Designation: E1340 – 05 (Reapproved 2010)

An American National Standard

Standard Guide for Rapid Prototyping of Information Systems[1]

This standard is issued under the fixed designation E1340; the number immediately following the designation indicates the year of original adoption or, in the case of revision, the year of last revision. A number in parentheses indicates the year of last reapproval. A superscript epsilon (ε) indicates an editorial change since the last revision or reapproval.

1. Scope

1.1 This guide covers a rapid prototyping method for developing information systems that is particularly relevant to systems for the healthcare sector. Intended readers of this guide are people who develop information systems, and students and teachers of system development methods.

1.2 Rapid prototyping is an approach to developing information systems which produce a working model more quickly than conventional approaches. Where conventional methods concentrate on preparing Requirements and design documents that describe the needed system, rapid prototyping methods concentrate on preparing a working prototype. Users and developers learn the functional requirements and an appropriate system design by interacting with a series of prototypes, each of which is rapidly produced from a starting framework or from an earlier version. A prototype can evolve into an operational system, it can serve as an exact behavioral specification of an operational system, or it can be used to explore the feasibility of a new idea or design which can be incorporated in a larger system. The method is rapid in preparing each version of the prototype, but the overall time required for system development may be more or less than the time required with conventional methods.

1.3 Rapid prototyping is most appropriate when the Requirements or design for a system are not well understood, or when experimentation is required to explore some aspect of system behavior. It is not appropriate in hazardous settings, or when the requirements are well understood.

1.4 The guide recommends use of prototyping tools, but it is not a standard for the tools themselves. It does not cover executable specification tools. Transforming a prototype that is used to clarify Requirements into an operational system is discussed briefly in Section 8 and in detail in other referenced standards (see 2.1).

1.5 *This standard does not purport to address all of the safety concerns, if any, associated with its use. It is the responsibility of the user of this standard to establish appropriate safety and health practices and determine the applicability of regulatory limitations prior to use.*

2. Referenced Documents

2.1 *ANSI Standards:*

ANSI/MIL-STD-1815A Ada Programming Language[2]
ANSI X3.9 Programming Language FORTRAN[2]
ANSI X3.159 Programming Language C[2]
ANSI/X11.1 MUMPS Programming Language[2]
ANSI/IEEE 610.12 Glossary of Software Engineering Terminology[2]
ANSI/IEEE 770 X3.97 Pascal Programming Language[2]
ANSI/IEEE 830 Recommended Practice for Software Requirement Specifications[3]
ANSI/IEEE 1016 Recommended Practice for Software Design Descriptions[3]
ANSI/IEEE 1058 Standard for Software Project Management Plans[3]
ANSI/IEEE 1059 Guide for Software Verification and Validation Plans[3]
ANSI/IEEE 1063 User Documentation for Computer Software[3]
ANSI/IEEE 1074 Software Life Cycle Processes[3]

2.2 *ISO Standards:*

IS 12207 Information Technology-Software Life Cycle Processes
IS 15288 System Life Cycle Processes
IS 15440 Guide for Life Cycle Processes
IS 11756 MUMPS Programming Language

3. Terminology

3.1 *Definitions—*

3.1.1 For definitions of terms relating to information systems, refer to ANDIP[4] and ANSI/IEEE 610.12.

[1] This guide is under the jurisdiction of ASTM Committee E31 on Healthcare Informatics and is the direct responsibility of Subcommittee E31.25 on Healthcare Data Management, Security, Confidentiality, and Privacy.

Current edition approved March 1, 2010. Published August 2010. Originally approved in 1990. Last previous edition approved in 2005 as E1340–05. DOI: 10.1520/E1340-05R10.

[2] Available from American National Standards Institute (ANSI), 25 W. 43rd St., 4th Floor, New York, NY 10036, http://www.ansi.org.

[3] Available from Institute of Electrical and Electronics Engineers, Inc. (IEEE), 445 Hoes Ln., P.O. Box 1331, Piscataway, NJ 08854-1331, http://www.ieee.org.

[4] American National Dictionary for Information Processing Systems, Information Processing Systems Technical Report X3/TR-1-82, Dow Jones-Irwin, Homewood, IL.

Copyright © ASTM International, 100 Barr Harbor Drive, PO Box C700, West Conshohocken, PA 19428-2959. United States

Documents
Document Sharing Hub

Full document details: ASTM E1340-05(2010)

3.1.2 *fourth generation language, n*—a high-level computer language that incorporates data structures and procedures for a specific problem domain.

3.1.3 *prototype, n*—an original or model from which a system is copied.

3.1.4 *prototype, v*—to create an original or model.

3.1.5 *prototyping, n*—the activities that create an original or model.

3.1.6 *rapid prototyping, n*—an iterative method for developing prototypes of components, subsystems, or complete computerized systems, in which the time between successive versions of the prototype is short.

3.1.7 *RP, n*—rapid prototyping.

3.1.8 *third generation language, n*—a procedural high-level computer language, such as COBOL, FORTRAN, or Pascal.

4. Significance and Use

4.1 Rapid Prototyping (RP) is a specific Life Cycle Model used to develop an information system which produces a working model of the system very quickly. The RP process shown in Fig. 1 has many similarities, and some differences from the conventional system (Waterfall Life Cycle Model) development process shown in Fig. 2. RP replaces the Requirements and Design processes of the conventional method with an iterative process of prototype refinement. Where the phases of the conventional method produce a set of documents that describe the system, RP produces a prototype. The prototype is tested and refined through several iterations, with intense interaction between system users and developers. RP is an experimental approach to system development which provides a learning device, the prototype, for users and developers. A prototype can be used as a tool for clarifying Requirements for the operational system, as a means of evaluating a design approach, or as a developing series of versions of the operational system. A prototype is sometimes used as an exact behavioral specification for an operational system which replaces it. Quality characteristics are often sacrificed during RP for the sake of rapid development and low cost; robustness, efficiency, generality, portability, and maintainability are commonly ignored but none of these aspects need to be neglected. However, documentation needed to use the system cannot be ignored but none of these aspects need to be neglected. A "Throwaway" prototype is used specifically to define Requirements which are used to implement a final system. An "Evolutionary" prototype is a prototypical system used for ongoing refinement of Requirements while operational versions at specific milestones are used in production settings.

4.1.1 *Rapid* in RP means that the time between successive versions of the prototype is relatively short. It should be short enough that (*1*) both users and developers can remember how each version relates to the previous one without written notes,

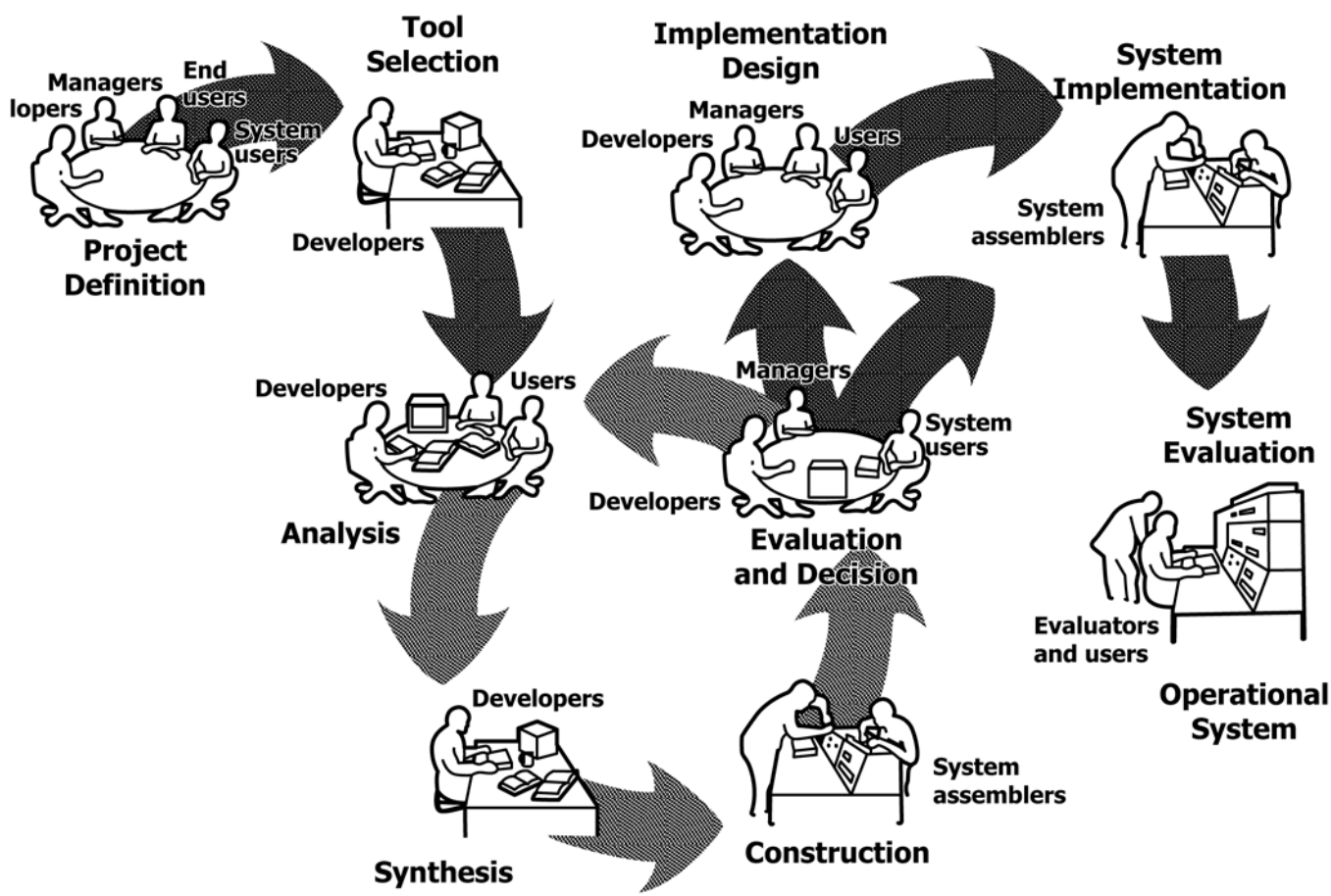

FIG. 1 Rapid Prototyping of An Information System

Documents
Document Sharing Hub
Full document details: ASTM E1340-05(2010)

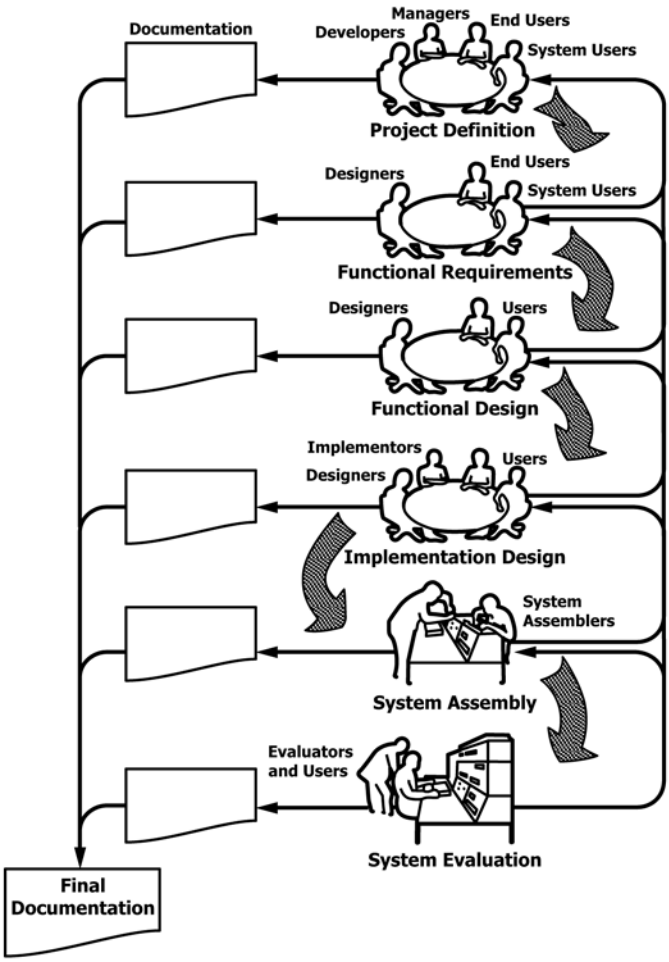

FIG. 2 Conventional Development of An Information System

(*2*) user requirements do not change significantly while a version is being developed, (*3*) the prototyping team will remain in the project through the RP phase, and (*4*) total time to develop the system is acceptable. (Expected project duration should be stated in the project management planning document. See Section 6 and ANSI/IEEE 1058 and ANSI/IEEE 1074.) A few days between versions is adequate and a few weeks may be acceptable. If the time needed to produce a new version is longer, then it may be necessary to produce that version using a conventional system development method with full documentation of requirements and design (see Appendix X3).

4.1.2 RP integrates analysis, design and construction, and defines Requirements during the process. It is especially appropriate for dealing with problems which are not well understood or are rapidly changing. The prototype focuses communication between users and developers.

4.2 For large systems, a RP approach can be used at a high level to explore the overall system architecture or feasibility. It can also be used to develop subsystems and components whose requirements are not fully understood (see Section 11). RP is especially well suited for developing user-system interfaces.

4.2.1 *What to Prototype*—The ill-structured system development problems that yield best to RP include:

4.2.1.1 Decision support systems in areas where the state of knowledge changes rapidly, for example, research or clinical practice,

4.2.1.2 Systems whose users need to access and organize data in ways not foreseen when the system is created, for example, strategic decision support and exploration of cognitive processes,

4.2.1.3 Systems that consist entirely of software,

4.2.1.4 Instructional or experimental systems, and

4.2.1.5 User-system interfaces.

4.2.2 *Ways to use RP*—Three ways that are widely used are (*1*) evolutionary, (*2*) experimental, and (*3*) build-and-replace. In evolutionary prototyping, the developers rapidly produce an initial version as a framework to learn user requirements, and then satisfy the requirements incrementally through a series of versions to produce the operational system. In experimental prototyping, the developers explore the feasibility of selected capabilities or components with a series of versions that serve to test concepts and designs. In build-and-replace prototyping, the developers assemble a series of versions to establish what the system should do and how it should do it, then the prototype is used as a behavioral specification for building the operational system. Build-and-replace is sometimes called throw-away prototyping, but the prototype should not be thrown away.

A RAPID PROTOTYPING METHOD

5. Introduction

5.1 The following sections describe a system development method that uses RP. It is based on, and shares some features with, the methods described in ANSI/IEEE 1074 and other ISO and IEEE standards which are listed in 2.1 (See IS 15440). Instead of producing documents that describe Requirements (ANSI/IEEE 830, Section 7), or Designs (ANSI/IEEE 1016) for the required system, this method produces a prototype of the system and refines it through an iterative process of analysis, synthesis, and evaluation.

6. Project Definition

6.1 In any system development project, whether done by RP or conventional means, it is important to have a definition of what is to be accomplished, when, where, why, by whom, and for about how much effort. It is also important to identify everyone who must be satisfied with the results, and especially everyone who will use the system. These things are determined in the preliminary discussions and negotiations about a project. A written statement of them is a Project Management Plan (PMP, see ANSI/IEEE 1058) document. A Project Management Plan is an agreement among everyone involved with a project. Agreement on project goals is essential for project success.

6.2 A Project Management Plan must be in writing. A written document is concrete and changes to it are explicit and documented. A written PMP does not drift like a verbal one. As new people become involved with the project, they can read the original document and can either become parties to it or renegotiate it.

AVA Documents
Document Sharing Hub

Full document details: ASTM E1340-05(2010)

6.3 The PMP document should be brief, and structured preferably no more than a few pages. If it exceeds several pages, provide a one page summary.

6.4 The PMP document should state what is to be prototyped and it should be specific about the goals of the project. If a prototype is to be developed to learn the true requirements for a system, the project definition should say so. If a prototype is to explore feasibility of a new idea or design, that should be stated. The PMP document should say whether the prototype is to evolve into an operational version, or is to be replaced by an operational version which is to be rebuilt from scratch. If this is not known at the start of the project, then the PMP should state the criteria for deciding.

6.5 The PMP document should briefly say what functions the prototype is to perform. It should clarify and limit the scope of the project, the prototype, and the system that is to be based on the prototype.

6.6 Everyone involved with the project should indicate agreement by signing the PMP document, or the one page summary of it.

7. Tool Selection

7.1 The PMP and the users' preliminary statements about what the system needs to do, guide the developers in selecting appropriate RP tools. The initial choice is crucial because it immediately constrains what can be accomplished. By selecting the right tools, the developers minimize the time and effort required for each RP cycle and maximize the likelihood of success. If the wrong tools are selected, then the project may be immediately doomed to failure. Most of the tools discussed in this section are software tools, that is, computer programs, but the principles also apply to hardware tools.

7.2 Good tools for RP share the following characteristics:

7.2.1 They help produce the prototype quickly,

7.2.2 They allow changes to the prototype to be made quickly,

7.2.3 They are general enough to permit more than one solution to most problems,

7.2.4 They encourage developers to try different approaches,

7.2.5 They are used interactively,

7.2.6 The person using them need not be a highly trained specialist,

7.2.7 They simulate real time, and

7.2.8 They are available for use now rather than proposed or promised for future use.

7.3 *RP languages:*

7.3.1 Many fourth generation languages are appropriate for RP. A good RP language uses the natural terminology of the problem domain, and it incorporates enough knowledge about programming to eliminate most of the need to write procedures. A good RP language should incorporate concepts which are in general use by the people who are to use it. It should provide high-level data constructs that are appropriate to the problem.

7.3.2 The user-system interface for a good RP language should incorporate features which allow development team members to use descriptive techniques that they use with human colleagues, that is, graphic, tabular, and other nonverbal techniques.

7.3.3 Tools that make the job easy for an inexperienced person may be tedious for a skilled system engineer. There should be a terse dialog mode for the experienced user who can abstract more features than the inexperienced user.

7.3.4 Good languages for RP are usually directly executed or interpreted in some fashion. The ideal language for an evolutionary prototype would fit the problem domain, would have an interpretive processor with good diagnostic facilities, and would be compilable (for efficiency of the operational version).

7.3.5 If the prototype is to evolve into the operational system, then the RP language should have reasonable efficiency. MUMPS (ANSI/X11.1, IS 11756), LISP, Pyton, Ruby, Perl, Scheme, PHP, and APL are widely used RP languages. These languages make RP easier than strongly typed languages like Pascal (ANSI/IEEE 770 X3.97) and Ada (ANSI/MIL-STD-1815A).

7.3.6 General purpose third generation programming languages like FORTRAN (ANSI X3.9) and C (ANSI X3.159) are not appropriate for RP unless powerful libraries can be used to eliminate the bulk of the programming effort. These languages may be useful if the prototype is to evolve into an operational system, but they are not appropriate for RP unless a large body of code is available and well-suited for reuse in the prototype. These languages may also be useful in an environment where the prototype can use several languages, and language selection is less critical.

7.4 RP is especially useful for developing user-system interfaces, even when other system functions are developed by a conventional approach. Mock-ups and simulators are powerful tools for exploring how people will use a system to accomplish their tasks. The interface between a system and its users can determine whether the system is accepted and successful. Tools for prototyping user-system interfaces should allow system developers to describe dialog attributes and style in plain language. These kinds of tools are useful for RP of user-system interfaces:

7.4.1 Screen managers, screen layout simulators, and forms managers, to rapidly produce user displays. These should incorporate graphics which are appropriate for the problem domain.

7.4.2 Dialog generator such as a transition diagram interpreter. This allows RP of a dialog that involves several different user displays.[5]

7.4.3 Dialog recording, to capture the conversation between user and system for later playback and analysis.

7.4.4 Report generator to rapidly produce printed output.

7.5 *There are two kinds of databases in RP*—(*1*) Data used by the system itself. Development of this database may be one of the goals of a RP project. (*2*) Data about the project

[5] Wasserman, A. I., and Shewmake, T., "Rapid Prototyping of Interactive Information Systems,'' Working Paper from ACM SIGSOFT Workshop on Rapid Prototyping. *ACM Software Engineering Notes*, Vol 7, No. 5, December 1982, pp. 171–180.

AVA Documents
Document Sharing Hub

Full document details: ASTM E1340-05(2010)