on the mainstream Containers and DevOps space.

Following are some of the advantages of using Solaris Containers, let us now take a look at each and every one of them:

Advantages of Solaris Containers:

Configuration is pretty easy, until and unless you are able to point and click your way through the Enterprise Manager Ops Center to manage the Solaris Containers.

Virtual resources are managed pretty well and easy with Solaris Containers as compared to Docker and CoreOS.

10. CloudSlang:

CloudSlang, an open-source software tool that finds its usage in the orchestration space is one of the cutting-edge technologies available for Organizations with DevOps implementations. It is one such tool that can perform the orchestration activity on almost anything that you can imagine in an ageless manner.

There is a possibility that an individual can re-use a ready-made workflow or design a custom workflow altogether – which can further be reusable, shareable, and are also very easy to understand as well.

Following are some of the advantages of using CloudSlang, let us now take a look at each and every one of them:

Advantages of CloudSlang:

One of the biggest advantages of using CloudSlang is that it is an Open source tool that is available for orchestrating cutting-edge technologies.

Use, Re-use, or Customize the readymade YAML-based workflows.

These workflows are further powerful, shareable amongst members, and are extremely easy to understand by others.

The content that is available with CloudSlang is easy to understand as it uses YAML-based DSL.

It is an Open source-based orchestration tool with readymade workflows.

5.9 Continuous Deployment – Continuous Integration

The CI/CD pipeline , Figure 5.12 is one of the best practices for devops teams to implement, for delivering code changes more frequently and reliably

Continuous integration (CI) and continuous delivery (CD) embody a culture, set of operating principles, and collection of practices that enable application development teams to deliver code changes more frequently and reliably. The implementation is also known as the CI/CD pipeline.

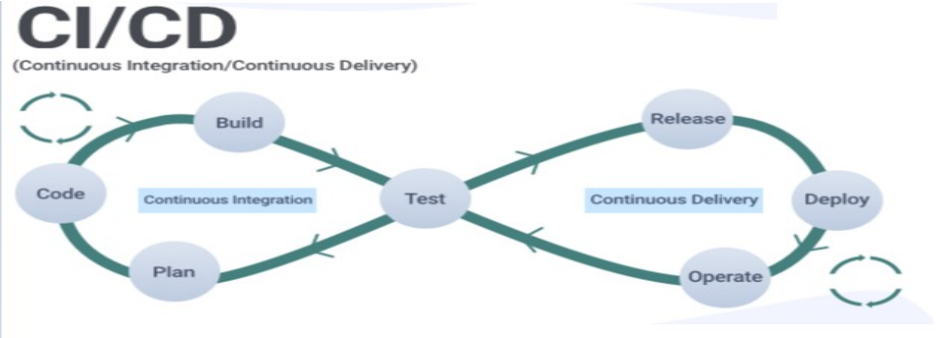


Fig 5.12 **CI/CD(Continuous Integration/Continuous Delivery)**

Continuous Integration is a prerequisite for CI/CD, and requires:

- Developers to merge their changes to the main code branch many times per day.
- Each code merge to trigger an automated code build and test sequence. Developers ideally receive results in less than 10 minutes, so that they can stay focused on their work.

The job of Continuous Integration is to produce an artifact that can be deployed. The role of automated tests in CI is to verify that the artifact for the given version of code is safe to be deployed, Figure 5.13.

Tools that you can use to start testing UI:

- Jasmine (JavaScript)
- Selenium (Frontend Testing)
- CasperJS (Frontend Testing)
- Cucumber or Lettuce (Behaviour Driven Testing for Ruby and Python)

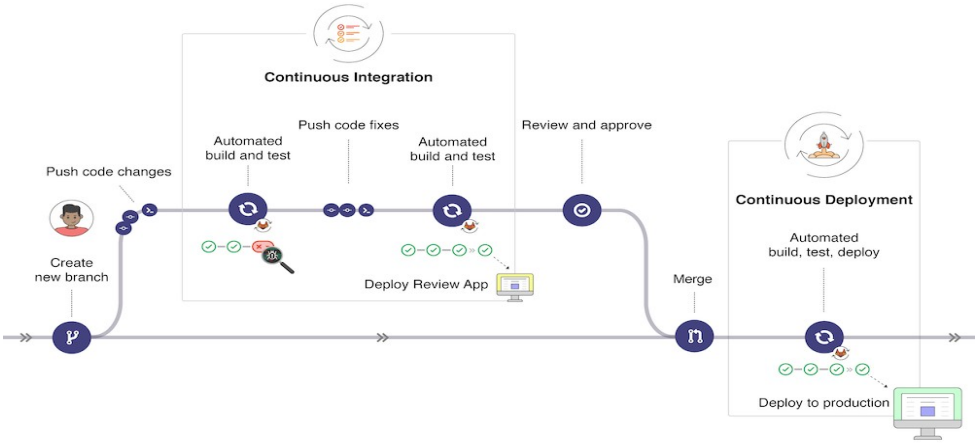


Fig 5.13 Process flow of Continuous integration (CI)/ continuous delivery (CD)

CI/CD principles

Continuous Delivery practices take CI further by describing principles for successful production deployments:

- Architect the system in a way that supports iterative releases. Avoid tight coupling between components. Implement metrics that help detect issues in real-time.
- Practice test-driven development to always keep the code in a deployable state. Maintain a comprehensive and healthy automated test suite. Build in monitoring, logging, and fault-tolerance by design.
- Work in small iterations. For example, if you develop in feature branches, they should live no longer than a day. When you need more time to develop new features, use feature flags.
- Developers can push the code into production-like staging environments. This ensures that the new version of the software will work when it gets in the hands of users.
- Anyone can deploy any version of the software to any environment on demand, at a push of a button. If you need to consult a wiki on how to deploy, it's game over.
- If you build it, you run it. Autonomous engineering teams should be responsible for the quality and stability of the software they build. This breaks down the silos between traditional developers and operations groups, as they work together to achieve high-level goals.

5.10 Continuous Testing:

Continuous Testing, Figure 5.14 in DevOps is a software testing type that involves testing the software at every stage of the software development life cycle. The goal of Continuous testing is evaluating the quality of software at every step of the Continuous Delivery Process by testing early and testing often.

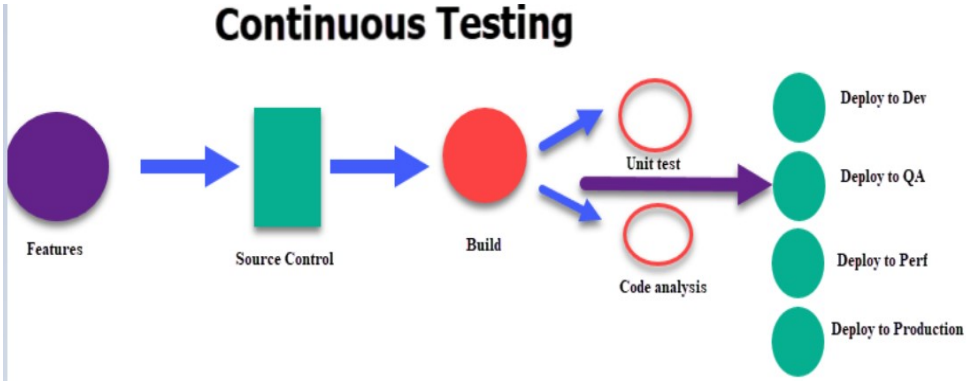


Fig 5.14 Continuous Testing

5.11 Continuous Monitoring

Continuous Monitoring comes in at the end of the DevOps pipeline. Once the software is released into production, Continuous Monitoring will notify dev and QA teams in the event of specific issues arising in the prod environment. It provides feedback on what is going wrong, which allows the relevant people to work on necessary fixes as soon as possible.

Continuous Monitoring basically assists IT organizations, DevOps teams in particular, with procuring real-time data from public and hybrid environments. This is especially helpful with implementing and fortifying various security measures – incident response, threat assessment,

computers, and database forensics, and root cause analysis. It also helps provide general feedback on the overall health of the IT setup, including offsite networks and deployed software.

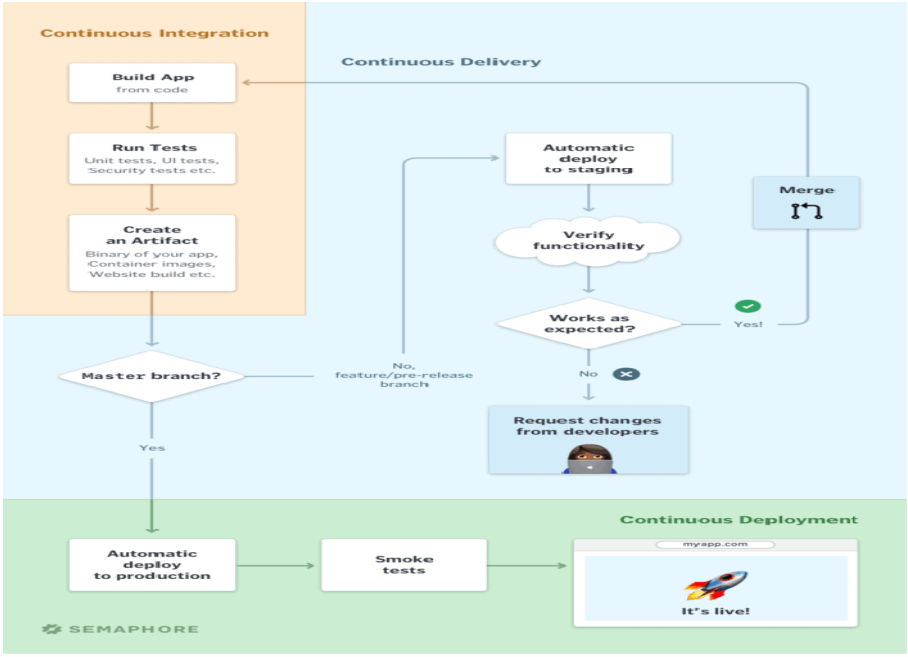


Fig 5.15 Continuous Monitoring

Goals of Continuous Monitoring in DevOps

- Enhance transparency and visibility of IT and network operations, especially those that can trigger a security breach, and resolve it with a well-timed alert system.
- Help monitor software operation, especially performance issues, identify the cause of the error, and apply appropriate solutions before significant damage to uptime and revenue.
- Help track user behaviour, especially right after an update to a particular site or app has been pushed to prod. This monitors if the update has a positive, negative, or neutral effect on user experience.

TEXT /REFERENCE BOOKS

1. MaurícioVianna, YsmarVianna, Brenda Lucena and Beatriz Russo," Design thinking : Business innovation", MJV Technologies and innovation press, 2011.
2. Design Thinking: Integrating Innovation, Customer Experience, and Brand Valueby Thomas Lockwood (Editor) Published February 16th 2010 by Allworth Press.
3. KalloriVikram, —Introduction to DevOps, 1 st Edition, KalloriVikram Publication, 2016.
4. Jaokim Verona, —Practical DevOps, 2 nd Edition, Packt. Publication, 2018.
5. Stephen Fleming, Pravin, —DevOps Handbook: Introduction of DevOps Resource Management—,1st Edition, Createspace Independent Pub. , 2010.
6. Len Bass, Ingo Weber, Liming Zhu, G., —DevOps: A Software Architect's Perspective, 1st Edition, Addison-Wesley Professional, 2015.
7. Alistair Cockburn, "Agile Software Development", 2nd ed, Pearson Education, 2007.