

# Using Agile Programming Techniques for the Enterprise Information System : A Case Study

Louis Coraggio – Troy State University, Florida & Western Region  
Wayne A. Lundeberg – University of Phoenix, Southern Arizona Campus

## ABSTRACT

*The authors describe the development of an Enterprise Information System (EIS) for an ISO 9001 manufacturing firm. The system is built using rapid application development tools with the method known as extreme programming. An overview of the EIS development process, the system design goals, and a chronological narrative of EIS development are presented. Included are additional requirements and recommendations for those considering agile methods.*

## AGILE PROGRAMMING CONCEPTS

Information management has a cyclical history that resembles the fashion cycle of men's ties; hold on to a tie long enough and it comes back in style. In the 1960s mainframe era, centralized processing ruled. Controlling access to resources and code efficiency were key IS goals. The introduction of the personal computer ushered in the distributed processing age. Users customized the environment and hardware became relatively cheap. Now that server based peer networks dominate the firm, centralized principles are being revisited as data integrity and controlling access are once again a priority.

In early application development focus was on getting the programming done, then worrying about documentation. The introduction of the Systems Development Life Cycle (SDLC) model and Systems Analysis and Design (SAD) methods focused attention on assessing user needs and constructing performance specifications prior to coding. The introduction of CASE development tools and object oriented programming techniques drastically reduced the expense and time for programming. The SAD methods are capital intensive, requiring a substantial investment of resources before producing a usable piece of code.

In the late 1990s, a movement toward "agile" software development gained popular momentum. The principles of the agile development are:

- Individuals and interactions over processes and tools;
- Working software over comprehensive documentation;
- Customer collaboration over contract negotiation; and
- Responding to change over following a plan. (Beck, 3)

Agile methods assume a constantly changing business and user environment. Focus is on cooperative efforts between programmers and the user community. Economic justification is the immediate benefit of the application, i.e. savings by the users will offset any increased cost for later maintenance.

## Extreme Programming

The most widespread form of agile concept is Extreme Programming (XP) was first proposed by Beck (2). The formalizing of XP principles arose from the development of the

Chrysler Comprehensive Compensation system headed by Beck in 1997. Paulk (6) best summarizes the principles of XP:

1. *Planning the game* -- quickly determine the scope of the next release, combining business priorities and technical estimates.
2. *Small releases* -- put a simple system into production quickly. Release new versions on a very short cycle.
3. *Metaphor* -- guides all development with a simple, shared story of how the whole system works.
4. *Simple design* -- designed as simply as possible at any given moment.
5. *Testing* -- continually write unit tests that must run flawlessly; customers write tests to demonstrate functions are finished. "Test then code" means a failed test case is an entry criterion for writing code.
6. *Refactoring* -- restructure the system without changing behavior to remove duplication, improve communication, simplify, or add flexibility.
7. *Pair programming* -- all production code written by two programmers at one machine.
8. *Collective ownership* -- anyone can improve any code anywhere in the system at any time.
9. *Continuous integration* -- integrate and build the system frequently, every time a task is finished. Continual regression testing means no regressions in functionality as a result of changed requirements.
10. *40-hour week* -- work no more than 40 hours per week as a rule; never work overtime two weeks in a row.
11. *On-site customer* -- real, live user on the team full-time to answer questions.
12. *Coding standards* -- rules emphasizing communication throughout the code.

While neither of the authors had formally embraced the XP creed, both had an extensive history of collaborating on development of commercial applications. The authors had independently concluded that many of these principles worked. Pair programming, collective ownership, simple design, continuous integration, refactoring and coding standards had proven effective in past projects. With an on-site customer, much of the XP model was in place. The remainder of this paper describes the development of an Enterprise Information System (EIS) using (and discarding some) XP principles.

## **EIS DESIGN GOALS**

### **Background**

Catalina Tool & Mold (CTM) is an ISO 9001 manufacturer of precision plastic injection molds and molded plastic products. Historically, CTM has competed primarily on a quality and delivery basis. CTM is an industry leader in manufacturing technology, rapid delivery, and product design. The emergence of cheaper mold shops in the Far East has eroded the domestic market for traditionally price-sensitive customers. As a consequence, CTM focused on those customers for whom speedy delivery and/or extreme precision are primary concerns.

With annual sales of \$5-\$9 million and 60-90 employees, CTM was one of the larger independent tool and mold shops in the U.S. The typical job is a “one of”, make to order contract, done on a fixed price basis. On average, a job is in the shop for 6 weeks with \$150,000 in revenue. Repairs and modifications to prior jobs also constitute a substantial portion of sales.

In 1998, Mr. Lundeberg took over as CEO. Typical products at the time were cell phones, integrated circuit assembly trays, and industrial sprinklers. Customers pushed for faster turnaround times, experimental plastics, and closer tolerances. Investments in robotics, computer controlled machining and sophisticated CAD software could no longer be managed with the existing batch mode, report driven information system. The existing system was a patchwork of a Unix based, mainframe managerial/ financial accounting system, Excel spreadsheets, and user developed Access applications.

Much of Beck’s model fit circumstances at CTM. The network platform was clearly moving to Microsoft Windows NT based servers. CAD applications that had once required specialized UNIX workstations could now be run on Intel based Windows machines. It was clear that CTM needed an information system that would allow for enterprise resource planning and paperless manufacturing using real-time data. Strategic plans also called for re-inventing the business into new markets and products.

## **Objectives for the EIS**

Extensive discussions took place with key operation people, the Board of Directors, and financial managers. After spirited debate, the following general goals were identified.

***Job Estimation*** – Many design and manufacturing jobs lost money due to misquoting the project at its inception. A quick, clean, collaborative method for quoting new work was identified as the top priority. The new estimating tools should incorporate lessons learned from prior mistakes.

***Real Time Resource Monitoring and Allocation*** – In general, labor and three key processes were identified as bottlenecks. Individual operations often vary greatly from budgeted time. This was true for both engineering and manufacturing operations. Additionally, management priorities changed based on new work; customer modifications; or errors in design and machining. Managers needed real time monitoring of shop floor resources.

***Preparation for ISO 9001-2000 Standards*** – To maintain certification, ISO requires that: “(t)he organization shall plan and implement the monitoring, measurement, analysis and improvement processes needed to demonstrate: ... conformity of the product; ...; the conformity of the quality management system; ... and the continuous improvement of the quality management system.” (ASQ,1) The new EIS should include monitoring and exception reporting for non-conformance as well as archiving capabilities.

***Universal Visibility*** – All workstations should have at least read privileges to most major aspects of the job.

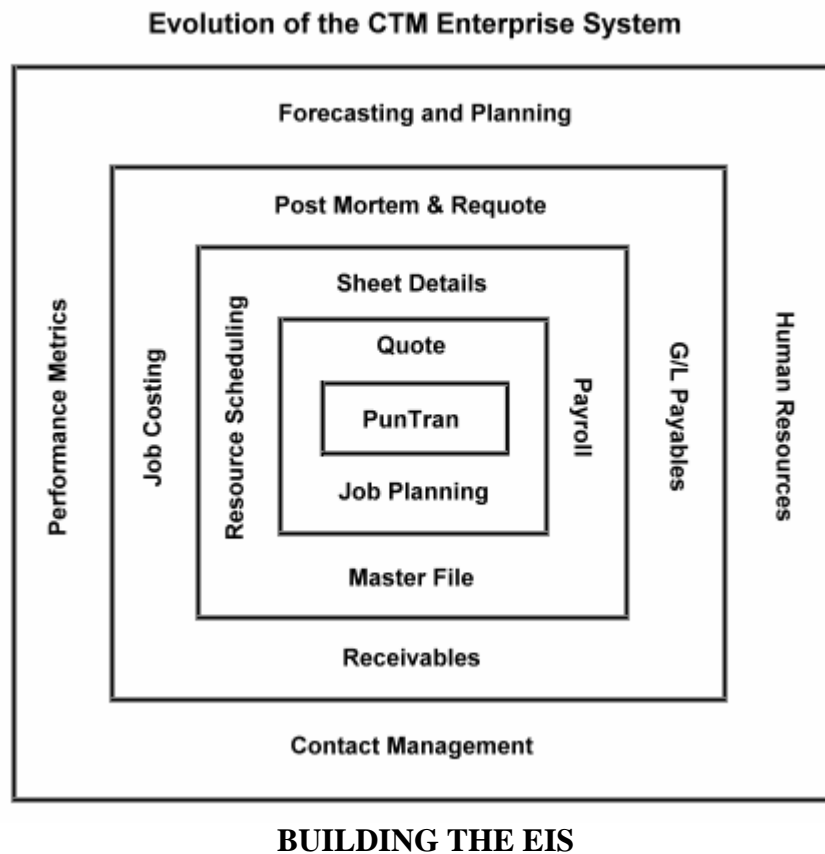
**Cultural Compatibility** – CTM had established cost centers, local terminology, and shop floor reporting formats. User acceptance and minimized training time were deemed mission critical.

**Reduction of Redundant Entry** – Much of the administrative information flow was paper to electronic to paper (e.g. time reporting.) Eliminating the initial paper step would reduce transcription errors and streamline accounting processes.

**Post Production Analysis** – Retrospectives on “what we did” to improve quoting skills, elevate quality and for use in performance evaluation. Analysis of the “as built” project would form the basis of a modular library of operations to be used in future job estimating.

Work began on the EIS in the summer of 1998. (Figure 1 depicts an Overview of the EIS development sequence.)

Figure 1



### Development Environment

The Clarion environment is a 4<sup>th</sup> generation, object oriented (OO) development environment. Applications are based on the initial construction of a relational database with file relationships, formatting, and changeable referential integrity constraints. Basic forms, queries, tables, and reports are then constructed from the database design – much like Microsoft Access. Unlike Access, Clarion allows the developer to mix and match multiple file formats in the same

application and produce a stand-alone executable application that manipulates an independent set of data files. Also included is an independent report writer application that allows users to construct (and save) ad-hoc queries as formatted reports. Reports added in this manner do not require modifications of the database or recompile and reissue of applications. With a mixture of Windows 95, 98 and NT machines, Clarion's independent data and application provided maximum flexibility.

The interchangeable file formats allowed us to prototype quickly and rapidly convert live data. The proprietary "TopSpeed" file format was chosen for the final database. This format stores each table and all of its keys in a single DOS file and uses the application as a DBMS. Record hold/lock semantics, sharing, filtering and referential integrity constraints are enforced at the application level. Using this format allowed us to add, format, and manipulate individual tables without revising the entire database. It also places responsibility for maintaining database integrity and validity squarely on the developers.

The OO structure of Clarion provides an inherent consistency of formats, screens and reports. Much of the "code standard" advocated by XP is easily accomplished by customizing the templates used as a starting point for forms and tables.

### **Project Management for the EIS**

In Clarion, the "Dictionary" construction is done independently from the "Application" development. When a Dictionary is changed, the programmer must close it, reopen the Application, and instruct the environment to integrate the changes. The Application must then be recompiled to produce the standalone "Program". Since the database is independent, each change of the Dictionary requires a conversion of the live database. With two developers, version control (and data file conversion) becomes a critical issue. To mitigate this issue, the authors created a "Master" zip disk. Only the programmer with the Master disk was allowed to change code and database fields. The official version was contained only on the master disk. Despite the availability of remote network connections, this master disk approach resulted in better personal time management and forced the developers to review and agree on changes before going live.

Each day, live data was backed up from the server to a development workstation. The live data and current versions of all Dictionaries, Applications and Programs were then archived to offline media. Each version revision was placed on a separate disk to allow the opportunity to roll back to a prior version if necessary.

While both developers worked on most aspects of the EIS, a comfortable division of labor was soon reached. As the "resident" member, Lundeberg was primarily responsible for defining business rules, final formatting of screens and reports, setting priorities, and developing program flow. Lundeberg had exclusive control of posting new releases and acted as project manager. As the "off-site" developer, Coraggio did most of the hard coding and prototyping. Database design and relationships were done together. The major benefits of paired programming were seen in developing complex database relationships and logically intensive batch updates.

User requests for changes and features rapidly got out of hand. Eventually, a formal request procedure was established to field requests. Each was reviewed by Lundeberg and typically forwarded to Coraggio for implementation in the next release. As the system grew, releases became less frequent. Many of the requested features were aesthetic or the result of disparities in local workstations. Since Lundeberg was both Project Manager and CEO, he had both the technical background and the authority to implement (or deny) changes in CTM EIS.

## **Phase 1 – Time Reporting**

Customer pressure to reduce delivery times and price suggested the immediate need was an effective means of managing jobs in the factory. The existing Shop floor job management used a mainframe system driven by paper records of worker activity. Timecards served as both a payroll vehicle and as the primary record of cost center activity and progress. Workers typically put off filling out cards until days end, often with less than perfect recall. Transcription time added another 3-6 hours to the reporting process. As a consequence, a computer based timecard system (PunTran) was chosen as the initial system module.

The PunTran application drives most of the shop floor management. It was also the first time many shop floor workers would be asked to use PC's for cost accounting purposes. PunTran needed to be easy to use and relatively bullet proof. Workers punch in at the beginning of each shift. When punching out, net working time is calculated. Before a time record is accepted, the worker must completely allocate their time to current jobs and to cost centers. The construction of the cost center codes allowed the punch records to generate job utilization for key machine centers as well. A worker could punch in and out as often as they wished; eliminating the need to remember what he had done.

Introduction of PunTran was done formally in a company meeting. A general outline of the EIS was presented, and the initial PunTran release was demonstrated on 10 computers spread throughout the factory. A one-week period of testing was initiated. Workers entered time on both paper cards and in PunTran. At the end of the test period, worker comments were solicited.

Most found the system easy to use but slow. The prototype used list boxes and pull downs for validating choices. Most thought typing (with validation) should be an option. Workers also wanted to see a listing of their hours; to request personal time off; and to leave notes for management. All changes were implemented and the system went "live."

After a couple of weeks, behavioral issues came to light. Workers typically used one machine for punching in and another for punching out at days end. Word spread quickly that some PCs had slow clocks and others had fast clocks. In the morning, they used slow machines; in the afternoon they used the fast PC's resulting in as much as 15 minutes per day of extra time. (This was remedied by synchronizing local stations with the server clock)

Though individual workers had no edit privileges, several key administrators could fix "mistakes". It became apparent that someone was padding the records. Several security

measures were considered. Eventually, transaction logging was added for any edit changes and included fields that identified the date, time, workstation, and original values. Armed with transaction record logs, the culprit was identified and confronted. Word spread that the system had “spy-ware.” After about a month, most of the initial gamesmanship and “testing” was done. PunTran became part of daily life at CTM. PunTran freed one full-time clerical worker from time card processing.

## **Phase 2 Estimating and Job Planning**

The job-monitoring database structure is the heart of this EIS. All resource scheduling, cost accounting, ISO documentation, resource utilization, engineering designs and purchasing are tracked and allocated to the job. Determining the granularity of the job database consumed many hours of analyst time.

Integrating estimating (“quotes”) with work orders (“jobs”) needed to be as seamless as possible. The Access based estimating system (already in use) was accepted and effective. Using it as a basis, the basic screen formats, program flow and logic were duplicated in Clarion. Using the Access structure also allowed us to convert live data into the TopSpeed format. This was particularly important because of the value of using historical quote information to aid in quoting similar new projects.

Both quotes and jobs may have several “items”, each having its own budget. (*Typical quote items include tasks like product engineering, mold design, mold building and plastic part production.*) Quote items formed the basis for job items, but a job may be assembled from a variety of quotes to the same customer.

Frequently, a job requires a many:many relationship with the customer file. CTM may ship to several locations around the world, and have several different contacts for billing, project management, or technical decisions. Some jobs (like repairs or enhancements) may actually be completed before written authorization arrives. As part of job planning system, contacts within a company were spawned into a child file. Also added was a communication log that attached to each contact and to each job. This allowed project managers to track the history of engineering changes and project anomalies.

Each job consists of a parent job record and two sets of children; the job items and one record for each cost center. When a new job is created, the basic information is copied from various quotes and transferred into job items. At the same time, one child record for each cost center is created. In the quote system, individual cost centers are carried as fields in the quote record. Thus each quote field instantiates one child record in the job. Multiple items are aggregated to their individual cost center. (Appendix 1 shows a partial list of CTM Cost Centers)

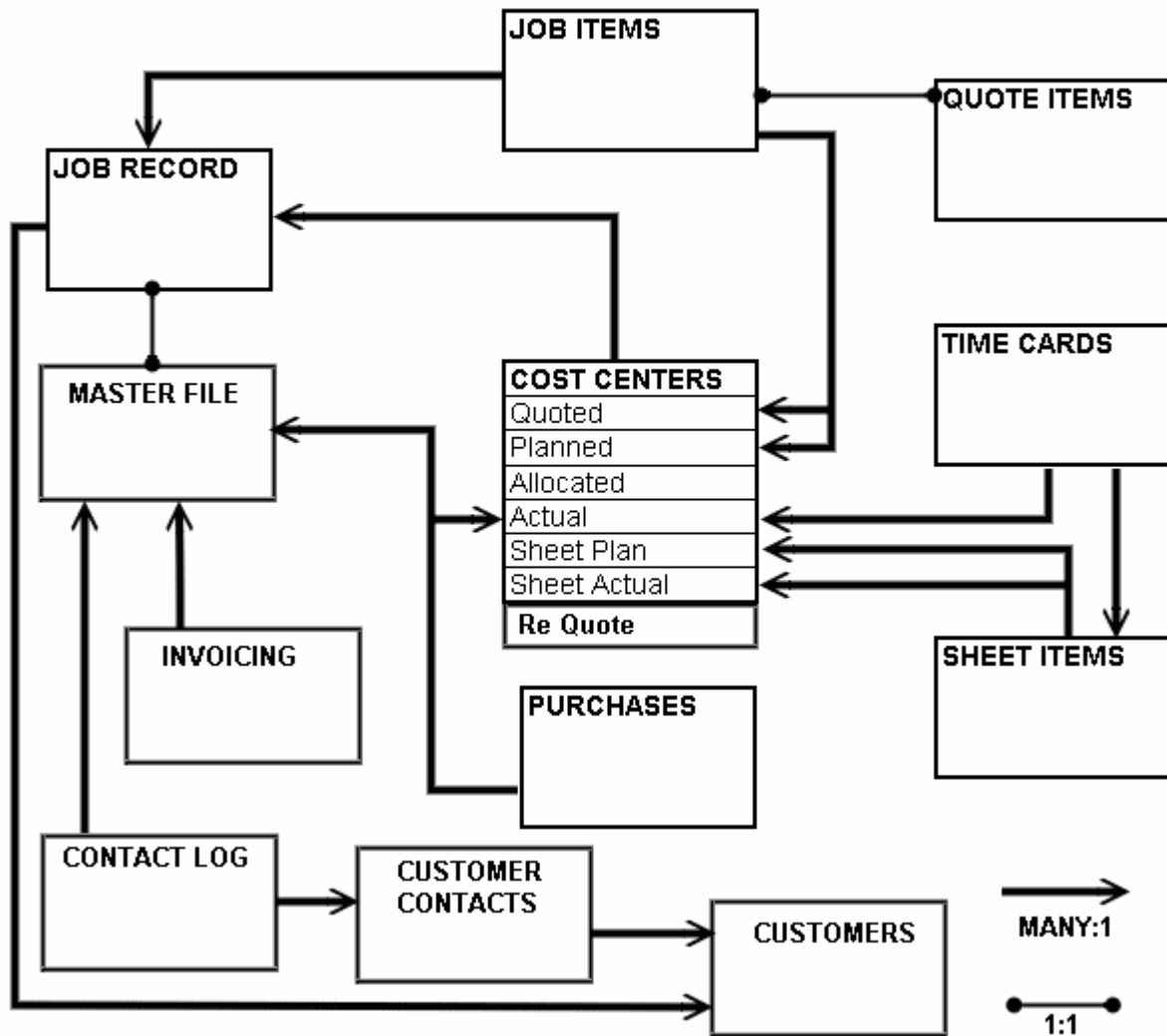
For each cost center, four fields representing the “hours” or dollars are included. There are five separate “hours” kept for each cost center activity.

- Quote – Hours directly from the quote;
- Planned – Revisions based on conditions at the time the job is created;
- Allocated – Budget hours to be publicly visible (may be revised);

- Actual – Hours tracked by the punch system; and
  - Requote – Analysis of the actual time spent per process restated to account for specific process events occurring on that job. The value of this is for quoting a repeat job.
- Fields for start/end dates, remaining hours and management-only “fudge” fields round out the cost center record. (Figure 2 depicts the file relationships for the primary job files)

Figure 2

## EIS JOB FILE RELATIONSHIPS



### Phase 3 – ISO 9001 Subsystem and Documentation

The completion of Phase 3 stabilized the database and established the central transform for the EIS. It allowed the developers to divide work independently as well. ISO 9001 compliance standards require considerable documentation of meetings, reviews, customer contacts and error correction processes. Most of these are tracked to the job level. In a

normalized database, most of the ISO requirements should be part of the main job record as most requirements are 1:1. However, the sheer number of fields, and their infrequent access, led us to create a separate “master file” record for each job. This record also became the parent for most of the administrative tasks including purchasing, shipping, inventory movement and customer invoicing.

From a culture standpoint, most of the ISO documentation is after the fact. Consequently it becomes a nuisance. Much of the development work was targeted to making the job tracking record a push button operation. The CTM ISO System calls for a series of reviews and meetings throughout the project. When the master file record is created, tentative dates and participants are instantiated for these reviews. Based on the ISO plan and job characteristics, appropriate fields are displayed or hidden from users. If the program detects that a target date has past, users are prompted for an explanation or asked to fill in details.

Splitting the production and administrative sides into separate records also enhanced network performance. Typically, managers worked on the job tracking side while manufacturing floor people worked in the cost center side.

#### **Phase 4 -- Resource Scheduling**

Rolling up all budgeted and actual hours to cost centers provides a good overview of individual jobs and resource loading. However, it does not provide sufficient detail for scheduling. Since short delivery time is a major competitive advantage for CTM, scheduling is the mission critical function of the EIS. In manufacturing, tasks are identified on individual sheets of the design drawings. A prototype mold may have only one cavity; a production mold may have several cavities. The estimates, specifications and CNC programming for a single cavity might constitute a “Sheet”. For a six- cavity mold, each sheet detail would be repeated six times. Thus the sheet detail becomes the atomic unit for purposes of resource scheduling.

In a perfect world, sheet details would roll up to job items. Job items would roll up to the main job record. CTM history had proven otherwise. Competition for resources, customer changes, and rework all create a dynamically changing manufacturing floor. So another set of child records were created to accommodate shop floor scheduling. Sheet detail records are logically the children of the main job record. As cross check, sheet hours are also rolled up to their respective cost center records.

With the complex relationships and evolutionary development cycle, job records are not truly real-time. Job records are actually batch updated on-demand at the user level. Each time a user asks for details on a job, a batch process checks for changes and additions in sheet and timecard files, updates the job record in the database, then the record is displayed for that user. Since Clarion does all database management at the application level, this approach offers improved performance for all users, without forcing every entry in the PunTran system to update multiple tables.

Several versions of the resource scheduling process were modeled. Scheduling simulations were tried (based on PERT and Shortest Processing Time). While theoretically

attractive, these proved impractical due to uncertainty of individual workstation times; on-the-fly redesign of the product (concurrent engineering); and resource availability. These prototypes were built with sample data, and demonstrated in focus groups with line level managers. This departure from “simple design” principles proved to be a major waste of time. This was a definite win for the XP camp.

The final version uses a list based priority system that employs a group technology approach. Resources (labor and machinery) were classified into groups based on interchangeability. Cost centers were first aggregated into groups based on similarity of skills and workstation capabilities. Each resource was assigned to one or more groups based on capabilities. For each job, tasks are also assigned to the same group designators. This allows managers to look at subsets of resources and tasks that match up. The assignment is by drag and drop. The sequencing of tasks is determined on a daily basis by the plant managers through assignment of a priority.

### **Phase 5 - Financial Accounting**

For the financial accounting functions, a set of Clarion “templates” was purchased from another developer. These included General Ledger, Payables, Receivables, Payroll, and Purchase Orders. The authors spent about 150 hours customizing these templates. Much of the time was devoted to figuring out the template logic and data structures used by the original programmer. Importing time card records for payroll and invoicing for receivables are the primary interface with the main job based system. The authors found dealing with “foreign” code frustrating at times, the time and manpower savings from the purchase of these templates clearly justified the decision.

Payroll was the first subsystem brought on line. With PunTran in use, all of the pertinent information could be electronically accessed. Once the payroll system was customized for local taxes, direct deposits and payroll deductions, a batch process extracted payroll hours from the PunTran database to produce paychecks.

Receivables are handled within the Master File records. CTM agreements typically contain progress payments, and recurring invoices that required tie-ins to job items, quote items, and authorizations already contained within the main job system. Only summary information is exported to the General Ledger. Payables are handled within the financial system by the purchased template system. Balance Sheets, Income Statements and other standard financial reports are maintained in standard spreadsheet programs.

CTM operates using a revolving line of credit secured by capital assets and receivables. Periodic reviews of key financial ratios and performance indicators are part of the bank covenants that dictate interest rates and fees. Since CTM has a mix of recurring production and one-time projects, determining revenues, costs and work in process is a quite complex. Accurate measures of true output are performed on a monthly basis. Sales taken for a month are determined on a job-by-job basis by management. Once sales are determined, an extensive batch process rolls up all payables, invoices, time records and job progress. From this Shipment/Work In Process (WIP) report, financial ratios are calculated and the labor overhead burden is

determined. Each quarter the shop labor rates (containing the overhead burden) and machine rates are reviewed and adjusted based on Ship/WIP results. The completion of the Ship/WIP process was the final link in the enterprise system. It ties the managerial accounting system to the financial accounting system and generates most of the performance metrics required by management.

## **Phase 6 - Marketing and Performance Evaluation**

As the EIS evolved and manufacturing floor management became more streamlined, the primary focus turned to marketing. By mid 2001, the CTM client base had shifted. Jobs were smaller, with a wider customer base. CTM had opened a subsidiary operation in Queretaro, Mexico to service major manufacturers in the area. CTM marketing efforts turned toward soliciting ideas for new products from patent holders and high margin specialty manufacturers. This became known as Concept To Market (C2M).

Contact management, follow-ups, forecasting and long term monitoring of market segments became critical to the success of the C2M strategy. For the EIS, a contact management scheme was added to track personal responsibilities, dates, and prospect qualification information. Likelihood estimators for quoted jobs were added to improve sales forecasting. Much of the development effort was directed to making contact system easy to use and convenient. Reminders, and contact information buttons were tied to customer files, quotes, job progress, Rolodex and calendar functions. Wherever possible, field values were instantiated based on login, program location, and customer information. Exception reports and color-coded browse lists enable senior management to assign responsibilities and set priorities for both the corporate and individual to-do lists.

Postmortem job analysis and ISO documentation include both “as built” information and mitigation for any quality issues encountered during construction. This provided an opportunity to add individual performance appraisals for employees. These became the basis for the HR sub system. In combination with time summaries from PunTran, wage/benefits from payroll, and personal information, these became the Human Resources system. The addition of user-defined queries enables management to look at histories for individuals or groups, as well as a cross-section for any time period.

## **Security**

While most of the EIS information is visible to all employees, there are areas limited to senior management, and accounting. Three methods are used to secure access to sensitive information, passwords; separate applications, and hidden menus. A master “switchbox” provides access to all of the individual applications. When the user logs into the switchbox with an ID and password, only those menu choices available to that user are displayed. The switchbox in turn calls individual applications and logs the date, time, station, and user in a transaction file. Individual applications also look for specific “flag files” in other folder locations to verify that they are being accessed from the CTM server. Absence of these flag files will cause an immediate shutdown.

Within an application, there are selected menus that only appear when the user executes a specific key sequence. This method is primarily used for senior management to “adjust” budgets, editing timecards, or printing sensitive reports.

The structure of the TopSpeed data files also allows the database to be copied while in use. Thus mirrored copies can be quickly generated for tracking and comparison purposes. The EIS resides entirely on a single server, with firewall protection to the outside world provided by a Microsoft Proxy Server. Source code is kept on a single local machine. While a knowledgeable employee could conceivably make copies of the data and applications, it’s unlikely that he/she could reproduce the necessary local flag files required for operation.

## **Refactoring Examples**

As more and more of the daily activities at CTM were included in the EIS, user performance expectations also rose. Increased dependency on the EIS created higher demands on the applications. Requests for unusual reports, local use features, faster data input and different layouts dominated requests. Many of these requests were not judged to be worth the developers’ time, or implemented and never used. Though Clarion includes a “Report Writer” for ad-hoc queries, instructing users in its operation (and the structure of the database), proved impractical.

For recurring views, like receivables, export queries by example are employed. Users can tag individual fields in tables, and export the results to comma-delimited, ASCII text files. Resulting files are then imported to Excel or Access for further processing. For “one time” queries (e.g. a capital loan application), the developers created small programs to extract the data and process it with standard office software.

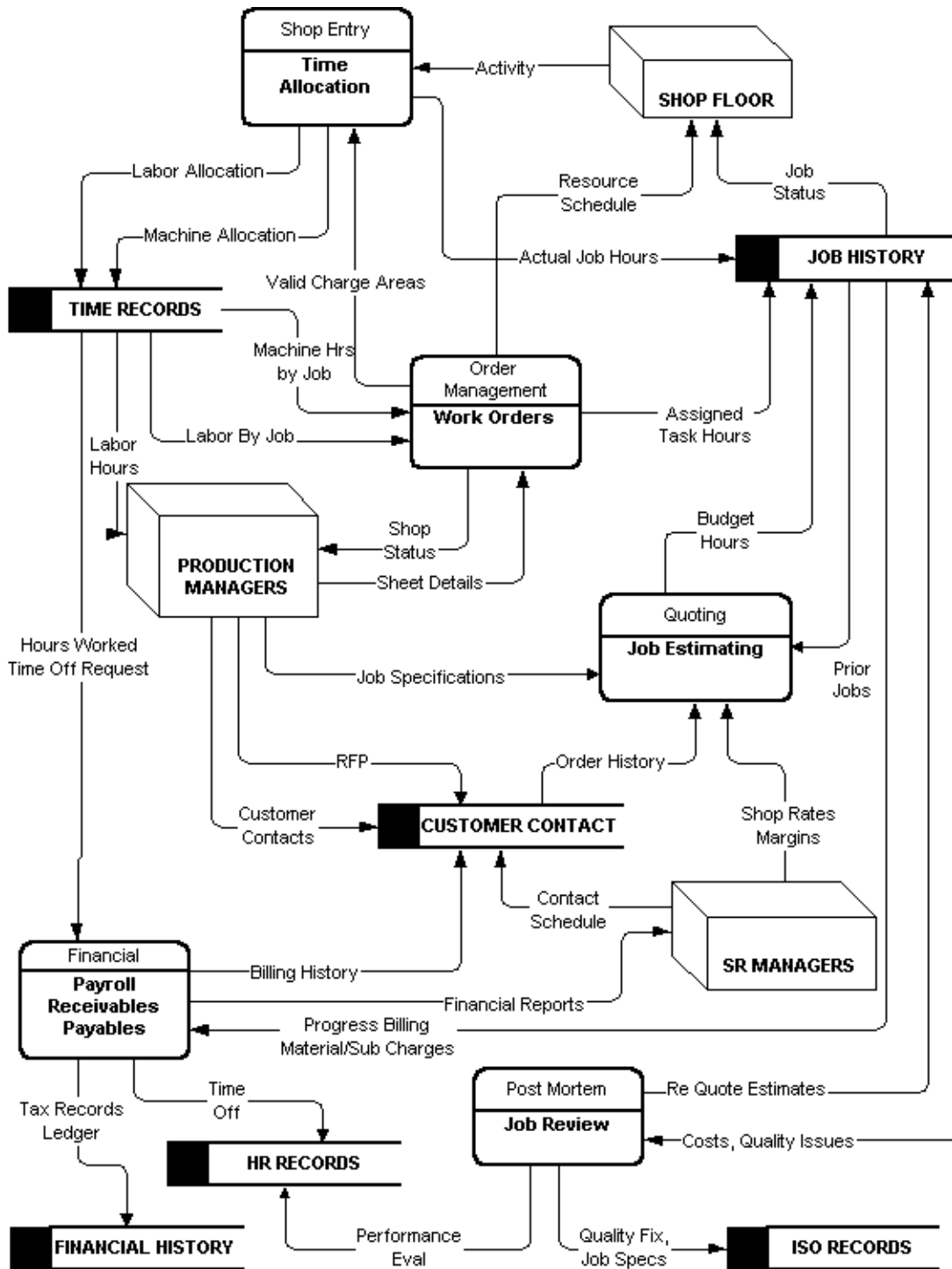
As the EIS grew, infrequently used reports and queries were moved into a separate application to facilitate code maintenance, and enhance performance and response-time of the core procedures. These procedures characteristically require massive sequential searches of tables, intense screen I/O, or creation of temporary files. Though this module can be called from other applications via embedded command line, it also acts as a standalone application. Thus long searches (like the Ship/WIP report) can be accomplished as a background task. Unused procedures and one-time transitional code was removed and archived.

## **Final Form of the EIS**

When the “final” version of the EIS was released in summer of 2002 perhaps half of the U.S. mold building capacity had disappeared. CTM had survived the exodus, and won ISO 9001-2000 certification – one of the first mold shops in the world to do so. Sales for custom machining, OEM products, and C2M projects have supplemented traditional mold building.

Sales for 2001 were only 85% of 1998 levels. Over the same period, revenues per direct labor hour rose from \$53.16 to \$63.62. Much of this rise in efficiency can be attributed to the EIS. Figure 3 shows a data flow diagram of the final system. In Figure 4, summary statistics on development are provided.

Figure 3 Final EIS System Overview



**Figure 4**  
CTM EIS Final Statistics (June 2002)

Procedures in Financial System	275
Procedures in Main System	303
Primary Tables in Data Base	150
Externally Generated Reports	94
Estimated Programmer Hours	1400
Approximate Development Time	22 months

### SO DOES XP REALLY WORK?

The CTM EIS must be considered a major success. Since the EIS development follows the XP model, the authors' answer would be a qualified yes. The biggest flaws in XP reasoning are the notions of the simple metaphor and simple coding. Stevens (7) makes a strong case for lack of planning being the major downfall in XP projects. The authors concur. Even if no formal documentation is created, time spent on the data dictionary, entity relationships and data flows are necessary planning stages. Most of the rework time was consumed in revisiting older code to add new functionality. Beck suggests that system specifications be assembled from "note cards" containing user requests. While that may be a good starting point, a trained analyst must convert those requests to fields, records and processes.

The simple coding concept ("exactly what is needed now") should be a guideline, not a rule. Adding database fields for future use, and hooks for future processing are usually viewed as major time savers. In XP style development they also serve as part of the development plan and reminders for procedures yet to be done. The inclusion of "to do" procedures also acts as a preview of coming attractions for the user community.

Paired programming has definite advantages for database development and complex logical modeling. Cockburn and Williams (4) suggest that the cost of a second programmer adds about 15% to the cost of development, offset by a 15% improvement in quality. Collaboration also ensures consistency in coding and data element naming conventions. Once the central transformation is functional (job planning), much of the development focuses on additional input and formatting output. Report writing and screen design are more modular tasks and could be delegated to additional programmers. However, the logistics of meeting, communicating, and integrating the pieces could rapidly consume any extra capacity. Having both resident and non-resident members proved valuable. Intricate and lengthy procedures are better accomplished off-site, with no interruptions. A resident developer can more effectively do testing, revisions, formatting, and refinements.

Based on the experience of Catalina's EIS the authors suggest the following additional precepts for those considering an XP approach.

***The development team needs decision-making authority.*** – To consider all of the user requests and potential policy changes, decisions must be made quickly and have force of law. For XP development, a benevolent dictator is more efficient than a democracy.

***Analyst and programmer roles must be tightly integrated.*** – Many of the flow and logic decisions require an understanding of user perception, operations reality and programming feasibility. Adding new capabilities often creates new software performance specifications and changes in business rules. The analyst’s role as user liaison is vital to acceptance.

***Anticipate transitional nulls.*** – As new tables, features and fields are added, the transition of older data must be incorporated into the code. The XP approach mandates checks for null data and some method for dealing with it. In most cases, this transitional code (or field instantiation) goes away over time. However, transition management will consume a substantial amount of coding time. The majority of incremental programming cost under XP (over SAD methods) occurs in this transition management.

***Create a release plan and adhere to it.*** – As the system grows, incorporating changes and making the user community aware of those changes becomes a management task in itself. At CTM all requests for changes had to be submitted in writing. These were batched together and released about every two weeks. A log of changes was attached to the application, and the current release date displayed on the application title bar. New system features became part of production planning meetings.

***A motivated user community is a must.*** – The technical sophistication of the workforce and the “can-do” corporate culture at CTM is largely responsible for the success of the EIS. Direct user requests and feedback were largely responsible for many of the EIS capabilities and refinements. Without an active, receptive user community XP development will flounder.

***Save a rollback version.*** – Despite thorough testing, every new feature added does not prove successful. A complete sequential backup of applications and data will be a major time saver. It’s often easier to start over with a prior version, than to undo changes.

***Be flexible in database design.*** – While the theoretical advantages of normalized database design are well documented, they may become restrictive in practice. The addition of 1:1 file relationships, extra fields, and multiple key relationships may result in major performance gains and greatly simplify development efforts.

***Allow adequate time for organization adaptation.*** – The default setting for the user community is “I don’t know what I want, but I know that isn’t it.” Introduction of a new report or subsystem will inevitably meet resistance. During the CTM development, users were told that changes would take one to three weeks for completion. Many initial reactions disappeared after people lived with the new report for a while. The lag between activation and utilization will be longer when there is a culture change involved. Though considerable effort went into simplifying the contact management subsystem, many users simply don’t use it.

***Hold on to back up systems as long as possible.*** – Some bugs may take weeks or months to surface. Issues with triggered events (like year end closings or quarterly taxes) may reveal problems that were not caught during routine operations or pre release testing. Trial closings with backup data are highly recommended.

**Consider batch, on-demand processing.** – Many reports are infrequently needed, or are limited to very few users. Using spawned, batch-style processes, (with locally generated files), for these reports will save extensive rework of database keys and relationships. Batch processes are typically easier to modify for transitional nulls.

**Limit end user computing.** – While major development is going on, users will insist that provided reports and screens don't meet their needs. Some will create shadow systems without recognizing changes in business rules and the effects of transitional nulls on data integrity. If the need for ad hoc figures arises, the development team should provide it. Agile development will exacerbate reconciliation problems.

**Consider user documentation issues.** – Most programmers and analysts despise doing user documentation. The accelerated pace of XP methods exaggerates this problem. At CTM several key users act as testers and become “Local Resident Experts” (LRE.) While formal training was regularly scheduled, the LREs provided the majority of instruction. In commercial, “shrink wrap” situations, this approach may not be feasible.

## References

1. American Society for Quality, *American National Standard Quality Management Systems – Requirements for ISO 9001-2000*, December 13, 2000, Quality Press, Milwaukee, WI
2. Beck, K., *Extreme Programming Explained: Embrace Change*, Addison-Wesley, Reading, MA, 1999
3. Beck, K. et.al., *Manifesto for Agile Software Development*, <http://www.agilemanifesto.org/>, Confirmed October, 2002.
4. Cockburn, A., Williams, L., *The Costs and Benefits of Pair Programming*, XP2002, <http://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF>
5. Hoffer, J.A., George, J.F., Valacich, J.S., *Modern Systems Analysis and Design, 3<sup>rd</sup> Edition*, Prentice Hall, Reading, 2001.
6. Paulk, M.C., *Extreme Programming from a CMM Perspective*, IEEE Software, Vol. 18, No. 6, November/December 2001, pp. 19-26.
7. Stephens, M., *The Case Against Extreme Programming*, [http://www.softwarereality.com/lifecycle/xp/case\\_against\\_xp.jsp](http://www.softwarereality.com/lifecycle/xp/case_against_xp.jsp), Feb 3, 2002

“Clarion”, and “TopSpeed” are registered trademarks of SoftVelocity Inc.

“Access”, and “Windows” are registered trademarks of MicroSoft Corporation.

Appendix 1 -- PARTIAL LIST OF CTM COST CENTERS

<b>General Labor</b>		<b>Machine Centers</b>		<b>Materials</b>	
1	Layout	141	CNC Electrodes	201	Mold Base
2	Detail	150	EDM Sinker	202	Steel/MB Plates
3	Checking	151	Mold Base CNC	203	Pins/Comp
4	Electrode Design	152	CNC Machining	204	Graph/Wire
5	Programming	157	Mold Sample	205	Hot Runner Sys
6	Training	158	Production Molding	206	Misc
7	Run Prints	160	Wire Steel	207	Tools
8	Database Processing	165	Wire Electrodes	208	Resin
11	CNC MB				
12	Conventional MB				
21	CNC Machining				
22	Conventional Machining				
23	Grinding				
			<b>Administrative</b>		<b>Subcontracting</b>
41	CNC Electrodes	506	Inspection Admin	301	Heat Treat
42	Conventional Electrodes	510	Engineer Admin	302	Thr. OD Grind
50	EDM Sinker	511	Indirect	303	Turning
60	Wire Steel	512	Supervision	304	Jig Grinding
65	Wire Electrodes	514	Meeting/Training	305	Plating
70	Honing	517	Shop PLT	306	Gundrilling
75	Heat Treating	527	Project Management	307	Polish
80	Polishing	618	Admin Salary	308	Mold Design
90	Assemble	622	Admin Holiday	309	Text Engrave
91	Meeting	623	Admin PLT	310	Sample
111	Production Molding	624	Admin PTO	311	Freight
112	Mold Sample	217	Personal Time Off	312	Misc Subcontract
120	Inspection	218	Shop Holiday	313	Wire - EDM
121	First Article Inspection			314	Machining