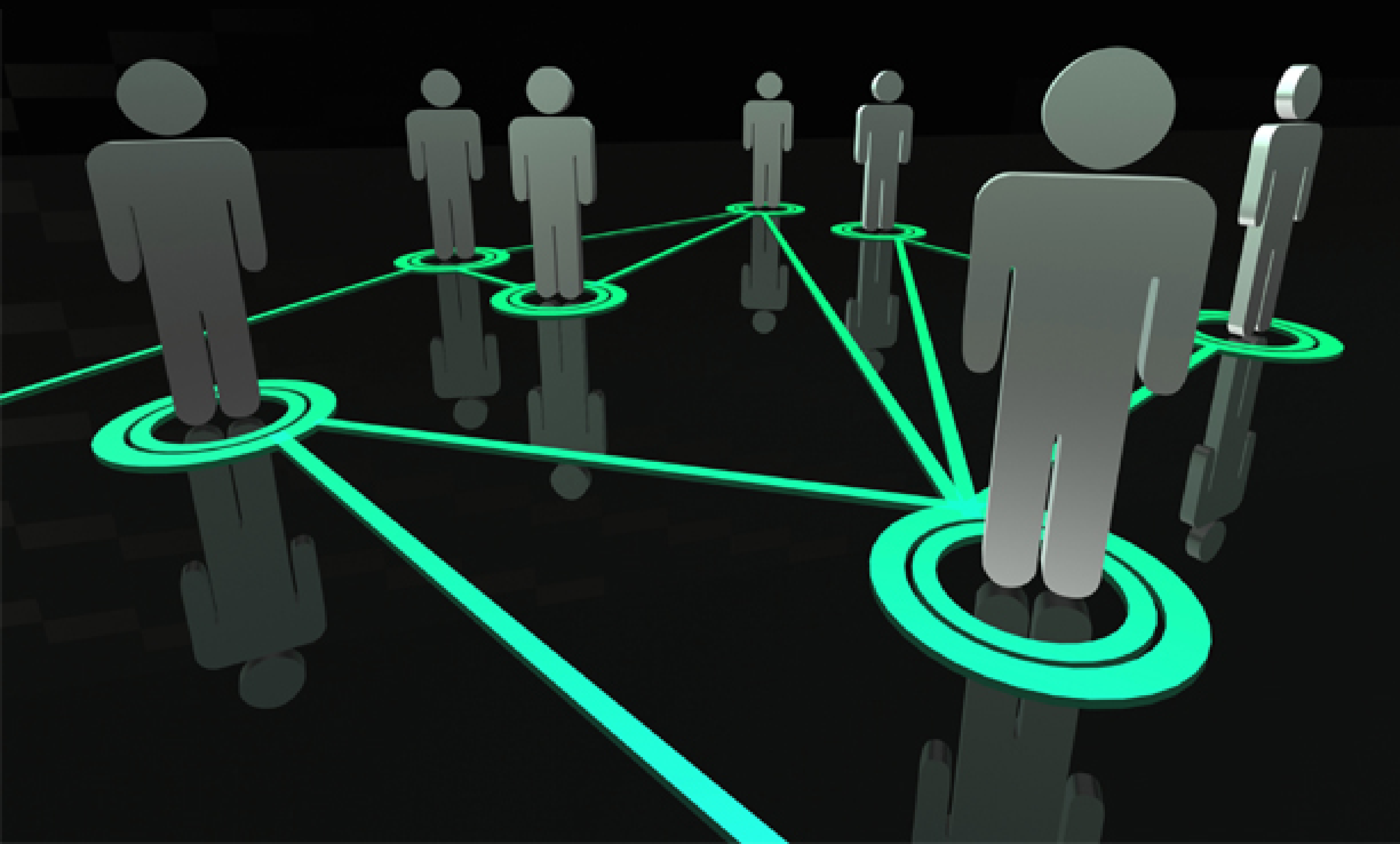
Knowledge-Based Engineering (Concepts & Applications)

Marisa Shay



First Edition, 2012

ISBN 978-81-323-3611-2

Published by:
University Publications
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter 1 - Knowledge-Based Engineering

Chapter 2 - Product Lifecycle Management

Chapter 3 - Systems Engineering

Chapter 4 - Reliability Engineering

Chapter 5 - Tools of Systems Engineering

Chapter 6 - Primary Areas of Product Lifecycle Management

Chapter 7 - Computer-Aided Technologies

Chapter 8 - Computational Fluid Dynamics & Computer-Aided Manufacturing

Chapter- 1

Knowledge-Based Engineering

Knowledge-based engineering (KBE) is a discipline with roots in computer-aided design (CAD) and knowledge-based systems but has several definitions and roles depending upon the context. An early role was support tool for a design engineer generally within the context of product design. Success of early KBE prototypes was remarkable; eventually this led to KBE being considered as the basis for generative design with many expectations for hands-off performance where there would be limited human involvement in the design process.

Overview

KBE can be defined as engineering on the basis of electronic knowledge models. Such knowledge models are the result of knowledge modeling that uses knowledge representation techniques to create the computer interpretable models. The knowledge models can be imported in and/or stored in specific engineering applications that enable engineers to specify requirements or create designs on the basis of the knowledge in such models. There are various methods available for the development of knowledge models, most of them are system dependent. An example of a system-independent language for the development machine-readable ontology databases, including support for basic engineering knowledge, is called Gellish English. An example of a CAD-specific system that can store knowledge and use it for design is the CATIA program through its KnowledgeWare module. An example of a CAD-independent, language-based KBE system with full compiler and support for runtime application deployment is General-purpose Declarative Language (GDL) from Genworks.

KBE can have a wide scope that covers the full range of activities related to Product Lifecycle Management and Multidisciplinary design optimization. KBE's scope would include design, analysis (computer-aided engineering – CAE), manufacturing, and support. In this inclusive role, KBE has to cover a large multi-disciplinary role related to many computer aided technologies (CAx).

KBE also has more general overtones. One of its roles is to bridge knowledge management and design automation. Knowledge processing is a recent advance in computing. It has played a successful role in engineering and is now undergoing modifications (to be explained). An example of KBE's role is generative mechanical design. There are others. KBE can be thought of as an advanced form of computer applications (in some forms with an extreme end-user computing flavor) that support PLM and CAx.

There are similar techniques, such as electronic design automation. AAAI provides a long list of engineering applications. some of which are within the KBE umbrella. At some point, the concept of KBE might split into several sub-categories as MCAD and ECAD are just two of many possible types of design automation.

History

KBE essentially was a complementary development to CAx and can be dated from the 1980s. CAx has been developing along with the computer after making large strides in the 1970s.

As with any bit of progress, KBE flashed on the horizon, lit the sky for a while, and then experienced a downslide. KBE had sufficient success stories that sustained it long enough into the 1990s to get attention. Some prime contributors to the hiatus of KBE were unmanageable expectations, increasing tedium associated with forming completion of results, and some notion that the architecture for KBE was not sufficiently based upon the newer technology.

KBE continued to exist in pockets. With the prevalence of object-oriented methods, systems advanced enough to allow re-implementation. This reconstruction has been on-going for several years and has been frustratingly slow. Now, with the basis for this discipline becoming more robust, it starts to get interesting again.

KBE, as implemented with ICAD can be thought of as an advanced form of computer applications (in some forms with an extreme end-user computing flavor) that support PLM and CAx.

KBE and product lifecycle management

The scope of PLM involves all the steps that exist within any industry that produces goods. KBE at this level will deal with product issues of a more generic nature than it will with CAx. Some might call this level 'assembly' in orientation. However, it's much more than that as PLM covers both the technical and the business side of a product.

KBE then needs to support the decision processes involved with configuration, trades, control, management, and a number of other areas, such as optimization.

Recently the Object Management Group released a RFP document and requested feedback.

KBE and CAX

CAx crosses many disciplinary bounds and provides a sound basis for PLM. In a sense, CAx is a form of applied science that uses most of the disciplines of engineering and their associated fields. Materials science comes to mind.

KBE's support of CAx may have some similarities with its support of PLM but, in a sense, the differences are going to be larger.

The KBE flavor at the CAx level may assume a strong behavioral flavor. Given the underlying object oriented focus, there is a natural use of entities possessing complicated attributes and fulfilling non-trivial roles. One vendor's approach provides a means via workbenches to embed attributes and methods within sub-parts (object) or within a joining of sub-parts into a part.

As an aggregate, the individual actions, that are event driven, can be fairly involved. This fact identifies one major problem, namely control of what is essentially a non-deterministic mixture. This characteristic of the decision problem will get more attention as the KBE systems subsume more levels and encompasses a broader scope of PLM.

KBE and knowledge management

KBE is related to knowledge management which has many levels itself. Some approaches to knowledge are reductionistic, as well they ought to be given the pragmatic focus of knowledge modeling. However, due to KBE dealing with aggregates that can be quite complicated both in structure and in behavior, some holistic notions (note link to complexity theory) might be apropos.

Also, given all the layers of KBE and given the fact that one part of an associated space is heavily mathematical (namely, manifold in nature), KBE is extremely interesting from the knowledge viewpoint (or one would hope).

All one has to do is note that the KBE process's goal is to produce results in the 'real world' via artifacts and to do so using techniques that are highly computational. That, in essence, is the epitome of applied science/engineering, and it could never be non-interesting.

KBE methodology

The development of KBE applications concerns the requirements to identify, capture, structure, formalize and finally implement knowledge. Many different so-called KBE platforms support only the implementation step which is not always the main bottleneck in the KBE development process. In order to limit the risk associated with the development and maintenance of KBE application there is a need to rely on an appropriate methodology for managing the knowledge and maintaining it up to date. As example of such KBE methodology the EU project MOKA "Methodology and tools Oriented to Knowledge based Applications" propose solutions which focus on the structuration and formalization steps as well as links to the implementation.

An alternative to MOKA is to use a general methodology for developing knowledge bases for expert systems and for intranet pages. Such a methodology is described in "Knowledge Acquisition in Practice: A Step-by-step Guide" by Nick Milton.

Languages for KBE

Some questions can be asked in regard to KBE implementation: can we represent knowledge in a vendor-neutral format? can the knowledge in our designs be retained for decades, long after a vendor system (such as CATIA) has disappeared?

These questions are addressed in a 2005 Aerospace COE presentation A Proposal for CATIA V6 by Walter Wilson of Lockheed Martin.

Mr. Wilson advocates using a type of programming language to define design data—operations, parameters, formulas, etc. -- instead of a proprietary file format (such as Dassault's CATIA). One's data would no longer be tied to a specific CAD system. Unlike STEP, which inevitably lags commercial CAD systems in the features it supports, programmability would allow the definition of new design features.

A logic programming language is proposed as the basis for the engineering design language because of its simplicity and extensibility. The geometric engine for the language features would be open source to give engineers control over approximation algorithms and to better guarantee long-term accessibility of the data.

Meanwhile, the commercially available General-purpose Declarative Language (GDL) from Genworks International addresses the issue of application longevity by providing a high-level declarative language kernel which is a superset of a standard dialect of the Lisp programming language (currently ANSI Common Lisp, or CL).

The GDL kernel follows a concise, pragmatic language specification representing something akin to a *de-facto* neutral format for representing KBE-style knowledge. It consists of the same Smalltalk-inspired declarative object-oriented (and object-centric) message-passing format which been a common thread among classical KBE systems for more than two decades. While CL is a multi-paradigm language (supporting procedural, object-oriented, and functional programming), the core features of KBE tend to share several aspects with Functional programming languages "under the hood" (e.g. lazy evaluation, immutable data). However there is a consensus that pure Functional programming is too esoteric for typical engineers to practice, so one of the purposes of a declarative KBE language such as GDL is to provide an "engineer-friendly" object-centric front-end on what is essentially a Functional language programming environment.

Because GDL applications are written as a strict superset of a standard Lisp dialect, only the high-level declarative surface syntax is GDL-specific. The bulk of application code is purely compliant with the underlying language standard. And because of Lisp's inherent (and unique) support for code transformation macros, even this surface syntax is subject to straightforward automated conversion among other variations of the de-facto standard. It is reasonable to expect that implementations following this approach will eventually converge on a true vendor-neutral Standard KBE language specification.

The Pacelab Suite adresses the problem that functional programming languages are still not widely accepted by potential KBE users. Engineers are usually trained in procedural and object-oriented programming languages; in quite a few software environments (Excel, MATLAB and FORTRAN) used by engineers the procedural programming style clearly dominates. Therefore the Pacelab Suite rebases the inherent technological advantages offered by a development and runtime environment like Common Lisp, on a new technological paradigm (.NET Framework) equally supporting procedural, object-oriented and functional languages.

KBE in Academia

- Design of Aircraft and Rotorcraft, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands, has adopted GDL from Genworks International as a basis for KBE and their Multi-model Generator (MMG).
- Knowledge-based engineering at the Norwegian University of Science and Technology (NTNU)
- Knowledge Based Engineering department at the Faculty of Aerospace Engineering of the Delft University of Technology

Implementations

The following KBE development packages are commercially available:

For CAD

- Adaptive Modeling Language from TechnoSoft Inc.
- DriveWorks A SolidWorks Certified Gold Partner
- GDL from Genworks International
- Kadviser from NIMTOTH previously edited by Kade-Tech
- KBEWorks by VisionKBE
- Knowledge Fusion from Siemens PLM Software
- Knowledgeware from Dassault Systemes
- Magix by Navitech
- Pro/ENGINEER Expert Framework from Parametric Technology Corporation
- SmartAssembly for Pro/ENGINEER from Sigmaxim Inc
- TactonWorks Interactive design automation inside SolidWorks
- YVE - Your Variant Engineer from tecneos software-engineering

- ICAD from Dassault Systemes (no longer available)
- KBMax Configurator by Citius Corporation

For General-purpose development of Web-deployed applications

- GDL from Genworks International

For analysis, design and engineering processes

- GDL from Genworks International
- Pacelab Suite by PACE Aerospace Engineering and Information Technology GmbH
- PCPACK by Tacit Connexions
- Quaestor by Qnowledge Modeling Technologies

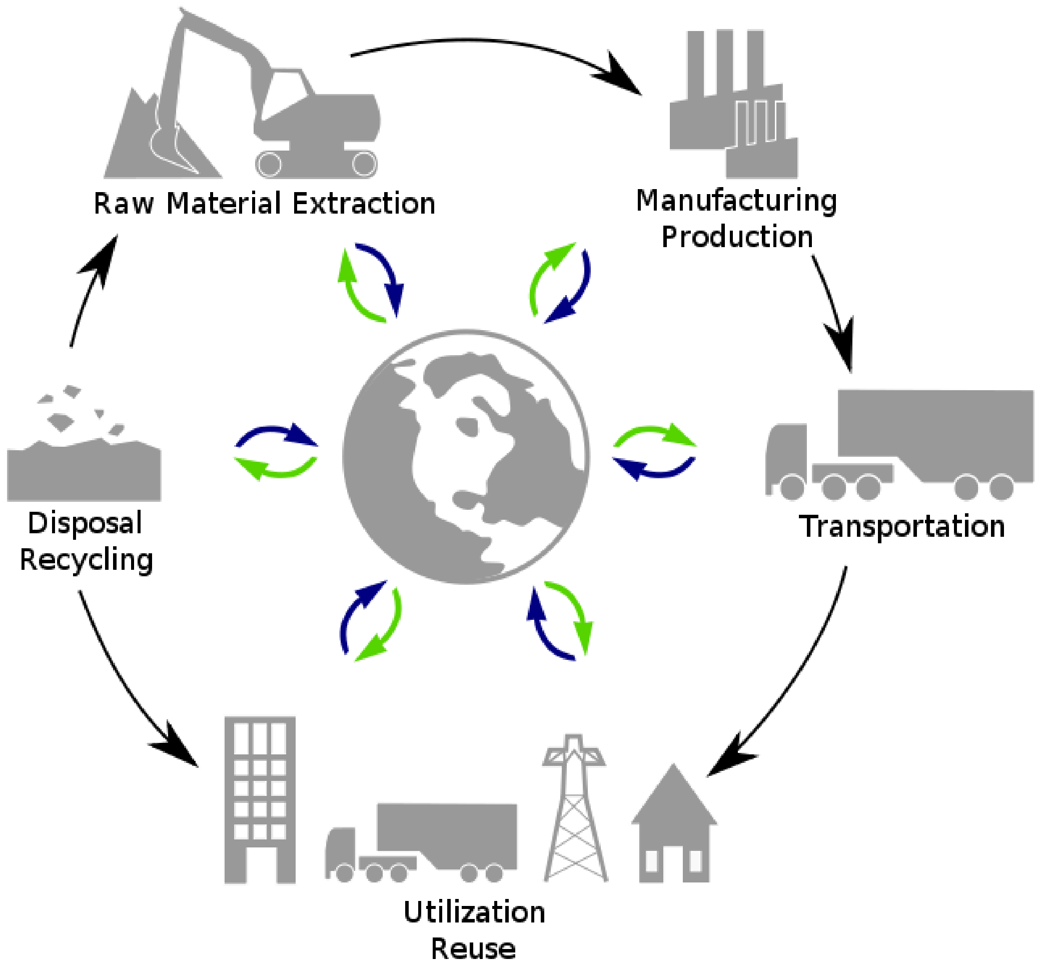
KBE futures, KBE theory

KBE, as a particular example of KS, is a multi-disciplinary framework that has more than practical considerations. Not only will KBE require successful handling of issues of the computational (Ontology, Artificial Intelligence, Entscheidungsproblem, Interactive computation, Category Theory, ...) and logic (non-monotonic issues related to the qualification, frame, and ramification problems)), it will touch upon all sciences that deal with matter, its manipulations, and the related decisions. In a sense, PLM allows us to have the world as a large laboratory for experimental co-evolution of our knowledge and our artificial co-horts. As noted in the ACM Communications, "Computers will grow to become scientists in their own right, with intuitions and computational variants of fascination and curiosity." What better framework is there to explore the "increasingly complicated mappings between the human world and the computational"?

In terms of methodology and their associated means, KBE offers support via several paradigms. These range from the home-grown all the way to strategically defined and integrated tools that cover both breadth and depth. A continuing theme will be resolving the contextual definitions for KBE into a coherent discipline (or at least attempting this) and keeping a handle on managing the necessary quantitative comparisons. One issue of importance considers what limits there may be to the computational; this study requires a multi-disciplinary focus and an understanding of the quasi-empirical. Given the knowledge focus of KBE, another issue involves what limits there might be to a computational basis for knowledge and whether these are overcome with the more advanced types of human-machine interface.

Chapter- 2

Product Lifecycle Management



A generic lifecycle of products

In industry, **product lifecycle management** (**PLM**) is the process of managing the entire lifecycle of a product from its conception, through design and manufacture, to service and disposal. PLM integrates people, data, processes and business systems and provides a product information backbone for companies and their extended enterprise.

'Product lifecycle management' (PLM) should be distinguished from 'Product life cycle management (marketing)' (PLCM). PLM describes the engineering aspect of a product, from managing descriptions and properties of a product through its development and useful life; whereas, PLCM refers to the commercial management of life of a product in the business market with respect to costs and sales measures.

Product lifecycle management is one of the four cornerstones of a corporation's information technology structure. All companies need to manage communications and information with their customers (CRM-Customer Relationship Management), their suppliers (SCM-Supply Chain Management), their resources within the enterprise (ERP-Enterprise Resource Planning) and their planning (SDLC-Systems Development Life Cycle). In addition, manufacturing engineering companies must also develop, describe, manage and communicate information about their products.

A form of PLM called people-centric PLM. While traditional PLM tools have been deployed only on release or during the release phase, people-centric PLM targets the design phase.

Recent (as of 2009) ICT development (EU funded PROMISE project 2004-2008) has allowed PLM to extend beyond traditional PLM and integrate sensor data and real time 'lifecycle event data' into PLM, as well as allowing this information to be made available to different players in the total lifecycle of an individual product (closing the information loop). This has resulted in the extension of PLM into Closed Loop Lifecycle Management (CL_2M).

Benefits

Documented benefits of product lifecycle management include:

- Reduced time to market
- Improved product quality
- Reduced prototyping costs
- More accurate and timely Request For Quote generation
- Ability to quickly identify potential sales opportunities and revenue contributions
- Savings through the re-use of original data
- A framework for product optimization
- Reduced waste
- Savings through the complete integration of engineering workflows
- Documentation that can assist in proving Compliance for RoHS or Title 21 CFR Part 11
- Ability to provide Contract Manufacturers with access to a centralized product record

Areas of PLM

Within PLM there are five primary areas;

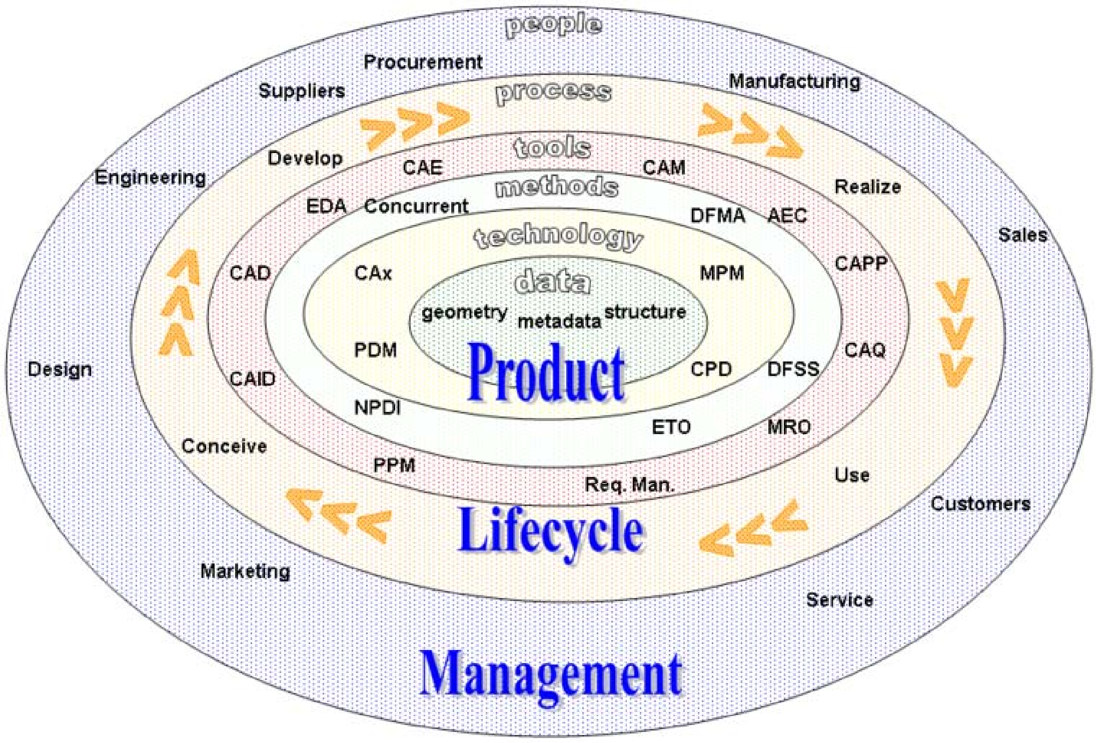
1. Systems Engineering (SE)
2. Product and Portfolio Management (PPM)
3. Product Design (CAx)
4. Manufacturing Process Management (MPM)
5. Product Data Management (PDM)

Note: While application software is not required for PLM processes, the business complexity and rate of change requires organizations execute as rapidly as possible.

Systems Engineering is focused on meeting all requirements, primary meeting customer needs, and coordinating the Systems Design process by involving all relevant disciplines. Product and Portfolio Management is focused on managing resource allocation, tracking progress vs. plan for projects in the new product development projects that are in process (or in a holding status). Portfolio management is a tool that assists management in tracking progress on new products and making trade-off decisions when allocating scarce resources. Product Data Management is focused on capturing and maintaining information on products and/or services through their development and useful life.

Introduction to development process

The core of PLM (product lifecycle management) is in the creations and central management of all product data and the technology used to access this information and knowledge. PLM as a discipline emerged from tools such as CAD, CAM and PDM, but can be viewed as the integration of these tools with methods, people and the processes through all stages of a product's life. It is not just about software technology but is also a business strategy.



For simplicity the stages described are shown in a traditional sequential engineering workflow. The exact order of event and tasks will vary according to the product and industry in question but the main processes are:

- Conceive
    - Specification
    - Concept design
- Design
    - Detailed design
    - Validation and analysis (simulation)
    - Tool design
- Realize
    - Plan manufacturing
    - Manufacture
    - Build/Assemble
    - Test (quality check)
- Service
    - Sell and Deliver
    - Use
    - Maintain and Support
    - Dispose

The major key point events are:

- Order
- Idea
- Kick-off
- Design freeze
- Launch

The reality is however more complex, people and departments cannot perform their tasks in isolation and one activity cannot simply finish and the next activity start. Design is an iterative process, often designs need to be modified due to manufacturing constraints or conflicting requirements. Where exactly a customer order fits into the time line depends on the industry type, whether the products are for example Build to Order, Engineer to Order, or Assemble to Order.

History

Inspiration for the burgeoning business process now known as PLM came when American Motors Corporation (AMC) was looking for a way to speed up its product development process to compete better against its larger competitors in 1985, according to François Castaing, Vice President for Product Engineering and Development. After introducing its compact Jeep Cherokee (XJ), the vehicle that launched the modern sport utility vehicle (SUV) market, AMC began development of a new model, that later came out as the Jeep Grand Cherokee. The first part in its quest for faster product development was computer-aided design (CAD) software system that make engineers more productive. The second part in this effort was the new communication system that allowed conflicts to be resolved faster, as well as reducing costly engineering changes because all drawings and documents were in a central database. The product data management was so effective, that after AMC was purchased by Chrysler, the system was expanded throughout the enterprise connecting everyone involved in designing and building products. While an early adopter of PLM technology, Chrysler was able to become the auto industry's lowest-cost producer, recording development costs that were half of the industry average by the mid-1990s.

Phases of product lifecycle and corresponding technologies

Many software solutions have developed to organize and integrate the different phases of a product's lifecycle. PLM should not be seen as a single software product but a collection of software tools and working methods integrated together to address either single stages of the lifecycle or connect different tasks or manage the whole process. Some software providers cover the whole PLM range while others a single niche application. Some applications can span many fields of PLM with different modules within the same data model. An overview of the fields within PLM is covered here. It should be noted however that the simple classifications do not always fit exactly, many areas overlap and many software products cover more than one area or do not fit easily into one category. It should also not be forgotten that one of the main goals of PLM is to collect knowledge that can be reused for other projects and to coordinate simultaneous concurrent development of many products. It is about business processes, people and methods as much as software application solutions. Although PLM is mainly associated with engineering tasks it also involves marketing activities such as Product Portfolio Management (PPM), particularly with regards to New product introduction (NPI).

Phase 1: Conceive

Imagine, specify, plan, innovate

The first stage in idea is the definition of its requirements based on customer, company, market and regulatory bodies' viewpoints. From this specification of the products major technical parameters can be defined. Parallel to the requirements specification the initial concept design work is carried out defining the visual aesthetics of the product together with its main functional aspects. For the Industrial Design, Styling, work many different media are used from pencil and paper, clay models to 3D CAID Computer-aided industrial design software.

Phase 2: Design

Describe, define, develop, test, analyze and validate

This is where the detailed design and development of the product's form starts, progressing to prototype testing, through pilot release to full product launch. It can also involve redesign and ramp for improvement to existing products as well as planned obsolescence. The main tool used for design and development is CAD Computer-aided design. This can be simple 2D Drawing / Drafting or 3D Parametric Feature Based Solid/Surface Modeling. Such software includes technology such as Hybrid Modeling, Reverse Engineering, KBE (Knowledge-Based Engineering), NDT (Nondestructive testing), Assembly construction.

This step covers many engineering disciplines including: Mechanical, Electrical, Electronic, Software (embedded), and domain-specific, such as Architectural, Aerospace, Automotive, ... Along with the actual creation of geometry there is the analysis of the components and product assemblies. Simulation, validation and optimization tasks are carried out using CAE (Computer-aided engineering) software either integrated in the CAD package or stand-alone. These are used to perform tasks such as:- Stress analysis, FEA (Finite Element Analysis); Kinematics; Computational fluid dynamics (CFD); and mechanical event simulation (MES). CAQ (Computer-aided quality) is used for tasks such as Dimensional Tolerance (engineering) Analysis. Another task performed at this stage is the sourcing of bought out components, possibly with the aid of Procurement systems.

Phase 3: Realize

Manufacture, make, build, procure, produce, sell and deliver

Once the design of the product's components is complete the method of manufacturing is defined. This includes CAD tasks such as tool design; creation of CNC Machining instructions for the product's parts as well as tools to manufacture those parts, using integrated or separate CAM Computer-aided manufacturing software. This will also involve analysis tools for process simulation for operations such as casting, molding, and die press forming. Once the manufacturing method has been identified CPM comes into play. This involves CAPE (Computer-aided Production Engineering) or CAP/CAPP – (Production Planning) tools for carrying out Factory, Plant and Facility Layout and Production Simulation. For example: Press-Line Simulation; and Industrial Ergonomics; as well as tool selection management. Once components are manufactured their geometrical form and size can be checked against the original CAD data with the use of Computer Aided Inspection equipment and software. Parallel to the engineering tasks, sales product configuration and marketing documentation work will be taking place. This could include transferring engineering data (geometry and part list data) to a web based sales configurator and other Desktop Publishing systems.

Phase 4: Service

Use, operate, maintain, support, sustain, phase-out, retire, recycle and disposal

The final phase of the lifecycle involves managing of in service information. Providing customers and service engineers with support information for repair and maintenance, as well as waste management/recycling information. This involves using such tools as Maintenance, Repair and Operations Management (MRO) software.

All phases: product lifecycle

Communicate, manage and collaborate

None of the above phases can be seen in isolation. In reality a project does not run sequentially or in isolation of other product development projects. Information is flowing between different people and systems. A major part of PLM is the co-ordination of and management of product definition data. This includes managing engineering changes and release status of components; configuration product variations; document management; planning project resources and timescale and risk assessment.

For these tasks graphical, text and metadata such as product bills of materials (BOMs) needs to be managed. At the engineering departments level this is the domain of PDM – (Product Data Management) software, at the corporate level EDM (Enterprise Data Management) software, these two definitions tend to blur however but it is typical to see two or more data management systems within an organization. These systems are also linked to other corporate systems such as SCM, CRM, and ERP. Associated with these system are Project Management Systems for Project/Program Planning.

This central role is covered by numerous Collaborative Product Development tools which run throughout the whole lifecycle and across organizations. This requires many technology tools in the areas of Conferencing, Data Sharing and Data Translation. The field being Product visualization which includes technologies such as DMU (Digital Mock-Up), Immersive Virtual Digital Prototyping (virtual reality) and Photo realistic Imaging.

User skills

The broad array of solutions that make up the tools used within a PLM solution-set (e.g., CAD, CAM, CAx...) were initially used by dedicated practitioners who invested time and effort to gain the required skills. Designers and engineers worked wonders with CAD systems, manufacturing engineers became highly skilled CAM users while analysts, administrators and managers fully mastered their support technologies. However, achieving the full advantages of PLM requires the participation of many people of various skills from throughout an extended enterprise, each requiring the ability to access and operate on the inputs and output of other participants.

Despite the increased ease of use of PLM tools, cross-training all personnel on the entire PLM tool-set has not proven to be practical. Now, however, advances are being made to address ease of use for all participants within the PLM arena. One such advance is the availability of “role” specific user interfaces. Through Tailorable UIs, the commands that are presented to users are appropriate to their function and expertise.

Product development processes and methodologies

A number of established methodologies have been adopted by PLM and been further advanced. Together with PLM digital engineering techniques, they have been advanced to meet company goals such as reduced time to market and lower production costs. Reducing lead times is a major factor as getting a product to market quicker than the competition will help with higher revenue and profit margins and increase market share.

These techniques include:-

- Concurrent engineering workflow
- Industrial Design
- Bottom-up design
- Top-down design
- Front loading design workflow
- Design in context
- Modular design
- NPD New product development
- DFSS Design for Six Sigma

- DFMA Design for manufacture / assembly
- Digital simulation engineering
- Requirement driven design
- Specification managed validation
- Configuration Management

Concurrent engineering workflow

Concurrent engineering (British English: **simultaneous engineering**) is a workflow that, instead of working sequentially through stages, carries out a number of tasks in parallel. For example: starting tool design before the detailed designs of the product are finished, or starting on detail design solid models before the concept design surfaces models are complete. Although this does not necessarily reduce the amount of manpower required for a project, it does drastically reduce lead times and thus time to market. Feature-based CAD systems have for many years allowed the simultaneous work on 3D solid model and the 2D drawing by means of two separate files, with the drawing looking at the data in the model; when the model changes the drawing will associatively update. Some CAD packages also allow associative copying of geometry between files. This allows, for example, the copying of a part design into the files used by the tooling designer. The manufacturing engineer can then start work on tools before the final design freeze; when a design changes size or shape the tool geometry will then update. Concurrent engineering also has the added benefit of providing better and more immediate communication between departments, reducing the chance of costly, late design changes. It adopts a problem prevention method as compared to the problem solving and re-designing method of traditional sequential engineering.

Bottom-up design

Bottom-up design (CAD Centric) occurs where the definition of 3D models of a product starts with the construction of individual components. These are then virtually brought together in sub-assemblies of more than one level until the full product is digitally defined. This is sometimes known as the review structure showing what the product will look like. The BOM contains all of the physical (solid) components; it may (but not also) contain other items required for the final product BOM such as paint, glue, oil and other materials commonly described as 'bulk items'. Bulk items typically have mass and quantities but are not usually modelled with geometry.

Bottom-up design tends to focus on the capabilities of available real-world physical technology, implementing those solutions which this technology is most suited to. When these bottom-up solutions have real-world value, bottom-up design can be much more efficient than top-down design. The risk of bottom-up design is that it very efficiently provides solutions to low-value problems. The focus of Bottom-Up design is "what can we most efficiently do with this technology?" rather than the focus of Top-Down which is "What is the most valuable thing to do?"

Top-down design

Top-Down design is focused on high-level functional requirements, with relatively less focus on existing implementation technology. A top level spec is decomposed into lower and lower level structures and specifications, until the physical implementation layer is reached. The risk of a top-down design is that it will not take advantage of the most efficient applications of current physical technology, especially with respect to hardware implementation. Top-Down design sometimes results in excessive layers of lower-level abstraction and inefficient performance when the Top-Down model has followed an abstraction path which does not efficiently fit available physical-level technology. The positive value of Top-Down design is that it preserves a focus on the optimum solution requirements.

A Part-Centric Top-down design may eliminate some of the risks of Top-Down design. This starts with a layout model, often a simple 2D sketch defining basic sizes and some major defining parameters. Industrial Design, brings creative ideas to product development. Geometry from this is associatively copied down to the next level, which represents different sub-systems of the product. The geometry in the sub-systems is then used to define more detail in levels below. Depending on the complexity of the product, a number of levels of this assembly are created until the basic definition of components can be identified, such as position and principal dimensions. This information is then associatively copied to component files. In these files the components are detailed; this is where the classic bottom-up assembly starts.

The top down assembly is sometime known as a control structure. If a single file is used to define the layout and parameters for the review structure it is often known as a skeleton file.

Defense engineering traditionally develops the product structure from the top down. The system engineering process prescribes a functional decomposition of requirements and then physical allocation of product structure to the functions. This top down approach would normally have lower levels of the product structure developed from CAD data as a bottom up structure or design.

Both-Ends-Against-The-Middle design

Both-Ends-Against-The-Middle (BEATM) design is a design process that endeavors to combine the best features of Top-Down design, and Bottom-Up design into one process. A BEATM design process flow may begin with an emergent technology which suggests solutions which may have value, or it may begin with a top-down view of an important problem which needs a solution. In either case the key attribute of BEATM design methodology is to immediately focus at both ends of the design process flow: a top-down view of the solution requirements, and a bottom-up view of the available technology which may offer promise of an efficient solution. The BEATM design process proceeds from both ends in search of an optimum merging somewhere between the top-down requirements, and bottom-up efficient implementation. In this fashion, BEATM has been shown to genuinely offer the best of both methodologies. Indeed some of the best success stories from either top-down or bottom-up have been successful because of an intuitive, yet unconscious use of the BEATM methodology. When employed consciously, BEATM offers even more powerful advantages.

Front loading design and workflow

Front loading is taking top-down design to the next stage. The complete control structure and review structure, as well as downstream data such as drawings, tooling development and CAM models, are constructed before the product has been defined or a project kick-off has been authorized. These assemblies of files constitute a template from which a family of products can be constructed. When the decision has been made to go with a new product, the parameters of the product are entered into the template model and all the associated data is updated. Obviously predefined associative models will not be able to predict all possibilities and will require additional work. The main principle is that a lot of the experimental/investigative work has already been completed. A lot of knowledge is built into these templates to be reused on new products. This does require additional resources "up front" but can drastically reduce the time between project kick-off and launch. Such methods do however require organizational changes, as considerable engineering efforts are moved into "offline" development departments. It can be seen as an analogy to creating a concept car to test new technology for future products, but in this case the work is directly used for the next product generation.

Design in context

Individual components cannot be constructed in isolation. CAD; CAiD models of components are designed within the context of part or all of the product being developed. This is achieved using assembly modelling techniques. Other components' geometry can be seen and referenced within the CAD tool being used. The other components within the sub-assembly, may or may not have been constructed in the same system, their geometry being translated from other CPD formats. Some assembly checking such as DMU is also carried out using Product visualization software.

Product and process lifecycle management (PPLM)

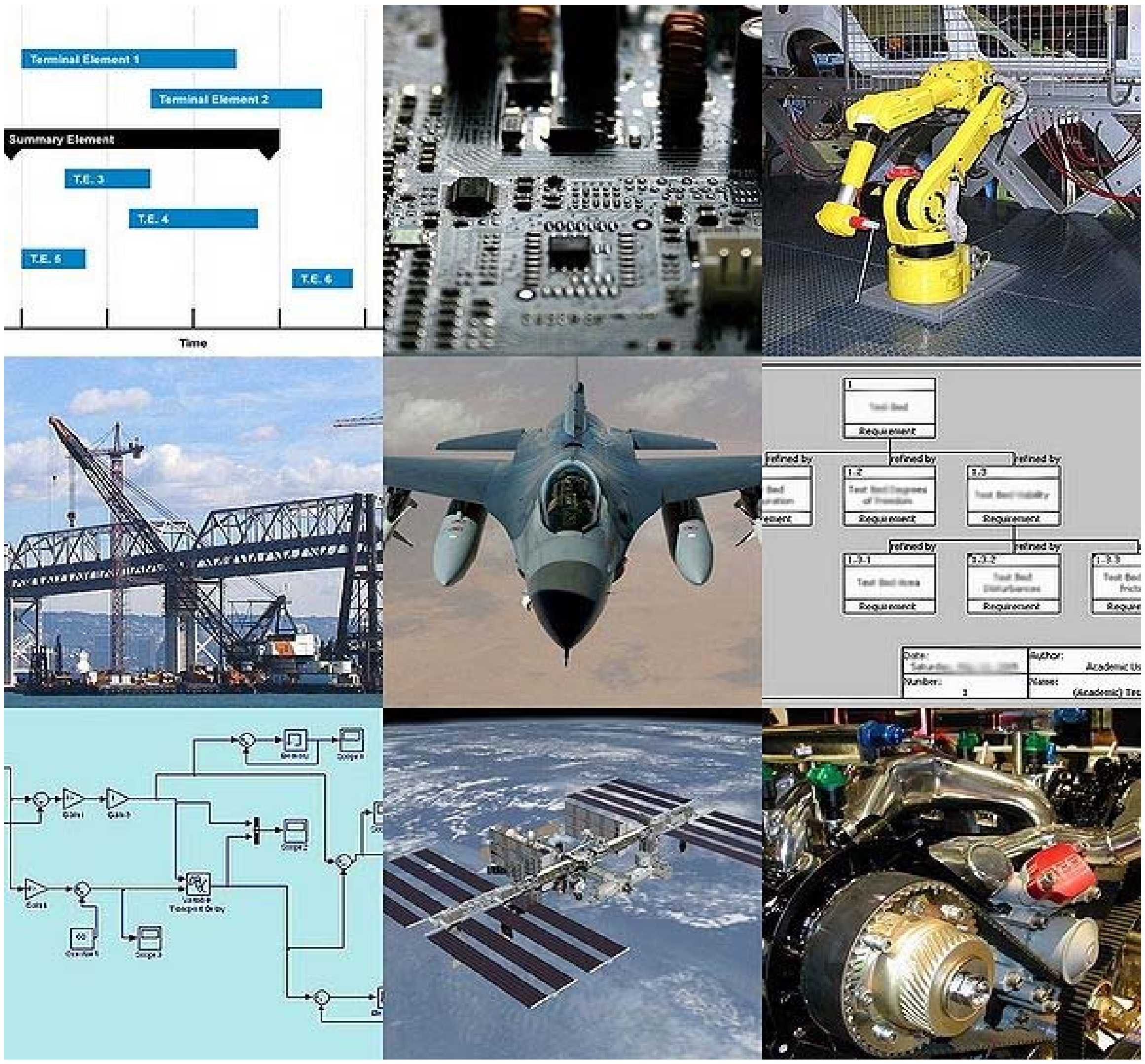
Product and process lifecycle management **(PPLM)** is an alternate genre of PLM in which the process by which the product is made is just as important as the product itself. Typically, this is the life sciences and advanced specialty chemicals markets. The process behind the manufacture of a given compound is a key element of the regulatory filing for a new drug application. As such, PPLM seeks to manage information around the development of the process in a similar fashion that baseline PLM talks about managing information around development of the product.

Market size

Total spending on PLM software and services was estimated in 2006 to be above $15 billion a year, but it is difficult to find any two market analysis reports that agree on figures. Market growth estimates are in the 10% area.

Chapter- 3

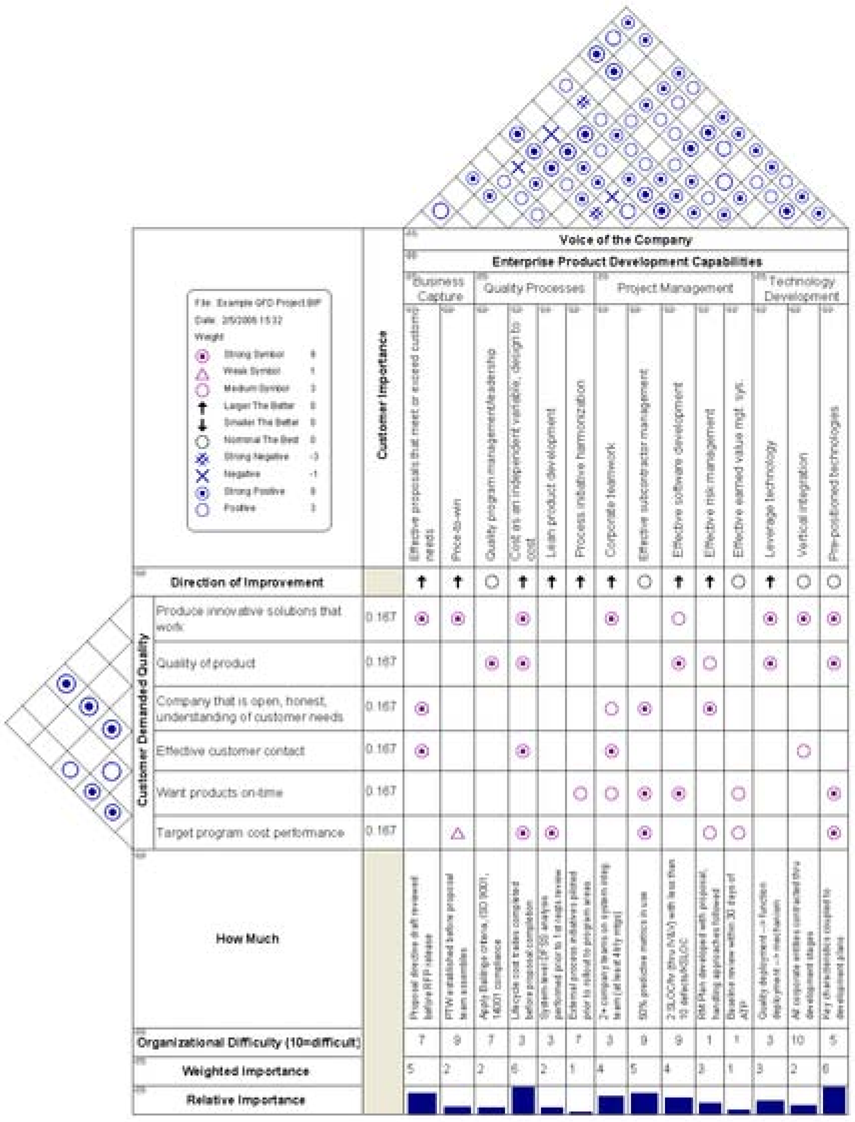
Systems Engineering



Systems engineering techniques are used in complex projects: spacecraft design, computer chip design, robotics, software integration, and bridge building. Systems engineering uses a host of tools that include modeling and simulation, requirements analysis and scheduling to manage complexity.

Systems engineering is an interdisciplinary field of engineering that focuses on how complex engineering projects should be designed and managed over the life cycle of the project.. Issues such as logistics, the coordination of different teams, and automatic control of machinery become more difficult when dealing with large, complex projects. Systems engineering deals with work-processes and tools to handle such projects, and it overlaps with both technical and human-centered disciplines such as control engineering, industrial engineering, organizational studies, and project management.

History



Voice of the Company
Enterprise Product Development Capabilities
Business Capture
Quality Processes
Project Management
Technology Development
Customer Importance
Customer Demanded Quality
Direction of Improvement
Produce innovative solutions that work
Quality of product
Company that is open, honest, understanding of customer needs
Effective customer contact
Want products on-time
Target program cost performance
How Much
Organizational Difficulty (10=difficult)
Weighted Importance
Relative Importance

QFD House of Quality for Enterprise Product Development Processes

The term *systems engineering* can be traced back to Bell Telephone Laboratories in the 1940s. The need to identify and manipulate the properties of a system as a whole, which in complex engineering projects may greatly differ from the sum of the parts' properties, motivated the Department of Defense, NASA, and other industries to apply the discipline.

When it was no longer possible to rely on design evolution to improve upon a system and the existing tools were not sufficient to meet growing demands, new methods began to be developed that addressed the complexity directly. The evolution of systems engineering, which continues to this day, comprises the development and identification of new methods and modeling techniques. These methods aid in better comprehension of engineering systems as they grow more complex. Popular tools that are often used in the systems engineering context were developed during these times, including USL, UML, QFD, and IDEF0.

In 1990, a professional society for systems engineering, the *National Council on Systems Engineering* (NCOSE), was founded by representatives from a number of U.S. corporations and organizations. NCOSE was created to address the need for improvements in systems engineering practices and education. As a result of growing involvement from systems engineers outside of the U.S., the name of the organization was changed to the International Council on Systems Engineering (INCOSE) in 1995. Schools in several countries offer graduate programs in systems engineering, and continuing education options are also available for practicing engineers.

Concept

Systems engineering signifies both an approach and, more recently, as a discipline in engineering. The aim of education in systems engineering is to simply formalize the approach and in doing so, identify new methods and research opportunities similar to the way it occurs in other fields of engineering. As an approach, systems engineering is holistic and interdisciplinary in flavour.

Origins and traditional scope

The traditional scope of engineering embraces the design, development, production and operation of physical systems, and systems engineering, as originally conceived, falls within this scope. "Systems engineering", in this sense of the term, refers to the distinctive set of concepts, methodologies, organizational structures (and so on) that have been developed to meet the challenges of engineering functional physical systems of unprecedented complexity. The Apollo program is a leading example of a systems engineering project.

The use of the term " system engineer " has evolved over time to embrace a wider, more holistic concept of "systems" and of engineering processes. This evolution of the definition has been a subject of ongoing controversy , and the term continues to be applied to both the narrower and broader scope.

Holistic view

Systems engineering focuses on analyzing and eliciting customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem, the system lifecycle. Oliver *et al.* claim that the systems engineering process can be decomposed into

- a *Systems Engineering Technical Process*, and
- a *Systems Engineering Management Process*.

Within Oliver's model, the goal of the Management Process is to organize the technical effort in the lifecycle, while the Technical Process includes *assessing available information*, *defining effectiveness measures*, to *create a behavior model*, *create a structure model*, *perform trade-off analysis*, and *create sequential build & test plan*.

Depending on their application, although there are several models that are used in the industry, all of them aim to identify the relation between the various stages mentioned above and incorporate feedback. Examples of such models include the Waterfall model and the VEE model.

Interdisciplinary field

System development often requires contribution from diverse technical disciplines. By providing a systems (holistic) view of the development effort, systems engineering helps mold all the technical contributors into a unified team effort, forming a structured development process that proceeds from concept to production to operation and, in some cases, to termination and disposal.

This perspective is often replicated in educational programs in that systems engineering courses are taught by faculty from other engineering departments which, in effect, helps create an interdisciplinary environment.

Managing complexity

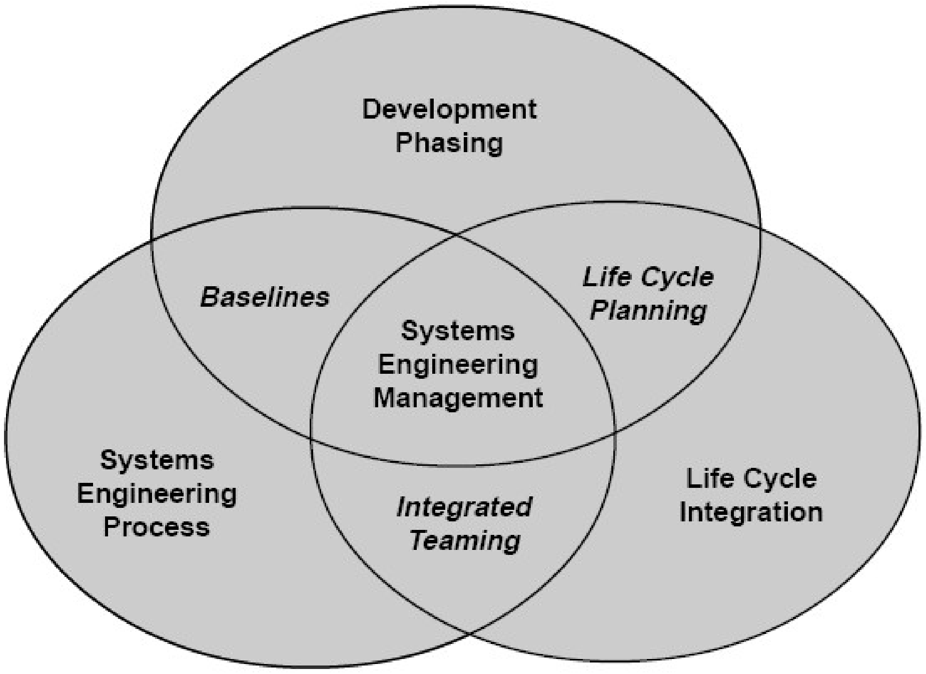
The need for systems engineering arose with the increase in complexity of systems and projects. When speaking in this context, complexity incorporates not only engineering systems, but also the logical human organization of data. At the same time, a system can become more complex due to an increase in size as well as with an increase in the amount of data, variables, or the number of fields that are involved in the design. The International Space Station is an example of such a system.

The development of smarter control algorithms, microprocessor design, and analysis of environmental systems also come within the purview of systems engineering. Systems engineering encourages the use of tools and methods to better comprehend and manage complexity in systems. Some examples of these tools can be seen here:

- *System model, Modeling, and Simulation*,
- System architecture,
- *Optimization*,
- *System dynamics*,
- *Systems analysis*,
- *Statistical analysis*,
- *Reliability analysis*, and
- *Decision making*

Taking an interdisciplinary approach to engineering systems is inherently complex since the behavior of and interaction among system components is not always immediately well defined or understood. Defining and characterizing such systems and subsystems and the interactions among them is one of the goals of systems engineering. In doing so, the gap that exists between informal requirements from users, operators, marketing organizations, and technical specifications is successfully bridged.

Scope



The scope of systems engineering activities

One way to understand the motivation behind systems engineering is to see it as a method, or practice, to identify and improve common rules that exist within a wide variety of systems. Keeping this in mind, the principles of systems engineering —

holism, emergent behavior, boundary, et al. — can be applied to any system, complex or otherwise, provided systems thinking is employed at all levels. Besides defense and aerospace, many information and technology based companies, software development firms, and industries in the field of electronics & communications require systems engineers as part of their team.

An analysis by the INCOSE Systems Engineering center of excellence (SECOE) indicates that optimal effort spent on systems engineering is about 15-20% of the total project effort. At the same time, studies have shown that systems engineering essentially leads to reduction in costs among other benefits. However, no quantitative survey at a larger scale encompassing a wide variety of industries has been conducted until recently. Such studies are underway to determine the effectiveness and quantify the benefits of systems engineering.

Systems engineering encourages the use of modeling and simulation to validate assumptions or theories on systems and the interactions within them.

Use of methods that allow early detection of possible failures, in safety engineering, are integrated into the design process. At the same time, decisions made at the beginning of a project whose consequences are not clearly understood can have enormous implications later in the life of a system, and it is the task of the modern systems engineer to explore these issues and make critical decisions. There is no method which guarantees that decisions made today will still be valid when a system goes into service years or decades after it is first conceived but there are techniques to support the process of systems engineering. Examples include the use of soft systems methodology, Jay Wright Forrester's System dynamics method and the Unified Modeling Language (UML), each of which are currently being explored, evaluated and developed to support the engineering decision making process.

Education

Education in systems engineering is often seen as an extension to the regular engineering courses, reflecting the industry attitude that engineering students need a foundational background in one of the traditional engineering disciplines (e.g. mechanical engineering, industrial engineering, computer engineering, electrical engineering) plus practical, real-world experience in order to be effective as systems engineers. Undergraduate university programs in systems engineering are rare.

INCOSE maintains a continuously updated Directory of Systems Engineering Academic Programs worldwide. As of 2006, there are about 75 institutions in United States that offer 130 undergraduate and graduate programs in systems engineering. Education in systems engineering can be taken as *SE-centric* or *Domain-centric*.

- *SE-centric* programs treat systems engineering as a separate discipline and all the courses are taught focusing on systems engineering practice and techniques.

- *Domain-centric* programs offer systems engineering as an option that can be exercised with another major field in engineering.

Both these patterns cater to educate the systems engineer who is able to oversee interdisciplinary projects with the depth required of a core-engineer.

Systems engineering topics

Systems engineering tools are strategies, procedures, and techniques that aid in performing systems engineering on a project or product. The purpose of these tools vary from database management, graphical browsing, simulation, and reasoning, to document production, neutral import/export and more.

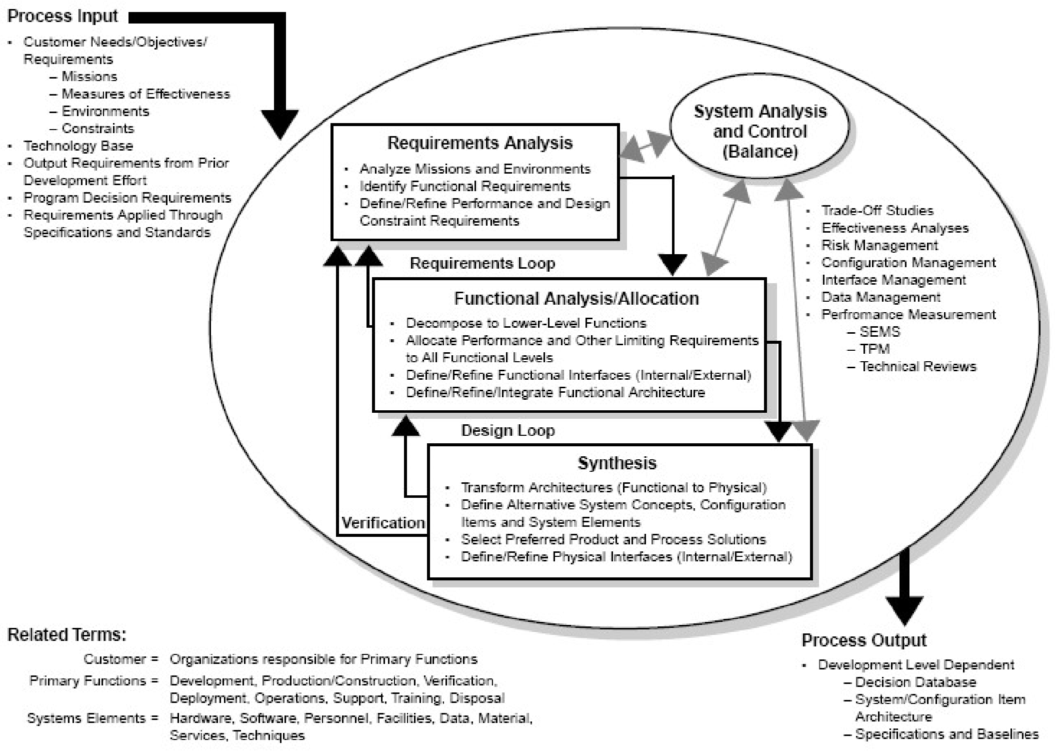
System

There are many definitions of what a system is in the field of systems engineering. Below are a few authoritative definitions:

- ANSI/EIA-632-1999: "An aggregation of end products and enabling products to achieve a given purpose."
- IEEE Std 1220-1998: "A set or arrangement of elements and processes that are related and whose behavior satisfies customer/operational needs and provides for life cycle sustainment of the products."
- ISO/IEC 15288:2008: "A combination of interacting elements organized to achieve one or more stated purposes."
- NASA Systems Engineering Handbook: "(1) The combination of elements that function together to produce the capability to meet a need. The elements include all hardware, software, equipment, facilities, personnel, processes, and procedures needed for this purpose. (2) The end product (which performs operational functions) and enabling products (which provide life-cycle support services to the operational end products) that make up a system."
- INCOSE Systems Engineering Handbook: "homogeneous entity that exhibits predefined behavior in the real world and is composed of heterogeneous parts that do not individually exhibit that behavior and an integrated configuration of components and/or subsystems."
- INCOSE: "A system is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce systems-level results. The results include system level qualities, properties, characteristics, functions, behavior and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected."

The systems engineering process

Depending on their application, tools are used for various stages of the systems engineering process:


Process Input
• Customer Needs/Objectives/ Requirements
– Missions
– Measures of Effectiveness
– Environments
– Constraints
• Technology Base
• Output Requirements from Prior Development Effort
• Program Decision Requirements
• Requirements Applied Through Specifications and Standards
Requirements Analysis
• Analyze Missions and Environments
• Identify Functional Requirements
• Define/Refine Performance and Design Constraint Requirements
System Analysis and Control (Balance)
• Trade-Off Studies
• Effectiveness Analyses
• Risk Management
• Configuration Management
• Interface Management
• Data Management
• Perfromance Measurement
– SEMS
– TPM
– Technical Reviews
Requirements Loop
Functional Analysis/Allocation
• Decompose to Lower-Level Functions
• Allocate Performance and Other Limiting Requirements to All Functional Levels
• Define/Refine Functional Interfaces (Internal/External)
• Define/Refine/Integrate Functional Architecture
Design Loop
Synthesis
• Transform Architectures (Functional to Physical)
• Define Alternative System Concepts, Configuration Items and System Elements
• Select Preferred Product and Process Solutions
• Define/Refine Physical Interfaces (Internal/External)
Verification
Related Terms:
Customer = Organizations responsible for Primary Functions
Primary Functions = Development, Production/Construction, Verification, Deployment, Operations, Support, Training, Disposal
Systems Elements = Hardware, Software, Personnel, Facilities, Data, Material, Services, Techniques
Process Output
• Development Level Dependent
– Decision Database
– System/Configuration Item Architecture
– Specifications and Baselines

Using models

Models play important and diverse roles in systems engineering. A model can be defined in several ways, including:

- An abstraction of reality designed to answer specific questions about the real world
- An imitation, analogue, or representation of a real world process or structure; or
- A conceptual, mathematical, or physical tool to assist a decision maker.

Together, these definitions are broad enough to encompass physical engineering models used in the verification of a system design, as well as schematic models like a functional flow block diagram and mathematical (i.e., quantitative) models used in the trade study process.

The main reason for using mathematical models and diagrams in trade studies is to provide estimates of system effectiveness, performance or technical attributes, and cost from a set of known or estimable quantities. Typically, a collection of separate models is needed to provide all of these outcome variables. The heart of any mathematical model is a set of meaningful quantitative relationships among its inputs and outputs. These relationships can be as simple as adding up constituent quantities to obtain a total, or as complex as a set of differential equations describing the trajectory of a spacecraft in a gravitational field. Ideally, the relationships express causality, not just correlation.

Tools for graphic representations

Initially, when the primary purpose of a systems engineer is to comprehend a complex problem, graphic representations of a system are used to communicate a system's functional and data requirements. Common graphical representations include:

- Functional Flow Block Diagram (FFBD)
- VisSim
- Data Flow Diagram (DFD)
- N2 (N-Squared) Chart
- IDEF0 Diagram
- UML Use case diagram
- UML Sequence diagram
- USL Function Maps and Type Maps.
- Enterprise Architecture frameworks, like TOGAF, MODAF, Zachman Frameworks etc.

A graphical representation relates the various subsystems or parts of a system through functions, data, or interfaces. Any or each of the above methods are used in an industry based on its requirements. For instance, the N2 chart may be used where interfaces between systems is important. Part of the design phase is to create structural and behavioral models of the system.

Once the requirements are understood, it is now the responsibility of a systems engineer to refine them, and to determine, along with other engineers, the best technology for a job. At this point starting with a trade study, systems engineering encourages the use of weighted choices to determine the best option. A decision matrix, or Pugh method, is one way (QFD is another) to make this choice while considering all criteria that are important. The trade study in turn informs the design which again affects the graphic representations of the system (without changing the requirements). In an SE process, this stage represents the iterative step that is carried out until a feasible solution is found. A decision matrix is often populated using techniques such as statistical analysis, reliability analysis, system dynamics (feedback control), and optimization methods.

At times a systems engineer must assess the existence of feasible solutions, and rarely will customer inputs arrive at only one. Some customer requirements will produce no feasible solution. Constraints must be traded to find one or more feasible solutions. The customers' wants become the most valuable input to such a trade and cannot be assumed. Those wants/desires may only be discovered by the customer once the customer finds that he has overconstrained the problem. Most commonly, many feasible solutions can be found, and a sufficient set of constraints must be defined to produce an optimal solution. This situation is at times advantageous because one can present an opportunity to improve the design towards one or many ends, such as cost or schedule. Various modeling methods can be used to solve the problem including constraints and a cost function.

Systems Modeling Language (SysML), a modeling language used for systems engineering applications, supports the specification, analysis, design, verification and validation of a broad range of complex systems.

Universal Systems Language (USL) is a systems oriented object modeling language with executable (computer independent) semantics for defining complex systems, including software.

Closely related fields

Many related fields may be considered tightly coupled to systems engineering. These areas have contributed to the development of systems engineering as a distinct entity.

Cognitive systems engineering

Cognitive systems engineering (CSE) is a specific approach to the description and analysis of human-machine systems or sociotechnical systems. The three main themes of CSE are how humans cope with complexity, how work is accomplished by the use of artefacts, and how human-machine systems and socio-technical systems can be described as joint cognitive systems. CSE has since its beginning become a recognised scientific discipline, sometimes also referred to as Cognitive Engineering. The concept of a Joint Cognitive System (JCS) has in particular become widely used as a way of understanding how complex socio-technical systems can be described with varying degrees of resolution. The experience with CSE has been described in two books that summarises the field after more than 20 years of work, namely and.

Configuration Management

Like systems engineering, Configuration Management as practiced in the defence and aerospace industry is a broad systems-level practice. The field parallels the taskings of systems engineering; where systems engineering deals with requirements development, allocation to development items and verification, Configuration Management deals with requirements capture, traceability to the development item, and audit of development item to ensure that it has achieved the desired functionality that systems engineering and/or Test and Verification Engineering have proven out through objective testing.

Control engineering

Control engineering and its design and implementation of control systems, used extensively in nearly every industry, is a large sub-field of systems engineering. The cruise control on an automobile and the guidance system for a ballistic missile are two examples. Control systems theory is an active field of applied mathematics involving the investigation of solution spaces and the development of new methods for the analysis of the control process.

Industrial engineering

Industrial engineering is a branch of engineering that concerns the development, improvement, implementation and evaluation of integrated systems of people, money, knowledge, information, equipment, energy, material and process. Industrial engineering draws upon the principles and methods of engineering analysis and synthesis, as well as mathematical, physical and social sciences together with the principles and methods of engineering analysis and design to specify, predict and evaluate the results to be obtained from such systems.

Interface design

Interface design and its specification are concerned with assuring that the pieces of a system connect and inter-operate with other parts of the system and with external systems as necessary. Interface design also includes assuring that system interfaces be able to accept new features, including mechanical, electrical and logical interfaces, including reserved wires, plug-space, command codes and bits in communication protocols. This is known as extensibility. Human-Computer Interaction (HCI) or Human-Machine Interface (HMI) is another aspect of interface design, and is a critical aspect of modern systems engineering. Systems engineering principles are applied in the design of network protocols for local-area networks and wide-area networks.

Mechatronic engineering

Mechatronic engineering, like Systems engineering, is a multidisciplinary field of engineering that uses dynamical systems modeling to express tangible constructs. In that regards it is almost indistinguishable from Systems Engineering, but what sets it apart is the focus on smaller details rather than larger generalizations and relationships. As such, both fields are distinguished by the scope of their projects rather than the methodology of their practice.

Operations research

Operations research supports systems engineering. The tools of operations research are used in systems analysis, decision making, and trade studies. Several schools teach SE courses within the operations research or industrial engineering department, highlighting the role systems engineering plays in complex projects. Operations research, briefly, is concerned with the optimization of a process under multiple constraints.

Performance engineering

Performance engineering is the discipline of ensuring a system will meet the customer's expectations for performance throughout its life. Performance is usually defined as the speed with which a certain operation is executed or the capability of executing a number of such operations in a unit of time. Performance may be degraded when an operations queue to be executed is throttled when the capacity is of the system is limited. For example, the performance of a packet-switched network would be characterised by the end-to-end packet transit delay or the number of packets switched within an hour. The design of high-performance systems makes use of analytical or simulation modeling, whereas the delivery of high-performance implementation involves thorough performance testing. Performance engineering relies heavily on statistics, queueing theory and probability theory for its tools and processes.

Program management and project management.

Program management (or programme management) has many similarities with systems engineering, but has broader-based origins than the engineering ones of systems engineering. Project management is also closely related to both program management and systems engineering.

Proposal engineering

Proposal engineering is the application of scientific and mathematical principles to design, construct, and operate a cost-effective proposal development system. Basically, proposal engineering uses the "systems engineering process" to create a cost effective proposal and increase the odds of a successful proposal.

Reliability engineering

Reliability engineering is the discipline of ensuring a system will meet the customer's expectations for reliability throughout its life; i.e. it will not fail more frequently than expected. Reliability engineering applies to all aspects of the system. It is closely associated with maintainability, availability and logistics engineering. Reliability engineering is always a critical component of safety engineering, as in failure modes and effects analysis (FMEA) and hazard fault tree analysis, and of security engineering. Reliability engineering relies heavily onstatistics, probability theory and reliability theory for its tools and processes.

Safety engineering

The techniques of safety engineering may be applied by non-specialist engineers in designing complex systems to minimize the probability of safety-critical failures. The "System Safety Engineering" function helps to identify "safety hazards" in emerging designs, and may assist with techniques to "mitigate" the effects of (potentially) hazardous conditions that cannot be designed out of systems.

Security engineering

Security engineering can be viewed as an interdisciplinary field that integrates the community of practice for control systems design, reliability, safety and systems engineering. It may involve such sub-specialties as authentication of system users, system targets and others: people, objects and processes.

Software engineering

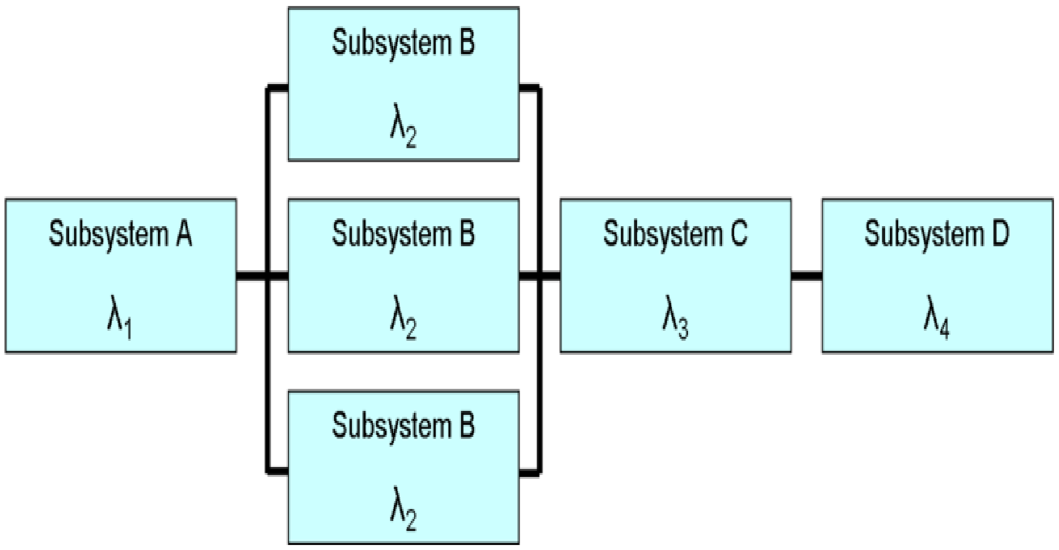
From its beginnings, software engineering has helped shape modern systems engineering practice. The techniques used in the handling of complexes of large software-intensive systems has had a major effect on the shaping and reshaping of the tools, methods and processes of SE.

Chapter- 4

Reliability Engineering

Reliability engineering is an engineering field, that deals with the study of reliability: the ability of a system or component to perform its required functions under stated conditions for a specified period of time. It is often reported as a probability.

Overview



A Reliability Block Diagram

Reliability may be defined in several ways:

- The idea that something is fit for a purpose with respect to time;
- The capacity of a device or system to perform as designed;
- The resistance to failure of a device or system;
- The ability of a device or system to perform a required function under stated conditions for a specified period of time;
- The probability that a functional unit will perform its required function for a specified interval under stated conditions.
- The ability of something to "fail well" (fail without catastrophic consequences)

Reliability engineers rely heavily on statistics, probability theory, and reliability theory. Many engineering techniques are used in reliability engineering, such as reliability prediction, Weibull analysis, thermal management, reliability testing and accelerated life testing. Because of the large number of reliability techniques, their expense, and the varying degrees of reliability required for different situations, most projects develop a reliability program plan to specify the reliability tasks that will be performed for that specific system.

The function of reliability engineering is to develop the reliability requirements for the product, establish an adequate reliability program, and perform appropriate analyses and tasks to ensure the product will meet its requirements. These tasks are managed by a reliability engineer, who usually holds an accredited engineering degree and has additional reliability-specific education and training. Reliability engineering is closely associated with maintainability engineering and logistics engineering. Many problems from other fields, such as security engineering, can also be approached using reliability engineering techniques. Here we, provides an overview of some of the most common reliability engineering tasks.

Many types of engineering employ reliability engineers and use the tools and methodology of reliability engineering. For example:

- System engineers design complex systems having a specified reliability
- Mechanical engineers may have to design a machine or system with a specified reliability
- Automotive engineers have reliability requirements for the automobiles (and components) which they design
- Electronics engineers must design and test their products for reliability requirements.
- In software engineering and systems engineering the **reliability engineering** is the subdiscipline of ensuring that a system (or a device in general) will perform its intended function(s) when operated in a specified manner for a specified length of time. Reliability engineering is performed throughout the entire life cycle of a system, including development, test, production and operation.

Reliability theory

Reliability theory is the foundation of reliability engineering. For engineering purposes, reliability is defined as:

> **the probability that a device will perform its intended function during a specified period of time under stated conditions.**

Mathematically, this may be expressed as,

$$R(t) = Pr\{T > t\} = \int_t^{\infty} f(x)\, dx,$$

where $f(x)$ is the failure probability density function and t is the length of the period of time (which is assumed to start from time zero).

Reliability engineering is concerned with four key elements of this definition:

- First, reliability is a probability. This means that failure is regarded as a random phenomenon: it is a recurring event, and we do not express any information on individual failures, the causes of failures, or relationships between failures, except that the likelihood for failures to occur varies over time according to the given probability function. Reliability engineering is concerned with meeting the specified probability of success, at a specified statistical confidence level.
- Second, reliability is predicated on "intended function:" Generally, this is taken to mean operation without failure. However, even if no individual part of the system fails, but the system as a whole does not do what was intended, then it is still charged against the system reliability. The system requirements specification is the criterion against which reliability is measured.
- Third, reliability applies to a specified period of time. In practical terms, this means that a system has a specified chance that it will operate without failure before time t. Reliability engineering ensures that components and materials will meet the requirements during the specified time. Units other than time may sometimes be used. The automotive industry might specify reliability in terms of miles, the military might specify reliability of a gun for a certain number of rounds fired. A piece of mechanical equipment may have a reliability rating value in terms of cycles of use.
- Fourth, reliability is restricted to operation under stated conditions. This constraint is necessary because it is impossible to design a system for unlimited conditions. A Mars Rover will have different specified conditions than the family car. The operating environment must be addressed during design and testing. Also, that same rover, may be required to operate in varying conditions requiring additional scrutiny.

Reliability program plan

Many tasks, methods, and tools can be used to achieve reliability. Every system requires a different level of reliability. A commercial airliner must operate under a wide range of conditions. The consequences of failure are grave, but there is a correspondingly higher budget. A pencil sharpener may be more reliable than an airliner, but has a much different set of operational conditions, insignificant consequences of failure, and a much lower budget.

A reliability program plan is used to document exactly what tasks, methods, tools, analyses, and tests are required for a particular system. For complex systems, the reliability program plan is a separate document. For simple systems, it may be combined with the systems engineering management plan or integrated Logistics Support management plan. The reliability program plan is essential for a successful reliability program and is developed early during system development. It specifies not only what the reliability engineer does, but also the tasks performed by others. The reliability program plan is approved by top program management.

Reliability requirements

For any system, one of the first tasks of reliability engineering is to adequately specify the reliability requirements. Reliability requirements address the system itself, test and assessment requirements, and associated tasks and documentation. Reliability requirements are included in the appropriate system/subsystem requirements specifications, test plans, and contract statements.

System reliability parameters

Requirements are specified using reliability parameters. The most common reliability parameter is the mean time between failures (MTBF), which can also be specified as the failure rate or the number of failures during a given period. These parameters are very useful for systems that are operated frequently, such as most vehicles, machinery, and electronic equipment. Reliability increases as the MTBF increases. The MTBF is usually specified in hours, but can also be used with other units of measurement such as miles or cycles.

In other cases, reliability is specified as the probability of mission success. For example, reliability of a scheduled aircraft flight can be specified as a dimensionless probability or a percentage. refer to system safety engineering.

A special case of mission success is the single-shot device or system. These are devices or systems that remain relatively dormant and only operate once. Examples include automobile airbags, thermal batteries and missiles. Single-shot reliability is specified as a probability of success, or is subsumed into a related parameter. Single-shot missile reliability may be incorporated into a requirement for the probability of hit.

For such systems, the probability of failure on demand (PFD) is the reliability measure. This PFD is derived from failure rate and mission time for non-repairable systems. For repairable systems, it is obtained from failure rate and mean-time-to-repair (MTTR) and test interval. This measure may not be unique for a given system as this measure depends on the kind of demand. In addition to system level requirements, reliability requirements may be specified for critical subsystems. In all cases, reliability parameters are specified with appropriate statistical confidence intervals.

Reliability modelling

Reliability modelling is the process of predicting or understanding the reliability of a component or system. Two separate fields of investigation are common: The physics of failure approach uses an understanding of the failure mechanisms involved, such as crack propagation or chemical corrosion; The parts stress modelling approach is an empirical method for prediction based on counting the number and type of components of the system, and the stress they undergo during operation.

For systems with a clearly defined failure time (which is sometimes not given for systems with a drifting parameter), the empirical distribution function of these failure times can be determined. This is done in general in an accelerated experiment with increased stress. These experiments can be divided into two main categories:

Early failure rate studies determine the distribution with a decreasing failure rate over the first part of the bathtub curve. Here in general only moderate stress is necessary. The stress is applied for a limited period of time in what is called a censored test. Therefore, only the part of the distribution with early failures can be determined.

In so-called zero defect experiments, only limited information about the failure distribution is acquired. Here the stress, stress time, or the sample size is so low that not a single failure occurs. Due to the insufficient sample size, only an upper limit of the early failure rate can be determined. At any rate, it looks good for the customer if there are no failures.

In a study of the intrinsic failure distribution, which is often a material property, higher stresses are necessary to get failure in a reasonable period of time. Several degrees of stress have to be applied to determine an acceleration model. The empirical failure distribution is often parametrised with a Weibull or a log-normal model.

It is a general praxis to model the early failure rate with an exponential distribution. This less complex model for the failure distribution has only one parameter: the constant failure rate. In such cases, the Chi-square distribution can be used to find the goodness of fit for the estimated failure rate. Compared to a model with a decreasing failure rate, this is quite pessimistic. Combined with a zero-defect experiment this becomes even more pessimistic. The effort is greatly reduced in this case: one does not have to determine a second model parameter (e.g., the shape parameter of a Weibull distribution) or its confidence interval (e.g., by an MLE / Maximum likelihood approach) - and the sample size is much smaller.

Reliability test requirements

Because reliability is a probability, even highly reliable systems have some chance of failure. However, testing reliability requirements is problematic for several reasons. A single test is insufficient to generate enough statistical data. Multiple tests or long-duration tests are usually very expensive. Some tests are simply impractical. Reliability engineering is used to design a realistic and affordable test program that provides enough evidence that the system meets its requirement. Statistical confidence levels are used to address some of these concerns. A certain parameter is expressed along with a corresponding confidence level: for example, an MTBF of 1000 hours at 90% confidence level. From this specification, the reliability engineer can design a test with explicit criteria for the number of hours and number of failures until the requirement is met or failed.

The combination of reliability parameter value and confidence level greatly affects the development cost and the risk to both the customer and producer. Care is needed to select the best combination of requirements. Reliability testing may be performed at various levels, such as component, subsystem, and system. Also, many factors must be addressed during testing, such as extreme temperature and humidity, shock, vibration, and heat. Reliability engineering determines an effective test strategy so that all parts are exercised in relevant environments. For systems that must last many years, reliability engineering may be used to design an accelerated life test as well.

Reliability prediction

A prediction of reliability is an important element in the process of selecting equipment for use by telecommunications service providers and other buyers of electronic equipment. Reliability is a measure of the frequency of equipment failures as a function of time. Reliability has a major impact on maintenance and repair costs and on the continuity of service. Reliability predictions:

- Help assess the effect of product reliability on the maintenance activity and on the quantity of spare units required for acceptable field performance of any particular system. For example, predictions of the frequency of unit level maintenance actions can be obtained. Reliability prediction can be used to size spare populations.
- Provide necessary input to system-level reliability models. System-level reliability models can subsequently be used to predict, for example, frequency of system outages in steady-state, frequency of system outages during early life, expected downtime per year, and system availability.
- Provide necessary input to unit and system-level Life Cycle Cost Analyses. Life cycle cost studies determine the cost of a product over its entire life. Therefore, how often a unit will have to be replaced needs to be known. Inputs to this process include unit and system failure rates. This includes how often units and systems fail during the first year of operation as well as in later years.
- Assist in deciding which product to purchase from a list of competing products. As a result, it is essential that reliability predictions be based on a common procedure.
- Can be used to set factory test standards for products requiring a reliability test. Reliability predictions help determine how often the system should fail.
- Are needed as input to the analysis of complex systems such as switching systems and digital cross-connect systems. It is necessary to know how often different parts of the system are going to fail even for redundant components.
- Can be used in design trade-off studies. For example, a supplier could look at a design with many simple devices and compare it to a design with fewer devices that are newer but more complex. The unit with fewer devices is usually more reliable.

- Can be used to set achievable in-service performance standards against which to judge actual performance and stimulate action.

The telecommunications industry has devoted much time over the years to concentrate on developing reliability models for electronic equipment. One such tool is the Automated Reliability Prediction Procedure (ARPP), which is an Excel-spreadsheet software tool that automates the reliability prediction procedures in SR-332, Reliability Prediction Procedure for Electronic Equipment. FD-ARPP-01 provides suppliers and manufacturers with a tool for making Reliability Prediction Procedure (RPP) calculations. It also provides a means for understanding RPP calculations through the capability of interactive examples provided by the user.

The RPP views electronic systems as hierarchical assemblies. Systems are constructed from units that, in turn, are constructed from devices. The methods presented predict reliability at these three hierarchical levels:

1. *Device*: A basic component (or part)
2. *Unit*: Any assembly of devices. This may include, but is not limited to, circuit packs, modules, plug-in units, racks, power supplies, and ancillary equipment. Unless otherwise dictated by maintenance considerations, a unit will usually be the lowest level of replaceable assemblies/devices. The RPP is aimed primarily at reliability prediction of units.
3. *Serial System*: Any assembly of units for which the failure of any single unit will cause a failure of the system.

Requirements for reliability tasks

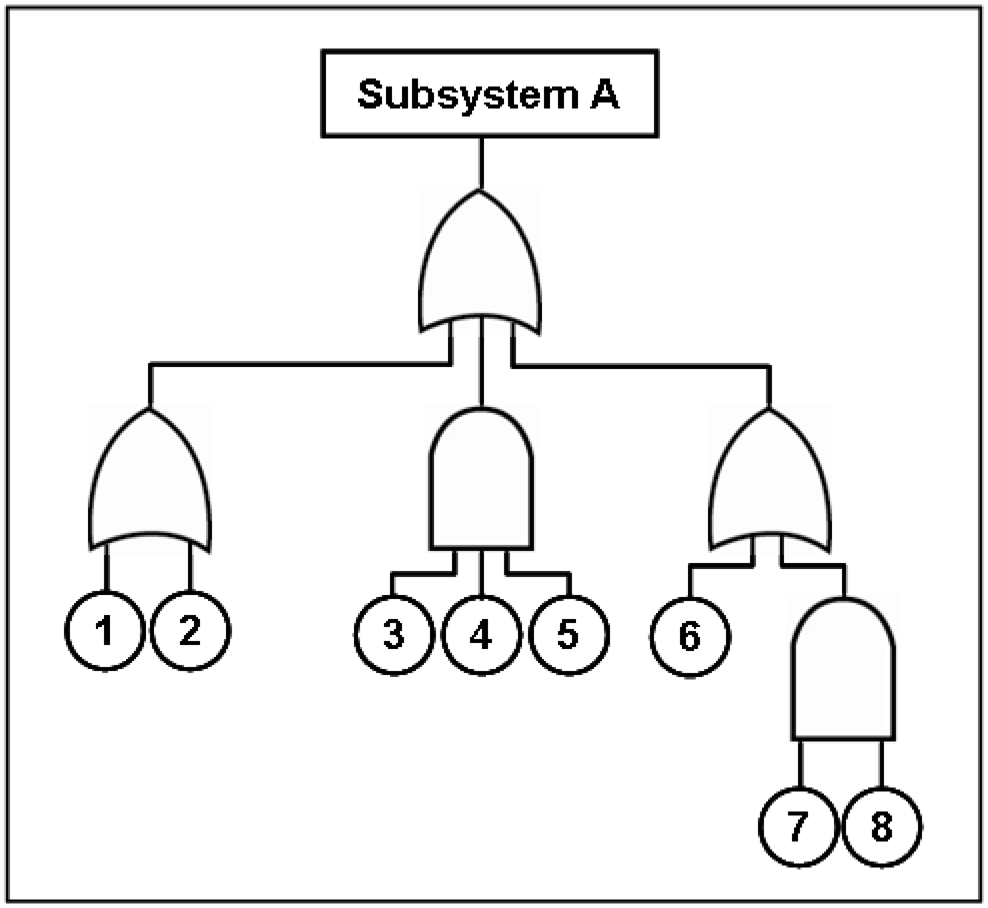
Reliability engineering must also address requirements for various reliability tasks and documentation during system development, test, production, and operation. These requirements are generally specified in the contract statement of work and depend on how much leeway the customer wishes to provide to the contractor. Reliability tasks include various analyses, planning, and failure reporting. Task selection depends on the criticality of the system as well as cost. A critical system may require a formal failure reporting and review process throughout development, whereas a non-critical system may rely on final test reports. The most common reliability program tasks are documented in reliability program standards, such as MIL-STD-785 and IEEE 1332. Failure reporting analysis and corrective action systems are a common approach for product/process reliability monitoring.

Design for reliability

Design For Reliability (DFR), is an emerging discipline that refers to the process of designing reliability into products. This process encompasses several tools and practices and describes the order of their deployment that an organization needs to have in place to drive reliability into their products. Typically, the first step in the DFR process is to set the system's reliability requirements. Reliability must be "designed in" to the system.

During system design, the top-level reliability requirements are then allocated to subsystems by design engineers and reliability engineers working together.

Reliability design begins with the development of a model. Reliability models use **block diagrams** and **fault trees** to provide a graphical means of evaluating the relationships between different parts of the system. These models incorporate predictions based on parts-count failure rates taken from historical data. While the predictions are often not accurate in an absolute sense, they are valuable to assess relative differences in design alternatives.



A Fault Tree Diagram

One of the most important design techniques is **redundancy**. This means that if one part of the system fails, there is an alternate success path, such as a backup system. An automobile brake light might use two light bulbs. If one bulb fails, the brake light still operates using the other bulb. Redundancy significantly increases system reliability, and is often the only viable means of doing so. However, redundancy is difficult and expensive, and is therefore limited to critical parts of the system. Another design technique, **physics of failure**, relies on understanding the physical processes of stress, strength and failure at a very detailed level. Then the material or component can be re-designed to reduce the probability of failure. Another common design technique is

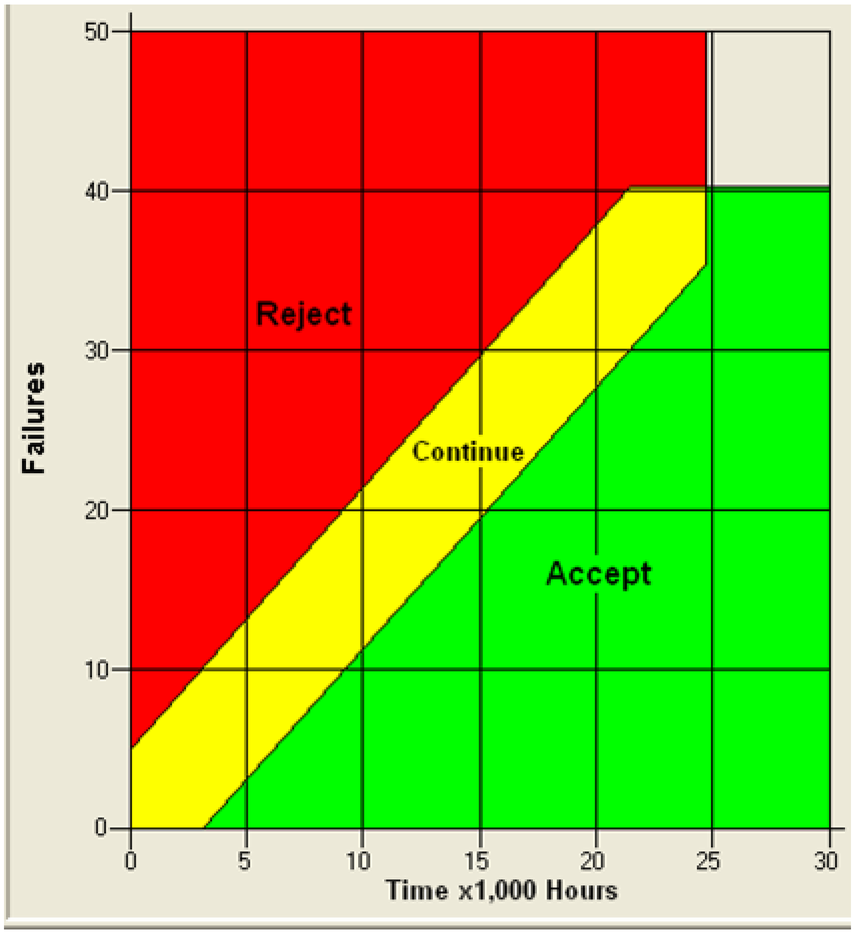
component derating: Selecting components whose tolerance significantly exceeds the expected stress, as using a heavier gauge wire that exceeds the normal specification for the expected electrical current.

Many tasks, techniques and analyses are specific to particular industries and applications. Commonly these include:

- Built-in test (BIT)
- Failure mode and effects analysis (FMEA)
- Reliability simulation modeling
- Thermal analysis
- Reliability Block Diagram analysis
- Fault tree analysis
- Root cause analysis
- Sneak circuit analysis
- Accelerated Testing
- Reliability Growth analysis
- Weibull analysis
- Electromagnetic analysis
- Statistical interference
- Avoid Single Point of Failure

Results are presented during the system design reviews and logistics reviews. Reliability is just one requirement among many system requirements. Engineering trade studies are used to determine the optimum balance between reliability and other requirements and constraints.

Reliability testing



A Reliability Sequential Test Plan

The purpose of reliability testing is to discover potential problems with the design as early as possible and, ultimately, provide confidence that the system meets its reliability requirements.

Reliability testing may be performed at several levels. Complex systems may be tested at component, circuit board, unit, assembly, subsystem and system levels. (The test level nomenclature varies among applications.) For example, performing environmental stress screening tests at lower levels, such as piece parts or small assemblies, catches problems before they cause failures at higher levels. Testing proceeds during each level of integration through full-up system testing, developmental testing, and operational testing, thereby reducing program risk. System reliability is calculated at each test level. Reliability growth techniques and failure reporting, analysis and corrective active systems (FRACAS) are often employed to improve reliability as testing progresses. The drawbacks to such extensive testing are time and expense. Customers may choose to accept more risk by eliminating some or all lower levels of testing.

It is not always feasible to test all system requirements. Some systems are prohibitively expensive to test; some failure modes may take years to observe; some complex interactions result in a huge number of possible test cases; and some tests require the use of limited test ranges or other resources. In such cases, different approaches to testing can be used, such as accelerated life testing, design of experiments, and simulations.

The desired level of statistical confidence also plays an important role in reliability testing. Statistical confidence is increased by increasing either the test time or the number of items tested. Reliability test plans are designed to achieve the specified reliability at the specified confidence level with the minimum number of test units and test time. Different test plans result in different levels of risk to the producer and consumer. The desired reliability, statistical confidence, and risk levels for each side influence the ultimate test plan. Good test requirements ensure that the customer and developer agree in advance on how reliability requirements will be tested.

A key aspect of reliability testing is to define "failure". Although this may seem obvious, there are many situations where it is not clear whether a failure is really the fault of the system. Variations in test conditions, operator differences, weather, and unexpected situations create differences between the customer and the system developer. One strategy to address this issue is to use a **scoring conference** process. A scoring conference includes representatives from the customer, the developer, the test organization, the reliability organization, and sometimes independent observers. The scoring conference process is defined in the statement of work. Each test case is considered by the group and "scored" as a success or failure. This scoring is the official result used by the reliability engineer.

As part of the requirements phase, the reliability engineer develops a test strategy with the customer. The test strategy makes trade-offs between the needs of the reliability organization, which wants as much data as possible, and constraints such as cost, schedule, and available resources. Test plans and procedures are developed for each reliability test, and results are documented in official reports.

Accelerated testing

The purpose of accelerated life testing is to induce field failure in the laboratory at a much faster rate by providing a harsher, but nonetheless representative, environment. In such a test the product is expected to fail in the lab just as it would have failed in the field—but in much less time. The main objective of an accelerated test is either of the following:

- To discover failure modes
- To predict the normal field life from the high stress lab life

An **Accelerated testing** program can be broken down into the following steps:

- Define objective and scope of the test

- Collect required information about the product
- Identify the stress(es)
- Determine level of stress(es)
- Conduct the Accelerated test and analyse the accelerated data.

Common way to determine a life stress relationship are

- Arrhenius Model
- Eyring Model
- Inverse Power Law Model
- Temperature-Humidity Model
- Temperature Non-thermal Model

Software reliability

Software reliability is a special aspect of reliability engineering. System reliability, by definition, includes all parts of the system, including hardware, software, operators and procedures. Traditionally, reliability engineering focuses on critical hardware parts of the system. Since the widespread use of digital integrated circuit technology, software has become an increasingly critical part of most electronics and, hence, nearly all present day systems. There are significant differences, however, in how software and hardware behave. Most hardware unreliability is the result of a component or material failure that results in the system not performing its intended function. Repairing or replacing the hardware component restores the system to its original unfailed state. However, software does not fail in the same sense that hardware fails. Instead, software unreliability is the result of unanticipated results of software operations. Even relatively small software programs can have astronomically large combinations of inputs and states that are infeasible to exhaustively test. Restoring software to its original state only works until the same combination of inputs and states results in the same unintended result. Software reliability engineering must take this into account.

Despite this difference in the source of failure between software and hardware — software doesn’t wear out — some in the software reliability engineering community believe statistical models used in hardware reliability are nevertheless useful as a measure of software reliability, describing what we experience with software: the longer you run software, the higher the probability you’ll eventually use it in an untested manner and find a latent defect that results in a failure (Shooman 1987), (Musa 2005), (Denney 2005).

As with hardware, software reliability depends on good requirements, design and implementation. Software reliability engineering relies heavily on a disciplined software engineering process to anticipate and design against unintended consequences. There is more overlap between software quality engineering and software reliability engineering than between hardware quality and reliability. A good software development plan is a key aspect of the software reliability program. The software development plan describes the design and coding standards, peer reviews, unit tests, configuration management, software metrics and software models to be used during software development.

A common reliability metric is the number of software faults, usually expressed as faults per thousand lines of code. This metric, along with software execution time, is key to most software reliability models and estimates. The theory is that the software reliability increases as the number of faults (or fault density) goes down. Establishing a direct connection between fault density and mean-time-between-failure is difficult, however, because of the way software faults are distributed in the code, their severity, and the probability of the combination of inputs necessary to encounter the fault. Nevertheless, fault density serves as a useful indicator for the reliability engineer. Other software metrics, such as complexity, are also used.

Testing is even more important for software than hardware. Even the best software development process results in some software faults that are nearly undetectable until tested. As with hardware, software is tested at several levels, starting with individual units, through integration and full-up system testing. Unlike hardware, it is inadvisable to skip levels of software testing. During all phases of testing, software faults are discovered, corrected, and re-tested. Reliability estimates are updated based on the fault density and other metrics. At system level, mean-time-between-failure data are collected and used to estimate reliability. Unlike hardware, performing exactly the same test on exactly the same software configuration does not provide increased statistical confidence. Instead, software reliability uses different metrics such as test coverage.

Eventually, the software is integrated with the hardware in the top-level system, and software reliability is subsumed by system reliability. The Software Engineering Institute's Capability Maturity Model is a common means of assessing the overall software development process for reliability and quality purposes. However, actual software reliability is served through SAE standards JA1002 and JA1003.

Reliability Operational Assessment

After a system is produced, reliability engineering monitors, assesses, and corrects deficiencies. Monitoring includes electronic and visual surveillance of critical parameters identified during the fault tree analysis design stage. The data are constantly analyzed using statistical techniques, such as Weibull analysis and linear regression, to ensure the system reliability meets requirements. Reliability data and estimates are also key inputs for system logistics. Data collection is highly dependent on the nature of the system. Most large organizations have quality control groups that collect failure data on vehicles, equipment, and machinery. Consumer product failures are often tracked by the number of returns. For systems in dormant storage or on standby, it is necessary to establish a formal surveillance program to inspect and test random samples. Any changes to the system, such as field upgrades or recall repairs, require additional reliability testing to ensure the reliability of the modification. Since it is not possible to anticipate all the failure modes of a given system, especially ones with a human element, failures will occur. The reliability program also includes a systematic root cause analysis that identifies the causal relationships involved in the failure such that effective corrective actions may be implemented. When possible, system failures and corrective actions are reported to the reliability engineering organization.

One of the most common methods to apply a Reliability Operational Assessment are Failure Reporting, Analysis and Corrective Action Systems (FRACAS). This systematic approach develops a reliability, safety and logistics assessment based on Failure / Incident reporting, management, analysis and corrective/preventive actions. Organizations today are adopting this method and utilize commercial systems such as a Web based FRACAS application enabling and organization to create a failure/incident data repository from which statistics can be derived to view accurate and genuine reliability, safety and quality performances.

Some of the common outputs from a FRACAS system includes: Field MTBF, MTTR, Spares Consumption, Reliability Growth, Failure/Incidents distribution by type, location, part no., serial no, symptom etc.

Reliability organizations

Systems of any significant complexity are developed by organizations of people, such as a commercial company or a government agency. The reliability engineering organization must be consistent with the company's organizational structure. For small, non-critical systems, reliability engineering may be informal. As complexity grows, the need arises for a formal reliability function. Because reliability is important to the customer, the customer may even specify certain aspects of the reliability organization.

There are several common types of reliability organizations. The project manager or chief engineer may employ one or more reliability engineers directly. In larger organizations, there is usually a product assurance or specialty engineering organization, which may include reliability, maintainability, quality, safety, human factors, logistics, etc. In such case, the reliability engineer reports to the product assurance manager or specialty engineering manager.

In some cases, a company may wish to establish an independent reliability organization. This is desirable to ensure that the system reliability, which is often expensive and time consuming, is not unduly slighted due to budget and schedule pressures. In such cases, the reliability engineer works for the project day-to-day, but is actually employed and paid by a separate organization within the company.

Because reliability engineering is critical to early system design, it has become common for reliability engineers, however the organization is structured, to work as part of an integrated product team.

Certification

The American Society for Quality has a program to become a Certified Reliability Engineer, CRE. Certification is based on education, experience, and a certification test: periodic recertification is required. The body of knowledge for the test includes: reliability management, design evaluation, product safety, statistical tools, design and development, modeling, reliability testing, collecting and using data, etc.

Another highly respected certification program is the CRP (Certified Reliability Professional). To achieve certification, candidates must complete a series of courses focused on important Reliability Engineering topics, successfully apply the learned body of knowledge in the workplace and publicly present this expertise in an industry conference or journal.

Reliability engineering education

Some Universities offer graduate degrees in Reliability Engineering (e.g, University of Tennessee, Knoxville, University of Maryland, College Park, Concordia University, Montreal, Canada, Monash University, Australia and Tampere University of Technology, Tampere, Finland). Other reliability engineers typically have an engineering degree, which can be in any field of engineering, from an accredited university or college program. Many engineering programs offer reliability courses, and some universities have entire reliability engineering programs. A reliability engineer may be registered as a Professional Engineer by the state, but this is not required by most employers. There are many professional conferences and industry training programs available for reliability engineers. Several professional organizations exist for reliability engineers, including the IEEE Reliability Society, the American Society for Quality (ASQ), and the Society of Reliability Engineers (SRE).

Chapter- 5

Tools of Systems Engineering

System model

A **system model** is the conceptual model that describes and represents a system. A system comprises multiple views such as planning, requirement, design, implementation, deployment, operational, structure, behavior, input data, and output data views. A system model is required to describe and represent all these multiple views.

The system model describes and represents the multiple views possibly adopting two different approaches. The first one is the non-architectural approach and the second one is the architectural approach.

The non-architectural approach respectively picks a model for each view. For example, Structured Systems Analysis and Design Method (SSADM), picking the Structure Chart (SC) for structure description and the Data Flow Diagram (DFD) for behavior description, is categorized into the non-architectural approach.

The architectural approach, instead of picking many heterogeneous and unrelated models, will use only one single coalescence model. For example, System architecture, using the Architecture Description Language (ADL) for both structure and behavior descriptions, is categorized into the architectural approach.

System model may refer to:

- Community Climate System Model, a coupled Global Climate Model
- Human visual system model, a human visual system model used by image processing, video processing and computer vision
- Internal Family Systems Model, an integrative approach to psychotherapy, relationship counseling and family therapy
- Solar system model, a model that illustrate the relative positions and motions of the planets and stars
- Viable System Model, a model of the organisational structure of any viable or autonomous system

Systems architecture

A **system architecture** or **systems architecture** is the conceptual model that defines the structure, behavior, and more views of a system.

An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structure of the system which comprises system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them, and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system. The language for architecture description is called the architecture description language (ADL).

Overview

There is no universally agreed definition of which aspects constitute a system architecture, and various organizations define it in different ways, including:

- The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.
- The composite of the design architectures for products and their life cycle processes.
- A representation of a system in which there is a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components.
- An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline.
- An architecture is the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.
- A description of the design and contents of a computer system. If documented, it may include information such as a detailed inventory of current hardware, software and networking capabilities; a description of long-range plans and priorities for future purchases, and a plan for upgrading and/or replacing dated equipment and software.
- A formal description of a system, or a detailed plan of the system at component level to guide its implementation.
- The structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time.

Systems architecture can best be thought of as a representation of an existent (or To Be Created) system, and the process and discipline for effectively implementing the design(s) for such a system. The set of relations (that is, embedded information) which an architecture describes may be expressed in hardware, software, or something else (for example, organizational management describes many architectures whose nodes are people, knowledge management systems use nodes of metadescription).

- It is a *representation* because it is used to convey the informational content of the elements comprising a system, the relationships among those elements, and the rules governing those relationships.
- It is a *process* because a sequence of steps is prescribed to produce or change the architecture, and/or a design from that architecture, of a system within a set of constraints.
- It is a *discipline* because a body of knowledge is used to inform practitioners as to the most effective way to design the system within a set of constraints.

A systems architecture is primarily concerned with the internal interfaces among the system's components or subsystems, and the interface between the system and its external environment, especially the user. (In the specific case of computer systems, this latter, special interface, is known as the computer human interface, *AKA* human computer interface, or CHI; formerly called the man-machine interface.)

History

It is important to keep in mind that the modern systems architecture did not appear out of nowhere. Systems architecture depends heavily on practices and techniques which were developed over thousands of years in many other fields most importantly being, perhaps, civil architecture.

Prior to the advent of digital computers, the electronics and other engineering disciplines used the term system as it is still commonly used today. However, with the arrival of digital computers and the development of software engineering as a separate discipline, it was often necessary to distinguish among engineered hardware artifacts, software artifacts, and the combined artifacts. A programmable hardware artifact, or computing machine, that lacks its software program is impotent; even as a software artifact, or program, is equally impotent unless it can be used to alter the sequential states of a suitable (hardware) machine. However, a hardware machine and its software program can be designed to perform an almost illimitable number of abstract and physical tasks. Within the computer and software engineering disciplines (and, often, other engineering disciplines, such as communications), then, the term system came to be defined as containing all of the elements necessary (which generally includes both hardware and software) to perform a useful function.

Consequently, within these engineering disciplines, a system generally refers to a programmable hardware machine and its included program. And a systems engineer is defined as one concerned with the complete device, both hardware and software and, more particularly, all of the interfaces of the device, including that between hardware and software, and especially between the complete device and its user (the CHI). The hardware engineer deals (more or less) exclusively with the hardware device; the software engineer deals (more or less) exclusively with the software program; and the systems engineer is responsible for seeing that the software program is capable of properly running within the hardware device, and that the system composed of the two entities is capable of properly interacting with its external environment, especially the user, and performing its intended function.

By analogy, then, a systems architecture makes use of elements of both software and hardware and is used to enable design of such a composite system. A good architecture may be viewed as a 'partitioning scheme,' or algorithm, which partitions all of the system's present and foreseeable requirements into a workable set of cleanly bounded subsystems with nothing left over. That is, it is a partitioning scheme which is exclusive, inclusive, and exhaustive. A major purpose of the partitioning is to arrange the elements in the sub systems so that there is a minimum of communications needed among them. In both software and hardware, a good sub system tends to be seen to be a meaningful "object". Moreover, a good architecture provides for an easy mapping to the user's requirements and the validation tests of the user's requirements. Ideally, a mapping also exists from every least element to every requirement and test.

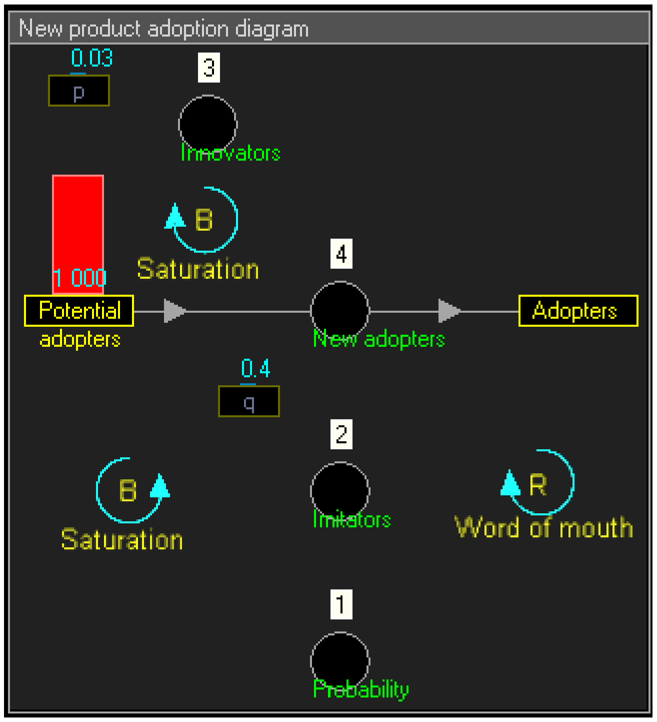
A *robust architecture* is said to be one that exhibits an optimal degree of fault-tolerance, backward compatibility, forward compatibility, extensibility, reliability, maintainability, availability, serviceability, usability, and such other quality attributes as necessary and/or desirable.

Types of systems architectures

Several types of systems architectures (underlain by the same fundamental principles) have been identified as follows:

- Collaborative Systems (such as the Internet, intelligent transportation systems, and joint air defense systems)
- Manufacturing Systems
- Social Systems
- Software and Information Technology Systems
- Strategic Systems Architecture

System dynamics



Dynamic stock and flow diagram of model *New product adoption* (model from article by John Sterman 2001)

System dynamics is an approach to understanding the behaviour of complex systems over time. It deals with internal feedback loops and time delays that affect the behaviour of the entire system. What makes using system dynamics different from other approaches to studying complex systems is the use of feedback loops and stocks and flows. These elements help describe how even seemingly simple systems display baffling nonlinearity.

Overview

System dynamics is a methodology and computer simulation modeling technique for framing, understanding, and discussing complex issues and problems. Originally developed in the 1950s to help corporate managers improve their understanding of industrial processes, system dynamics is currently being used throughout the public and private sector for policy analysis and design.

System dynamics is an aspect of systems theory as a method for understanding the dynamic behavior of complex systems. The basis of the method is the recognition that the structure of any system — the many circular, interlocking, sometimes time-delayed relationships among its components — is often just as important in determining its behavior as the individual components themselves. Examples are chaos theory and social dynamics. It is also claimed that because there are often properties-of-the-whole which cannot be found among the properties-of-the-elements, in some cases the behavior of the whole cannot be explained in terms of the behavior of the parts.

History

System dynamics was created during the mid-1950s by Professor Jay Forrester of the Massachusetts Institute of Technology. In 1956, Forrester accepted a professorship in the newly-formed MIT School of Management. His initial goal was to determine how his background in science and engineering could be brought to bear, in some useful way, on the core issues that determine the success or failure of corporations. Forrester's insights into the common foundations that underlie engineering and management, which led to the creation of system dynamics, were triggered, to a large degree, by his involvement with managers at General Electric (GE) during the mid-1950s. At that time, the managers at GE were perplexed because employment at their appliance plants in Kentucky exhibited a significant three-year cycle. The business cycle was judged to be an insufficient explanation for the employment instability. From hand simulations (or calculations) of the stock-flow-feedback structure of the GE plants, which included the existing corporate decision-making structure for hiring and layoffs, Forrester was able to show how the instability in GE employment was due to the internal structure of the firm and not to an external force such as the business cycle. These hand simulations were the beginning of the field of system dynamics.

During the late 1950s and early 1960s, Forrester and a team of graduate students moved the emerging field of system dynamics from the hand-simulation stage to the formal computer modeling stage. Richard Bennett created the first system dynamics computer modeling language called SIMPLE (Simulation of Industrial Management Problems with Lots of Equations) in the spring of 1958. In 1959, Phyllis Fox and Alexander Pugh wrote the first version of DYNAMO (DYNAmic MOdels), an improved version of SIMPLE, and the system dynamics language became the industry standard for over thirty years. Forrester published the first, and still classic, book in the field titled Industrial Dynamics in 1961.

From the late 1950s to the late 1960s, system dynamics was applied almost exclusively to corporate/managerial problems. In 1968, however, an unexpected occurrence caused the field to broaden beyond corporate modeling. John Collins, the former mayor of Boston, was appointed a visiting professor of Urban Affairs at MIT. The result of the Collins-Forrester collaboration was a book titled Urban Dynamics. The Urban Dynamics model presented in the book was the first major non-corporate application of system dynamics.

The second major noncorporate application of system dynamics came shortly after the first. In 1970, Jay Forrester was invited by the Club of Rome to a meeting in Bern, Switzerland. The Club of Rome is an organization devoted to solving what its members describe as the "predicament of mankind" -- that is, the global crisis that may appear sometime in the future, due to the demands being placed on the Earth's carrying capacity (its sources of renewable and nonrenewable resources and its sinks for the disposal of pollutants) by the world's exponentially growing population. At the Bern meeting, Forrester was asked if system dynamics could be used to address the predicament of mankind. His answer, of course, was that it could. On the plane back from the Bern meeting, Forrester created the first draft of a system dynamics model of the world's socioeconomic system. He called this model WORLD1. Upon his return to the United States, Forrester refined WORLD1 in preparation for a visit to MIT by members of the Club of Rome. Forrester called the refined version of the model WORLD2. Forrester published WORLD2 in a book titled World Dynamics.

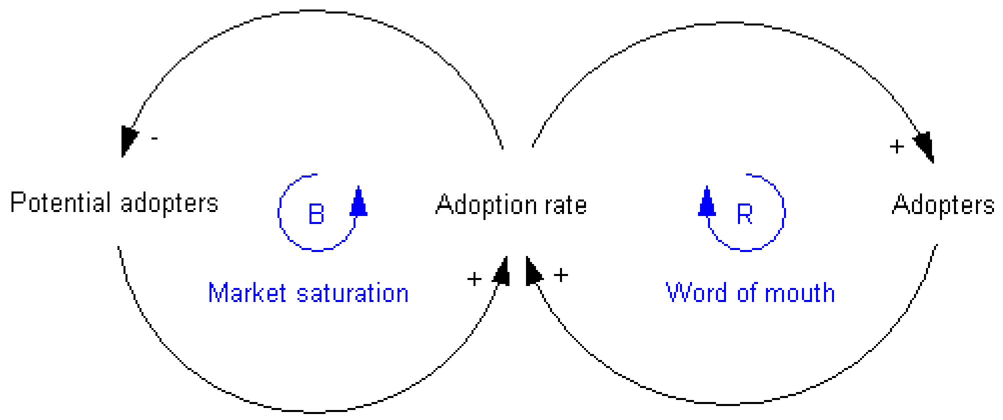
Topics in systems dynamics

The elements of system dynamics diagrams are feedback, accumulation of flows into stocks and time delays.

As an illustration of the use of system dynamics, imagine an organisation that plans to introduce an innovative new durable consumer product. The organisation needs to understand the possible market dynamics in order to design marketing and production plans.

Causal loop diagrams

A causal loop diagram is a visual representation of the feedback loops in a system. The causal loop diagram of the new product introduction may look as follows:

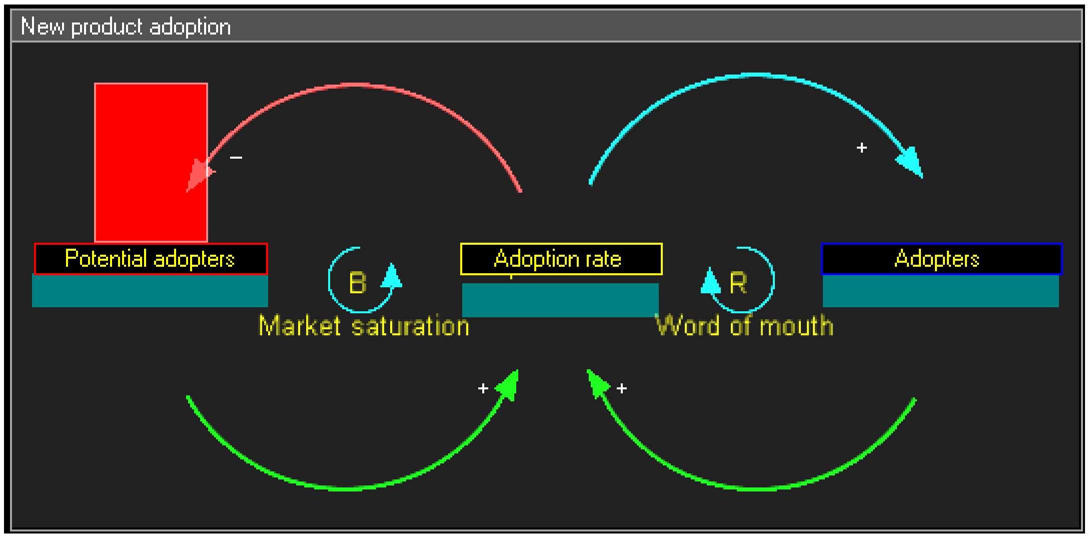


Causal loop diagram of *New product adoption* model

There are two feedback loops in this diagram. The positive reinforcement (labeled R) loop on the right indicates that the more people have already adopted the new product, the stronger the word-of-mouth impact. There will be more references to the product, more demonstrations, and more reviews. This positive feedback should generate sales that continue to grow.

The second feedback loop on the left is negative reinforcement (or "balancing" and hence labeled B). Clearly growth can not continue forever, because as more and more people adopt, there remain fewer and fewer potential adopters.

Both feedback loops act simultaneously, but at different times they may have different strengths. Thus one would expect growing sales in the initial years, and then declining sales in the later years.



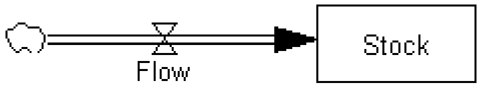
Causal loop diagram of *New product adoption* model with nodes values after calculus

In this dynamic causal loop diagram :

- step1 : (+) green arrows show that *Adoption rate* is function of *Potential Adopters* and *Adopters*
- step2 : (-) red arrow shows that *Potential adopters* decreases by *Adoption rate*
- step3 : (+) blue arrow shows that *Adopters* increases by *Adoption rate*

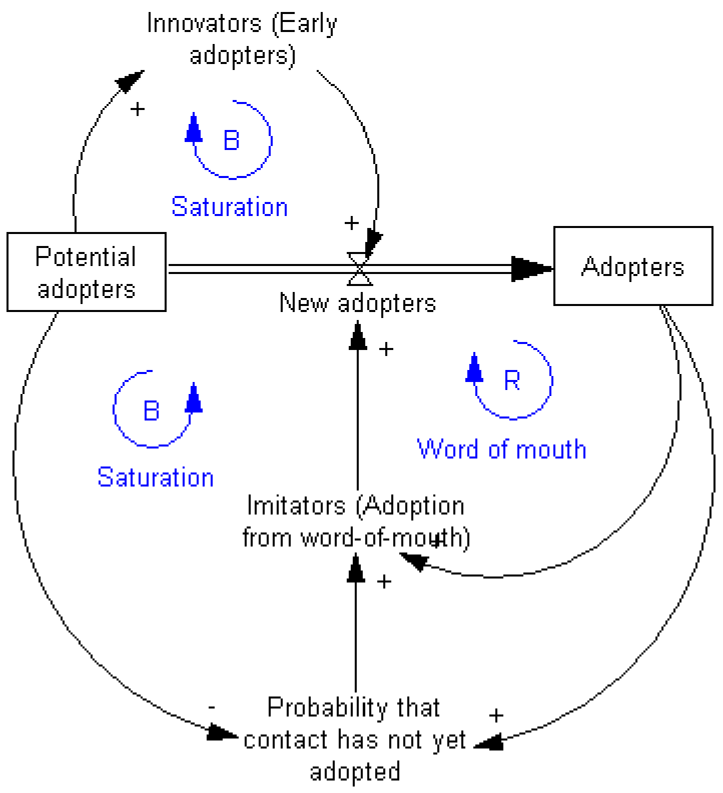
Stock and flow diagrams

The next step is to create what is termed a stock and flow diagram. A stock is the term for any entity that accumulates or depletes over time. A flow is the rate of change in a stock.



A flow is the rate of accumulation of the stock

In our example, there are two stocks: Potential adopters and Adopters. There is one flow: New adopters. For every new adopter, the stock of potential adopters declines by one, and the stock of adopters increases by one.


Innovators (Early adopters)
+
B
Saturation
+
Potential adopters
New adopters
Adopters
+
R
Word of mouth
B
Saturation
Imitators (Adoption from word-of-mouth)
+
+
-
Probability that contact has not yet adopted
+

Stock and flow diagram of *New product adoption* model

Equations

The real power of system dynamics is utilised through simulation. Although it is possible to perform the modeling in a spreadsheet, there are a variety of software packages that have been optimised for this.

The steps involved in a simulation are:

- Define the problem boundary
- Identify the most important stocks and flows that change these stock levels
- Identify sources of information that impact the flows
- Identify the main feedback loops
- Draw a causal loop diagram that links the stocks, flows and sources of information
- Write the equations that determine the flows
- Estimate the parameters and initial conditions. These can be estimated using statistical methods, expert opinion, market research data or other relevant sources of information.
- Simulate the model and analyse results

In this example, the equations that change the two stocks via the flow are:

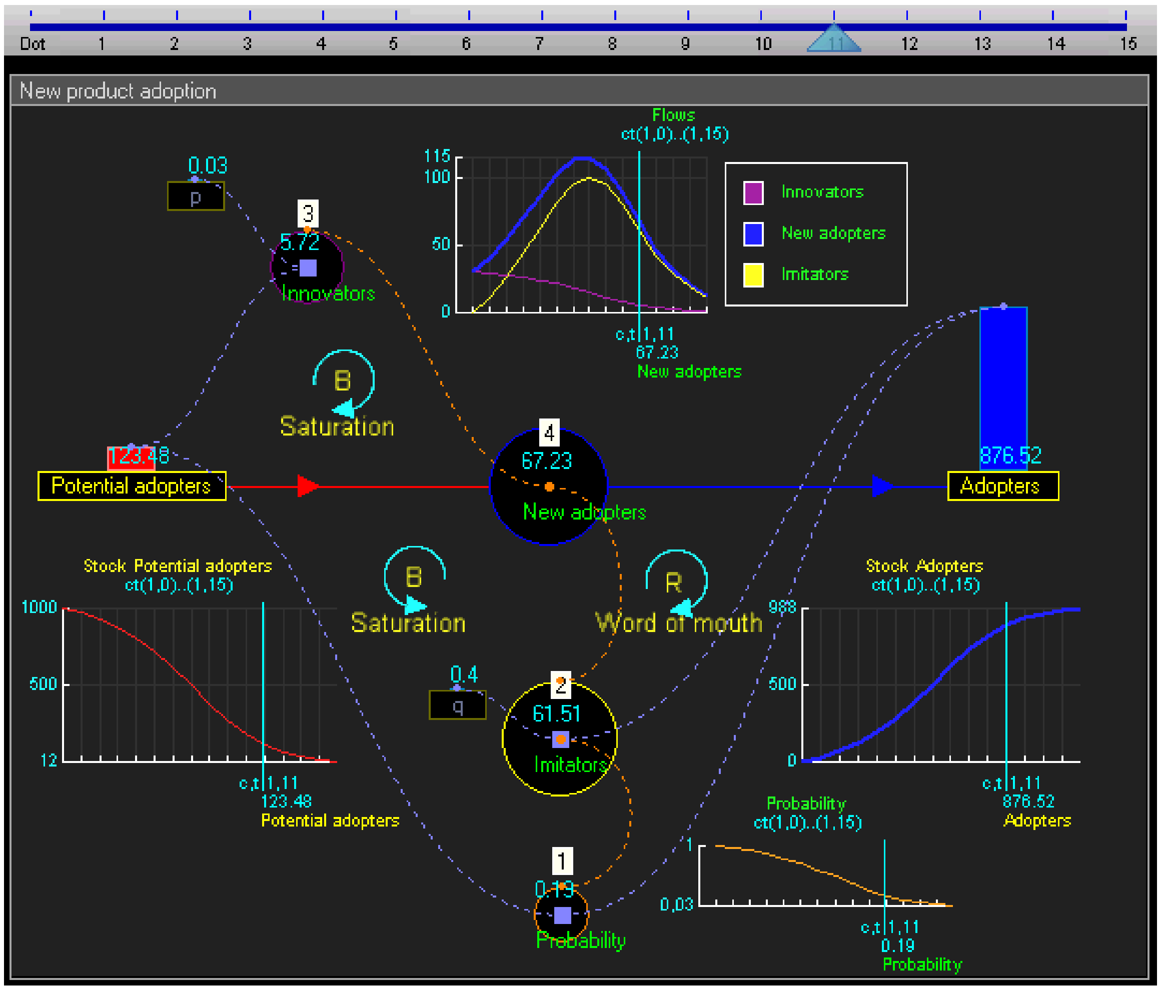
$$\text{Potential adopters} = \int_0^t \text{-New adopters } dt$$
$$\text{Adopters} = \int_0^t \text{New adopters } dt$$

List of all the equations, in their order of execution in each year, from year 1 to 15:

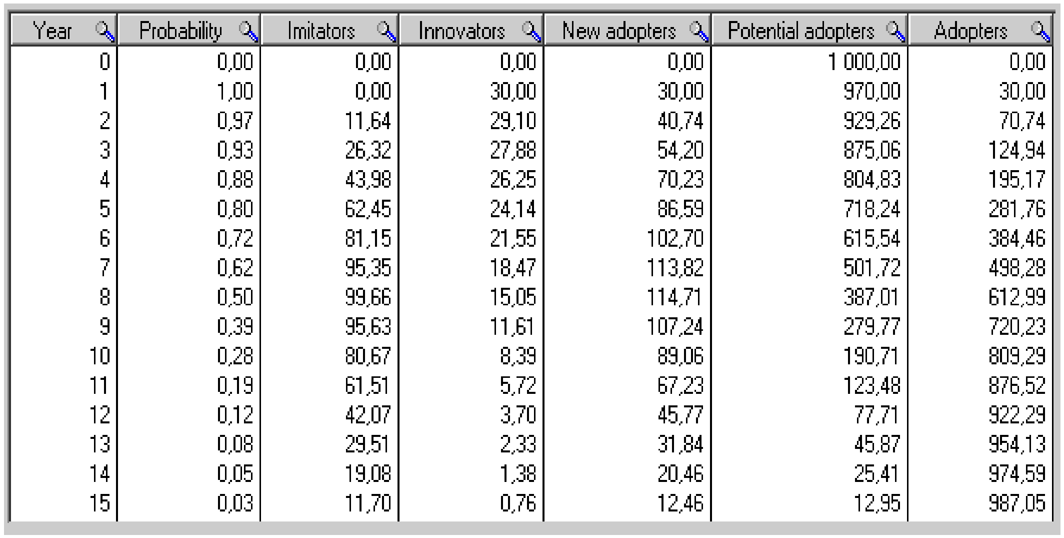
$$1)\ \text{Probability that contact has not yet adopted} = \frac{\text{Potential adopters}}{\text{Potential adopters } + \text{ Adopters}}$$
$$2)\ \text{Imitators} = q \cdot \text{Adopters} \cdot \text{Probability that contact has not yet adopted}$$
$$3)\ \text{Innovators} = p \cdot \text{Potential adopters}$$
$$4)\ \text{New adopters} = \text{Innovators} + \text{Imitators}$$
$$4.1)\ \text{Potential adopters} - = \text{New adopters}$$
$$4.2)\ \text{Adopters} + = \text{New adopters}$$
$$p = 0.03$$
$$q = 0.4$$

Dynamic simulation results

The dynamic simulation results show that the behaviour of the system would be to have growth in **Adopters** that follows a classical s-curve shape.
The increase in **Adopters** is very slow initially, then exponential growth for a period, followed ultimately by saturation.



Dynamic stock and flow diagram of *New product adoption* model



Year	Probability	Imitators	Innovators	New adopters	Potential adopters	Adopters
0	0,00	0,00	0,00	0,00	1 000,00	0,00
1	1,00	0,00	30,00	30,00	970,00	30,00
2	0,97	11,64	29,10	40,74	929,26	70,74
3	0,93	26,32	27,88	54,20	875,06	124,94
4	0,88	43,98	26,25	70,23	804,83	195,17
5	0,80	62,45	24,14	86,59	718,24	281,76
6	0,72	81,15	21,55	102,70	615,54	384,46
7	0,62	95,35	18,47	113,82	501,72	498,28
8	0,50	99,66	15,05	114,71	387,01	612,99
9	0,39	95,63	11,61	107,24	279,77	720,23
10	0,28	80,67	8,39	89,06	190,71	809,29
11	0,19	61,51	5,72	67,23	123,48	876,52
12	0,12	42,07	3,70	45,77	77,71	922,29
13	0,08	29,51	2,33	31,84	45,87	954,13
14	0,05	19,08	1,38	20,46	25,41	974,59
15	0,03	11,70	0,76	12,46	12,95	987,05

Stocks and flows values for years = 0 to 15

Application

System dynamics has found application in a wide range of areas, for example population, ecological and economic systems, which usually interact strongly with each other.

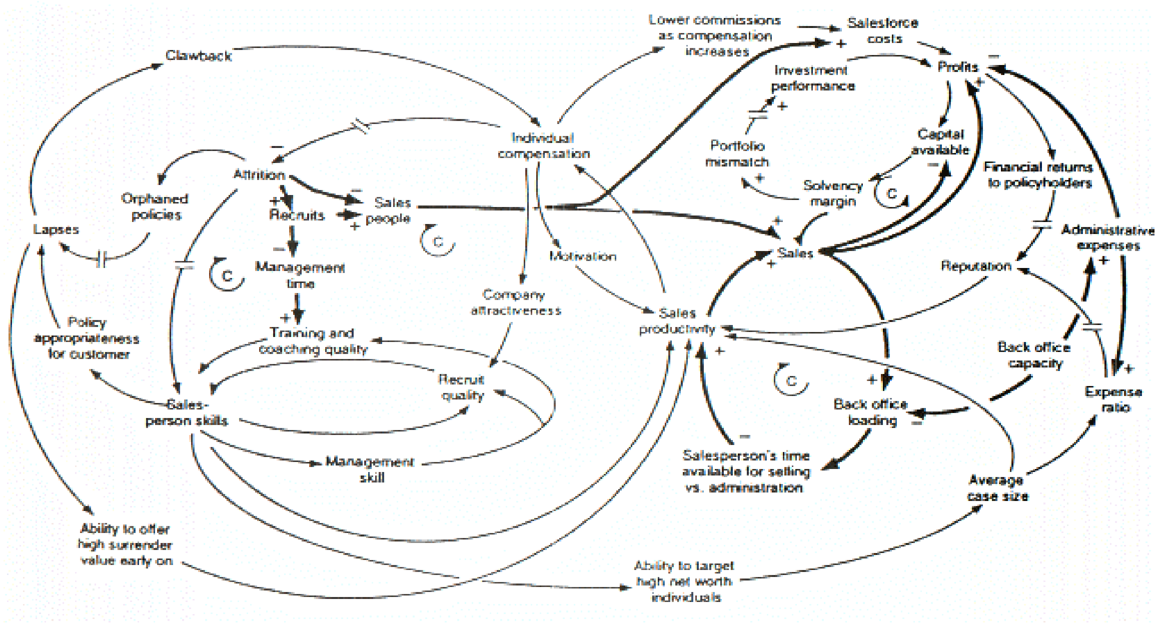
System dynamics have various "back of the envelope" management applications. They are a potent tool to:

- Teach system thinking reflexes to persons being coached
- Analyze and compare assumptions and mental models about the way things work
- Gain qualitative insight into the workings of a system or the consequences of a decision
- Recognize archetypes of dysfunctional systems in everyday practice

Computer software is used to simulate a system dynamics model of the situation being studied. Running "what if" simulations to test certain policies on such a model can greatly aid in understanding how the system changes over time. System dynamics is very similar to systems thinking and constructs the same causal loop diagrams of systems with feedback. However, system dynamics typically goes further and utilises simulation to study the behaviour of systems and the impact of alternative policies.

System dynamics has been used to investigate resource dependencies, and resulting problems, in product development.

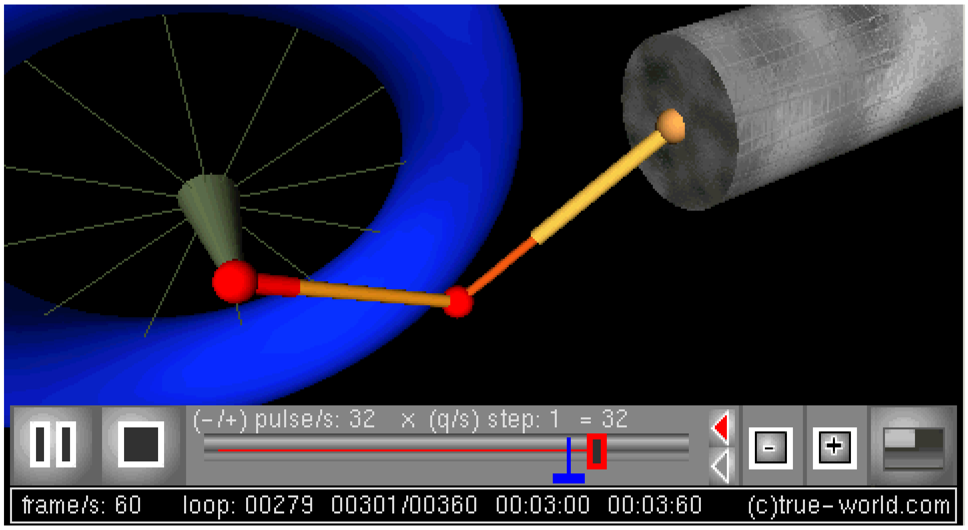
Example



Causal loop diagram of a model examining the growth or decline of a life insurance company.

The figure above is a causal loop diagram of a system dynamics model created to examine forces that may be responsible for the growth or decline of life insurance companies in the United Kingdom. A number of this figure's features are worth mentioning. The first is that the model's negative feedback loops are identified by "C's," which stand for "Counteracting" loops. The second is that double slashes are used to indicate places where there is a significant delay between causes (i.e., variables at the tails of arrows) and effects (i.e., variables at the heads of arrows). This is a common causal loop diagramming convention in system dynamics. Third, is that thicker lines are used to identify the feedback loops and links that author wishes the audience to focus on. This is also a common system dynamics diagramming convention. Last, it is clear that a decision maker would find it impossible to think through the dynamic behavior inherent in the model, from inspection of the figure alone.

Example of 4D piston motion


(-/+) pulse/s: 32 x (q/s) step: 1 = 32
frame/s: 60 loop: 00279 00301/00360 00:03:00 00:03:60 (c)true-world.com

Piston motion equations

This animation was made with the 3D modeler of a system dynamics software.
The calculated values are associated with parameters of the rod and crank.
In this example the crank is driving, we vary both the speed of rotation, its radius and the length of the rod, the piston follows.

Systems analysis

Systems analysis is the study of sets of interacting entities, including computer systems analysis. This field is closely related to operations research. It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made."

Overview

The terms analysis and synthesis come from Greek where they mean respectively "to take apart" and "to put together". These terms are in scientific disciplines from mathematics and logic to economy and psychology to denote similar investigative procedures. Analysis is defined as the procedure by which we break down an intellectual or substantial whole into parts or . Synthesis is defined as the : to basant combine separate elements or components in order to form a coherent whole. Systems analysis researchers apply methodology to the analysis of systems involved to form an overall picture.

Information technology

The development of a computer-based information system includes a systems analysis phase which produces or enhances the data model which itself is a precursor to creating or enhancing a database. There are a number of different approaches to system analysis. When a computer-based information system is developed, systems analysis (according to the Waterfall model) would constitute the following steps:

- The development of a feasibility study, involving determining whether a project is economically, socially, technologically and organizationally feasible.
- Conducting fact-finding measures, designed to ascertain the requirements of the system's end-users. These typically span interviews, questionnaires, or visual observations of work on the existing system.
- Gauging how the end-users would operate the system (in terms of general experience in using computer hardware or software), what the system would be used for etc.

Another view outlines a phased approach to the process. This approach breaks systems analysis into 5 phases:

- Scope definition
- Problem analysis
- Requirements analysis
- Logical design
- Decision analysis

Use cases are a widely-used systems analysis modeling tool for identifying and expressing the functional requirements of a system. Each use case is a business scenario or event for which the system must provide a defined response. Use cases evolved out of object-oriented analysis; however, their use as a modeling tool has become common in many other methodologies for system analysis and design.

Practitioners

Practitioners of systems analysis are often called up to dissect systems that have grown haphazardly to determine the current components of the system. This was shown during the year 2000 re-engineering effort as business and manufacturing processes were examined as part of the Y2K automation upgrades. Employment utilizing systems analysis include systems analyst, business analyst, manufacturing engineer, enterprise architect, etc.

While practitioners of systems analysis can be called upon to create new systems they often modify, expand or document existing systems (processes, procedures and methods). A set of components interact with each other to accomplish some specific purpose. Systems are all around us. Our body is itself a system. A business is also a system. Men, money, machine, market and material are the components of business system that work together that achieve the common goal of the organization.

Characteristics: >> Systems are on-going never ending and continuous process. >> A system may be closed or open. All on-going system are open thus closed systems exist only as a concept. >> A system must have: Standard for acceptable performance

```
      + A method of comparing actual performance with standard
performance
      + A method of measuring actual performance.
      + A method for feedback.
```

Chapter- 6

Primary Areas of Product Lifecycle Management

Project portfolio management

Project Portfolio Management (**PPM**) is a term used by project managers and project management (PM) organizations, (or PMO's), to describe methods for analyzing and collectively managing a group of current or proposed projects based on numerous key characteristics. The fundamental objective of PPM is to determine the optimal mix and sequencing of proposed projects to best achieve the organization's overall goals - typically expressed in terms of hard economic measures, business strategy goals, or technical strategy goals - while honoring constraints imposed by management or external real-world factors. Typical attributes of projects being analyzed in a PPM process include each project's total expected cost, consumption of scarce resources (human or otherwise) expected timeline and schedule of investment, expected nature, magnitude and timing of benefits to be realized, and relationship or inter-dependencies with other projects in the portfolio.

The key challenge to implementing an effective PPM process is typically securing the mandate to do so. Many organizations are culturally inured to an informal method of making project investment decisions, which can be compared to political processes observable in the U.S. legislature. However this approach to making project investment decisions has led many organizations to unsatisfactory results, and created demand for a more methodical and transparent decision making process. That demand has in turn created a commercial marketplace for tools and systems which facilitate such a process.

Some commercial vendors of PPM software emphasize their products' ability to treat projects as part of an overall investment portfolio. PPM advocates see it as a shift away from one-off, ad hoc approaches to project investment decision making. Most PPM tools and methods attempt to establish a set of values, techniques and technologies that enable visibility, standardization, measurement and process improvement. PPM tools attempt to enable organizations to manage the continuous flow of projects from concept to completion.

Treating a set of projects as a portfolio would be, in most cases, an improvement on the ad hoc, one-off analysis of individual project proposals. The relationship between PPM techniques and existing investment analysis methods is a matter of debate. While many are represented as "rigorous" and "quantitative", few PPM tools attempt to incorporate established financial portfolio optimization methods like modern portfolio theory or Applied Information Economics, which have been applied to project portfolios, including even non-financial issues.

Controversy over the "investment discipline" of PPM

Developers of PPM tools see their solutions as borrowing from the financial investment world. However, other than using the word "portfolio", few can point to any specific portfolio optimization methods implemented in their tools.

A project can be viewed as a composite of resource investments such as skilled labour and associated salaries, IT hardware and software, and the opportunity cost of deferring other project work. As project resources are constrained, business management can derive greatest value by allocating these resources towards project work that is objectively and relatively determined to meet business objectives more so than other project opportunities. Thus, the decision to invest in a project can be made based upon criteria that measures the relative benefits (eg. supporting business objectives) and its relative costs and risks to the organization.

In principle, PPM attempts to address issues of resource allocation, e.g., money, time, people, capacity, etc. In order for it to truly borrow concepts from the financial investment world, the portfolio of projects and hence the PPM movement should be grounded in some financial objective such as increasing shareholder value, top line growth, etc. Equally important, risks must be computed in a statistically, actuarially meaningful sense. Optimizing resources and projects without these in mind fails to consider the most important resource any organization has and which is easily understood by people throughout the organization whether they be IT, finance, marketing, etc and that resource is money.

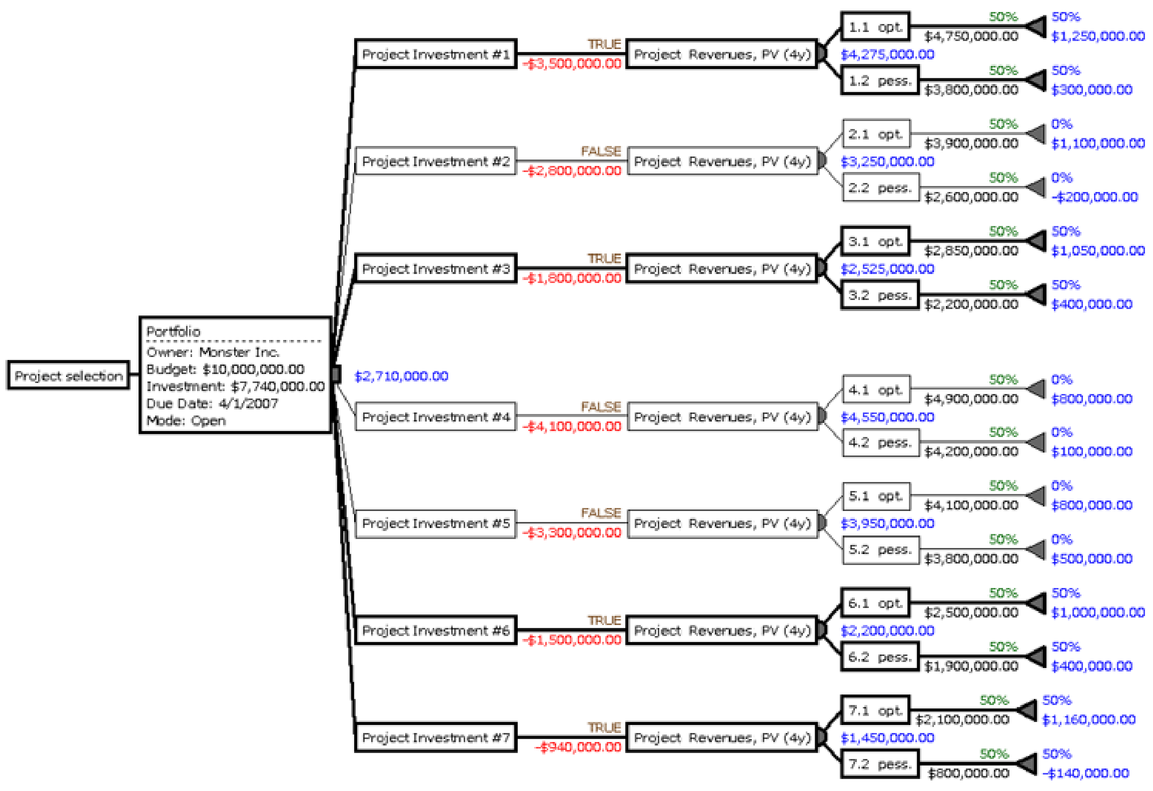
While being tied largely to IT and fairly synonymous with IT portfolio management, PPM is ultimately a subset of corporate portfolio management and should be exportable/utilized by any group selecting and managing discretionary projects. However, most PPM methods and tools opt for various subjective weighted scoring methods, not quantitatively rigorous methods based on options theory, modern portfolio theory, Applied Information Economics or operations research.

Beyond the project investment decision, PPM aims to support ongoing measurement of the project portfolio so each project can be monitored for its relative contribution to business goals. If a project is either performing below expectations (cost overruns, benefit erosion) or is no longer highly aligned to business objectives (which change with natural market and statutory evolution), management can choose to decommit from a project and redirect its resources elsewhere. This analysis, done periodically, will "refresh" the portfolio to better align with current states and needs.

Historically, many organizations were criticized for focusing on "doing the wrong things well." PPM attempts to focus on a fundamental question: "Should we be doing this project or this portfolio of projects at all?" One litmus test for PPM success is to ask "Have you ever canceled a project that was on time and on budget?" With a true PPM approach in place, it is much more likely that the answer is "yes." As goals change so should the portfolio mix of what projects are funded or not funded no matter where they are in their individual lifecycles. Making these portfolio level business investment decisions allows the organization to free up resources, even those on what were before considered "successful" projects, to then work on what is really important to the organization.

Optimizing for payoff

One method PPM tools or consultants might use is the use of decision trees with decision nodes that allow for multiple options and optimize against a constraint. The organization in the following example has options for 7 projects but the portfolio budget is limited to $10,000,000. The selection made are the projects 1, 3, 6 and 7 with a total investment of $7,740,000 - the optimum under these conditions. The portfolio's payoff is $2,710,000.



Presumably, all other combinations of projects would either exceed the budget or yield a lower payoff. However, this is an extremely simplified representation of risk and is unlikely to be realistic. Risk is usually a major differentiator among projects but it is difficult to quantify risk in a statistically and actuarially meaningful manner (with probability theory, Monte Carlo Method, statistical analysis, etc.). This places limits on the deterministic nature of the results of a tool such as a decision tree (as predicted by modern portfolio theory).

An approach to include uncertainty and risk in portfolio optimization takes a decision centric view of the project portfolio optimization process . In this view there are five key decisions that must be made:

- Decision D1: Decide what strategic initiatives, benefits, and resource limitation criteria (i.e. measures) to use for project filtering and portfolio ranking.
- Decision D2: Decide which criteria are most important to achieve.
- Decision D3: Decide which project ideas or needs are worth developing into business cases?
- Decision D4: Decide which Business Cases should be considered as part of the portfolio
- Decision D5: Decide which projects to fund

For each decision is imperative to fuse both qualitative and quantitative evaluations of alternatives where each measure includes uncertainty. It is the uncertainty in the strategic alignment, value benefits and resources demand that cause risk and thus drives all five decisions.

Resource allocation

Resource allocation is a critical component of PPM. Once it is determined that one or many projects meet defined objectives, the available resources of an organization must be evaluated for its ability to meet project demand (aka as a demand "pipeline" discussed below). Effective resource allocation typically requires an understanding of existing labor or funding resource commitments (in either business operations or other projects) as well as the skills available in the resource pool. Project investment should only be made in projects where the necessary resources are available during a specified period of time.

Resources may be subject to physical constraints. For example, IT hardware may not be readily available to support technology changes associated with ideal implementation timeframe for a project. Thus, a holistic understanding of all project resources and their availability must be conjoined with the decision to make initial investment or else projects may encounter substantial risk during their lifecycle when unplanned resource constraints arise to delay achieving project objectives.

Beyond the project investment decision, PPM involves ongoing analysis of the project portfolio so each investment can be monitored for its relative contribution to business goals versus other portfolio investments. If a project is either performing below expectations (cost overruns, benefit erosion) or is no longer aligned to business objectives (which change with natural market and statutory evolution), management can choose to decommit from a project to stem further investment and redirect resources towards other projects that better fit business objectives. This analysis can typically be performed on a periodic basis (eg. quarterly or semi-annually) to "refresh" the portfolio for optimal business performance. In this way both new and existing projects are continually monitored for their contributions to overall portfolio health. If PPM is applied in this manner, management can more clearly and transparently demonstrate its effectiveness to its shareholders or owners.

Implementing PPM at the enterprise level faces a challenge in gaining enterprise support because investment decision criteria and weights must be agreed to by the key stakeholders of the organization, each of whom may be incentivised to meet specific goals that may not necessarily align with those of the entire organization. But if enterprise business objectives can be manifested in and aligned with the objectives of its distinct business unit sub-organizations, portfolio criteria agreement can be achieved more easily. (Assadourian 2005)

From a requirements management perspective Project Portfolio Management can be viewed as the upper-most level of business requirements management in the company, seeking to understand the business requirements of the company and what portfolio of projects should be undertaken to achieve them. It is through portfolio management that each individual project should receive its allotted business requirements (Denney 2005).

Pipeline management

In addition to managing the mix of projects in a company, Project Portfolio Management must also determine whether (and how) a set of projects in the portfolio can be executed by a company in a specified time, given finite development resources in the company. This is called *pipeline management*. Fundamental to pipeline management is the ability to measure the planned allocation of development resources according to some strategic plan. To do this, a company must be able to estimate the effort planned for each project in the portfolio, and then roll the results up by one or more strategic project types e.g., effort planned for research projects. (Cooper et al. 1998); (Denney 2005) discusses project portfolio and pipeline management in the context of use case driven development.

Organizational applicability

The complexity of PPM and other approaches to IT projects (e.g., treating them as a capital investment) may render them not suitable for smaller or younger organizations. An obvious reason for this is that a few IT projects doesn't make for much of a portfolio selection. Other reasons include the cost of doing PPM—the data collection, the analysis, the documentation, the education, and the change to decision-making processes.

Product design

Example of designed product - Roomba robotic vacuum cleaner.

Product design is concerned with the efficient and effective generation and development of ideas through a process that leads to new products.

Product Designers conceptualize and evaluate ideas, making them tangible through products in a more systematic approach. Their role is to combine art, science and technology to create tangible three-dimensional goods. This evolving role has been facilitated by digital tools that allow designers to communicate, visualize and analyze ideas in a way that would have taken greater manpower in the past.

Product design is sometimes confused with industrial design, industrial design is concerned with the aspect of that process that brings that sort of artistic form and usability usually associated with craft design to that of mass produced goods.

Process

Product designers follow various methodology that requires a specific skill set to complete.

Initial Stage

- **Idea Generation** can be from imagination, observation, or research.
- **Need Based Generation** can be from the need to solve a problem, the need to follow the popular trends, or the need for a product to do a specific task.

Mid Stage

- **Design Solutions** arise from meeting user needs, concept development, form exploration, ergonomics, prototyping, materials, and technology.
- **Production** involves fabrication and manufacturing the design.

Final Stage

- **Marketing** involves selling the product. It can either be client based which mean the a client buys the design and manufactures it and then sells it to customers. Or it can be user based where the product is sold directly to the user by the designer.

Application

Product design ranges from furniture, electronics, lighting, tools, toys, and general everyday objects.

Manufacturing process management

Manufacturing process management (**MPM**) is a collection of technologies and methods used to define how products are to be manufactured. MPM differs from ERP/MRP which is used to plan the ordering of materials and other resources, set manufacturing schedules, and compile cost data.

A cornerstone of MPM is the central repository for the integration of all these tools and activities aids in the exploration of alternative production line scenarios; making assembly lines more efficient with the aim of reduced lead time to product launch, shorter product times and reduced work in progress (WIP) inventories as well as allowing rapid response to product or product changes.

Topics and technology

- Production process planning
    - Manufacturing concept planning
    - Factory layout planning and analysis
        - work flow simulation.
        - walk-path assembly planning
        - plant design optimization
    - Mixed model line balancing.
    - Workloads on multiple stations.

    - Process simulation tools e.g. die press lines, manufacturing lines
    - Ergonomic simulation and assessment of production assembly tasks
    - Resource planning
- Computer-aided manufacturing (CAM)
    - Numerical control CNC
    - Direct Numerical Control (DNC)
    - Tooling/equipment/fixtures development
    - Tooling and Robot work-cell setup and offline programming (OLP)
- Generation of shop floor work instructions
- Time and cost estimates
    - ABC - Manufacturing activity-based costing
    - Production, costs, and pricing
- Quality Computer-aided quality assurance (CAQ)
    - FMEA Failure mode and effects analysis
    - SPC Statistical process control
    - Computer aided inspection with coordinate-measuring machine (CMM)
    - Tolerance stack-up analysis using PMI models.
- Success Measurements
    - Overall Equipment Effectiveness (OEE),
- Communication with other systems
    - Enterprise resource planning (ERP)
    - Manufacturing Operations Management (MOM)
    - Product Data Management (PDM)
    - SCADA (Supervisory Control and Data Acquisition) real time process monitoring and control
    - Human-machine interface (HMI) (or *man-machine interface* (MMI))
    - Distributed control system (DCS)

Product data management

Product data management (**PDM**) is the business function often within product lifecycle management that is responsible for the creation, management and publication of product data.

Introduction

Product data management **(PDM)** is the use of software or other tools to track and control data related to a particular product. The data tracked usually involves the technical specifications of the product, specifications for manufacture and development, and the types of materials that will be required to produce goods. The use of product data management allows a company to track the various costs associated with the creation and launch of a product. Product data management is part of product life cycle management, and is primarily used by engineers.

Within PDM the focus is on managing and tracking the creation, change and archive of all information related to a product. The information being stored and managed (on one or more file servers) will include engineering data such as Computer-aided design (CAD) models, drawings and their associated documents.

Product data management (PDM) serves as a central knowledge repository for process and product history, and promotes integration and data exchange among all business users who interact with products — including project managers, engineers, sales people, buyers, and quality assurance teams.

The central database will also manage metadata such as owner of a file and release status of the components. The package will: control check-in and check-out of the product data to multi-user; carry out engineering change management and release control on all versions/issues of components in a product; build and manipulate the product structure bill of materials (BOM) for assemblies; and assist in configurations management of product variants.

This enables automatic reports on product costs, etc. Furthermore, PDM enables companies producing complex products to spread product data into the entire PLM launch-process. This significantly enhances the effectiveness of the launch process.

Product data management is focused on capturing and maintaining information on products and/or services through its development and useful life. Typical information managed in the PDM module include

- Part number
- Part description
- Supplier/vendor
- Vendor part number and description
- Unit of measure
- Cost/price
- Schematic or CAD drawing
- Material data sheets

PDM Advantages:

- Track and manage all changes to product related data
- Accelerate return on investment with easy setup;
- Spend less time organizing and tracking design data;
- Improve productivity through reuse of product design data;
- Enhance collaboration.

History of PDM

PDM stems from traditional engineering design activities that created product drawings and schematics on paper and using CAD tools to create parts lists (Bills of Material structures - BOM). The PDM and BOM data is used in enterprise resource planning (ERP) systems to plan and coordinate all transactional operations of a company (sales order management, purchasing, cost accounting, logistics, etc.)

PDM is a subset of a larger concept of product lifecycle management (PLM). PLM encompasses the processes needed to launch new products (NPI), manage changes to existing products (ECN/ECO) and retire products at the end of their life (End of Life) .

Capabilities

PDM systems vary in their functionality, but some of their common capabilities are described below.

- Access control

Access control to each element in the product definition data base can be specified. Read only access can be given to personnel not directly involved with the design, development and planning process. Creation and maintenance access can be given to the individuals responsible for product and process design. As Product Data Management systems evolve towards Collaborative Product Commerce (CPC) systems which are used across multiple enterprises in a supply chain, access control becomes more critical and requires control to limit access to specific projects, products or parts for a specific supplier or customer

- Component / Material Classification

Components and materials can be classified and organized and attributes assigned. This supports standardization by identifying similar components/materials, eliminating redundancy, and establishing a preferred parts list. Establishing classes and subclasses with attributes allows a designer to search and select a needed material, component or assembly with minimal effort thereby avoiding having to re-specifying an existing or similar component or material

- Product Structure

Since the relationship of a product's parts is a logical one maintained by the information system rather than a fixed physical relationship as represented on a drawing, it is possible to readily maintain more than one relationship. This will allow different views of part relationships in assemblies to correspond to the various departmental needs (e.g., engineering and manufacturing product structures), while maintaining rigor and consistency of the product's definition through this single data base. Thus, this one logical data base can support product and process design requirements as well as maintain part relationships to serve as a manufacturing bill of materials for MRP II/ERP. In other words, PDM provides the ability to hold not just the physical relationships between parts in an assembly but also other kinds of structures; for instance, manufacturing, financial, maintenance or document relationships. So, it is possible for specialist team members to see the product structured from their point of view. Product data can be accessed via this complete Bill of Materials. This access includes assemblies, parts and related documents.An integrated approach to developing, organizing and maintaining part and product definition data facilitates the design process, makes design data more readily usable and enhances integration with process requirements

- Engineering Changes

Engineering changes can be facilitated with this configuration management and administrative control embedded within the system. CAE/CAD tools will enable engineering changes to be more thoroughly developed and analyzed to better define change impact. Once a design has been created, it can be checked-out electronically to a workstation for engineering changes. When the changes have been made, it can be returned to the central database and placed in a queue or an email notification sent for approval by designated parties. In this manner, a Change Control Board (CCB) can even "convene" and provide individual member's input electronically. In addition to supporting engineering analysis, information related to procurement, inventory, manufacturing and cost is available for members of the CCB to evaluate, designate the effectivity of the change and determine the disposition of existing items.

- Process Management and Workflow

PDM systems support process management by defining process steps related to the development, distribution and use of product data. The process is defined in the form of specified process steps and release or promotion levels that the data must achieve. The manner in which the process is defined varies with every PDM system. Within a project, responsibilities are defined for the process steps - who needs to approve the data or work on the data before it moves to the next release or promotion level. While, the current process is defined in a company's configuration management or engineering change procedures and in its new product development process, often changes have to be made to take advantage of the communication and coordination capabilities of the PDM system. This new data is moved to the next person's "in basket" within PDM or an email notification is sent.

- Collaboration

Collaboration can be supported in several ways. First, a PDM system may be the gateway that a team uses to access the information under discussion avoiding the need to copy and distribute a series of paper documents. Second, the PDM system may provide a synchronous or asynchronous collaboration environment for team members to access, present, review and produce feedback on product and process information. Further, this collaboration tool may incorporate viewing and mark-up capability, as well as provide the ability to store marked-up files or documents submitted by collaborators. Third, what are now described as collaborative product commerce systems (CPC), provide extended PDM functionality and access control outside the enterprise for customers, suppliers and interested third parties (e.g., regulatory agencies). This speeds the distribution of information, enhances coordination, and speeds the capture of feedback.

- Storage and retrieval of product information
- Product structure modeling and management
  - Bill of materials
  - Product configurations
  - Product variations
  - Product versions
- Project tracking
- Resource planning

Chapter- 7

Computer-Aided Technologies

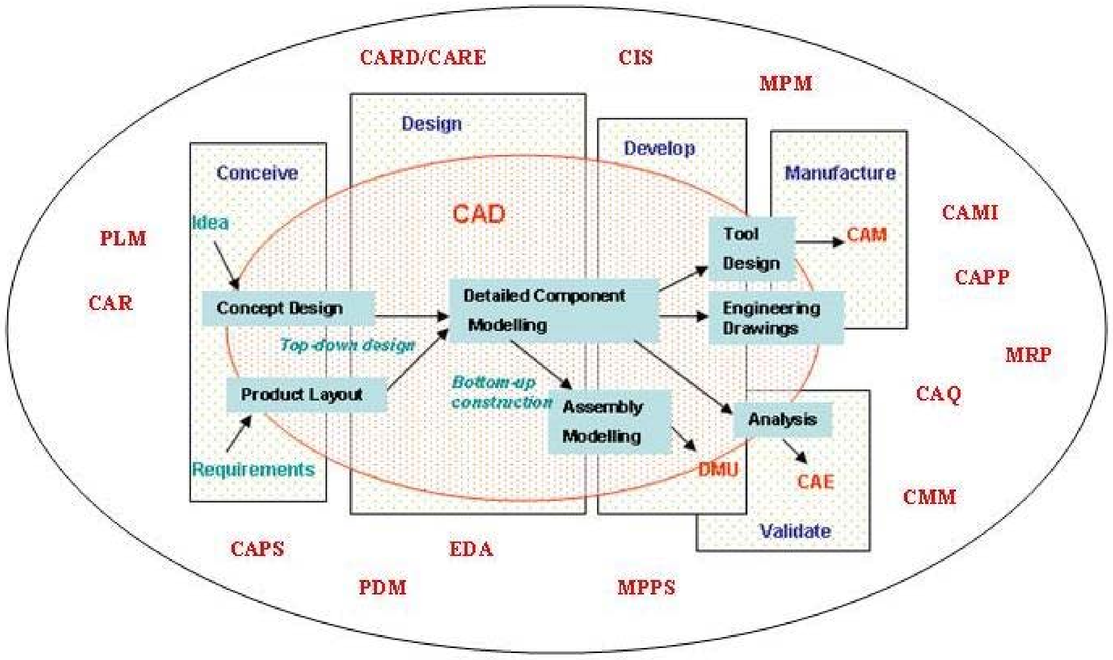
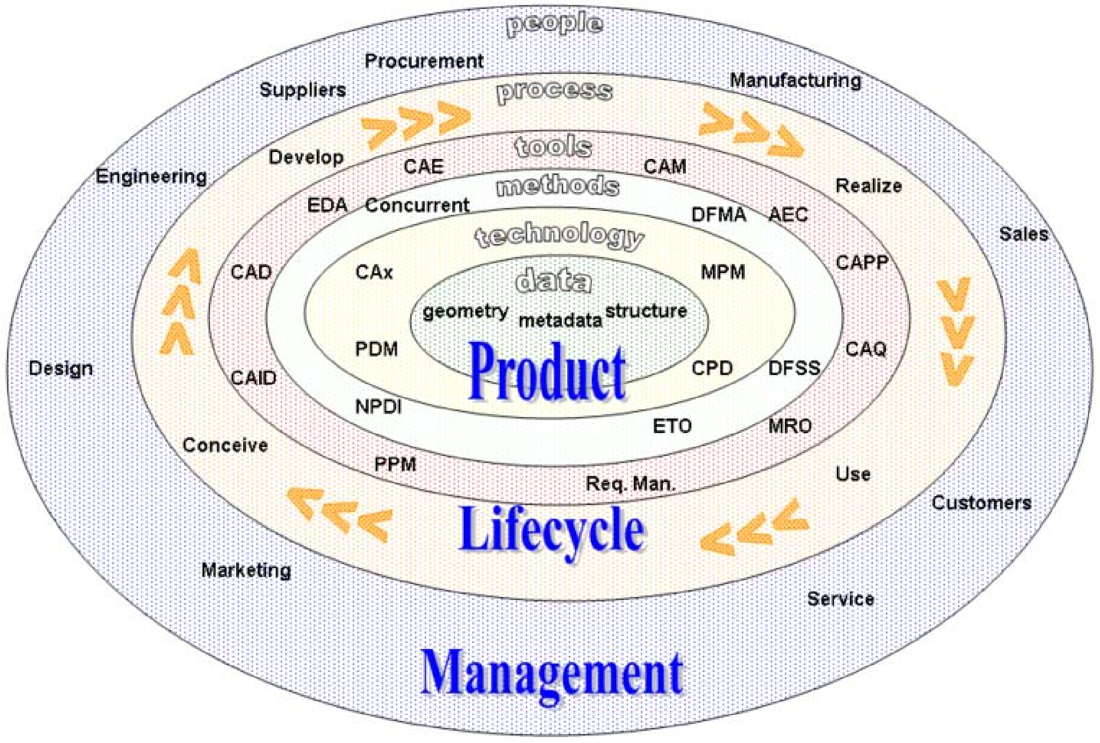


Illustration of the interaction of the various computer-aided technologies.



CAx tools in the context of product lifecycle management

Computer-aided technologies (**CAx**) is a broad term describing the use of computer technology to aid in the design, analysis, and manufacture of products.

Advanced CAx tools merge many different aspects of the product lifecycle management (PLM), including design, analysis using finite element analysis (FEA), manufacturing, production planning, product testing using virtual lab models and visualization, product documentation, product support, etc. CAx encompasses a broad range of tools, both those commercially available and those which are proprietary to the engineering firm.

The term **CAD/CAM** (computer-aided design and computer-aided manufacturing) is also often used in the context of a software tool covering a number of engineering functions.

List of computer-aided technologies

Computer-aided design

Computer-aided design (**CAD**), also known as **computer-aided design and drafting** (**CADD**), is the use of computer technology for the process of design and design-documentation. Computer Aided Drafting describes the process of drafting with a computer. CADD software, or environments, provide the user with input-tools for the purpose of streamlining design processes; drafting, documentation, and manufacturing processes. CADD output is often in the form of electronic files for print or machining operations. The development of CADD-based software is in direct correlation with the processes it seeks to economize; industry-based software (construction, manufacturing, etc.) typically uses vector-based (linear) environments whereas graphic-based software utilizes raster-based (pixelated) environments.

CADD environments often involve more than just shapes. As in the manual drafting of technical and engineering drawings, the output of CAD must convey information, such as materials, processes, dimensions, and tolerances, according to application-specific conventions.

CAD may be used to design curves and figures in two-dimensional (2D) space; or curves, surfaces, and solids in three-dimensional (3D) objects.

CAD is an important industrial art extensively used in many applications, including automotive, shipbuilding, and aerospace industries, industrial and architectural design, prosthetics, and many more. CAD is also widely used to produce computer animation for special effects in movies, advertising and technical manuals. The modern ubiquity and power of computers means that even perfume bottles and shampoo dispensers are designed using techniques unheard of by engineers of the 1960s. Because of its enormous economic importance, CAD has been a major driving force for research in computational geometry, computer graphics (both hardware and software), and discrete differential geometry.

The design of geometric models for object shapes, in particular, is often called *computer-aided geometric design* (*CAGD*).

Overview

Current computer-aided design software packages range from 2D vector-based drafting systems to 3D solid and surface modellers. Modern CAD packages can also frequently allow rotations in three dimensions, allowing viewing of a designed object from any desired angle, even from the inside looking out. Some CAD software is capable of dynamic mathematic modeling, in which case it may be marketed as **CADD** — *computer-aided design and drafting*.

CAD is used in the design of tools and machinery and in the drafting and design of all types of buildings, from small residential types (houses) to the largest commercial and industrial structures (hospitals and factories).

CAD is mainly used for detailed engineering of 3D models and/or 2D drawings of physical components, but it is also used throughout the engineering process from conceptual design and layout of products, through strength and dynamic analysis of assemblies to definition of manufacturing methods of components. It can also be used to design objects.

CAD has become an especially important technology within the scope of computer-aided technologies, with benefits such as lower product development costs and a greatly shortened design cycle. CAD enables designers to lay out and develop work on screen, print it out and save it for future editing, saving time on their drawings.

Uses

Computer-aided design is one of the many tools used by engineers and designers and is used in many ways depending on the profession of the user and the type of software in question.

CAD is one part of the whole Digital Product Development (DPD) activity within the Product Lifecycle Management (PLM) process, and as such is used together with other tools, which are either integrated modules or stand-alone products, such as:

- Computer-aided engineering (CAE) and Finite element analysis (FEA)
- Computer-aided manufacturing (CAM) including instructions to Computer Numerical Control (CNC) machines
- Photo realistic rendering
- Document management and revision control using Product Data Management (PDM).

CAD is also used for the accurate creation of photo simulations that are often required in the preparation of Environmental Impact Reports, in which computer-aided designs of intended buildings are superimposed into photographs of existing environments to represent what that locale will be like were the proposed facilities allowed to be built. Potential blockage of view corridors and shadow studies are also frequently analyzed through the use of CAD.

Types

There are several different types of CAD. Each of these different types of CAD systems require the operator to think differently about how he or she will use them and he or she must design their virtual components in a different manner for each.

There are many producers of the lower-end 2D systems, including a number of free and open source programs. These provide an approach to the drawing process without all the fuss over scale and placement on the drawing sheet that accompanied hand drafting, since these can be adjusted as required during the creation of the final draft.

3D wireframe is basically an extension of 2D drafting. Each line has to be manually inserted into the drawing. The final product has no mass properties associated with it and cannot have features directly added to it, such as holes. The operator approaches these in a similar fashion to the 2D systems, although many 3D systems allow using the wireframe model to make the final engineering drawing views.

3D "dumb" solids (programs incorporating this technology include AutoCAD) are created in a way analogous to manipulations of real world objects. Basic three-dimensional geometric forms (prisms, cylinders, spheres, and so on) have solid volumes added or subtracted from them, as if assembling or cutting real-world objects. Two-dimensional projected views can easily be generated from the models. Basic 3D solids don't usually include tools to easily allow motion of components, set limits to their motion, or identify interference between components.

3D parametric solid modeling require the operator to use what is referred to as "design intent". The objects and features created are adjustable. Any future modifications will be simple, difficult, or nearly impossible, depending on how the original part was created. One must think of this as being a "perfect world" representation of the component. If a feature was intended to be located from the center of the part, the operator needs to locate it from the center of the model, not, perhaps, from a more convenient edge or an arbitrary point, as he could when using "dumb" solids. Parametric solids require the operator to consider the consequences of his actions carefully.

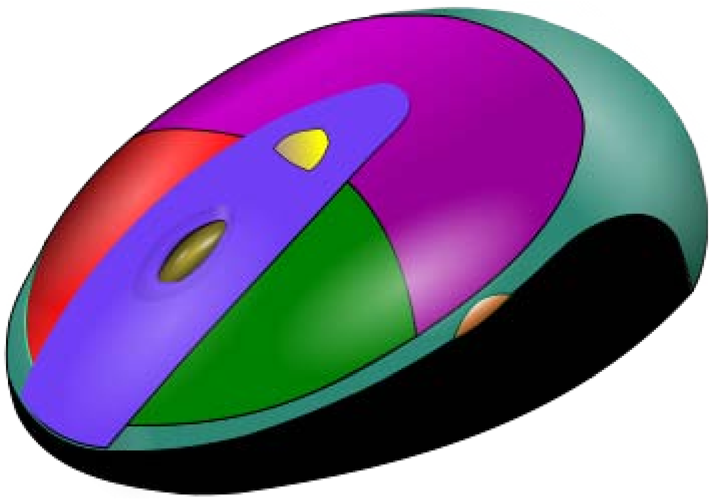
Some software packages provide the ability to edit parametric and non-parametric geometry without the need to understand or undo the design intent history of the geometry by use of direct modeling functionality. This ability may also include the additional ability to infer the correct relationships between selected geometry (e.g., tangency, concentricity) which makes the editing process less time and labor intensive while still freeing the engineer from the burden of understanding the model's design intent history. These kind of non history based systems are called Explicit Modellers or Direct CAD Modelers. The first Explicit Modeling system was introduced to the world at the end of 80's by Hewlett-Packard under the name SolidDesigner.

Draft views are able to be generated easily from the models. Assemblies usually incorporate tools to represent the motions of components, set their limits, and identify interference. The tool kits available for these systems are ever increasing; including 3D piping and injection mold designing packages.

Mid range software are integrating parametric solids more easily to the end user: integrating more intuitive functions (SketchUp), using the best of both 3D dumb solids and parametric characteristics (VectorWorks), making very real-view scenes in relative few steps (Cinema4D) or offering all-in-one (form•Z).

Top end systems offer the capabilities to incorporate more organic, aesthetics and ergonomic features into designs (Catia, GenerativeComponents). Freeform surface modelling is often combined with solids to allow the designer to create products that fit the human form and visual requirements as well as they interface with the machine.

Technology



A CAD model of a mouse.

Originally software for Computer-Aided Design systems was developed with computer languages such as Fortran, but with the advancement of object-oriented programming methods this has radically changed. Typical modern parametric feature based modeler and freeform surface systems are built around a number of key C modules with their own APIs. A CAD system can be seen as built up from the interaction of a graphical user interface (GUI) with NURBS geometry and/or boundary representation (B-rep) data via a geometric modeling kernel. A geometry constraint engine may also be employed to manage the associative relationships between geometry, such as wireframe geometry in a sketch or components in an assembly.

Unexpected capabilities of these associative relationships have led to a new form of prototyping called digital prototyping. In contrast to physical prototypes, which entail manufacturing time in the design.

Today, CAD systems exist for all the major platforms (Windows, Linux, UNIX and Mac OS X); some packages even support multiple platforms.

Right now, no special hardware is required for most CAD software. However, some CAD systems can do graphically and computationally expensive tasks, so good graphics card, high speed (and possibly multiple) CPUs and large amounts of RAM are recommended.

The human-machine interface is generally via a computer mouse but can also be via a pen and digitizing graphics tablet. Manipulation of the view of the model on the screen is also sometimes done with the use of a spacemouse/SpaceBall. Some systems also support stereoscopic glasses for viewing the 3D model.

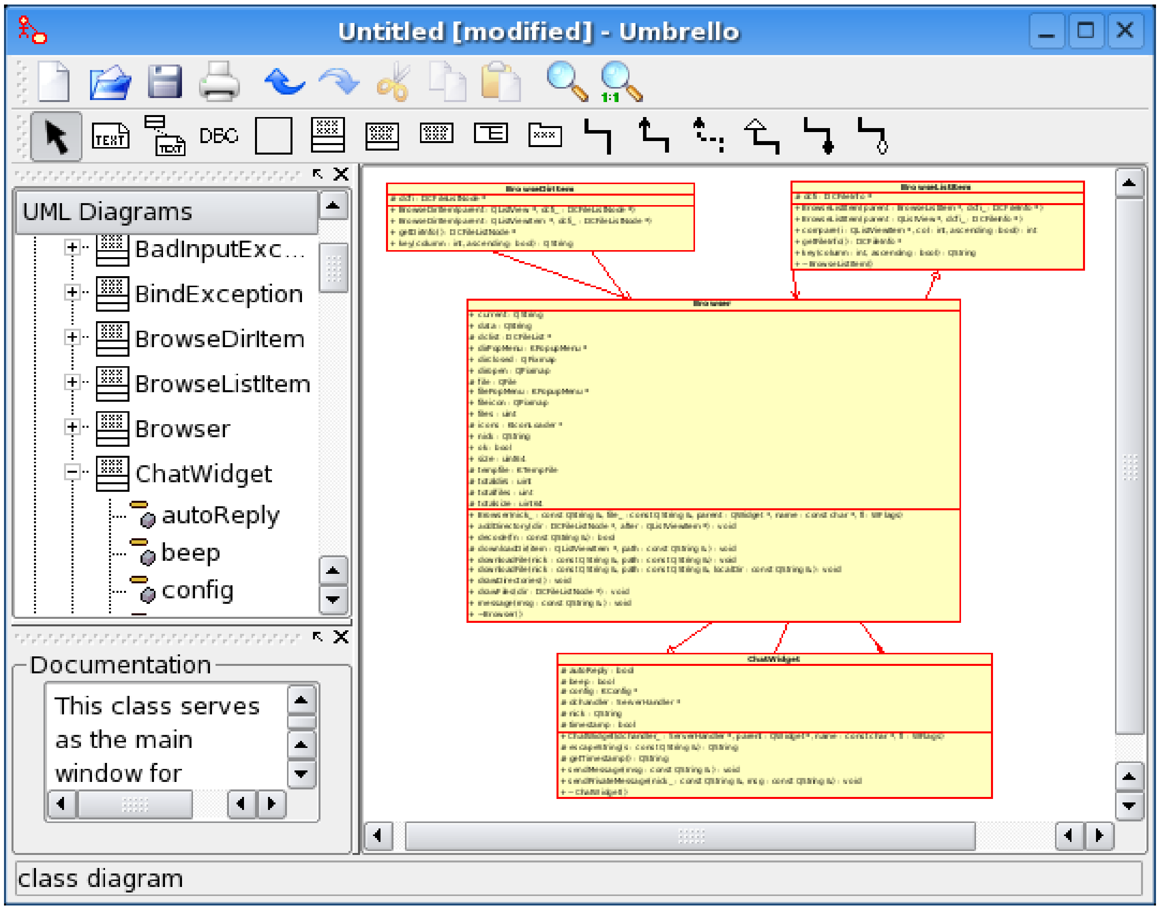
Effects

Beginning in the 1980s Computer-Aided Design programs reduced the need of draftsmen significantly, especially in small to mid-sized companies. Their affordability and ability to run on personal computers also allowed engineers to do their own drafting work, eliminating the need for entire departments. In today's world most, if not all, students in universities do not learn drafting techniques because they are not required to do so. The days of hand drawing for final drawings are almost obsolete. Universities such as New Jersey Institute of Technology no longer require the use of protractors and compasses to create drawings, instead there are several classes that focus on the use of CAD software such as Pro Engineer or IDEAS-MS.

Another consequence had been that since the latest advances were often quite expensive, small and even mid-size firms often could not compete against large firms who could use their computational edge for competitive purposes. Today, however, hardware and software costs have come down. Even high-end packages work on less expensive platforms and some even support multiple platforms. The costs associated with CAD implementation now are more heavily weighted to the costs of training in the use of these high level tools, the cost of integrating a CAD/CAM/CAE PLM using enterprise across multi-CAD and multi-platform environments and the costs of modifying design work flows to exploit the full advantage of CAD tools. CAD vendors have effectively lowered these training costs. These methods can be split into three categories:

1. Improved and simplified user interfaces. This includes the availability of “role” specific tailorable user interfaces through which commands are presented to users in a form appropriate to their function and expertise.
2. Enhancements to application software. One such example is improved design-in-context, through the ability to model/edit a design component from within the context of a large, even multi-CAD, active digital mockup.
3. User oriented modeling options. This includes the ability to free the user from the need to understand the design intent history of a complex intelligent model.

Computer-aided software engineering



This class serves as the main window for

Example of a CASE tool.

Computer-aided software engineering (**CASE**) is the scientific application of a set of tools and methods to a software system which is meant to result in high-quality, defect-free, and maintainable software products. It also refers to methods for the development of information systems together with automated tools that can be used in the software development process.

Overview

The term "Computer-aided software engineering" (CASE) can refer to the software used for the automated development of systems software, i.e., computer code. The CASE functions include analysis, design, and programming. CASE tools automate methods for designing, documenting, and producing structured computer code in the desired programming language.

Computer Aided Software Engineering (CASE) is the name given to the software used to support the software process activities such as requirement engineering, design, program development and testing. Therefore, CASE tools include design editors, data dictionaries, compilers, debuggers, system building tools, etc.

Computer Aided Software Engineering (CASE) refers to the methods dedicated to an engineering discipline for the development of information system using automated tools.

The case is mainly used for the developement of quality softwares which will perform effectively.

History

The ISDOS project at the University of Michigan initiated a great deal of interest in the whole concept of using computer systems to help analysts in the very difficult process of analysing requirements and developing systems. Several papers by Daniel Teichroew fired a whole generation of enthusiasts with the potential of automated systems development. His PSL/PSA tool was a CASE tool although it predated the term. His insights into the power of meta-meta-models was inspiring, particularly to a former student, Dr. Hasan Sayani, currently Professor, Program Director at University of Maryland University College.

Another major thread emerged as a logical extension to the DBMS directory. By extending the range of meta-data held, the attributes of an application could be held within a dictionary and used at runtime. This "active dictionary" became the precursor to the more modern "model driven execution" (MDE) capability. However, the active dictionary did not provide a graphical representation of any of the meta-data. It was the linking of the concept of a dictionary holding analysts' meta-data, as derived from the use of an integrated set of techniques, together with the graphical representation of such data that gave rise to the earlier versions of I-CASE.

The term CASE was originally coined by software company Nastec Corporation of Southfield, Michigan in 1982 with their original integrated graphics and text editor GraphiText, which also was the first microcomputer-based system to use hyperlinks to cross-reference text strings in documents—an early forerunner of today's web page link. GraphiText's successor product, DesignAid, was the first microprocessor-based tool to logically and semantically evaluate software and system design diagrams and build a data dictionary.

Under the direction of Albert F. Case, Jr. vice president for product management and consulting, and Vaughn Frick, director of product management, the DesignAid product suite was expanded to support analysis of a wide range of structured analysis and design methodologies, notably Ed Yourdon and Tom DeMarco, Chris Gane & Trish Sarson, Ward-Mellor (real-time) SA/SD and Warnier-Orr (data driven).

The next entrant into the market was Excelerator from Index Technology in Cambridge, Mass. While DesignAid ran on Convergent Technologies and later Burroughs Ngen networked microcomputers, Index launched Excelerator on the IBM PC/AT platform. While, at the time of launch, and for several years, the IBM platform did not support networking or a centralized database as did the Convergent Technologies or Burroughs machines, the allure of IBM was strong, and Excelerator came to prominence. Hot on the heels of Excelerator were a rash of offerings from companies such as Knowledgeware (James Martin, Fran Tarkenton and Don Addington), Texas Instrument's IEF and Accenture's FOUNDATION toolset (METHOD/1, DESIGN/1, INSTALL/1, FCP).

CASE tools were at their peak in the early 1990s. At the time IBM had proposed AD/Cycle, which was an alliance of software vendors centered around IBM's Software repository using IBM DB2 in mainframe and OS/2:

> *The application development tools can be from several sources: from IBM, from vendors, and from the customers themselves. IBM has entered into relationships with Bachman Information Systems, Index Technology Corporation, and Knowledgeware, Inc. wherein selected products from these vendors will be marketed through an IBM complementary marketing program to provide offerings that will help to achieve complete life-cycle coverage.*

With the decline of the mainframe, AD/Cycle and the Big CASE tools died off, opening the market for the mainstream CASE tools of today. Nearly all of the leaders of the CASE market of the early 1990s ended up being purchased by Computer Associates, including IEW, IEF, ADW, Cayenne, and Learmonth & Burchett Management Systems (LBMS).

Supporting software

Alfonso Fuggetta classified CASE into 3 categories:

1. *Tools* support only specific tasks in the software process.
2. *Workbenches* support only one or a few activities.
3. *Environments* support (a large part of) the software process.

Workbenches and environments are generally built as collections of tools. Tools can therefore be either stand alone products or components of workbenches and environments.

Tools

CASE tools are a class of software that automate many of the activities involved in various life cycle phases. For example, when establishing the functional requirements of a proposed application, prototyping tools can be used to develop graphic models of application screens to assist end users to visualize how an application will look after development. Subsequently, system designers can use automated design tools to transform the prototyped functional requirements into detailed design documents. Programmers can then use automated code generators to convert the design documents into code. Automated tools can be used collectively, as mentioned, or individually. For example, prototyping tools could be used to define application requirements that get passed to design technicians who convert the requirements into detailed designs in a traditional manner using flowcharts and narrative documents, without the assistance of automated design software.

Existing CASE tools can be classified along 4 different dimensions :

1. Life-Cycle Support
2. Integration Dimension
3. Construction Dimension
4. Knowledge Based CASE dimension

Let us take the meaning of these dimensions along with their examples one by one :

Life-Cycle Based CASE Tools

This dimension classifies CASE Tools on the basis of the activities they support in the information systems life cycle. They can be classified as Upper or Lower CASE tools.

- Upper CASE Tools: support strategic, planning and construction of conceptual level product and ignore the design aspect. They support traditional diagrammatic languages such as ER diagrams, Data flow diagram, Structure charts, Decision Trees, Decision tables, etc.
- Lower CASE Tools: concentrate on the back end activities of the software life cycle and hence support activities like physical design, debugging, construction, testing, integration of software components, maintenance, reengineering and reverse engineering activities.

Integration dimension

Three main CASE Integration dimensions have been proposed :

1. CASE Framework
2. ICASE Tools
3. Integrated Project Support Environment(IPSE)

Workbenches

Workbenches integrate several CASE tools into one application to support specific software-process activities. Hence they achieve:

- a homogeneous and consistent interface (presentation integration).
- easy invocation of tools and tool chains (control integration).

- access to a common data set managed in a centralized way (data integration).

CASE workbenches can be further classified into following 8 classes:

1. Business planning and modeling
2. Analysis and design
3. User-interface development
4. Programming
5. Verification and validation
6. Maintenance and reverse engineering
7. Configuration management
8. Project management

Environments

An environment is a collection of CASE tools and workbenches that supports the software process. CASE environments are classified based on the focus/basis of integration

1. Toolkits
2. Language-centered
3. Integrated
4. Fourth generation
5. Process-centered

Toolkits

Toolkits are loosely integrated collections of products easily extended by aggregating different tools and workbenches. Typically, the support provided by a toolkit is limited to programming, configuration management and project management. And the toolkit itself is environments extended from basic sets of operating system tools, for example, the Unix Programmer's Work Bench and the VMS VAX Set. In addition, toolkits' loose integration requires user to activate tools by explicit invocation or simple control mechanisms. The resulting files are unstructured and could be in different format, therefore the access of file from different tools may require explicit file format conversion. However, since the only constraint for adding a new component is the formats of the files, toolkits can be easily and incrementally extended.

Language-centered

The environment itself is written in the programming language for which it was developed, thus enabling users to reuse, customize and extend the environment. Integration of code in different languages is a major issue for language-centered environments. Lack of process and data integration is also a problem. The strengths of these environments include good level of presentation and control integration. Interlisp, Smalltalk, Rational, and KEE are examples of language-centered environments.

Integrated

These environments achieve presentation integration by providing uniform, consistent, and coherent tool and workbench interfaces. Data integration is achieved through the *repository* concept: they have a specialized database managing all information produced and accessed in the environment. Examples of integrated environment are IBM AD/Cycle and DEC Cohesion.

Fourth-generation

Fourth-generation environments were the first integrated environments. They are sets of tools and workbenches supporting the development of a specific class of program: electronic data processing and business-oriented applications. In general, they include programming tools, simple configuration management tools, document handling facilities and, sometimes, a code generator to produce code in lower level languages. Informix 4GL, and Focus fall into this category.

Process-centered

Environments in this category focus on process integration with other integration dimensions as starting points. A process-centered environment operates by interpreting a process model created by specialized tools. They usually consist of tools handling two functions:

- Process-model execution
- Process-model production

Examples are East, Enterprise II, Process Wise, Process Weaver, and Arcadia.

Applications

All aspects of the software development life cycle can be supported by software tools, and so the use of tools from across the spectrum can, arguably, be described as CASE; from project management software through tools for business and functional analysis, system design, code storage, compilers, translation tools, test software, and so on.

However, tools that are concerned with analysis and design, and with using design information to create parts (or all) of the software product, are most frequently thought of as CASE tools. CASE applied, for instance, to a database software product, might normally involve:

- Modeling business / real-world processes and data flow
- Development of data models in the form of entity-relationship diagrams
- Development of process and function descriptions

Risks and associated controls

Common CASE risks and associated controls include:

- *Inadequate Standardization* : Linking CASE tools from different vendors (design tool from Company X, programming tool from Company Y) may be difficult if the products do not use standardized code structures and data classifications. File formats can be converted, but usually not economically. Controls include using tools from the same vendor, or using tools based on standard protocols and insisting on demonstrated compatibility. Additionally, if organizations obtain tools for only a portion of the development process, they should consider acquiring them from a vendor that has a full line of products to ensure future compatibility if they add more tools.

- *Unrealistic Expectations* : Organizations often implement CASE technologies to reduce development costs. Implementing CASE strategies usually involves high start-up costs. Generally, management must be willing to accept a long-term payback period. Controls include requiring senior managers to define their purpose and strategies for implementing CASE technologies.

- *Slow Implementation* : Implementing CASE technologies can involve a significant change from traditional development environments. Typically, organizations should not use CASE tools the first time on critical projects or projects with short deadlines because of the lengthy training process. Additionally, organizations should consider using the tools on smaller, less complex projects and gradually implementing the tools to allow more training time.

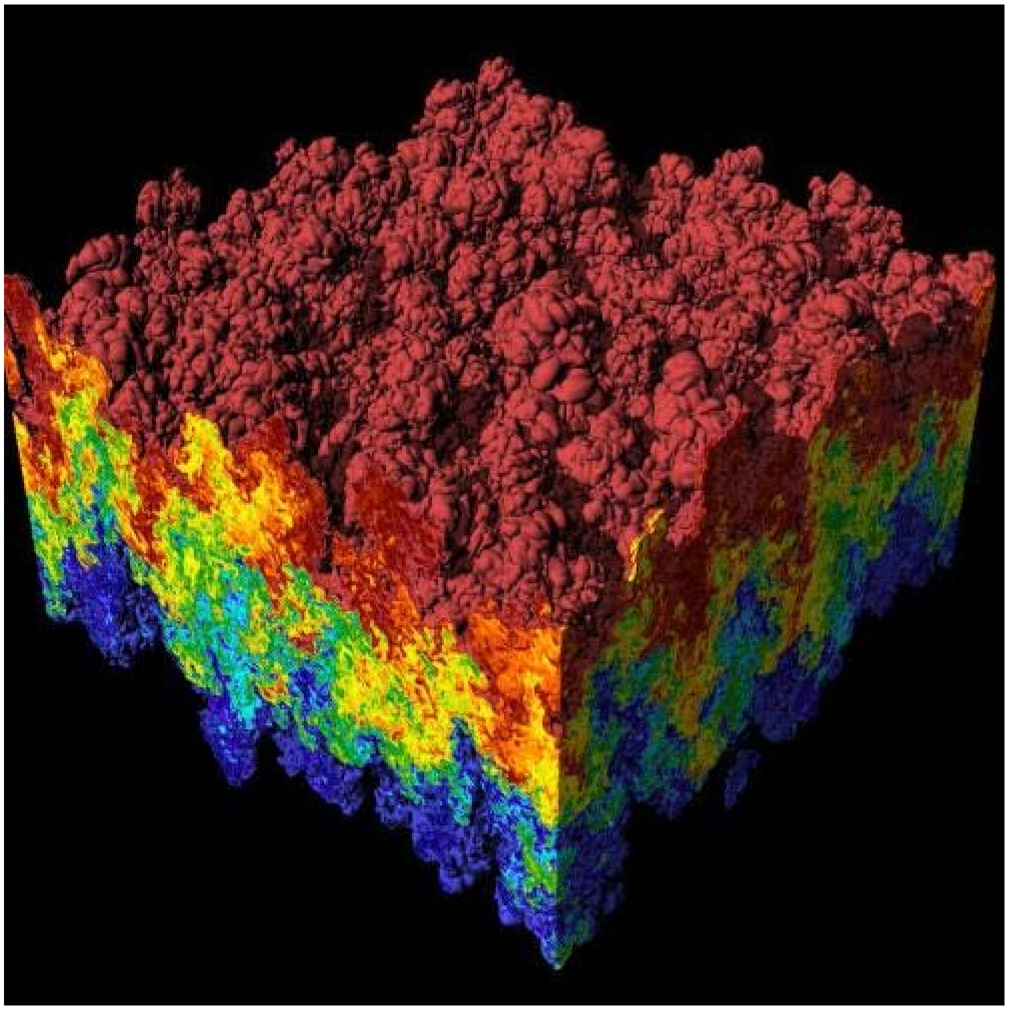
- *Weak Repository Controls* : Failure to adequately control access to CASE repositories may result in security breaches or damage to the work documents, system designs, or code modules stored in the repository. Controls include protecting the repositories with appropriate access, version, and backup controls.

Chapter- 8

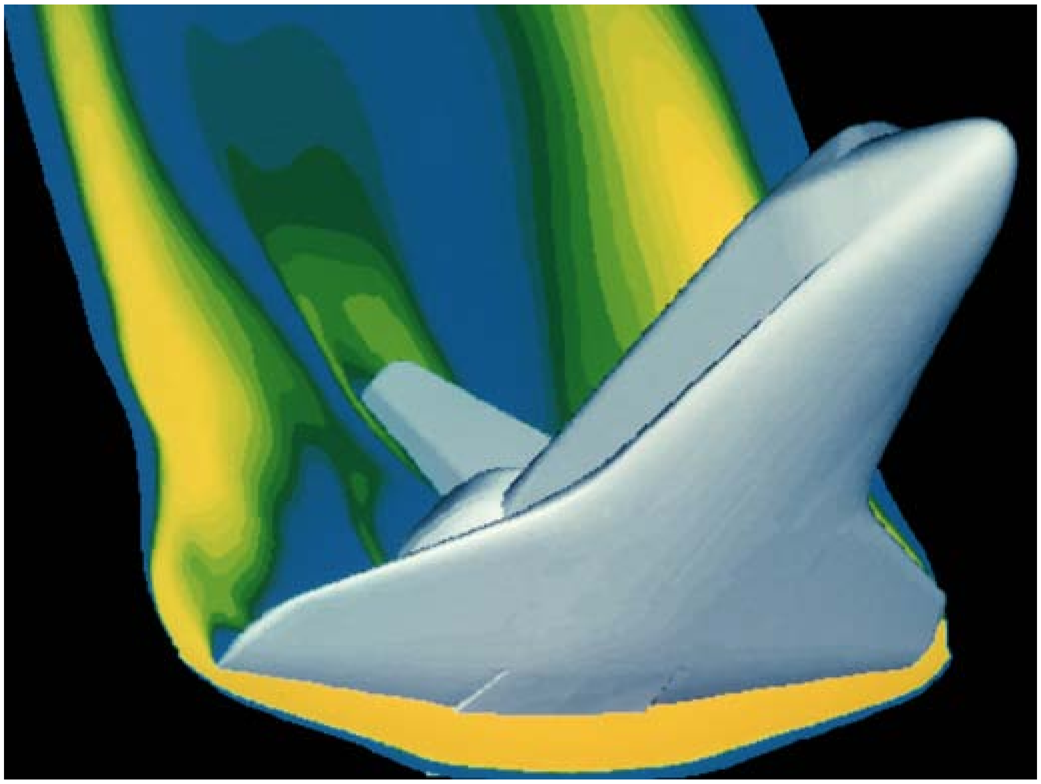
Computational Fluid Dynamics & Computer-Aided Manufacturing

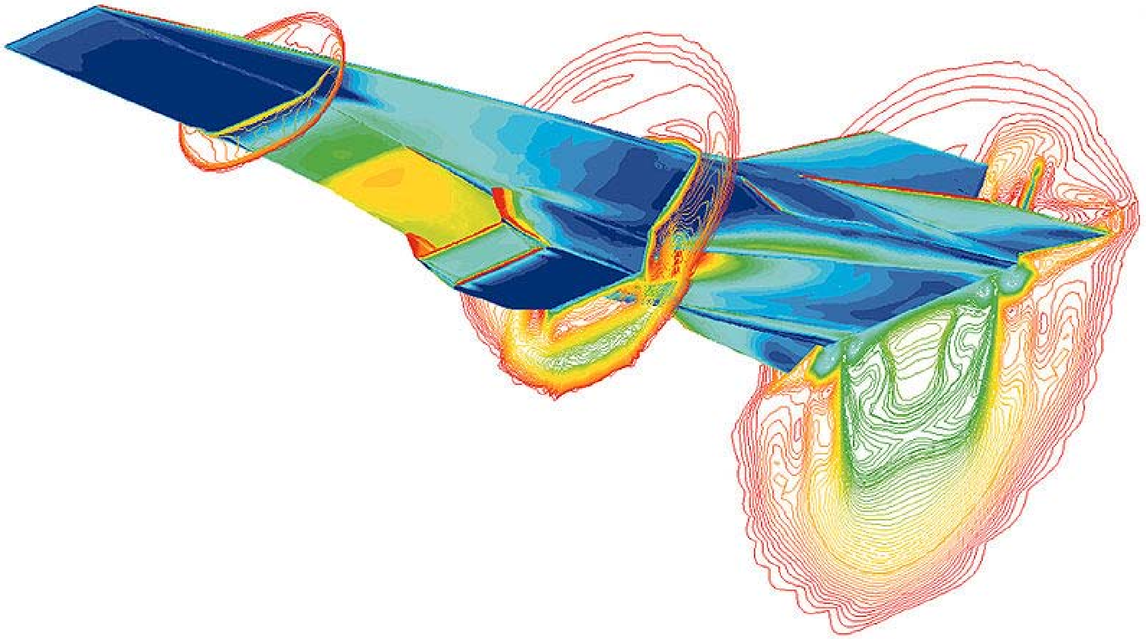
Computational fluid dynamics

Computational fluid dynamics (CFD) is a branch of fluid mechanics that uses numerical methods and algorithms to solve and analyze problems that involve fluid flows. Computers are used to perform the calculations required to simulate the interaction of liquids and gases with surfaces defined by boundary conditions. With high-speed supercomputers, better solutions can be achieved. Ongoing research, however, yield software that improves the accuracy and speed of complex simulation scenarios such as transonic or turbulent flows. Initial validation of such software is performed using a wind tunnel with the final validation coming in flight tests.



Background and history


A computer simulation of high velocity air flow around the Space Shuttle during re-entry.



A simulation of the Hyper-X scramjet vehicle in operation at Mach-7

The fundamental basis of almost all CFD problems are the Navier–Stokes equations, which define any single-phase fluid flow. These equations can be simplified by removing terms describing viscosity to yield the Euler equations. Further simplification, by removing terms describing vorticity yields the full potential equations. Finally, these equations can be linearized to yield the linearized potential equations.

Historically, methods were first developed to solve the Linearized Potential equations. Two-dimensional methods, using conformal transformations of the flow about a cylinder to the flow about an airfoil were developed in the 1930s. The computer power available paced development of three-dimensional methods. The first paper on a practical three-dimensional method to solve the linearized potential equations was published by John Hess and A.M.O. Smith of Douglas Aircraft in 1967. This method discretized the surface of the geometry with panels, giving rise to this class of programs being called Panel Methods. Their method itself was simplified, in that it did not include lifting flows and hence was mainly applied to ship hulls and aircraft fuselages. The first lifting Panel Code (A230) was described in a paper written by Paul Rubbert and Gary Saaris of Boeing Aircraft in 1968. In time, more advanced three-dimensional Panel Codes were developed at Boeing (PANAIR, A502), Lockheed (Quadpan), Douglas (HESS), McDonnell Aircraft (MACAERO), NASA (PMARC) and Analytical Methods (WBAERO, USAERO and VSAERO). Some (PANAIR, HESS and MACAERO) were higher order codes, using higher order distributions of surface singularities, while others (Quadpan, PMARC, USAERO and VSAERO) used single singularities on each surface panel. The advantage of the lower order codes was that they ran much faster on the computers of the time. Today, VSAERO has grown to be a multi-order code and is the most widely used program of this class. It has been used in the development of many submarines, surface ships, automobiles, helicopters , aircraft, and more recently wind turbines. Its sister code, USAERO is an unsteady panel method that has also been used for modeling such things as high speed trains and racing yachts. The NASA PMARC code from an early version of VSAERO and a derivative of PMARC, named CMARC, is also commercially available.

In the two-dimensional realm, a number of Panel Codes have been developed for airfoil analysis and design. The codes typically have a boundary layer analysis included, so that viscous effects can be modeled. Professor Richard Eppler of the University of Stuttgart developed the PROFIL code, partly with NASA funding, which became available in the early 1980s. This was soon followed by MIT Professor Mark Drela's XFOIL code. Both PROFIL and XFOIL incorporate two-dimensional panel codes, with coupled boundary layer codes for airfoil analysis work. PROFIL uses a conformal transformation method for inverse airfoil design, while XFOIL has both a conformal transformation and an inverse panel method for airfoil design. Both codes are used.

An intermediate step between Panel Codes and Full Potential codes were codes that used the Transonic Small Disturbance equations. In particular, the three-dimensional WIBCO code, developed by Charlie Boppe of Grumman Aircraft in the early 1980s has seen heavy use.

Developers turned to Full Potential codes, as panel methods could not calculate the non-linear flow present at transonic speeds. The first description of a means of using the Full Potential equations was published by Earll Murman and Julian Cole of Boeing in 1970. Frances Bauer, Paul Garabedian and David Korn of the Courant Institute at New York University (NYU) wrote a series of two-dimensional Full Potential airfoil codes that were widely used, the most important being named Program H. A further growth of Program H was developed by Bob Melnik and his group at Grumman Aerospace as Grumfoil. Antony Jameson, originally at Grumman Aircraft and the Courant Institute of NYU, worked with David Caughey to develop the important three-dimensional Full Potential code FLO22 in 1975. Many Full Potential codes emerged after this, culminating in Boeing's Tranair (A633) code, which still sees heavy use.

The next step was the Euler equations, which promised to provide more accurate solutions of transonic flows. The methodology used by Jameson in his three-dimensional FLO57 code (1981) was used by others to produce such programs as Lockheed's TEAM program and IAI/Analytical Methods' MGAERO program. MGAERO is unique in being a structured cartesian mesh code, while most other such codes use structured body-fitted grids (with the exception of NASA's highly successful CART3D code, Lockheed's SPLITFLOW code and Georgia Tech's NASCART-GT). Antony Jameson also developed the three-dimensional AIRPLANE code (1985) which made use of unstructured tetrahedral grids.

In the two-dimensional realm, Mark Drela and Michael Giles, then graduate students at MIT, developed the ISES Euler program (actually a suite of programs) for airfoil design and analysis. This code first became available in 1986 and has been further developed to design, analyze and optimize single or multi-element airfoils, as the MSES program. MSES sees wide use throughout the world. A derivative of MSES, for the design and analysis of airfoils in a cascade, is MISES, developed by Harold "Guppy" Youngren while he was a graduate student at MIT.

The Navier–Stokes equations were the ultimate target of developers. Two-dimensional codes, such as NASA Ames' ARC2D code first emerged. A number of three-dimensional codes were developed (OVERFLOW, CFL3D are two successful NASA contributions), leading to numerous commercial packages.

Methodology

In all of these approaches the same basic procedure is followed.

- During preprocessing
    - The geometry (physical bounds) of the problem is defined.
    - The volume occupied by the fluid is divided into discrete cells (the mesh). The mesh may be uniform or non uniform.
    - The physical modeling is defined – for example, the equations of motions + enthalpy + radiation + species conservation

    - Boundary conditions are defined. This involves specifying the fluid behaviour and properties at the boundaries of the problem. For transient problems, the initial conditions are also defined.
- The simulation is started and the equations are solved iteratively as a steady-state or transient.
- Finally a postprocessor is used for the analysis and visualization of the resulting solution.

Discretization methods

The stability of the chosen discretization is generally established numerically rather than analytically as with simple linear problems. Special care must also be taken to ensure that the discretization handles discontinuous solutions gracefully. The Euler equations and Navier–Stokes equations both admit shocks, and contact surfaces.

Some of the discretization methods being used are:

Finite Volume Method

The finite volume method (FVM) is a common approach used in CFD codes. The governing equations are solved over discrete control volumes. Finite volume methods recast the governing partial differential equations (typically the Navier-Stokes equations) in a conservative form, and then discretize the new equation. This guarantees the conservation of fluxes through a particular control volume. Though the overall solution will be conservative in nature, there is no guarantee that it is the actual solution. The finite volume equation yields governing equations in the form,

$$\frac{\partial}{\partial t}\iiint Q\,dV + \iint F\,d\mathbf{A} = 0,$$

where Q is the vector of conserved variables, F is the vector of fluxes, V is the volume of the control volume element, and $\mathbf{A}$is the surface area of the control volume element.

Finite Element Method

The finite element method (FEM) is used in structural analysis of solids, but is also applicable to fluids. However, the FEM formulation requires special care to ensure a conservative solution. The FEM formulation has been adapted for use with fluid dynamics governing equations. Although FEM must be carefully formulated to be conservative, it is much more stable than the finite volume approach However, FEM can require more memory than FVM.

In this method, a weighted residual equation is formed:

$$R_i = \iiint W_i Q \, dV^e$$

where R_i is the equation residual at an element vertex i, Q is the conservation equation expressed on an element basis, W_i is the weight factor, and V^e is the volume of the element.

Finite Difference Method

The finite difference method (FDM) has historical importance and is simple to program. It is currently only used in few specialized codes. Modern finite difference codes make use of an embedded boundary for handling complex geometries, making these codes highly efficient and accurate. Other ways to handle geometries include use of overlapping grids, where the solution is interpolated across each grid.

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = 0$$

where Q is the vector of conserved variables, and F, G, and H are the fluxes in the x, y, and z directions respectively.

Boundary Element Method

In the boundary element method, the boundary occupied by the fluid is divided into a surface mesh.

High Resolution Discretization Schemes

High-resolution schemes are used where shocks or discontinuities are present. Capturing sharp changes in the solution requires the use of second or higher-order numerical schemes that do not introduce spurious oscillations. This usually necessitates the application of flux limiters to ensure that the solution is total variation diminishing.

Turbulence models

In studying turbulent flows, the objective is to obtain a theory or a model that can yield quantities of interest, such as velocities. For turbulent flow, the range of length scales and complexity of phenomena make most approaches impossible. The primary approach in this case is to create numerical models to calculate the properties of interest. A selection of some commonly-used computational models for turbulent flows are presented in this section.

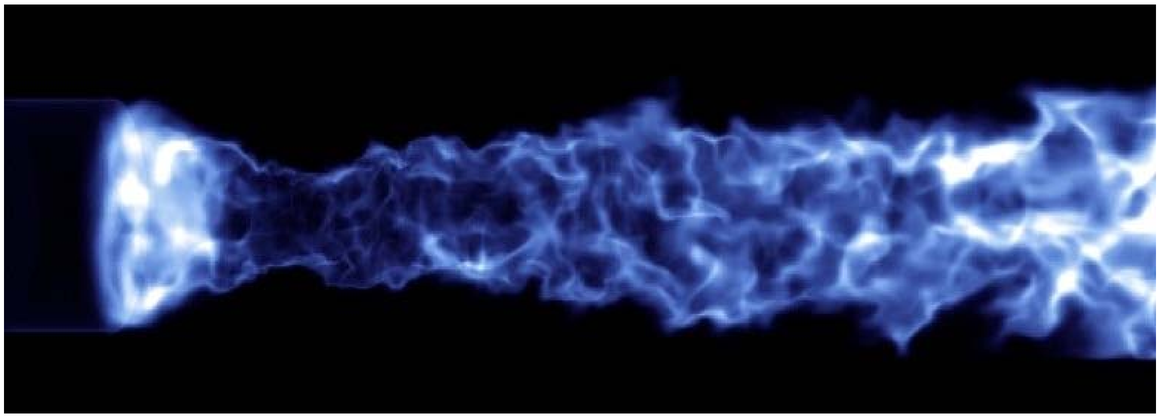
The chief difficulty in modeling turbulent flows comes from the wide range of length and time scales associated with turbulent flow. As a result, turbulence models can be classified based on the range of these length and time scales that are modeled and the range of length and time scales that are resolved. The more turbulent scales that are resolved, the finer the resolution of the simulation, and therefore the higher the computational cost. If a majority or all of the turbulent scales are modeled, the computational cost is very low, but the tradeoff comes in the form of decreased accuracy.

In addition to the wide range of length and time scales and the associated computational cost, the governing equations of fluid dynamics contain a non-linear convection term and a non-linear and non-local pressure gradient term. These nonlinear equations must be solved numerically with the appropriate boundary and initial conditions.

Direct numerical simulation

Direct numerical simulation (DNS) resolves the entire range of turbulent length scales. This marginalizes the effect of models, but is extremely expensive. The computational cost is proportional to Re^3. DNS is intractable for flows with complex geometries or flow configurations.

Large eddy simulation



Volume rendering of a non-premixed swirl flame as simulated by LES.

Large eddy simulation (LES) is a technique in which the smallest scales of the flow are removed through a filtering operation, and their effect modeled using subgrid scale models. This allows the largest and most important scales of the turbulence to be resolved, while greatly reducing the computational cost incurred by the smallest scales. This method requires greater computational resources than RANS methods, but is far cheaper than DNS.

Detached eddy simulation

Detached eddy simulations (DES) is a modification of a RANS model in which the model switches to a subgrid scale formulation in regions fine enough for LES calculations. Regions near solid boundaries and where the turbulent length scale is less than the maximum grid dimension are assigned the RANS mode of solution. As the turbulent length scale exceeds the grid dimension, the regions are solved using the LES mode.

Therefore the grid resolution for DES is not as demanding as pure LES, thereby considerably cutting down the cost of the computation. Though DES was initially formulated for the Spalart-Allmaras model (Spalart et al., 1997), it can be implemented with other RANS models (Strelets, 2001), by appropriately modifying the length scale which is explicitly or implicitly involved in the RANS model. So while Spalart-Allmaras model based DES acts as LES with a wall model, DES based on other models (like two equation models) behave as a hybrid RANS-LES model. Grid generation is more complicated than for a simple RANS or LES case due to the RANS-LES switch. DES is a non-zonal approach and provides a single smooth velocity field across the RANS and the LES regions of the solutions.

Reynolds-averaged Navier–Stokes

Reynolds-averaged Navier-Stokes (RANS) equations are the oldest approach to turbulence modeling. An ensemble version of the governing equations is solved, which introduces new *apparent stresses* known as Reynolds stresses. This adds a second order tensor of unknowns for which various models can provide different levels of closure. It is a common misconception that the RANS equations do not apply to flows with a time-varying mean flow because these equations are 'time-averaged'. In fact, statistically unsteady (or non-stationary) flows can equally be treated. This is sometimes referred to as URANS. There is nothing inherent in Reynolds averaging to preclude this, but the turbulence models used to close the equations are valid only as long as the time over which these changes in the mean occur is large compared to the time scales of the turbulent motion containing most of the energy.

RANS models can be divided into two broad approaches:

Boussinesq hypothesis

This method involves using an algebraic equation for the Reynolds stresses which include determining the turbulent viscosity, and depending on the level of sophistication of the model, solving transport equations for determining the turbulent kinetic energy and dissipation. Models include k-ε (Spalding), Mixing Length Model (Prandtl) and Zero Equation (Chen). The models available in this approach are often referred to by the number of transport equations associated with the method. For example, the Mixing Length model is a "Zero Equation" model because no transport equations are solved; the $k - \varepsilon$ is a "Two Equation" model because two transport equations (one for k and one for ε) are solved.

Reynolds stress model (RSM)

This approach attempts to actually solve transport equations for the Reynolds stresses. This means introduction of several transport equations for all the Reynolds stresses and hence this approach is much more costly in CPU effort.

Coherent vortex simulation

The coherent vortex simulation approach decomposes the turbulent flow field into a coherent part, consisting of organized vortical motion, and the incoherent part, which is the random background flow. This decomposition is done using wavelet filtering. The approach has much in common with LES, since it uses decomposition and resolves only the filtered portion, but different in that it does not use a linear, low-pass filter. Instead, the filtering operation is based on wavelets, and the filter can be adapted as the flow field evolves. Farge and Schneider tested the CVS method with two flow configurations and showed that the coherent portion of the flow exhibited the $-\frac{5}{3}$energy spectrum exhibited by the total flow, and corresponded to coherent structures (vortex tubes), while the incoherent parts of the flow composed homogeneous background noise, which exhibited no organized structures. Goldstein and Oleg applied the CVS model to large eddy simulation, but did not assume that the wavelet filter completely eliminated all coherent motions from the subfilter scales. By employing both LES and CVS filtering, they showed that the SFS dissipation was dominated by the SFS flow field's coherent portion.

PDF methods

Probability density function (PDF) methods for turbulence, first introduced by Lundgren, are based on tracking the one-point probability density function of the velocity, $f_V(\boldsymbol{v};\boldsymbol{x},t)d\boldsymbol{v}$, which gives the probability of the velocity at point $\boldsymbol{x}$being between $\boldsymbol{v}$and $\boldsymbol{v}+d\boldsymbol{v}$. This approach is analogous to the kinetic theory of gases, in which the macroscopic properties of a gas are described by a large number of particles. PDF methods are unique in that they can be applied in the framework of a number of different turbulence models; the main differences occur in the form of the PDF transport equation. For example, in the context of large eddy simulation, the PDF becomes the filtered PDF. PDF methods can also be used to describe chemical reactions, and are particularly useful for simulating chemically reacting flows because the chemical source term is unclosed and does not require a model. The PDF is commonly tracked by using Lagrangian particle methods; when combined with large eddy simulation, this leads to a Langevin equation for subfiler particle evolution.

Vortex method

The Vortex method is a grid-free technique for the simulation of turbulent flows. It uses vortices as the computational elements, mimicking the physical structures in turbulence. Vortex methods were developed as a grid-free methodology that would not be limited by the fundamental smoothing effects associated with grid-based methods. To be practical, however, vortex methods require means for rapidly computing velocities from the vortex elements – in other words they require the solution to a particular form of the N-body problem (in which the motion of N objects is tied to their mutual influences). A breakthrough came in the late 1980s with the development of the Fast Multipole Method (FMM), an algorithm by V. Rokhlin (Yale) and L. Greengard (Courant Institute) that has been heralded as one of the top ten advances in numerical science of the 20th century. This breakthrough paved the way to practical computation of the velocities from the vortex elements and is the basis of successful algorithms.

Software based on the Vortex method offer a new means for solving tough fluid dynamics problems with minimal user intervention. All that is required is specification of problem geometry and setting of boundary and initial conditions. Among the significant advantages of this modern technology;

- It is practically grid-free, thus eliminating numerous iterations associated with RANS and LES.
- All problems are treated identically. No modeling or calibration inputs are required.
- Time-series simulations, which are crucial for correct analysis of acoustics, are possible.
- The small scale and large scale are accurately simulated at the same time.

Vorticity Confinement method

The Vorticity confinement (VC) method is an Eulerian technique used in the simulation of turbulent wakes. It uses a solitary-wave like approach to produce stable solution with no numerical spreading. VC can capture the small scale features to over as few as 2 grid cells. Within these features, a nonlinear difference equation is solved as opposed to the finite difference equation. VC is similar to shock capturing methods, where conservation laws are satisfied, so that the essential integral quantities are accurately computed.

Two phase flow

The modeling of two-phase flow is still under development. Different methods have been proposed. The Volume of fluid method has received a lot of attention lately, for problems that do not have dispersed particles, but the Level set method and front tracking are also valuable approaches. Most of these methods are either good in maintaining a sharp interface or at conserving mass. This is crucial since the evaluation of the density, viscosity and surface tension is based on the values averaged over the interface. Lagrangian multiphase models, which are used for dispersed media, are based on solving the Lagrangian equation of motion for the dispersed phase.

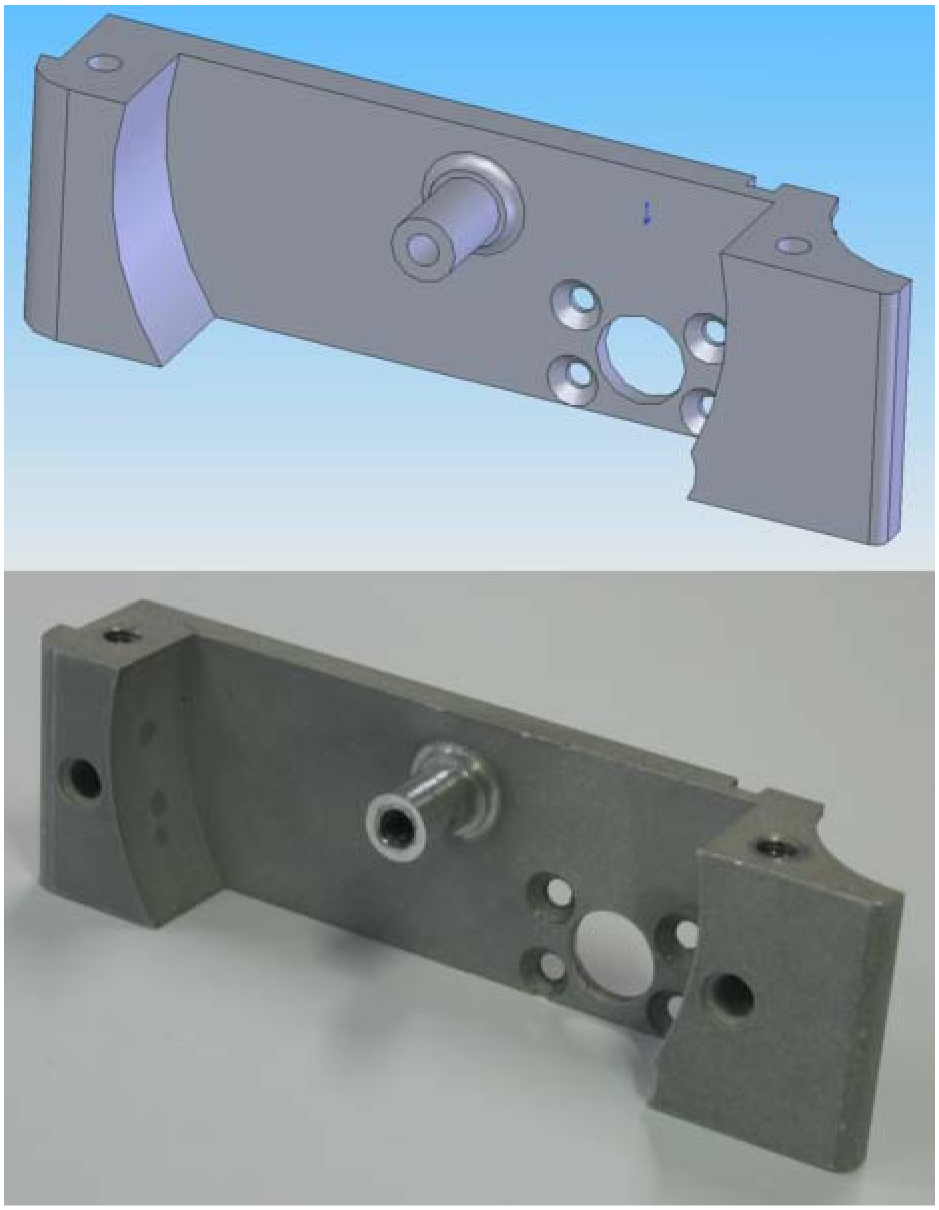
Solution algorithms

Discretization in space produces a system of ordinary differential equations for unsteady problems and algebraic equations for steady problems. Implicit or semi-implicit methods are generally used to integrate the ordinary differential equations, producing a system of (usually) nonlinear algebraic equations. Applying a Newton or Picard iteration produces a system of linear equations which is nonsymmetric in the presence of advection and indefinite in the presence of incompressibility. Such systems, particularly in 3D, are frequently too large for direct solvers, so iterative methods are used, either stationary methods such as successive overrelaxation or Krylov subspace methods. Krylov methods such as GMRES, typically used with preconditioning, operate by minimizing the residual over successive subspaces generated by the preconditioned operator.

Multigrid is especially popular , both as a solver and as a preconditioner, due to its asymptotically optimal performance on many problems. Traditional solvers and preconditioners are effective at reducing high-frequency components of the residual, but low-frequency components typically require many iterations to reduce. By operating on multiple scales, multigrid reduces all components of the residual by similar factors, leading to a mesh-independent number of iterations.

For indefinite systems, preconditioners such as incomplete LU factorization, additive Schwarz, and multigrid perform poorly or fail entirely, so the problem structure must be used for effective preconditioning. The traditional methods commonly used in CFD are the SIMPLE and Uzawa algorithms which exhibit mesh-dependent convergence rates, but recent advances based on block LU factorization combined with multigrid for the resulting definite systems have led to preconditioners that deliver mesh-independent convergence rates.

Computer-aided manufacturing


CAD model and CNC machined part

Computer-aided manufacturing (**CAM**) is the use of computer software to control machine tools and related machinery in the manufacturing of workpieces. This is not the only definition for CAM, but it is the most common; CAM may also refer to the use of a computer to assist in all operations of a manufacturing plant, including planning, management, transportation and storage. Its primary purpose is to create a faster production process and components and tooling with more precise dimensions and material consistency, which in some cases, uses only the required amount of raw material (thus minimizing waste), while simultaneously reducing energy consumption.

CAM is a subsequent computer-aided process after computer-aided design (CAD) and sometimes computer-aided engineering (CAE), as the model generated in CAD and verified in CAE can be input into CAM software, which then controls the machine tool.

Overview

Chrome-cobalt disc with dental implants manufactured using CAM

Traditionally, CAM has been considered as a numerical control (NC) programming tool, wherein two-dimensional (2-D) or three-dimensional (3-D) models of components generated in CAD software are used to generate G-code to drive computer numerically controlled (CNC) machine tools. Simple designs such as bolt circles or basic contours do not necessitate importing a CAD file.

As with other "Computer-Aided" technologies, CAM does not eliminate the need for skilled professionals such as manufacturing engineers, NC programmers, or machinists. CAM, in fact, leverages both the value of the most skilled manufacturing professionals through advanced productivity tools, while building the skills of new professionals through visualization, simulation and optimization tools.

History

The first commercial applications of CAM were in large companies in the automotive and aerospace industries for example UNISURF in 1971 at Renault for car body design and tooling.

Historically, CAM software was seen to have several shortcomings that necessitated an overly high level of involvement by skilled CNC machinists. Fallows created the first CAM software but this had severe shortcomings and was promptly taken back into the developing stage. CAM software would output code for the least capable machine, as each machine tool control added on to the standard G-code set for increased flexibility. In some cases, such as improperly set up CAM software or specific tools, the CNC machine required manual editing before the program will run properly. None of these issues were so insurmountable that a thoughtful engineer or skilled machine operator could not overcome for prototyping or small production runs; G-Code is a simple language. In high production or high precision shops, a different set of problems were encountered where an experienced CNC machinist must both hand-code programs and run CAM software.

Integration of CAD with other components of CAD/CAM/CAE Product lifecycle management (PLM) environment requires an effective CAD data exchange. Usually it had been necessary to force the CAD operator to export the data in one of the common data formats, such as IGES or STL, that are supported by a wide variety of software. The output from the CAM software is usually a simple text file of G-code, sometimes many thousands of commands long, that is then transferred to a machine tool using a direct numerical control (DNC) program.

CAM packages could not, and still cannot, reason as a machinist can. They could not optimize toolpaths to the extent required of mass production. Users would select the type of tool, machining process and paths to be used. While an engineer may have a working knowledge of g-code programming, small optimization and wear issues compound over time. Mass-produced items that require machining are often initially created through casting or some other non-machine method. This enables hand-written, short, and highly optimized g-code that could not be produced in a CAM package.

At least in the United States, there is a shortage of young, skilled machinists entering the workforce able to perform at the extremes of manufacturing; high precision and mass production. As CAM software and machines become more complicated, the skills required of a machinist or machine operator advance to approach that of a computer programmer and engineer rather than eliminating the CNC machinist from the workforce.

Typical areas of concern:

- High Speed Machining, including streamlining of tool paths
- Multi-function Machining
- 5 Axis Machining
- Feature recognition and machining

- Automation of Machining processes
- Ease of Use

Overcoming historical shortcomings

Over time, the historical shortcomings of CAM are being attenuated, both by providers of niche solutions and by providers of high-end solutions. This is occurring primarily in three arenas:

1. Ease of use
2. Manufacturing complexity
3. Integration with PLM and the extended enterprise

Ease in use

> For the user who is just getting started as a CAM user, out-of-the-box capabilities providing Process Wizards, templates, libraries, machine tool kits, automated feature based machining and job function specific tailorable user interfaces build user confidence and speed the learning curve.
> User confidence is further built on 3D visualization through a closer integration with the 3D CAD environment, including error-avoiding simulations and optimizations.

Manufacturing complexity

> The manufacturing environment is increasingly complex. The need for CAM and PLM tools by the manufacturing engineer, NC programmer or machinist is similar to the need for computer assistance by the pilot of modern aircraft systems. The modern machinery cannot be properly used without this assistance.
> Today's CAM systems support the full range of machine tools including: turning, 5 axis machining and wire EDM. Today’s CAM user can easily generate streamlined tool paths, optimized tool axis tilt for higher feed rates and optimized Z axis depth cuts as well as driving non-cutting operations such as the specification of probing motions.

Integration with PLM and the extended enterprise

> Today’s competitive and successful companies have used PLM to integrate manufacturing with enterprise operations from concept through field support of the finished product.
> To ensure ease of use appropriate to user objectives, modern CAM solutions are scalable from a stand-alone CAM system to a fully integrated multi-CAD 3D solution-set. These solutions are created to meet the full needs of manufacturing personnel including part planning, shop documentation, resource management and data management and exchange.

Machining process

Most machining progresses through four stages, each of which is implemented by a variety of basic and sophisticated strategies, depending on the material and the software available. The stages are:

Roughing

This process begins with raw stock, known as billet, and cuts it very roughly to shape of the final model. In milling, the result often gives the appearance of terraces, because the strategy has taken advantage of the ability to cut the model horizontally. Common strategies are zig-zag clearing, offset clearing, plunge roughing, rest-roughing.

Semi-finishing

This process begins with a roughed part that unevenly approximates the model and cuts to within a fixed offset distance from the model. The semi-finishing pass must leave a small amount of material so the tool can cut accurately while finishing, but not so little that the tool and material deflect instead of shearing. Common strategies are raster passes, waterline passes, constant step-over passes, pencil milling.

Finishing

Finishing involves a slow pass across the material in very fine steps to produce the finished part. In finishing, the step between one pass and another is minimal. Feed rates are low and spindle speeds are raised to produce an accurate surface.

Contour milling

In milling applications on hardware with five or more axes, a separate finishing process called contouring can be performed. Instead of stepping down in fine-grained increments to approximate a surface, the workpiece is rotated to make the cutting surfaces of the tool tangent to the ideal part features. This produces an excellent surface finish with high dimensional accuracy.

Software

The 10 largest CAM software products and companies, by end-user payments in year 2008 are, sorted alphabetically:

- CATIA from Dassault Systèmes
- Cimatron from Cimatron group
- Edgecam from Planit, formerly Pathtrace
- Mastercam from CNC Software
- NX from Siemens PLM Software
- Powermill from Delcam
- Pro/E from PTC
- Space-E/CAM from NDES
- Tebis from Tebis AG
- WorkNC from Sescoi

Other CAM products and companies are AlibreCAM, Alphacam, BobCAD, CAMWorks, Dolphin, ESPRIT, GCAM, GIBcam, GibbsCAM, GoElan, MazaCAM, MetaCAM, OneCNC, RhinoCAM, SolidCAM, SprutCAM, SUM3D, SurfCAM, T-FLEX, TopSolid, VisualMILL and VoluMill.