FOUNDATION - THE BROAD TOOL

Espen Brodin and Bjørn Ivar Danielsen

Arthur Andersen & Co.
Oslo
Norway

Most businesses of today are increasingly becoming dependent on computer systems in their daily operation. Businesses need to respond quickly and accurately to be able to compete in their rapidly changing environments.
It is important that organizations and businesses become aware of the gain that could be realized by applying Integrated Computer Aided Software Engineering to their systems design.
The major gains of using an I-CASE tool will be realized during the systems maintenance period by enabeling more timely, less costly and correct impementation of changes.
The incorporation of I-CASE in the organization requires not only conventional training of personell, but cultural changes. The cultural changes are needed to facilitate the life-long process that an I-CASE method includes.
Arthur Andersen & Co. has put in years of experience and development effort to reach the goal of a true I-CASE method. The product is named FOUNDATION and supports the entire life-cycle of systems design.

1 WHAT IS CASE?

1.1 An analogy

Historically, engineering has developed its products through a try and fail process. Pieces of machinery was often built before they could be tested in their environments, resulting in a long list of neccessary modifications or a total reassessment of needs and specifications. The result often was that new items had to be made or that the existing item had to be extensively modified.
The cost of repeating this process over and over pushed development of better planning systems, modeling and construction control. In today's computer-age the competitive industries all utilize some kind of automated tool to plan, design and implement complex products. One such tool is Computer Aided Design (CAD) systems that help car manufacturers design cars with thousands of parts. The manufacturers also use Computer Aided Manufacturing (CAM) to automate the actual manufacturing process.
The result is that the time from the idea of a new car emerges until the first car rolls off the assembly line is only a fraction of the time it took with the old approach to engineering.
The competitive environment has urged industries and their businesses to utilize this automated approach. Organizations that were late in applying new technology were forced into the automation process by a demand for better financial results.

The computer software industry is much younger than the traditional engineering industries. It does, however, share many of the properties and needs of traditional engineering.

1.2 CASE

The software industry, like traditional industries, realized the need for structured tools to help their product development and maintenance processes improve. In the seventies this lead to the automation of individual tasks of software development. This process was named Computer Aided Software Engineering (CASE).
As CASE developed, an increasing number of software developers and vendors realized the gain that could be made in utilizing this tool. The use of CASE spread, and made its way into software development environments, large and small, much like the spread of automation among traditional industries.

2 WHY CASE?

2.1 Computer systems - a critical part of the business

Since computer systems are increasingly becoming a critical part of the business environment, firms realize the need for making their systems more efficient. This includes traditional goals like increasing responsivness of

system alterations and improving the development of new systems.
It is critical that individual business manage to respond in ample time to take advantage of the ever changing business environments.

Today, businesses are completely dependent on their computer systems. It is not acceptable to have systems falter or misfunction. The demands for reliability, flexibility and dependency are increasing rapidly as organizations widen their use and complexity of computer-based solutions. As the dependency on computer systems increase, it becomes critical to the organizations that the systems work, and work right.

Until recently, too few businesses realized the need to invest in proper software developing tools. Maybe because software has been looked upon as something unrelated to traditional tangible products. The dependence on software has often been missed.
Today, software is critical to most business functions like planning, production, accounting, customer services, and a wealth of other administrative functions. It is therefore of great importance that time and effort is spent on synchronizing software strategies and development to both short and long run business-strategies.

The business environments get increasingly more competitive and the speed of change is also increasing. This stresses the need for businesses to be able to respond quickly and accurately in order to beat the competition in the marketplace. Automation of systems development is crucial for the ability to make timely changes.

2.2 Cost and time overruns

Even if business administrators have realized the need and criticality of software systems development, they are often discouraged by the numerous cost and time overruns before a system is completed. System developers are notoriously optimistic about their abilities to develop systems on time, on budget, and with the specified functionality. Too often, business administrators are faced with mediocre systems, slipping deadlines, cost overruns or project cancellations.

These discouraging situations can be avoided if proper attention and resources are allocated to systems development. But, it has to be done the right way. In many cases, systems development is undertaken without the help of a methodical, structured, standardized tool.
It is no longer enough to hire programming wizards to get the job done. Businesses of today operate far too complex and critical systems to leave anything to "quick and dirty" solutions. By applying a proper standardized method to systems development, large improvements can be made. System operators will have to adhere to a specified standard in their work. It is the only way to assure continuity, timeliness and control in a systems development environment; Past, present and in the future.

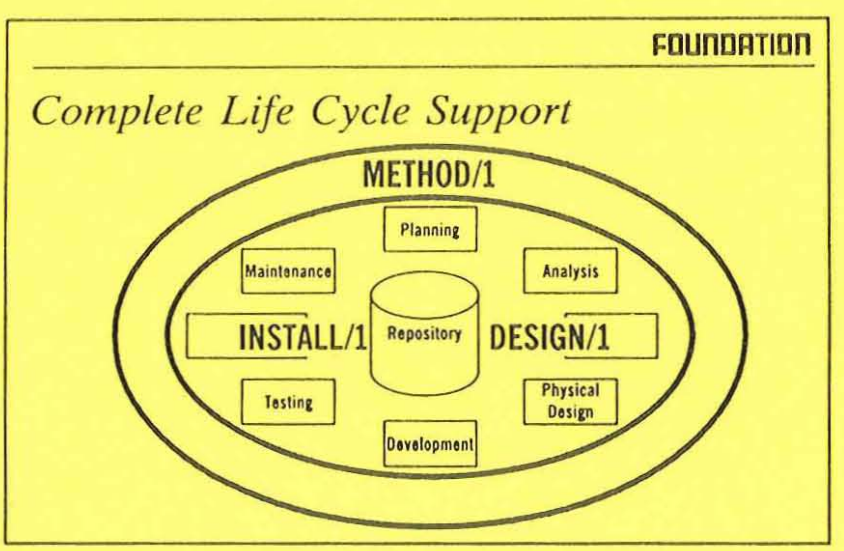
CASE is rapidly developing into encompassing more and more of the whole process of system development rather than just individual stages of the process.

3 INTEGRATED CASE

The most powerful property of some of today's CASE packages is the centralization of shared data in a repository. This enables all participants to access current data and eliminates the manual labor of moving data around. It can be referred to as a project data base which is accessible to several different tasks and users. This makes the process much more systematic since it operates in an environment rather than in individual partitions. The increased compatibility between automated tasks also enhance productivity and improves the learning curve.

By applying a continuous process, containing several tools, today's CASE tools should support systems design from the initial planning through development stages and all the way to maintenance of the finished product. The combination of a shared project-database with an extensive collection of tools that play together is called Integrated Computer Assisted Software Engineering (I-CASE). In I-CASE it should only be neccessary to enter the same piece of information once. I-CASE will help orchestrate the systems design and installation effort.

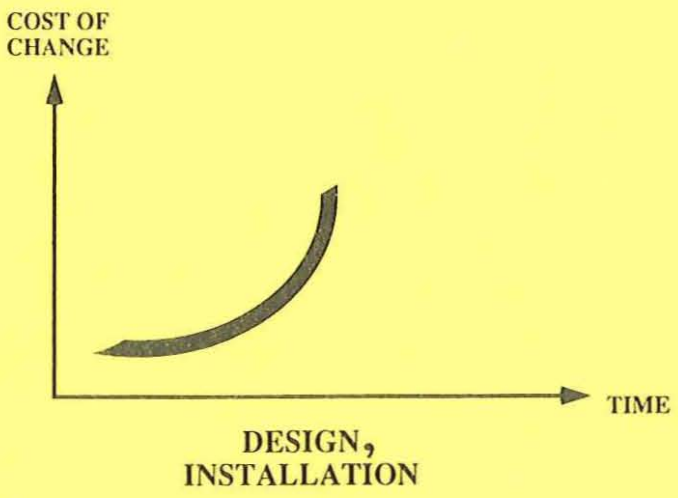
I-CASE assures a formal systematic and computerbased approach to systems development. It is a system for developing other systems, providing the system-crew with the opportunity for enhanced productivity, quality and compatibility with previous work. The I-CASE approach encourage group work, and makes it possible for relatively unexperienced people to produce useful results. It reduces the risk involved in big projects since it strongly encourages standards and proven techniques. The risk is also reduced by the fact that I-CASE lends itself to repeated use of previously completed work. I-CASE replaces the art of programming with the engineering science.



4 BENEFITS BY USING I-CASE

I-CASE encourages high quality and dependable systems

Dependability is improved by the repeated use of tested techniques and by the reuse of successful program modules from other systems.
Quality is improved by consitency in the implementation of the whole system, both in the user interface and the internal system structure. This makes the finished system easier and more predictable for the user. It also improves the ease of maintenance.
Quality is also enhanced by the use of prototyping, a process enabeling the end user a "look and feel" of the system before it has actually been implemented. Prototyping gives the end-users a better opportunity for input about their needs.



Affects accuracy of cost & time forecasts

The use of I-CASE increases the predictability of cost and time of implementing a system because of its highly automated structure and the integrated tools available for project management.

Overall management control

An I-CASE tool helps management control the system design effort by providing tools for assessing predictability, and quality assurance and control.

Orchestration of the system development process

I-CASE provides a tool for orchestration of the project environment, i.e. hardware, system software, and different tools to support the processes.

Sharing of project data

By arranging project data in a repository, data can be shared by system designers. This enhances standards and improves the communication process within the project environment.

Separation of unit testing and system test

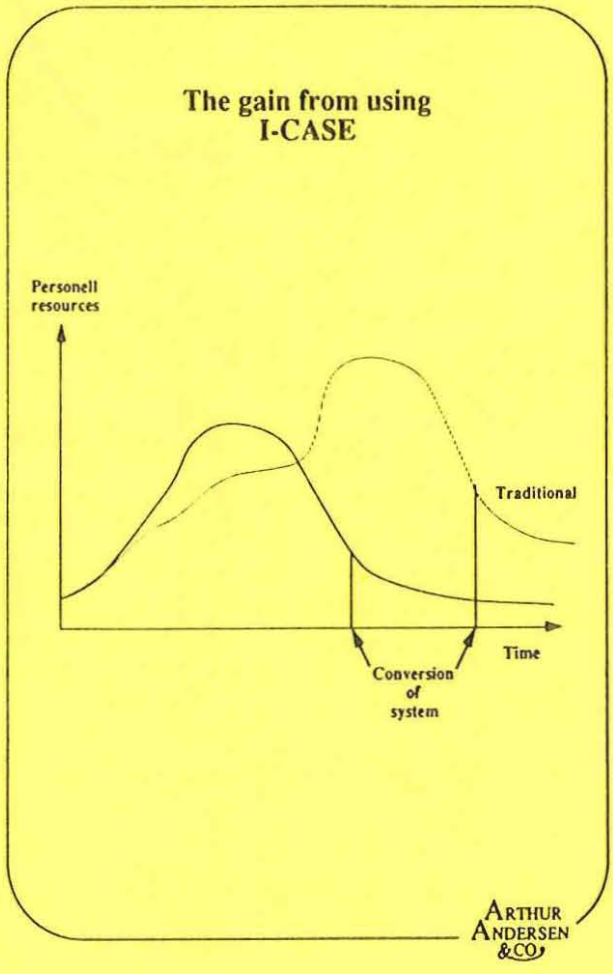
By enabeling design, coding and unit-testing to be performed on workstations in an I-CASE environment, mainframe resources will be utilized better. The system test will (and should be) performed in the production environment.

Larger gets easier

I-CASE enables organizations to undertake projects previously classified as too large or too risky and helps MIS departments produce higher quality systems, on-time and within budgets.

Cost efficiency

Experience tells us that development costs can be reduced with 15 to 50% by using I-CASE. These savings will usually be eaten up by the increased cost of training and the need for cultural changes in the host organization.
The large and meaningful savings occur during maintenance. By having a complete documentation of a system's development process, changes and modifications can be implemented swiftly. This will be the real savings of applying I-CASE in the organization.
One part of the savings will be the reduced resources needeed to implement changes. Another, and for most businesses the larger part, will be the savings (increased revenue) realized by constantly meeting business needs in the rapidly changing business environment.



5 FOUNDATION

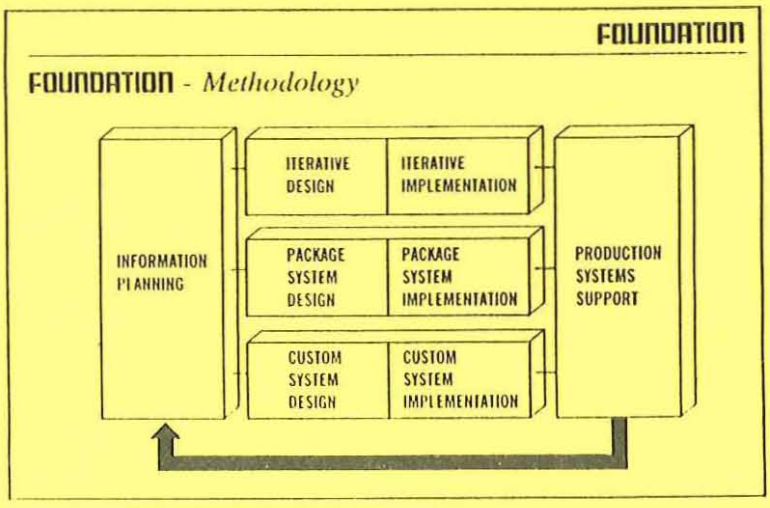
At Arthur Andersen & Co. we have put an extensive effort and more than 30 years of experience into developing I-CASE tools that will make businesses improve their efficiency and help them stay ahead of the competition.
We believe we have taken it further than our competitors by including an extensive variety of tools and making our system support the life of the business and not only that of a specific system.
The notion of life-long support necessitates the need for an evolutionary rather than a revolusionary approach in the development of systems design methods.

The key problem is to design a method for systems development that enables you to always produce a correct and predictable solution. The rigor of the solution is a very important factor when selecting systems development methods.
It is not enough to apply the limited "picture to code" terminology in order to facilitate a system to administer systems design efforts. There is actually no evidence that any vendor has achieved the "picture to code" for moderatly complex systems without significant manual intervention.
To be able to meet the requirements of tomorrow's need for automation, systems design organizations need to change their operative culture just like the traditional industries had to do when they applied automation.

Arthur Andersen & Co's contribution to I-CASE is named FOUNDATION. FOUNDATION consists of three integrated components that can be used individually or combined: METHOD/1 for project management, DESIGN/1 for planning and design, and INSTALL/1 for implementation and support. The combined use of these three tools enables a complete I-CASE that will significantly help productivity and predictability of systems development.

Because FOUNDATION operates on-line on a personal computer (IBM PC) it allows users to quickly access and review information. To promote even greater communication and productivity, FOUNDATION can operate in a Local Area Network (LAN).

Reality demands different implementation methods from a systems development method. FOUNDATION supports iterative design, package system design, and custom system design, and will incorporate expert system design and decision support systems design.



Among the functions and features of FOUNDATION are:

Full life-cycle methodology
The methodology provides a framework for developing complete systems. This is why FOUNDATION is process- and data-driven rather than data model-driven.

Project management
Tools for effective management of systems development efforts.

Diagramming tools
Used to enhance the conceptual integrity of systems design. They also improve the communication of the design concept among the users and the team.

Screen and report painters
Help the developer define layouts, validation and editing requirements.

Prototyping
Demonstrates proposed screen layouts and dialogs for end-users.

Checking and analysis
Checks design information as it is entered and analyzes the finished design for consistency and completeness.

Dictionary and reports
The dictionary contains information about the system entities, including screens, programs, etc. It was built using relational technology with a strong, flexible reporting facility. Once information is entered, it is made available to all users and consistently used by all components.

Code generation
Screen handling and dialog control logic are generated from dictionary resident, high-level, non-procedural specifications. Other features facilitate program generation with shell tailoring and statement generation. A complete file maintenance conversation may be generated from screen and database definitions.

Test data management
Manage test data as an asset by providing symbolic entrance and maintenance, version control and isolation of versions by programmer or team.

Production Systems support / configuration management
The dictionary provides the mechanism to analyze the impact of proposed maintenance changes and to control the movement of programs between production and development environments.

FOUNDATION-PAC

Services and support are often needed to help facilitate the adoption of FOUNDATION in an organization. FOUNDATION-PAC is a work-programmed set of services including additional training, help, and installation and adoption assistance. FOUNDATION-PAC will facilitate the adoption of FOUNDATION componenets and provide customized assistance to further the acceptance and effectiveness of the package.

6 METHOD/1

METHOD/1 contains a structured approach for managing systems development from planning and design through implementation and support. Because of the vast complexity and sophistication of today's information systems, this is a critical management problem. Advanced technologies are not a solution in itself. A framework for managing the systems development process is required.

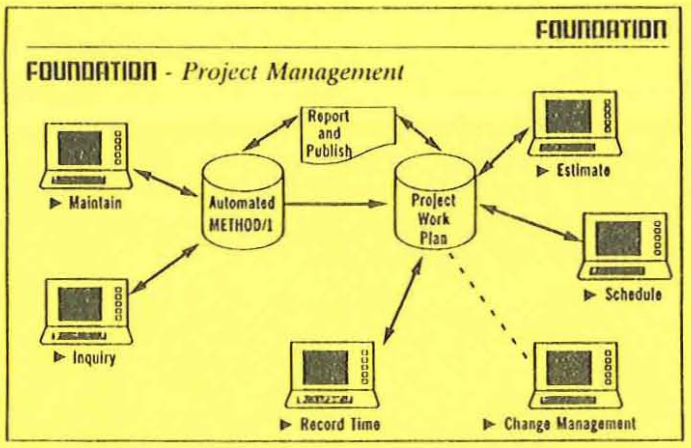
The method provides the framework to help optimize systems development and uses an approach for building information systems that meet long-range business needs and strategies.
By defining what to do and when to do it in each phase of the systems development life cycle, it provides a comprehensive, effective management approach for the systems development life cycle. Each phase produces and communicates valuable information to the next. Planning, development and maintenance are in this way managed as a process.

Tailoring to individual requirements

METHOD/1 can be tailored to reflect corporate culture and priorities in the host organization by incorporating a top-down management approach. The customization allows the MIS management to create a plan that communicates their objectives throughout the department.
The method can also be tailored to support one of three possible design approaches: custom systems, package systems or iterative development.

Facilitating project management

All procedures and functions in the systems development life cycle are incorporated within the methodology. This gives managers the information needed to estimate and schedule resources before any time is spent on design or programming.



Simplifying application development

By dividing necessary work to be done into successively smaller and more manageable units, the systems development is simplified. The units reflect the strategic objectives and goals of the organization.
The methodology defines all tasks required to build and implement an application, including objectives, work to be done, and deliverables.

Enhancing communication

METHOD/1 provides a means for management to communicate expectations to project teams giving everyone the same frame of reference. A reporting facility communicates work progress back to management.

Production systems support

The method helps anticipate potential changes, to prevent potential problems to become a reality. When revisions are required, they are scheduled directly into the planning phase for immediate modification.

Management checkpoints

At the completion of a phase within the method, management can examine completed work by help of a review capability. The review will help management to decide whether to proceed to the next phase of the project.

Quality assurance

METHOD/1 incorporates quality assurance reviews throughout the systems development process. This provides management with independent appraisals of completed work. The review are performed at specific checkpoints, allowing evaluation of the development process and completed work.

7 DESIGN/1

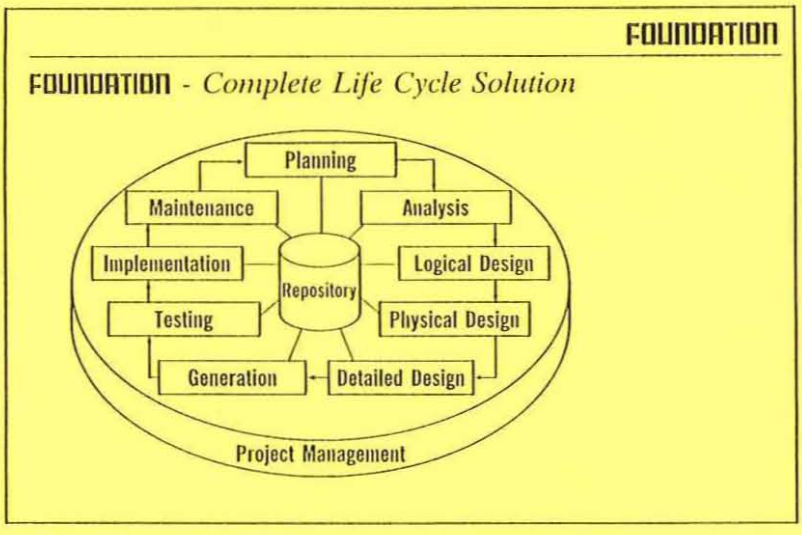
DESIGN/1 is an integrated workbench to supporting the analysis and design phase of the systems development life cycle. During the design phase it is important that systems analysts share specifications in order to develop consistent, high quality system design. Automation promotes the creation and management of such a shared repository of design information. Because design data is available to all members of the design team, more time can be spent on design development and less time on manual documentation.

DESIGN/1 consists of a variety of techniques needed in the design phase of systems development. By automating and integrating these techniques through word processing, modelling, screen and report design, data design, and prototyping tools, reusability is promoted and designers can share design specifications.

DESIGN/1, version 5.0, is the largest object oriented system in the world today.

Increasing communication, productivity and efficiency

A group-oriented approach is promoted by facilitating the sharing of data between system designers. This helps enhance productivity and minimizes the need for paperwork.
Documents that are stored in the shared repository are indexed and cross-referenced for each data element in the design.



Simplifying design development

A menu driven structure supports the simplification of the design software functions. Designers are able to define relationships between design objects, combine text and graphics within single objects, and move or copy from one object to another. On-line help and next-step instructions help accelerate the learning curve.

Improving design quality

DESIGN/1 incorporates verification and consistency facilities that review design data for completeness, accuracy, and consistency. These help designers detect, correct, and update all design documentation for improved design quality. Early detection of design errors and redundancies reduce total system time and resources required.

Repository providing a starting point for systems design

Within the repository, a number of pre-existing deliverables exist. These provide the basis for the project standards used to create design documents. Design data is automatically indexed and cross-referenced in the repository for every document. Images and diagrams are also storable.

Data modeller for creating entity relationship models

The data modeller diagrams conceptual data models and relational data structures. Diagramming enhances the conceptual integrety of systems design and improves communication between the project team and end-users. To model business functions, the data modeller uses pull-down menus, symbols, and icons to identify and define entity relationships.

Automated data flow diagrams connect system processes and data paths

The data flow diagrammer allows designers to create and modify data flow diagrams. A reporting facility checks models and documentation for consistency and completeness.

Automating the design of screens and reports

The screen/report design facility allows designers to create and modify screen and report layouts, using elements and literals stored in the design repository. Elements are selected and placed into the screen or report image. Cross-referencing between screen or report layouts and elements occurs automatically. The design facility also lets designers define display attributes for each element or literal.
The prototyping facility uses these screens, and the flow of data between them, to simulate conversations.

Prototyping

Prototyping allows designers and end-users to define a sequence of screens to prototype a conversation, using function keys to control screen flow. Prototyping allows designers to test and modify a system's conversation flow. End-users also become more involved in the design process, helping to ensure the quality of design.

Structure charts

Structure charts define the organization of a system's programs, modules, and functions from top-down. The structure chart editor allows your designers to create and modify structure charts. These are used to show the functional decomposition of program logic.

Data dictionary facilities

To help management review and audit design documentation, seven reporting facilities are available for summarizing the contents of the data design repository:

- o The element glossary produces a report of all data elements.
- o Index reports describe where elements and documents are used, and which elements are contained within each document.
- o Inventory reports list existing documents and their last revision date.
- o Cross-check reports identify duplicate documents.
- o Element verification reports highlight discrepancies between data elements and their attributes.
- o Segment reports provide detailed information on the characteristics and members of a data layout.
- o Consistency check reports identify exceptions to the consistency and completeness of a diagram's design documentation and data flow.

Code generation

DESIGN/1 has a COBOL copybook generation facility that translates design information into COBOL-formatted record descriptions. DESIGN/1 also has a CICS BS map generation facility to generate basic mapping support macros and COBOL copybooks from screen layouts.
Code will also be generated, from structure charts, to form shells.

Transferring of data to and from other software

DESIGN/1 provides designers with the ability to use relevant design documentation in the mainframe implementation phase. Information sharing increases the effectiveness of the design process with respect to design quality and productivity.

8 INSTALL/1

INSTALL/1 is an integrated set of facilities for the implementation and support phases of application development. It is a comprehensive set of mainframe-based software facilities which allows the creation and support of application systems.
INSTALL/1 is designed especially for the DB2 development environments, providing greater programming productivity control over the development process. In addition to providing code generation, INSTALL/1 automates development, streamlines testing, and enhances maintenance. It addresses all areas of application generation, including screen and conversation design, code generation, test data management, production systems support, data base administration, and technical support. Its application architecture enables developers to focus more on developing business solutions, and less on redundant design and implementation activities.

Integrating the development environment

INSTALL/1 has a DB2-based repository, which serves as a central location for all data elements, record layouts, copybooks, programs, files, and documentation. A published interface allows the population of the data repository with design data from DESIGN/1 or other design data dictionaries and design tools.
The data repository is key to the integration of FOUNDATION, ensuring referential integrity of data and consistency among all entities.

Promoting communication

The data repository is important in communicating design decisions between the project team members, and ensures that developers always have access to the most up-to-date development information.

Enhancing productivity

By generating 100% of the code required for basic application program components, INSTALL/1 simplifies coding. Developers no longer have to focus on redesigning, reimplementing, and retesting common program functions.

Increased control over the development process

INSTALL/1's repository-based reporting facility allows project managers to accurately asses the impact of a potential design change. Code generation, copybook generation, and screen modeling capabilities help enforce project standards to enhance ease of system use and maintenance. Security facilities control user access to INSTALL/1 functions, and data security protects the data repository and code tables. Configuration management isolates different repositories to control migration between environments.

A proven approach

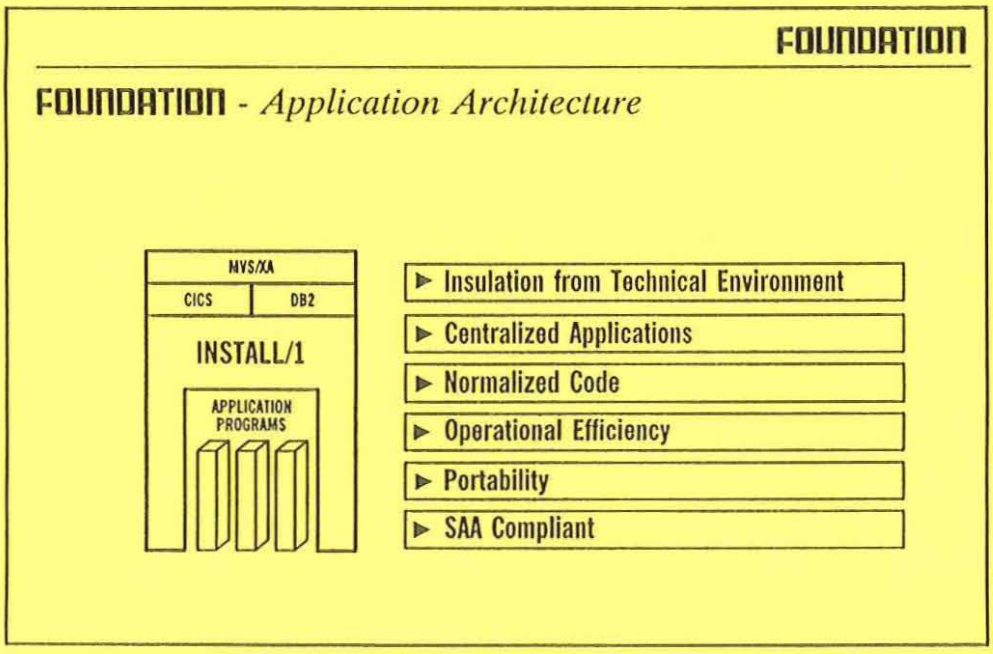
INSTALL/1 has been employed in more than 30 sites since the launch in April 1988. Its application architecture represents the newest generation of development platforms. The platform is based on AA&Co's experience on layered architecture which has been employed in hundreds of sites since the late seventies. The architecture allows developers to exploit the advantages of MVS/XA, DB2, CICS and COBOL II, and will be based on IBM's System Application Architecture (SAA).
In 1972, Arthur Andersen & Co. released the first commercial data dictionary and code generator.

The application architecture

The application architecture isolates the developers from the technical environment, allowing them to focus on developing business solutions. The programs generated by INSTALL/1 contain only application-specific code.
Major components of the application architecture include:

- The conversion control program detailes what work has been performed to date and identifies which program should be executed next.

- The message editing service interfaces the application program with the CICS Basic Mapping Support facility to perform initial validation and reformatting on screen data and output formatting operations.



The Application Architecture provides the following functions to developers, and allows them to distribute these functions to user applications:

- Security features help control access to functions and data for all components in the on-line environment.
- Suspend/resume allows developers to suspend work on one con versation, perform a task in another, and resume work in the first conversations.
- National language support allows developers to create and maintain screens in any language, providing geographic portability.

Quick and accurate definition of screens

The Screen Maintenance facility uses models and standard screen groups to ensure that screens adhere to established project standards. The design can also be uploaded from DESIGN/1.

Validation and editing rules

INSTALL/1 allows the assignment of validation rules for each field during screen definition rather than during programming. The rules can also be defined in DESIGN/1.

Automation of conversation definition

INSTALL/1 allows an analyst to define a conversation to establish the relationships between programs in the system. When DESIGN/1 is used with INSTALL/1, this data can simply be uploaded from the design repository.

Automated code generation

100% of the code can be generated for basic application program components such as basic COBOL requirements, environmental interfacing, advanced features, screen input and output, and dialog management.

Adding unique business logic

INSTALL/1 allows the creation and maintenance of COBOL code for unique business logic. Additional functions create code wich is not automatically generated. This increrases control and flexibility.

Plan administration

The Plan Administration Facility allows data administrators to bind and free plans based on information stored in the repository.

Reporting

The Reporting Facility assist in the analysis of different environments, system use, and entity relationships. Data administrators can review and perform impact analysis on data repository contents.

Testing

INSTALL/1 offers a number of tools to help streamline testing. The test Data Management facility allows each developer to create isolated versions of DB2 test data for different test sessions.

Re-engineering

The re-engineering facilitates the population of the INSTALL/1 repository with information about existing systems, which addresses the need for support of application maintenance.
Re-engineering also enables and simplifies the total remaking of a system by providing a better starting point.

9 INSTALL/1-PC

The PC version of INSTALL/1 allows applications to be designed, coded and unit tested on a workstation environment isolated from the target mainframe.
The coding is based on the INSTALL/1 layered architecture in conjunction with an animator, test data management and unit testing facilities.
By allowing design, coding and unit testing of the application to be done in the PC environment, only system test is left to be performed when the sytem is moved to the mainframe environment. It is of crucial importance that the system test is performed in the production environment.
The use of INSTALL/1-PC results in considerable savings of mainframe resources, allowing an increased resource allotment to be spent on system test, which improves the quality assurance and management without the need of acquiring additional resources.