Delivering E-Type Solutions through Requirements Validation

David Cohen, Gary Larson and Bill Ware
david@sente.com**,** gary@sente.com**,** bill@sente.com
sente Corporation
7209 Briarnoll Dr., Dallas, Texas 75252-6356 USA

Abstract

IT organizations are increasingly being expected to provide business-solutions faster and with better return on investments (ROIs). Their limited focus on software features fails to deliver predictable, relevant and cost effective solutions. The electronic customer contact management (*eccm*) toolkit was designed **to rectify the most pervasive problem in software development,** the known fact that requirements specifications are **always** incomplete, inaccurate and wrong. The toolkit was used to implement a **human-centric solution** in a wireless center. The solution produced **measurable improvements** in the robustness and effectiveness of center operations, providing **ROI** in less than 18 months. The *eccm* toolkit tackles this business challenge by delivering:

1. **Validated software requirements** in areas such as service, operations and marketing while guaranteeing the investment's ROI through faster cycle times.
2. **Reduction in the software complexity** of all relevant legacy systems because the implementation is based on validated requirements; increases the system life cycle and reduces maintenance costs while delivering business solutions faster and cheaper.
3. **The required work center tools** to professionally manage both center resources and enhancements to the team's process-maturity level.
4. **Effective change management capabilities** in order to preserve the center's productivity during the ongoing demand for new feature deployments.

Keywords: *Software Investment, Systems Engineering, ROI, Best Practices, Software Development Process, TL-9000, Wireless Network Operations and Center Productivity.*

1. Investment in Software Solutions

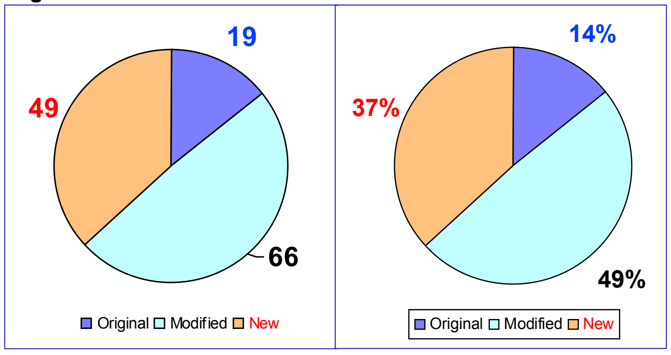
Businesses find themselves today under significant economic pressures to deliver products/services to generate new revenue streams. As a consequence, IT organizations are increasingly being expected to provide business-solutions faster and with better ROIs. The technology focus of IT organizations on **software** fails to deliver predictable, relevant and cost effective **solutions**. This has reached a crisis level in the communications industry. The process responsible for translating the business needs into requirements specifications is ad hoc, without well-defined success criteria for either completeness or accuracy [1]. This is a major challenge for two primary reasons:

1. There is great difficulty in translating high level business needs to detailed requirements specifications that drive the development process (e.g., increase share price by $1.5, improve the yearly productivity of a Rep by 30% in throughput and 15% in quality, improve cycle time of product delivery by 25%). This is further complicated by the lack of communication between the business and the technical staffs; *usually* the technical team does not have an understanding of the effective business use of the newly planned capabilities. Finally, the requirements are always specified under significant schedule pressures because they are a key bottleneck for budget and development planning [2,3].
2. The **specification-driven** (S-Type) investment paradigm incorporates the **wrong economic incentives** to deliver a business solution cost effectively. The business unit is responsible for both specifying the requirements and ROI delivery. The IT organization is responsible for delivering quality *software features* based on these requirements. The limitations of the deployed solution are clearly the business unit's "fault", generating an *economic model that rewards the IT organization (e.g., vendor) with additional development resources while it continues to deliver irrelevant business solutions with limited/no ROI.* This environment facilitates for example, new software releases that contain *only COTS updates* (e.g., OS or DBMS upgrades with no new user features) while causing operational disruptions with severe financial penalties to the bottom line (e.g., loss of Rep productivity caused by system down time, Rep over-time charges to recover lost processing time, and potential regulatory penalties for non-compliance). That is the primary reason why even system integrators (e.g., Accenture, Cap Gemini, IBM, EDS) with extensive business knowledge *prefer to deliver software* and not business solutions.

It is therefore not surprising that in **all** projects the requirements are **incomplete, inaccurate and/or wrong**.

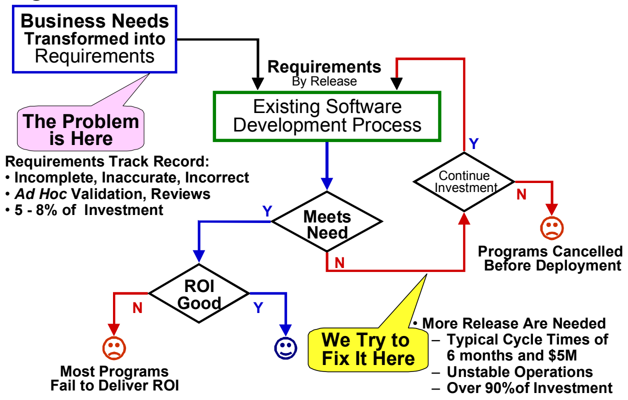
In **Figure 1** we demonstrate this point in regards to the business rules (BRs) that define field relationships associated with a particular order type used in the wireless center. The BRs were being maintained by a team of competent experts and were *considered accurate and complete* (the rules are posted on the company web site with the related customer order form). It took *five eccm* releases to achieve validated and correct BRs. When completed, only **14%** of the BRs were used as is; **49%** of the rules were modified and **49 new rules** (37% of the final total) had to be added as part of the validation process. The process has proven that BRs *cannot be validated* effectively

Figure 1: Prem to Prem MEGALINK BRs Validation



through the standard reviews of textual descriptions. The *eccm* toolkit has demonstrated that BR validation can be reduced from a typical cycle time of 2-3 weeks to 2-4 hours. **Figure 2** shows that business solutions delivered through the current software investment model are being "debugged" through *slow* and *expensive* development cycles making ROI delivery impractical (1:20 is the typical investment ratio between requirements specifications and development).

Figure 2: Current Software Investment Model


Business Needs Transformed into Requirements
The Problem is Here
Requirements Track Record:
• Incomplete, Inaccurate, Incorrect
• Ad Hoc Validation, Reviews
• 5 - 8% of Investment
Requirements By Release
Existing Software Development Process
Continue Investment
Y
N
Programs Cancelled Before Deployment
Meets Need
Y
N
ROI Good
N
Y
Most Programs Fail to Deliver ROI
We Try to Fix It Here
• More Release Are Needed
– Typical Cycle Times of 6 months and $5M
– Unstable Operations
– Over 90%of Investment

Though this phenomenon was recognized and studied by Boehm et al [4, 5, 6, 7] for the last two decades, the IT organizations prefer to maintain the status quo. Their analysis shows that over 50% of the projects terminated **before completion** were caused by requirements that failed to capture accurately what the business needed (e.g., incomplete requirements, lack of user involvement, changing requirements, absence of need, unrealistic expectations).

Software investments will start delivering ROI only after business units change current economic incentives, and **start demanding solutions** with measurable benefits (e.g., in areas such as operational processes, user productivity, training, effective change management, and new product cycle time). Change will happen faster in tough economic conditions that demand a more timely and relevant contribution by IT organizations [8, 9]. The recent **dot.com revolution** made a significant contribution in the "education" of both investors and technical staff on the limitations of narrow technology-based solutions in the market place [15, 16]. In the software educational system Denning [10, 11] and Dertouzos [12] recognize that human and customer centric success criteria must be addressed if we are to continue the phenomenal growth of the software industry.

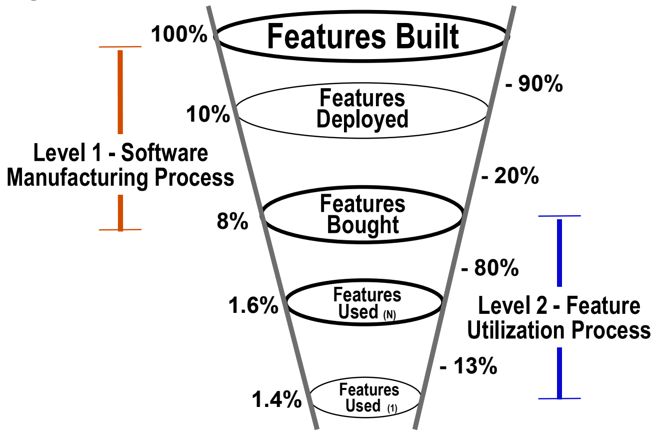
Most of the educational system is isolated from real world programs that are constrained by schedule and ROI. While Dertouzos [12] recognizes the need to *"finish the revolution through human centric solutions",* he dislikes **predictable and incremental solutions** *("If the project is safe, it is not worth tackling, for it makes only incremental improvements"*). He prefers projects that *"could fail just as easily as succeed,"* not recognizing the impact of such a statement on future software-engineers that will be working on these "unworthy" projects for the rest of their lives. **In business, the predictability of programs is priceless** [2, 8, 12, 14].

The need to meet the evolving business needs through e-capabilities of the software was presented first by Lehman [23] in the mid 80's. He reached these conclusions while investigating the software productivity of IBM software developers. His analysis, confirmed later by the actual evolution of these software systems, shows that the ongoing **specification-based (*S-Type*) feature expansion** results in their **"economic death,"** driven primarily by increased software complexity. Incrementally, it takes longer to deliver fewer features at an increased cost, making these systems increasingly irrelevant to the business. The severity of the situation is further compounded by the fact that most features are initially *developed based on incomplete and incorrect requirements,* which further reduces the effectiveness of the solution (e.g., timeliness, cost, robustness). *eccm* is able to meet these **evolving needs based feature-expansion** (***E**-Type)* through hundreds of configuration management (CM) capabilities that facilitate feature changes without software development. These e-capabilities were introduced incrementally to speed-up our responsiveness to the ongoing requirements' evolution. Our ability to validate the requirements reduces the unnecessary feature churning of legacy systems, expanding their life cycle and improving their software ROI. Corporations are littered with numerous *"frozen" legacy systems* that still perform productive functions but are no longer able to satisfy cost-effectively the new business needs. This creates an ongoing demand to introduce new software systems that are targeted to meet these new needs, for an ever increasing software-maintenance cost. The following two sections describe the Software Pollution and Business Knowledge Management models used by the ***eccm*** team to verify the requirements to deliver ROI based business solutions.

1.1. Software Pollution™ Model

The ***eccm*** toolkit was originally implemented as a system-engineering tool to validate the completeness of requirement specifications [1, 4]. In the late 1980's we recognized a phenomenon that we termed **software pollution™**. During a period of 15 years we examined over four hundred projects in sizes from 10-500 person- years per release. In every project evaluated there was a large deviation between the number of lines of code (LOC) developed and tested (e.g., investment) compared with the LOCs deployed in the field (see **Figure 3**).

Figure 3: Software Pollution Model



Software pollution is defined as the "scrap" by-product that occurs during the software (manufacturing) development process. For example, software pollution at level 10 means that we invested in the development of *ten LOCs* for every *one LOC* in the deployed release. The level of software pollution measured in these programs was in the range of 10-100, in line with the complexity, size and the team talent associated with each project. In general, larger and more complex projects had higher pollution levels. A similar effect was observed also in feature utilization. No project reviewed had a feature utilization (from all the features deployed) that exceeded 20%. This means that after all the effort, cost and **pain** to deliver a feature to the users, there is a high probability (e.g., over 80%) that no user will ever utilize it [21].

The analysis identified five major pollution sources:

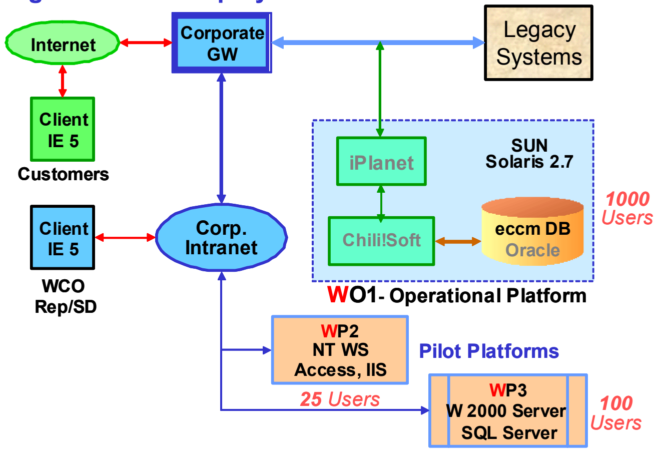
1. Incomplete understanding and capture of customer needs in the format of requirements specifications.
2. Traditional system engineering cannot effectively transfer the customer need to the development team.
3. The development team attempts to implement a system that will support a very large user base in a **long** single release.
4. The development team attempts premature utilization of "bleeding edge" technology ignoring its current level of expertise. Unknowingly, the project turns into a job-training program.
5. Users have a limited capability to absorb new features that do not fit into an effective operational process.

The *eccm* platform has evolved over the last four years into an effective *software pollution fighter*. *Cost effective product enhancements are possible to reflect the newly acquired insight*. This is not practical or even possible when the cycle times are 4-6 months or longer, and the cost of each release is millions of dollars [2, 3]. Software pollution becomes an effective quality metric that can be used to evaluate the quality of the development organization and to predict the useful life cycle of the software system. Increased pollution levels shorten system life cycle and reduce ROI.

The second benefit of this faster cycle time is the ability to *detect earlier* issues that are somehow missed and were not included in the project plan. These surprises that show up late in every program are key contributors to deployment delays and cost overruns [16, 17, 18].

The *eccm* architecture includes two pilot platforms and one operations platform (see **Figure 4**). It supports a robust change management process by avoiding damage to operational effectiveness through incremental scalable pilot deployments.

Figure 4: Wide Deployment Architecture



The toolkit, using generic Microsoft technology (e.g., W2000, W2000 SQL Server, IIS), allows scalability to support limited deployment (e.g., 100s of users) to validate the robustness of the solution (e.g., usability, productivity, system operations) while in use [22].

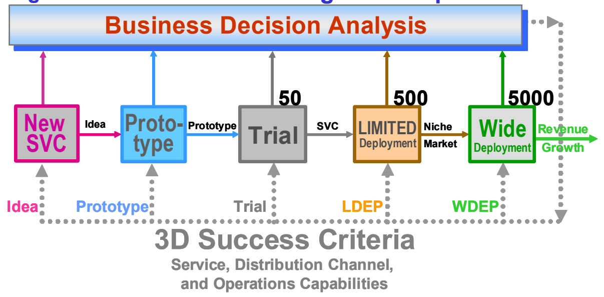
This proved to be an attractive approach to immediately start harvesting the productivity benefits of a solution while the IT organization develops a certified solution that meets the unique architectural guidelines of the corporation [20]. This guarantees that upon deployment, the IT provided solution will be both relevant to the business and have an extended successful life cycle.

1.2. Business Knowledge Development Model

During the 90's we had the opportunity to analyze over *sixty* telecommunications services that had met customer needs (e.g., successful customer trials) but failed in the market place, consequently representing an investment of over 800 million dollars. The results of this analysis were mapped into the traditional "idea pipeline model" for innovation [11, 19]; ideas flow sequentially through a pipeline comprised of peer reviews, prototype development, manufacturing and marketing. The model showed that this slow process required ongoing R&D investments as new success criteria were being encountered during service evolution.

The model was modified to monitor incremental progress using the *number of customers* as the primary success criteria; the updated "idea pipeline model" includes peer review of *new service, prototype development, user market-trial, limited deployment and wide deployment* (see **Figure 5**).

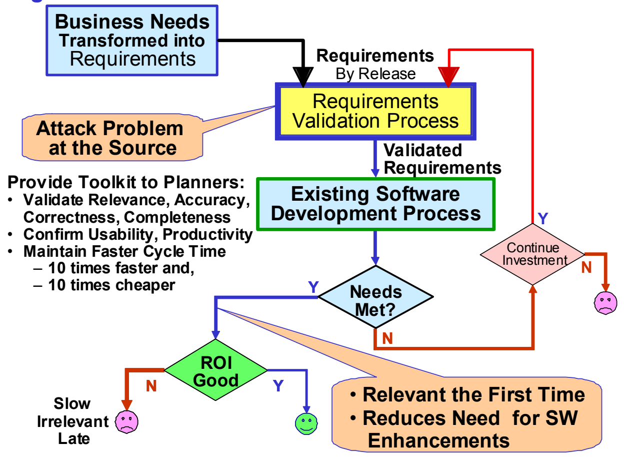
Figure 5: Business Knowledge Development Model



Ideas advance from state to state by meeting the success criteria we harvested from our analysis. This significantly improves the project's ROI because of *concurrent investments* in **service, operations and marketing** (the three dimensions - **3D**) capabilities which must be in line with both the number of customers using the service and the accumulated business insight. Repeatedly it was demonstrated that business insight/customer size scales by a **factor of ten** or less; that is, for example, if we can cost effectively meet the needs of 500 customers we will use this insight to develop the infrastructure for 5000 customers.

This model was used successfully to guide eight startups in industries such as advertising, education, healthcare and telecommunications [21]. Every one of these business entities was able to reach limited-deployment and to break-even financially within a year or less. The model was also used to upgrade the software-investment paradigm to be **ROI driven**, as shown in **Figure 6**. In practice we developed a cost effective method to expand the **e**-capabilities of systems ("E-Type") by

Figure 6: ROI Driven Investment Model



Business Needs Transformed into Requirements
Requirements By Release
Requirements Validation Process
Attack Problem at the Source
Validated Requirements
Provide Toolkit to Planners:
• Validate Relevance, Accuracy, Correctness, Completeness
• Confirm Usability, Productivity
• Maintain Faster Cycle Time
– 10 times faster and,
– 10 times cheaper
Existing Software Development Process
Continue Investment
Y
N
Needs Met?
Y
N
ROI Good
N
Y
Slow Irrelevant Late
• Relevant the First Time
• Reduces Need for SW Enhancements

reducing their software complexity caused by unnecessary software development. Software development is no longer required to figure out the "true requirements" that will deliver an ROI based solution [23, 24].

2. Wireless Center Operations

The Wireless Center is responsible for processing approximately 120,000 orders per year. Each order is for the provisioning of a transmission circuit that will provide interconnection between wireless network elements such as switches and antennas (e.g., AT&T Wireless, Cingular, Nextel, Verizon, etc.). Some orders are for a single trunk circuit while others may include a fiber ring within a city or a state.

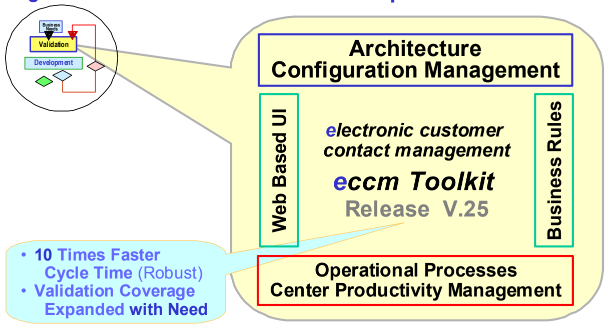
In the original operations environment without *eccm*, orders were submitted by over 500 customers using faxes and emails. The center had no effective tracking mechanism for the orders received. It was the responsibility of ~60 Reps and ~50 System Designers (SDs) to process the orders received in the center and issue orders to the network provisioning team responsible for the physical delivery (deployment, testing) of the service. The cycle time for circuit provisioning may be less than a week if facilities are in place, or it may take 4-6 months if new facilities are required.

Order processing requires the data entry of each order into a set of **legacy systems** such as service order systems (SOS) and billing systems (BS). Each system validates the order information for both accuracy and completeness (e.g., telephone numbers, addresses, contract information, etc.). In some cases error processing may require contacting the customer for clarifications ("clarifies") to ensure the accuracy of the data. These "clarifies" were transmitted in the past using telephone calls, faxes and/or emails. Similarly, customers unhappy with the progress in their order processing are provided the option, through "escalations", to increase the processing priority for a particular order.
The limited order tracking capability in the center combined with an ever-increasing number of escalations resulted in a crisis mode of operations with high stress levels and reduced Rep productivity. For example, it would take on the average 2-3 days to receive a response on an order status inquiry. Center management was provided only monthly quality and throughput (summary) reports delivered usually 2-3 weeks after the end of the month. On the average it takes six months to train a newly hired Rep. The yearly business growth of ~30% per year required a parallel staff growth to meet the order processing load. All previous software investments have failed to improve Reps' productivity. *The objective of this project was to increase the center's order-processing throughput without staff growth [22].*

3. *eccm* e-Capabilities in Support of Requirements Validation

The *eccm* toolkit supports requirements validation capabilities in six key areas related to the wireless center solution: architecture, configuration management (CM), web based user interface (UI), business rules (BRs), operational processes and center productivity management. **Figure 7** identifies these **key validation areas** that were expanded in conjunction with demand (a capability was usually added if we experienced churning of the requirements in an area).

Figure 7: *eccm* Toolkit Validation Capabilities



Validation
Development
Architecture
Configuration Management
Web Based UI
electronic customer contact management
eccm Toolkit
Release V.25
Business Rules
• 10 Times Faster Cycle Time (Robust)
• Validation Coverage Expanded with Need
Operational Processes
Center Productivity Management

eccm provides an environment in which users are delivered ***clearly defined and measurable benefits*** instead of ***software features*** [8, 21, 22].

The rest of this paper focuses primarily on validation capabilities in the area of configuration management (e.g., e-capabilities) and center productivity management. The other validation areas (architecture, web based user interface, business rules and operational processes) are only briefly summarized with detailed examples provided in [24].

3.1. Architecture

The architecture guarantees that all validated functionality is tested for robustness and that it delivers the productivity objectives specified. The two pilot platforms are used for both requirements and solution benefits validation. This eliminated most center disruptions associated with the deployment of new releases. For example, a typical two-hour outage in a 100-person center results in a productivity loss of at least 2 Rep/years (this estimate does not include impact on future customer orders).

3.2. Business Rules (BRs)

Business rules validation turned out to be a surprisingly difficult challenge. Initially BRs were provided to the development team in a textual description form. Once implemented in software we discovered that no previously processed order could pass verification. This was due to incorrect and missing information in the BR descriptions.

This complex BR knowledge is made available to the users as a standard product related report. The BR's textual description was also integrated with the browser's standard help feature. The user has direct access to this knowledge base just by positioning the mouse pointer over a selected field. This turned out to be one of *the most valuable aspects of the application* during order creation or verification. It also proved to be an effective "safety net" to the training program. The BR knowledge is updated continuously based on users' feedback without any service interruption. Several new users were able to create and submit complex orders after they were given a brief (10 to 15 minutes) introduction to the application (without training).

3.3. Web Based User Interface

Users access the *eccm* application through industry standard browsers (e.g., IE 5). The training of a Rep prior to *eccm* deployment in the complete WCO operations took over 6 months. Three human centric success criteria were identified for the solution [12]:

1) User training should take a day or less including hands on use of the software capabilities.
2) The application must support the baselining of user capabilities through measurements collected while processing a standard set of test scenarios (qualify Rep's performance before "going live" into order processing).
3) New computer literate users must be able to use the application without training, exposed only to a limited introduction to the application (e.g., 15 minutes).

All three of these objectives have been met.

The following user features made this possible [24]:

The BR knowledge base was turned into an effective "help" capability using the standard browser feature (used to create and correct orders). The system maintains a complete trace of *all order* content and status changes; each trace record identifies the initiator of the action as well as the date and a time stamp. The user interface includes a powerful two dimensional **sort capability**. First, any subset of the order fields can be "equipped" with a sort option. Second, a column sort is provided for all visible fields that could benefit from such a capability. The user interface **guides** the Rep to carry out work assignments based on their urgency. Priorities are set via table driven rules according to the order-processing performance objectives established by management.

3.4. Configuration Management (CM)

The following are the key *eccm* toolkit CM features (systems engineers use these features to validate requirements without any new software development):

1. Create a new product (or modify an existing one) and the related form/BRs. A new product can be made available to customers within a week; this is a competitive advantage since in the past it took 4-6 months to introduce a new product.

2. Partition order form into meaningful logical parts with effective **navigation-links** (e.g., some of the order forms include over 130 parameters). The toolkit supports a powerful order-form editor (e.g., moves fields with the related business rules from one logical section to another, modifies headers, modifies business rules, and automatically updates the navigation links to reflect structural changes, etc.). This capability was implemented to speed up new product introductions or make prompt product updates without the traditional "release planning" process. The implementation uses the "section title" concept of a word processor. Once the new structure is saved, the navigation links are automatically updated to reflect the new structure.

3. Classify data fields as **R**equired, **C**onditionally Required (a user must supply a Required field with an appropriate entry when creating/editing the associated form) or **O**ptional; field names can be modified as needed. Each data field can be assigned integrity constraints and/or BRs.

4. Modify a variety of system parameters (e.g., time out parameter of an inactive session - 30 minutes; time interval used in active/inactive order reports - 30 days).

5. The toolkit provides system administrators capabilities to maintain customer accounts, teams, team administrators, users, user sessions, user passwords, database backups, software release updates and projects (professional management of a collection of orders). Throughput and quality reports are provided to administrators to track these CM activities.
6. The toolkit supports an **UNDO** status capability. The user interface warns the user before an order is cancelled and the action is taken only *with direct user concurrence*. If the user makes a mistake in spite of the warning, the UNDO capability facilitates restoration of the database to the previous state while still tracking the UNDO action.

3.5. Operational Processes

The operational processes have a dominant influence on the organization's productivity. They define the organization's internal and external interfaces. Each one of these interfaces represents a handoff between organizations and/or people, and thus a *potential cause for delay and/or confusion*. Therefore, the *eccm* application allows monitoring of operations processes both before and after deployment. High priority was given to process definitions because they are required to ***verify completeness*** of the functional requirements. A typical project misses about **30-40%** of the needed requirement specifications if the baselined operations processes are not defined and documented. This is the primary reason that most requirement documents *do not include throughput and quality objectives* since they can be specified only in the context of an operations process.

Initially all business process changes required software development that negatively impacted operational productivity and robustness. Currently, **table-driven state machines** control all order management processes. Most process changes (e.g., create new states, or make changes in the state transition matrix) can be implemented through CM updates.

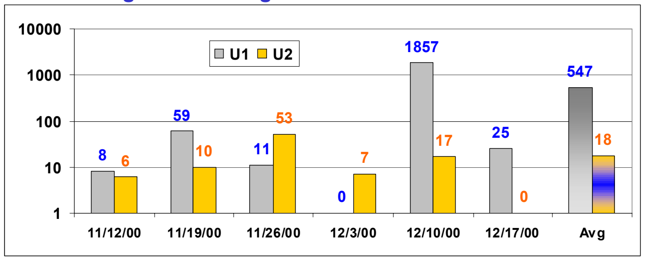
3.6. Center Productivity Management

The WCO is partitioned into 12 teams; each team is assigned a supervisor that has the role of center team administrator (CTA). Each team includes 10-14 Reps. Accounts are assigned to teams trying to load balance the center load across the teams. The CTA assigns each order to a specific Rep. It is also the CTA's responsibility to reassign orders between Reps if a particular Rep is absent or in over-load condition.
Periodically, changes in the business may result in an account transfer from one team to another to further balance the workload. Because of the unpredictable nature of the order work-load, a Rep from one team may be assigned temporarily

to assist another team for a period of time or to process a block of orders.

Figure 8 tracks the average time it takes a customer to **create a new order** and submit it for center processing. The report is also used to validate the *quality of the user interface* during requirements validation.

Figure 8: Average Create to Submit Interval



Users are able to submit a new order within 5-10 minutes taking advantage of the order copy-function that was developed primarily as a development tool. They discovered that a new order can be created more effectively by making a copy of a *completed order* and making a few required changes (orders from the same account naturally have a large number of fields in common).

Management reports are provided to track orders, clarifies, escalations, user sessions, business rules and process definitions. Similarly, the toolkit supports throughput, cycle time and quality reports at the user, team and the center levels.

The application currently supports about 150 validated reports. Based on the expertise we acquired, an average time to create a new report is about four hours; iterative validation with the users results in new reports ready to be deployed in the field within 2-3 calendar days. Getting feedback from key users is the gating factor for a new report deployment.

4. Conclusions

eccm has proven to be an effective toolkit to:

1. **Eliminate software bottlenecks in the delivery of business solutions**, *shifting the challenge from software development to change management.* Software releases are deployed monthly because of our limited ability to distribute changes effectively to a large number of physically distributed users. The development team still operates on weekly releases.

2. **Improve delivery of *relevant* and *complete* solutions to center users.** The cost of a software release has been reduced by at least two orders of magnitude compared to similar projects in the past. The weekly-release cycle time is on the average 10-20 times faster than on similar projects providing an environment in which newly acquired customer knowledge can be deployed in a timely fashion. The quality of an average release has improved by a factor of 50 using Severity 1 and 2 defects reported from the field. Software defects have not impacted either system availability or data integrity in operations.

3. **Enhance productivity at both the individual and team levels** using throughput, cycle time and quality analysis supported by real time and periodical reports. The center productivity improvements have delivered a return on the software investment of less than 18 months and a significant competitive advantage in the market place.

4. **Expand the life cycle of existing legacy systems** by eliminating the unnecessary churning of software features while concurrently improving their software ROI (e.g., using the *eccm* platform to validate business rules).

5. References

[1] B.W. Boehm, "Software Engineering Economics," Chapter 33, Prentice Hall (1981).

[2] R.S. Pressman, "Software Engineering A Practitioner's Approach," 5th Edition, McGraw Hill, 2001.

[3] S. McConnel, "Rapid Development," Microsoft Press, 1996.

[4] B.W. Boehm, "A Spiral Model of Software Development and Enhancement," IEEE Computer 21(5) pages 61-72 (1988).

[5] B.W. Boehm, "Software Management," IEEE Computer, September 2000, pages 94-97.

[6] B.W. Boehm, and V. Basili "Gaining Intellectual Control of Software Development," IEEE Computer, May 2000, pages 27-33.

[7] B.W. Boehm, and V. Basili "A COTS Based Systems - Top 10 List," IEEE Computer, May 2001, pages 91-93.

[8] J. Berry, "Tools Bring ROI Into Focus,"Internetweek.com, December 5, 2001.

[9] J. Berry, "Budgeting Reform Will Drag IT Along," Internetweek.com, December 10, 2001.

[10] P.J. Denning, "The Invisible Future," When IT Becomes a Profession, pages 295-325, McGraw Hill 2002.

[11] P.J. Denning, "The Core of the Third-Wave Professional," CACM November 2001, pages 21-25.

[12] M. Dertouzos, "The Unfinished Revolution," Harper Collins Business, 2001.

[13] G. Pour, M. Griss and M. Lutz, "The Push to make Software Engineering Respectable," IEEE Computer, May 2000, pages 35-43.

[14] Software Engineering Code of Ethics and Professional Practice (5.2), csciwww.etsu.edu/seeri/TheSECode.htm Sections 1.03, 2.06, 3.01, 3.04, **3.07, 3.08,** 3.09, 3.10, **5.04, 5.05,** 5.11, **6.07,** 6.08, 6.10, **6.12, 6.13** and 8.02.

[15] B. W. Boehm, "High Dependability Computing in a Competitive World," IEEE 26th SEW, November 2001.

[16] M. Lutz, "Software Engineering on Internet Time," IEEE Computer May 2001, page 36.

[17] R. Karpinsky, "Don't Get Nike-ed," Internet Week, March 7, 2001.

[18] M. Wagner, "Oracle Savings Don't Add Up," Internet Week, March 8, 2001.

[19] B. Latour, "Science in Action," Harvard Univ. Press 1987.

[20] R. Hellman, "The Future of the OS for Internet Applications," IEEE Computer, May 2000, pp. 12-15.

[21] D. Cohen, L. Berke, G. Larson, "Role of Interoperability in Business Application Development," ACM SIGMOD, May 1993, pages 487-490.

[22] D. Cohen, G. Larson, Bill Ware, "A Web Based Order Management Application," IEEE Proceedings of ICCNMC-01, Beijing, China, October 2001, pp. 27-34.

[23] M. Lehman, "Program Evolution: Processes of Software Change," Acad. Press, London, 1985.

[24] D. Cohen, G. Larson, Bill Ware, "Improving Software Investments Through Requirements Validation," IEEE Proceedings of 26th SEW IEEE/NASA, November 2001, pp. 106-114.