N94-11433

A RECENT CLEANROOM SUCCESS STORY: THE REDWING PROJECT

Philip A. Hausler

IBM Corporation
100 Lake Forest Blvd.
Gaithersburg, MD 20877
email: hausler@betasvm2.vnet.ibm.com
and
University of Maryland, Baltimore County
Department of Computer Science
Baltimore, MD 21228
email: hausler@umbc1.umbc.edu

ABSTRACT

Redwing is the largest completed Cleanroom software engineering project in IBM, both in terms of lines of code and project staffing. The product provides a decision-support facility that utilizes artificial intelligence (AI) technology for predicting and preventing complex operating problems in an MVS environment.

The project used the Cleanroom process for development, and realized a defect rate of 2.6 errors/KLOC, measured from first execution. This represents all errors ever found in all testing and installation at three field test sites. Development productivity was 486 LOC/PM, which included all development labor expended in design specification through completion of incremental testing. In short, the Redwing team produced a complex systems software product with an extraordinarily low error rate, while maintaining high productivity. All of this was accomplished by a project team using Cleanroom for the first time.

An "introductory implementation" of Cleanroom was defined and used on Redwing. This paper describes the quality and productivity results, the Redwing project, and how Cleanroom was implemented.

From Proceedings of Seventeenth Annual Software Engineering Workshop, Dec., 1992

CLEANROOM AND THE REDWING PROJECT

In the past few years, with the thrust in IBM on Market-Driven Quality and process improvement, the Cleanroom software engineering process has been broadly embraced. Cleanroom teams in IBM laboratories are developing software that exhibits dramatic quality results; consequently, they are meeting and often exceeding IBM's Market-Driven Quality objectives and are setting new standards and expectations of performance [Linger 1992].

Cleanroom software engineering is a methodology for developing ultra-high quality software. It defines a set of technical and management practices to be followed during software development. In the Cleanroom process, correctness is built in by the development team through formal specification, design and verification. Team correctness verification takes the place of unit testing and debugging, and software enters system testing directly, with no prior execution by the development team. All errors are tracked from first execution on, with no private unit testing permitted. The certification (test) team is responsible for certifying the quality of software through statistical usage testing that produces scientific estimates of product quality. Errors, if any, found in testing are returned to the development team for correction. If quality is not acceptable, the software is removed from testing and returned to the development team for rework and reverification.

The process of Cleanroom development and certification is carried out incrementally. Integration is top-down and continuous, with system functionality growing through addition of successive increments. When the final increment is complete, the system is complete.

The Redwing product is the largest completed Cleanroom project in IBM, both in terms of lines of code and project staffing. (Because the product has not yet reached the market, the internal name "Redwing" will be used in this paper.) The Redwing team was able to produce a complex systems software product with very high quality, while maintaining high productivity. Most impressively, this was the first development effort by this project team, and more notably, the first time the developers had used Cleanroom [Deck 1992]. How was this accomplished?

When the decision was first made to use Cleanroom, there were still many unanswered questions and concerns regarding its use on Redwing. Questions about the ability to train the personnel, apply it to the product domain, etc., were common. The concerns really fell into three primary areas: business, management and technical. The primary questions that needed to be addressed were:

Business:

> Could we introduce a new process and still maintain planned commitments for schedule and budget?

How would we use Cleanroom with a previously-defined checkpoint process, based on a sequential (waterfall) model?

Management:

How do we introduce a new methodology? Do performance and quality expectations differ in a Cleanroom development environment? Will functional management work?

Technical:

How do we train a new team? What consultation support will be required?

How do we define the incremental development plan?

Do we specify the system up-front or incrementally?

Can Cleanroom be extended to handle a rule-based system (AI expert system shell) and real-time tasking software in MVS?

This paper presents the Cleanroom quality and product results, describes the Redwing product and environment, explains how Cleanroom technology transfer and buy-in were accomplished, and defines Redwing's introductory implementation of Cleanroom. Finally, this paper summarizes the Redwing project with respect to the questions posed above.

CLEANROOM QUALITY RESULTS

After studying the Cleanroom methodology and examining available data from prior Cleanroom projects, the testing department projected defect rates for both the pre-ship and post-ship software. The projected defect rate for each incremental delivery was 4 defects/KLOC, plus 1 defect/KLOC for system testing, and 0.5 defects/KLOC after the product was shipped. Note that these numbers were significantly lower than those customarily used for comparable products, but it was felt that such aggressive goals should be set, even for a first-time Cleanroom team. Table 1 summarizes the development history of the product, with defect rates shown as measured from first execution. As a comparison, a projected defect number is shown using an average industrial defect rate of 30 defects/KLOC. The team produced the complex systems software with a defect rate an order of magnitude better than industry average, and a rate that even exceeded the projected rates of 4 and 1 defects/KLOC for incremental and system testing, respectively! At the time of this writing no post-ship customer data was available, however, three beta test installations at customer sites have reported no operational errors whatsoever.

Increment	KLOC	Defects @ 30/KLOC	Projected defects[1]	Actual software defects[2]	Defects/ KLOC
1	16	480	64	43	2.7
2	50	1500	200	41	0.8
3	41	1230	164	97	2.4
Subtotal	107	3210	428	181	1.7
System testing[3]			107	93	0.9
Total	107	3210	535	274	2.6

1 Increment testing defects projected @ 4/KLOC and system testing @ 1/KLOC.
2 Measured from first-ever execution.
3 Includes system, performance and field testing.

Table 1. Defect rates measured from first execution for Redwing.

The projected productivity was based on an assumption that it would improve with each increment. The productivity rates were projected as 300 LOC/person month for increment 1, 350 for increment 2, and 400 for the final increment. Table 2 shows the actual productivity rates for the three increments. The productivity rates are based on the total LOC developed, divided by the person months accumulated for design specification through testing of the final increment. The person months only include development resource, not testing; thus, they do not include the labor months incurred by the testers. Redwing development achieved very competitive productivity rates, exceeding the projected rates by 36% overall.

This dramatic increase in productivity was a significant factor in enabling Redwing to meet its schedule. The original code size estimate was 72 KLOC, but actual code size was significantly larger (107 KLOC), primarily due to unexpected growth in the workstation software (from 10 to 42 KLOC). The growth resulted from an unfamiliarity with Presentation Manager development and unanticipated requirements. Thus, while actual productivity was a 36% improvement over the projected rate, actual code size was 49% larger than planned. The improved productivity enabled the development team to stay on course during development.

Increment	KLOC	Projected Productivity LOC/PM	Actual[1] Productivity LOC/PM	Delta %
1	16	300	400	+33
2	50	350	500	+43
3	41	400	513	+28
Average		358	486	+36

1 Productivity = LOC/development person months from design specification through incremental testing.

Table 2. Productivity rates for three increments.

AN OVERVIEW OF THE REDWING PRODUCT

Redwing provides a decision-support facility that utilizes Artificial Intelligence (AI) technology for predicting and preventing complex operating problems in an MVS environment. Redwing is primarily a host-based product that runs in a NetView environment on MVS with interfaces to several other IBM program products. A workstation component runs on a PS/2 in the OS2 environment, providing the user interface needed for the management of business policy.

The architecture is designed to detect and handle problems through a series of independently running tasks that

1. monitor the operating environment for potential problems,
2. determine the nature and severity of potential problems,
3. recommend the best course of action based on the original problem and customer

policy, and
4. take action based on the best recommendation to resolve the problem.

The development environment was very complex as it required expertise in MVS and its subsystems, expert systems technology, real-time tasking, message passing, and windows-based programming for the workstation. Redwing was implemented using PL/I, TIRS, PL/X (an internal IBM system language), assembler, JCL and REXX for the host software, and C and Presentation Manager for the workstation code. To further complicate the environment, two major dependencies existed on IBM system management products that were developed by other labs.

Table 3 contains a breakdown of the various languages for implementation and the relative size of each.

Development Language	% of Total	Language Type
Assembler & PL/X	2	Low-level
C & Presentation Manager	40	High-level
JCL & REXX	3	Control code
PL/I	50	High-level
TIRS	5	AI: rule-based

Table 3. Implementation languages for Redwing.

Project Organization and Skills

The project commenced in July, 1989, with the first year and a half spent gathering, validating and documenting requirements for the product. A half-dozen customers participated in this activity. During the initial stages of the project, staffing for the development teams took place.

Redwing was a second-line organization with four first-line departments. The four departments were defined as follows:

1. Architecture: responsible for defining and documenting requirements;
2. Structure software development: responsible for all platform and workstation software;
3. Application software development: responsible for two applications running on platform; and
4. Testing: responsible for product testing.

In addition to this organization, there were several support organizations enlisted throughout product development that provided market development, quality assurance, information development, usability, business and legal services.

As this was a new project, the Redwing team had never developed software together. Experience of team members spanned the full gamut, from programmer retrainees to senior level programmers with twenty-five years of development experience. With respect to the product domain, there was considerable experience in application development and in AI, but very little with MVS and system programming. As it turned out, AI skills were utilized about 10% of the time during development, while MVS and system programming skills were needed 90% of the time.

As mentioned earlier, this was the first use of Cleanroom for all participants, with the exception of one development manager and two developers. The project had an average staff of fifty people throughout development. Consequently, extensive education and training were required to utilize the methodology. The overall project schedule, including an end-date, had been established in late 1989, prior to the decision to use Cleanroom. Given the schedule and amalgam of skills and experience levels, Cleanroom was first met with a fair share of raised eyebrows and healthy skepticism. This team had to simultaneously grapple with three important problems: unfamiliarity with team members since this was the first time they worked together, lack of experience in the subject domain and the introduction of a new (Cleanroom) methodology.

SELLING CLEANROOM TO THE REDWING PROJECT

In early 1990, a Quality Improvement Team (QIT) was formed on the project with the objective to make a recommendation on a development process to Redwing management. The QIT was comprised of members from all four first-line departments. It met weekly for about two months and reviewed several different models, such as waterfall, spiral, checkpoint, and Cleanroom.

A final position paper was written and presented to management with the recommendation to use Cleanroom. The reasons given for this choice were:

- The emphasis on spending more time up front in specification and design seemed critical in the new domain of the Redwing product.

- Since one of the objectives of Redwing was to maintain high availability of an MVS system, high reliability of Redwing was mandatory. Correctness verification was needed to produce ultra-high quality software.
- The Cleanroom approach would foster strong teamwork and enable others to become experts with each team member's code.
- Redwing personnel with prior Cleanroom experience could serve as project consultants.
- The incremental approach would allow for early testing, refinement of requirements, and provide definitive management checkpoints for progress.
- There was a general belief that Cleanroom would be successful.

At the same time the QIT was formulating its proposal, I made two briefings to the Redwing management team. The first was a general overview of Cleanroom, including the key technologies, past results and related management practices. Redwing management seemed generally interested and enthusiastic. Soon after, I gave a second presentation that outlined more concretely how Cleanroom should be applied to Redwing and what changes and differences management could expect over traditional development. In many ways, gaining management support was the most important activity that got Cleanroom accepted. By the time the QIT recommendation was made, both the management team and the technical team (as represented by the QIT) were in step.

The next step was to roll out Cleanroom to the remaining project members. A series of half-day Overview classes was scheduled, with attendance at one of the sessions mandatory. Following the Overview, a three-day Design and Verification class was given, with mandatory attendance for all project members, including management. Select technical personnel also participated in three-day classes for Specification and Certification (testing).

The Process Working Group

To further define and document the use of Cleanroom in the development process, a Process Working Group (PWG) was formed with the objective to define and document the Redwing development process using Cleanroom, establish and maintain project procedures, standards and conventions, establish and maintain a measurement and improvement subprocess, and provide a formal mechanism to resolve process issues and make improvements. Each major project functional area, such as architecture, host development, workstation development, test, configuration management and quality assurance, had representation on the PWG. This composition ensured each department had at least two members on the team.

The major achievement of the PWG was documenting the development process and standards, in the Redwing Project Development Procedures book [Redwing 1990]. The PWG documented a comprehensive set of procedures and standards for an integrated, Cleanroom-based software development process. More specifically, the Project Development Procedures book defined how Cleanroom would be used, software engineering standards and conventions, how to conduct reviews and design verifications, and the formal change process. This document and its subsequent use by the team was critical for employee ownership of the process and buy-in. Changes to the process had to be approved by the PWG and management. During the development of Redwing, several revisions to the process were the result of suggestions by project members during meetings held to improve the development process.

The Redwing managers required that all developers certify that they had read and would adhere to the Project Development Procedures. Not only did the document define a baseline of the process for team members, but it served as an effective guide for new employees as they were added to the project and to outside support personnel, such as vendors and quality assurance. Adherence to the process also was integrated into performance plans for everyone.

INTEGRATING THE CLEANROOM PROCESS ON REDWING

The PWG, primarily through the Project Development Procedures document, served a key role in integrating Cleanroom into the technical development process; however, using Cleanroom also required integration with an established set of business and planning checkpoints. The Cleanroom lifecycle, as depicted in figure 1, consists of a pipeline of accumulating functional increments, where each increment consists of design, verification, implementation and testing for a specific set of functions. As a result, in Cleanroom there are unique planning parameters and technical and business checkpoints required for project management. Pre-established project management guidelines existed in IBM based on the traditional waterfall model, where the activities of specification, design, coding and testing occur sequentially.

A phased checkpoint process based on the waterfall approach was expected to be used for project planning and accounting. Since Cleanroom development does not adhere to the traditional model, changes to the checkpoint model were required. This was primarily accomplished by working with the various groups responsible for the business planning and financial support teams. They attended Overview classes first, followed by meetings with Redwing management that served to explain what Cleanroom was and how it differed from the traditional model. The support groups, such as financial planning, quality assurance and legal, were very impressed with this methodology, and easily integrated the necessary changes into their pre-established processes. Most commented favorably that Cleanroom, due to the incremental approach, would provide early insight into project status. And, since this occurred early and frequently, would allow for appropriate contingency planning to maintain schedules and budgets.

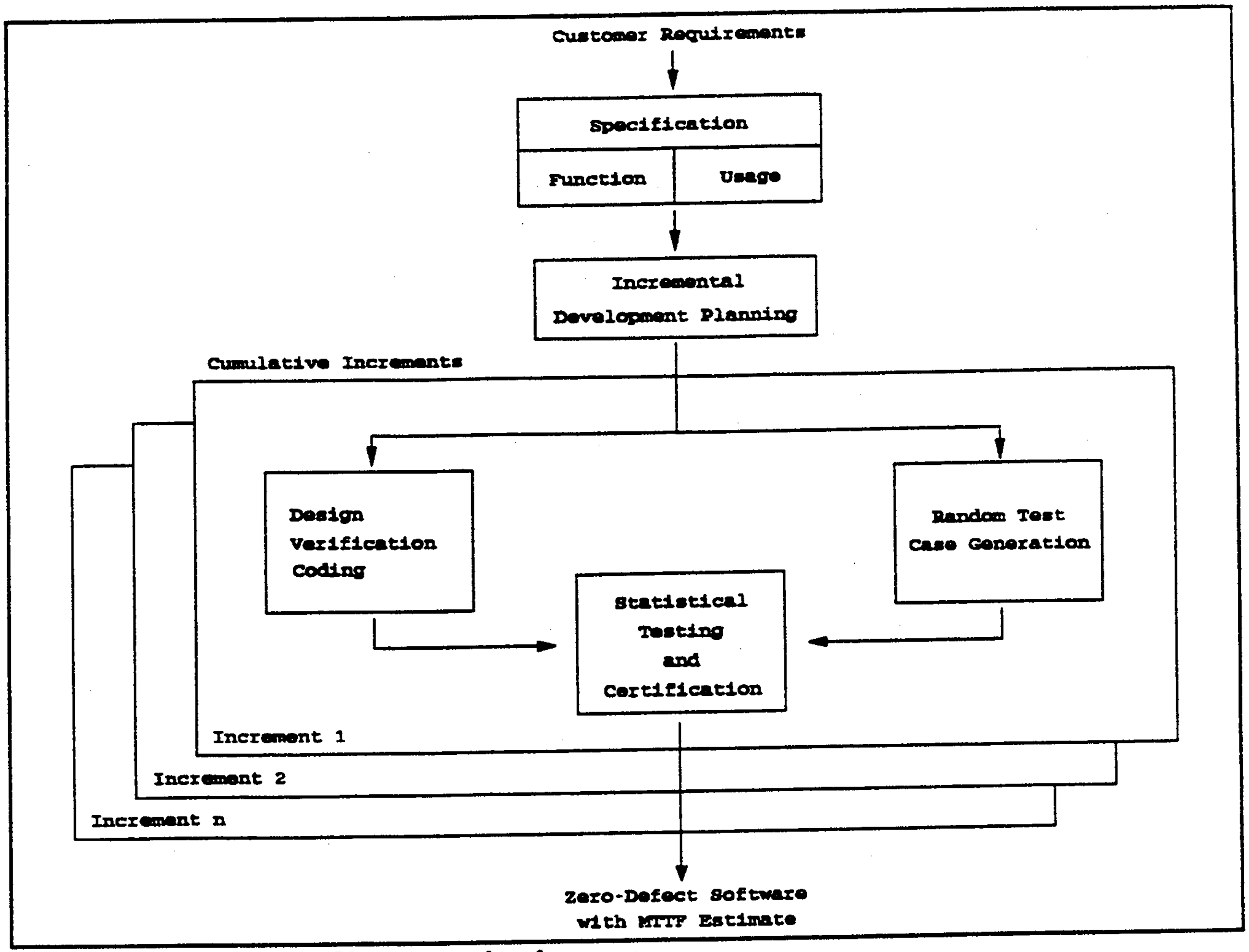
Customer Requirements
Specification
Function
Usage
Incremental Development Planning
Cumulative Increments
Design Verification Coding
Random Test Case Generation
Statistical Testing and Certification
Increment 1
Increment 2
Increment n
Zero-Defect Software with MTTF Estimate

Figure 1. The standard Cleanroom development process.

Defining An Introductory Implementation Of Cleanroom

Product requirements were developed and documented over a period of 15 months, from July, 1989 through October, 1990. The decision to use Cleanroom was made in the second quarter of 1990, and the Process Working Group was launched in June, 1990, with the first version of the Project Development Procedures Document completed in October, 1990. The Design and Verification class was held in October, 1990, and development commenced in late November, 1990.

Due to the project schedules, size of the organization, prior Cleanroom experience, and the amount of training time allotted, it was decided by the management and technical team to pursue a phased implementation of Cleanroom Software Engineering. In a phased implementation, increasing control over the software process can be achieved through stepwise introduction of Cleanroom practices [Trammell 1992].

Cleanroom development traditionally starts with a rigorous (formal) specification of the intended system behavior in all circumstances of use. It is recommended that the box structure specification technology be used for this. The decision to use Cleanroom was made fairly late in the cycle, after requirements analysis and system specification had occurred. By the third-quarter of 1990, the Product Functional Specification (PFS) document was almost completed. This document is required in IBM for program products, but is not an adequate replacement for a Cleanroom specification. Typically, it contains only a subset of the information required in a Cleanroom specification. The Redwing team decided to complete the PFS document, and modify the standard Cleanroom model in order to produce a more formal specification, following the less formal PFS. Consequently, as part of the introductory implementation, it was decided that Redwing would use the PFS document as the initial specification, and would add a formal specification phase in each increment. Figure 1 showed the recommended Cleanroom approach, where system specification occurs first, followed by the incremental design, verification, coding and testing. Redwing defined a model, as seen in figure 2, that used informal specification first, followed by incremental (formal) design specification, design, verification, coding and testing. The formal specification used crisp English descriptions in conjunction with intended functions to specify the external behavior of the increments.

The Redwing testing manager and testing team leaders attended a three-day class on Cleanroom Certification in October, 1990. While it was agreed that statistical testing would be a very effective way to test Redwing, it was not felt the test team could learn and apply the methodology in time for the first increment. Of greatest concern was the test team's ability to define the usage specification, normally done early in development at functional specification time. As it turned out, though statistical testing was not employed for the entire product, it was conducted for a significant subset of the product, namely, the workstation component that accounted for approximately 40% of the total product code.

In summary, the Redwing project team decided on an "introductory implementation" that included incremental development based on function requirements, incremental specification, design with intended functions, team verification of correctness, delivery of each increment to independent testers for first execution, and a measurement of process control against pre-established standards. The key elements of Redwing's introductory implementation are listed in Table 4.

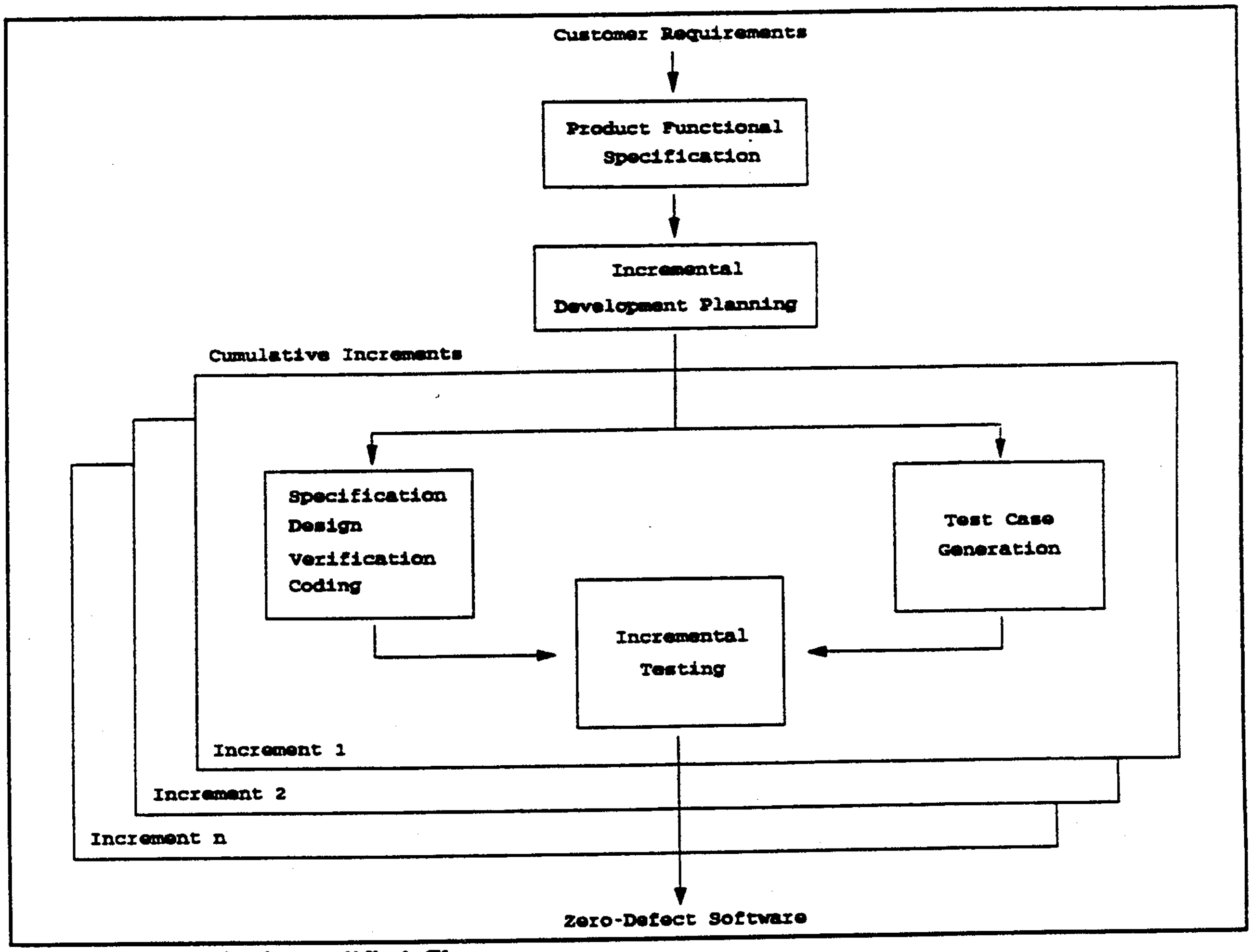

Figure 2. Redwing's modified Cleanroom process.

DEFINING THE INCREMENTAL DEVELOPMENT PLAN

Following the specification phase, an incremental development plan was developed that defined the functional content and number of increments, software development schedule and resources. Though the end-date for the project had essentially been established earlier, there was still substantial flexibility in defining the increments, team composition, and function for each increment. The overall development period, including testing, was approximately twelve months. It was decided that Redwing would build a plan where each increment was about two to three months in duration. The first increment would contain the least function in order to get everyone acclimated with the new process and development environment. Consequently, three increments were defined for Redwing. The first increment lasted two and a half months, and the second and third were approximately three and a half months each.

Cleanroom Technique	Used (Yes/ No/Partial)	Comment
Incremental planning	Yes	3 increments
Formal specification	Yes	PFS + formal incremental specification
Formal design	Yes	Used translatable PDL w/ intended functions
Verification	Yes	Strict adherence to rules
Statistical Testing	Partial	Incremental testing w/ statistical testing for workstation only

Table 4. An introductory implementation of Cleanroom for Redwing.

The project team reviewed historical productivity and defect rates for comparable commercial applications, and adjusted them for Cleanroom based on prior Cleanroom data, personal experience and confidence. The incremental development plan was primarily developed by the project technical leaders with review and approval by the technical team. When it was completed, it was presented to management and approved with only minor modifications.

Eight principal functional components were defined for Redwing, with each component assigned to a team of one to five developers. In addition, each team also had an architect and tester assigned to it. Team membership remained stable throughout development of all three increments. This ensured continuity and grew the expertise and capability of each team. A functional management approach was used for the teams because each team consisted of people from different departments. Since each team had a designated team leader, management ownership was assigned based on the team leader. Thus, a manager was responsible for all teams led by one of his department members. This scheme worked nicely but required daily communication between managers, usually in the form of morning status meetings in which schedules, plans, resources and performance were addressed.

USING CLEANROOM ON REDWING

Following the incremental planning phase, development commenced for increment 1. It became immediately obvious that the developers did not have a keen understanding of what the entry criteria were for design verification. Most understood how to perform verification, but underestimated the level of rigor and precision needed. Typical of most designs in increment 1 was that intended behavior was specified precisely for "normal" or "steady-state" processing, but failed to capture the intended function for error conditions, exception processing, and less likely scenarios.

To ameliorate this problem, it was decided that a demonstration review of an actual increment one design should be held as early as possible. A senior level programmer was asked to be the "guinea pig." When his design was "ready," his review team conducted a formal verification review with the rest of the Redwing organization in attendance as observers. Everyone in attendance had a copy of the material and followed along with the review team. The first review lasted about three hours, with the outcome that the design did not pass the verification review. This proved to be an invaluable teaching tool for the project team. Most were surprised that the design did not pass, and even more surprised at the changes required in order to make it verifiable. This turned out to be the most effective means to teach the team what was actually expected in a Cleanroom review, and certainly saved an immeasurable amount of time and frustration for the others. Since increment 1 was relatively small and straightforward, the team was able to learn how to correctly apply Cleanroom and still make the first delivery date.

Cleanroom Facilitators

The Redwing project did benefit from the project members with prior Cleanroom experience. They served dual roles as project team members and as Cleanroom methodology consultants. They were teachers and trainers of the methodology, providing guidance on how to write verifiable designs and conduct effective verification reviews. Equally important was the encouragement they gave and confidence they instilled in their peers through example and coaching. During the increment 1 reviews, one of the experts was present at every single review to ensure the methodology was being adhered to, especially with respect to the correctness verification conditions. During increments 2 and 3, others stepped forward to be Cleanroom experts as they learned the methodology. This core group of five to six facilitator served a key role for acceptance, application and improvement of Cleanroom.

Cleanroom Reviews

In order to ensure that reviews were effectve, that is, that entry criteria for reviews were met, advance preparation was conducted and the correctness conditions of each control structure were examined. The Redwing Project Development Procedures document outlined formal roles for review team members. A moderator was assigned who was usually one of the Cleanroom facilitator (experts) mentioned above. The moderator ensured that reviews were conducted properly, all issues recorded and all changes reverified. The author (writer of the design being verified) led the team through the review. Also present were a key reviewer, typically the component team leader who had a broad understanding of the component function, and other reviewers, typically other component team members or others whose code interfaced to that being reviewed. It was required that review materials be distributed to all participants at least 48 hours prior to the meeting; all reviewers were expected to have read the materials beforehand.

Early Results and Observations

Key to applying Cleanroom to an organization of this size was the small team approach used to partition the workload. Also, with multiple departments and managers, reliance on formal means of communication, especially for verification review results, status, schedules, etc., was essential. Weekly project status meetings were held and attended by management and team leaders. In addition, monthly project reviews were conducted with the project team, market development, quality assurance and finance.

Redwing's aggressive schedules did not allow for a pilot development effort in which the team could practice and polish their Cleanroom skills. As a result, it required one full increment for most developers to become proficient in Cleanroom.

At the beginning of the project until delivery and testing of increment 1, many developers and testers were somewhat skeptical about the effectiveness of Cleanroom. The real turnaround occurred after increment 1 was delivered and testing found so few errors. In fact, during increment 1 testing, several testers were upset and worried when they could not find any errors after running their test cases; ironically, so were the developers. But this soon changed for everyone -- defects quickly became the exception, not the rule.

Several important benefits, primarily derived from the team concept and correctness reviews, were experienced by the Redwing development team [Drew 1992].

- The teamwork was very strong and, consequently, verification reviews were "extremely positive experiences."
- Cleanroom reviews reduced the "lone eagle syndrome," where a programmer is hesitant to ask for help.
- Cleanroom helped produce "egoless code."
- No one on the team was indispensable; the entire team was familiar and capable with each other's components.
- Maintenance was less dependent upon code author.
- Cleanroom reviews provided an excellent forum for learning.

LESSONS LEARNED

As stated earlier, the challenges of using a new team in an unfamiliar environment were great: moreover, schedules and resources were extremely tight. Yet despite these odds, a new methodology was introduced, taught and implemented with tremendous success.

The primary success factors in this implementation of Cleanroom were:

- strong management support throught test,
- technical team participation in methodology selection (QIT) and implementation (PWG),
- joint management and technical team development of incremental development plan,
- adherence to correctness verification rules,
- demonstration review during increment 1,
- management attendance at training classes,
- three "consultants" on Redwing with prior Cleanroom experience,
- functional management scheme for teams of teams, and
- use of an introductory (phased) implementation of Cleanroom.

Recall, at the start of the Redwing project, several important questions needed to be answered.

Business

Q: Could we introduce a new process and still maintain planned commitments for schedule and budget?
A: Yes, a new process was introduced early during Redwing, and the team was able to accommodate the required start-up costs and still build the product successfully.

Q: How would we use Cleanroom with a previously-defined checkpoint process, based on a sequential (waterfall) model?
A: It was necessary to modify the checkpoint process to account for the incremental Cleanroom lifecycle, and to train the necessary groups that would be affected by the changes.

Management

Q: How do we introduce a new methodology?
A: First, management understanding and support must be gained, followed by technical buy-in and ownership. Also, the appropriate resources for education, training and consultation must be provided.

Q: Do performance and quality expectations differ in a Cleanroom development environment?
A: Yes, zero-defect software is the performance goal, not the hope. Developers are expected to produce zero- or near zero-defect software prior to first execution under test. All major milestones are based on the incremental approach. Completion of correctness verification

is a major milestone during an increment.

Q: Will functional management work?
A: Most definitely. The key is strong technical management of the team (or teams of teams for larger projects).

Technical

Q: How do we train a new team? What consultation support will be required?
A: Formal training in Cleanroom is essential. The entire project team should be trained together, including management. An experienced practitioner of Cleanroom (consultant) to support the project is essential.

Q: How do we define the incremental development plan?
A: The incremental plan should be built following software specification by the technical leaders and management. Much care should be given in defining the number of increments, the function and code of each increment, and the development teams.

Q: Do we specify the system up-front or incrementally?
A: Ideally, the system behavior should be specified up-front, but this is not always practical or feasible. An incremental specification can be used but needs to be developed carefully.

Q: Can Cleanroom be extended to handle a rule-based system (AI expert system shell) and real-time tasking software in MVS?
A: Extensions to Cleanroom were defined for both on Redwing. For the former, a rule-based design language was defined with a modification for verification. In the latter, one tasking design model was defined by the technical leaders and used by everyone on the project. Certain simplifications and assumptions to the tasking model were made; this enabled everyone to conduct design and verification with the same model.

REFERENCES

[Linger 1992] Linger, R.C. and Hausler, P.A., "The Journey to Zero Defects with Cleanroom Software Engineering," *Creativity!*, Vol. II, No. 3, September 1992.

[Deck 1992] Deck, M.D., Hausler, P. and Linger, R.C., "Recent Experience with Cleanroom Software Engineering," 2nd International IBM Conference on Software Engineering, May 1992.

[Redwing 1990] "Redwing Release 1 Version 1 Project Development Procedures," Document No. Redwing-0032, November, 1990.

[Trammell 1992] Trammell, C.J., Hausler, P.A. and Galbraith, C.E., "Incremental Implementation of Cleanroom Practices," *Proceedings of the 25th Hawaii International Conference on System*

Sciences, January 1992.

[Drew 1992] Drew, C.F. and Wade, G.P , "A Developer and Tester Report on Their Cleanroom Experiences," *IBM Software Engineering ITL*, March 1992.

A Recent Cleanroom Success Story: The Redwing Project

Philip A. Hausler

Cleanroom Software Technology Center
IBM Corporation
Gaithersburg, MD
email: hausler@betasvm2.vnet.ibm.com
and
Department of Computer Science
University of Maryland, Baltimore County
Baltimore, MD 21228
email: hausler@umbc1.umbc.edu

December 2, 1992

Agenda

- Introduction
- Cleanroom results
- Selling the process
- Implementing the process
- Summary

Introduction 1.1

SECTION 1.

Introduction

Introduction to Redwing 1.2

- Product

  Redwing provides a decision-support facility that utilizes artificial intelligence technology for predicting and preventing complex operating problems in an MVS environment.

- Project organization
  - 2nd-line organization with four 1st-line departments

    1 Architecture

    2 Development

    1 Test
  - Approximately 50 people

Development Environment 1.3

- Workstation Policy component under OS2
- Host Outage Avoidance under MVS
  - Interfaces to two Systems Management products developed by other labs
  - Real-time tasking
  - Rule-based processing

Redwing Development Languages 1.4

DEVELOPMENT LANGUAGE	% OF TOTAL	LANGUAGE TYPE
Assembler & PL/X	2	Low-level
C & Presentation Manager	40	High-level
JCL & REXX	3	Control code
PL/I	50	High-level
TIRS	5	AI: rule-based

Development Issues

1. New development team.

2. New product requiring extensive MVS and system programming skills.

3. New methodology (Cleanroom).

Development Highlights

◆ Fully-documented Cleanroom process

◆ Application of Cleanroom to large-scale development effort

◆ New Cleanroom discoveries

- TIRS design and verification
- Real-time tasking verification

◆ Dramatic quality results

Cleanroom Results

SECTION 2.

Cleanroom Results

Redwing Project Quality Results

INCR	KLOC	DEFECTS @ 30/KLOC	PROJECTED DEFECTS(1)	ACTUAL DEFECTS	ACTUAL DEFECTS/ KLOC
1	16	480	64	43	2.7
2	50	1500	200	41	0.8
3	41	1230	164	97	2.4
Subtot	107	3210	428	181	1.7
Sys(2) Test			107	93	0.9
TOTAL	107	3210	535	274	2.6

(1) Incr. testing defects projected @ 4/KLOC and system testing @ 1/KLOC.
(2) Includes system, performance and field testing.

Redwing Project Productivity Results

INCR	KLOC	PROJECTED PROD. LOC/PM	ACTUAL(1) PROD. LOC/PM	DELTA %
1	16	300	400	+33
2	50	350	500	+43
3	41	400	513	+28
Avg.		358	486	+36

(1) Productivity = LOC / development labor from spec through incr testing.

Selling Cleanroom

SECTION 3.

Selling Cleanroom

C-4

Key Activities for Selling Cleanroom 3.2

1. Quality Improvement Team (QIT) for development process recommendation.
2. Management overviews (2).
3. Team education, including management and support groups.
4. Process Working Group (PWG) for
   - Process definition and documentation
   - Continuous process improvement
5. Process certification for all project members.

QIT Recommendation 3.3

USE CLEANROOM METHODOLOGY BECAUSE ...

1. Cleanroom emphasizes specification and design.
2. Correctness verification would provide high quality for software intended for outage avoidance of MVS.
3. Cleanroom would foster strong teamwork.
4. Project personnel with prior Cleanroom experience could serve as consultants.
5. Incremental approach would provide early view of project progress and allow for refinement of requirements.
6. General belief that Cleanroom would be effective.

Project Education

1. Cleanroom Overview* (1/2 day)

2. PDL and PDL Translator Class (1/2 day)

3. Design and Verification Class* (3 days)

*Attended by all managers

Implementing Cleanroom

SECTION 4.

Implementing Cleanroom

Key Activities for Implementing Cleanroom 4.2

1. Defined an introductory (phased) implementation of Cleanroom.
2. Joint (mgmt and tech) development of Incremental Development Plan.
3. Small team of teams approach with functional management.
4. Appointed Cleanroom facilitators.
   - Held weekly "office hours" and "visits."
   - One expert present at every verification review.
5. Held "Demonstration" reviews for entire project.
6. Defined common "tasking model" for design and verification.

Introductory Implementation on Redwing 4.3

CLEANROOM TECHNIQUE	USED (Yes/ No/Partial)	COMMENT
Incr Planning	Yes	3 increments
Formal Spec	Yes	PFS + formal incremental spec
Formal design	Yes	Used translatable PDL w/ intended functions
Verification	Yes	Strict adherence to rules
Stat. Testing	Partial	Incremental testing w/ statistical testing for workstation

Summary

SECTION 5.

Summary

Observations From Project Members

- Teamwork very strong, with verification reviews as "extremely positive experiences."
- Reviews reduced the "lone eagle syndrome."
- Developed strong sense of "egoless code."
- Maintenance was simplified due to team expertise.
- Team structure and reviews provided excellent forum for learning.
- Testing team surprised when no errors found in several components.
- Zero-defects quickly became the expectation, not the exception.

Primary Success Factors

1. Strong management support.

2. Technical team participation in methodology selection (QIT) and implementation (PWG).

3. Joint management and technical team development of Incremental Development Plan.

4. Demonstration reviews during Increment 1.

Primary Success Factors ...

5. Strict adherence to correctness verification rules.

6. "Consultants" on Redwing with prior Cleanroom experience.

7. Functional management scheme for team of teams.

8. Use of an introductory (phased) implementation of Cleanroom.

What I Would Do Differently?

1. Select development process earlier and begin training at startup.

2. Allow time for a small demonstration project (1-2 months) for team training and buy-in.

3. Have more, but smaller increments.

4. Use a full-time, dedicated Cleanroom consultant.

5. Conduct statistical testing.