Software Process Improvement in Small Organizations: A Case Study

Kathleen Coleman Dangle, Patricia Larsen, and Michele Shaw, *Fraunhofer Center for Experimental Software Engineering*

Marvin V. Zelkowitz, *University of Maryland*

A case study of a five-year-old startup company looks at process improvement in the context of a small organization and why it's necessary for continued growth.

The Capability Maturity Model for Software (developed by Carnegie Mellon University's Software Engineering Institute[1]) has become a major force in software development process improvement. Companies strive to increase their CMM level from an initial level of 1 to levels 2 through 5 (see the sidebar). Each increase imposes stronger requirements on demonstrating that the organization is following a set of 18 *key process areas* (KPAs), which should improve the organization's capabilities to manage, develop, and deliver quality products.

We looked at the role of process improvement in the context of a small organization. Although the Capability Maturity Model Integration is replacing the CMM, we focused on the older CMM. We had to look at more than which CMM KPAs apply to small businesses. The problem became one of organizational culture and of helping the organization understand that, far from being an arbitrary set of rules, the CMM defines a framework that seeks to improve the development process regardless of company size.

The company's goal was to reach CMM Level 3 to secure new contracts. We used that goal as the hook to get inside the company to review their development practices. Although our overall goal was to institute good software development practices, we used the language of the CMM to gain the management's cooperation. Applying these practices is essential to managing growth, yet undertaking this effort without prior experience could impede a small company's innovative nature. Change management must be designed to fit the organization's culture and environment and presents challenges. This case study's purpose was to investigate the success factors of a software process improvement effort for a small software development organization.

The CMM in small organizations

Because of the requirement to be at a specified CMM level to bid on contracts, many organizations institute process improvement

The Capability Maturity Model

The CMM is based on a set of practices that organizations use to develop software products, and it organizes these practices into *key process areas*—for example, identifying configuration items such as designs, test cases, and source code. Controlling changes to those items and maintaining traceability of those items are common project activities; the CMM groups these configuration items in the Configuration Management KPA. Although every project must manage its set of artifacts, which artifacts are managed and how well organizations manage them vary widely.

The CMM rates organizations as to how well they succeed at meeting 18 KPAs, which are grouped into five categories or levels. An organization that succeeds at none of the KPAs is rated at the lowest level, called Level 1 (*Initial*).

Level 2 (*Repeatable*) consists of these KPAs:

- Requirements Management,
- Software Project Planning,
- Software Project Tracking and Oversight,
- Software Subcontract Management,
- Software Quality Management, and
- Software Configuration Management.

If an organization meets the goals of all six Level-2 KPAs, then the organization is rated as CMM Level 2.

Level 3 (*Defined*) consists of seven additional KPAs:

- Organization Process Focus,
- Organization Process Definition,
- Training Program,
- Integrated Software Management,
- Software Product Engineering,
- Intergroup Coordination, and
- Peer Reviews.

Level 4 (*Managed*) adds

- Quantitative Process Management and
- Software Quality Management.

Level 5 (*Optimized*) consists of all 18 KPAs, including

- Defect Prevention,
- Technology Change Management, and
- Process Change Management.

Fifty-two goals and 316 key practices (backed up by 500 pages of documentation) define the KPAs. An organization achieves configuration management, for example, not by using a specific tool but by demonstrating that it has implemented a process and institutionalized its use and that it manages its set of artifacts in a way commensurate with the goals of the Configuration Management KPA.[1]

Each level adds capabilities not present at a lower level, and an organization must meet lower-level KPAs before it can institute higher-level KPAs. One weakness in the CMM, however, is its focus on levels implied by assigning KPAs to specific levels. With a goal of achieving a particular CMM rating and a time frame often exceeding a year to succeed at meeting the required KPAs, organizations often ignore the KPAs for higher levels until they're trying to achieve those levels. This can seriously delay their progress as they develop processes one level at a time.

A second limitation is that the CMM's developers assigned KPAs to levels intuitively. Measurement is often considered in the Quantitative Process Management KPA (Level 4), so the CMM doesn't consider data collection important until the organization is fairly far along in its process improvement program. However, other approaches view measurement as important, and collecting data is often the first activity instituted in a process improvement program.[2] The Capability Maturity Model Integration has improved the area of measurement with the addition of the Measurement and Analysis KPA at Level 2.

Higher CMM levels generally improve software development for many organizations, and many organizations view the CMM as an important milestone. Requests for proposals often require a CMM level of 2 or higher to participate. So, adherence to the CMM is an important goal for many companies.

References

1. P. Byrnes and M. Phillips, *Software Capability Evaluation, Version 3.0, Method Description*, tech. report CMU/SEI-96-TR-002, Software Eng. Inst., Carnegie Mellon Univ., 1996; www.sei.cmu.edu/pub/documents/96.reports/pdf/tr002.96.pdf.
2. V. Basili et al., "Lessons Learned from 25 Years of Process Improvement: The Rise and Fall of the NASA Software Engineering Laboratory," *Proc. 24th Int'l Conf. Software Eng.* (ICSE 02), IEEE CS Press, 2002, pp. 69–79.

programs to raise their CMM levels. Organizations often view CMM Level 3 as an important milestone in these improvement programs. Many are content to remain at this level, while others want to achieve a rating of 5.

However, each KPA entails

- requirements to define processes, record results of executed processes, and provide evidence of achievement of that KPA;
- paying infrastructure costs for defining and instituting new processes for specific KPAs and for related training; and

If the primary activity isn't software development, they might not understand what software process improvement requires.

- overseeing the process to make sure that the organization follows the KPAs.

In addition, technology gurus in the organization must monitor the landscape for new applicable technologies and import them into the organization.[2] For an organization with hundreds of software developers, the costs of reporting requirements and hiring individuals to import new technology are manageable. Most large organizations have (either enthusiastically or reluctantly) built software process groups to manage their process improvement activities.

A small company, however, doesn't have the resources to build a software process group. A company with 30 employees might have only four or five developers in addition to management, business, finance, and administration staff. If the company's primary activity isn't software development—that is, if they produce a product or service that incidentally uses a computer—they might not really understand what software process improvement requires. So, we need to address how such a company can improve its development activities.

Researchers have proposed two process improvement approaches for small companies. Mark Paulk states that all companies, large and small, are interested in achieving CMM goals.[3] "Are meeting schedules, budgets, and requirements important to small projects? To small organizations?" He argues that the CMM satisfies all organizations' needs but that small organizations must adapt the KPA requirements: "The CMM is a tool that should be used in the context of a systematic approach to software process improvement."[3] Paulk also states that a "reasonable interpretation of the CMM practices must be made in any context in which the CMM is applied. … [T]he CMM was written to address the process for large, complex software efforts." He adds that "informed professional judgment" is essential when the context and the business environment differ.[3] The key to Paulk's approach is to recognize that KPAs are informative. While the concepts are appropriate as written for large organizations, KPA-specific practices are still relevant to small organizations if adapted to achieve adherence to the objectives in a simpler way. Adhering to the CMM, then, means applying the KPA-specific goals in a way that meets the small organization's needs.

Judith Brodman and Donna Johnson's approach is similar.[4] They want to produce a tailored CMM to allow for alternative practices that are applicable to a wide set of software organizations. For example, in the Software Configuration Management KPA, the CMM requires an organization to write an SCM policy, create an SCM plan for each project, and provide software quality assurance oversight to ensure the organization follows the plan. For a small organization, this additional work seems overwhelming. Brodman and Johnson revised the SCM KPA to let organizations combine the SCM plan with additional CMM-related documents to lessen the workload. As a result, their work modified 82 percent of the KPAs.

Although both of these studies address how to apply the CMM to meet small businesses' needs, neither addresses why an organization needs the CMM. If a small organization is successful (that is, profitable) and isn't required to follow any KPAs, what's the incentive to undertake a process improvement program?

A case study

In 1999, DataStream Content Solutions (DSCS) began providing a service that converts large data files from one format to another. Some of their contracts require immediate data turnaround (for example, within two hours), while others require them to convert data at set intervals (for example, monthly). From an initial staff of four, they've grown to 28, with plans to almost double in size over the next few years.

As with many small companies, the founders were experts in their application domain with little experience in computer software technology. From 1999 until 2003, they expanded their business, with the company president providing marketing support and the chief programmer providing programming and project management support. By early 2003, they realized that their company had grown too large to continue handling new business in this manner. Both realized that they needed to change their business model, but interestingly, each had a different reason for the change.

Why change a successful business?

At DSCS, the president secured contracts and the chief programmer built each system. Each system had the same basic architecture: read in textual data, error-check and convert the data, and write out text in a new format. Once the programmer had written the program, he gave it to an analyst, whose job was to process each day's textual input and produce the required output.

As the company grew, its founders gave little thought to development processes. An error during a day's run meant that either the input text or the software contained an error. The company handled each error the same way. If the input data was incorrect, the analyst could change it and continue processing. If fixing the data wasn't possible, the chief programmer had to fix and debug the source program. Because some contracts required the company to process data by 7:30 each morning, defects could be stressful for the staff. The company didn't keep information on program defects because they assumed all problems to be data errors and changed either the incorrect input for each occurrence or the program to routinely handle the incorrect data.

Both the president and chief programmer saw the need for change, but neither had an appreciation of why they needed to change and what change meant. The president wanted the organization to be compliant with CMM Level 3. New contracts might be available from customers who insisted on CMM Level 3 from their suppliers. Such a rating seemed like a good discriminator for a small company in their market niche.

On the other hand, the chief programmer saw a business with multiple independent systems. Each time a program failed, he had to modify and debug the program. He was an expert in the application domain but not in software development. Unlike large organizations with many software professionals, this company had no gatekeeper who understood software development, kept abreast of changing technology trends, and could import new technology to improve the company's software activities. The chief programmer saw himself as the potential single point of failure. For example, when he learned about syntax processing, context-free grammars, and programming tools such as Lex and Yacc, he was amazed at how much easier and more uniform they could make textual processing. He believed that a process improvement approach should eliminate the multiple versions of software conversion tools that the company was developing and would simplify development.

More important, the chief programmer seemed at a loss for handling some basic business decisions. As part of securing a contract, the president often needed an estimate of the effort involved in building another system. The programmer had little experience in how to estimate this effort because the development process was hidden from view.

As the company grew, it hired additional programmers and analysts, who were experts only in their particular sets of programs. If problems developed, only that system's developer could address a fix. The chief programmer was now on everyone's critical path. He wanted to change the system so that he wasn't the bottleneck on all activities. It was clear that the organization would soon become unmanageable as they added more systems to the mix.

Both the president and chief programmer saw the need for change, but neither had an appreciation of why they needed to change and what change meant.

Initial changes: Preparing to implement the CMM

As part of the State of Maryland program Maryland Industrial Partnerships, we at the University of Maryland and the Fraunhofer Center for Experimental Software Engineering, Maryland, received a grant to work with DSCS to enable the organization to manage an increasing workload as they expanded.

In analyzing DSCS, we observed three separate areas that the company needed to resolve to help themselves grow. All three were intermingled in the minds of the company staff, and separating them into distinct areas helped address specific needs.

System architecture. DSCS needed to manage the data they processed. They needed central servers to centralize data access, and they needed to archive files and institute a backup plan to ensure the integrity of their data and programs. If a programmer made an incorrect change to a program, undoing the change and reverting to an earlier version was difficult. The company needed a flexible network to ensure that programmers and analysts could access the data from anywhere in the company.

DSCS recognized the system architecture problem, so they hired a consultant to redesign their hardware architecture. The company installed central servers for the analysts to save data on, so each system is no longer on a separate machine. These centralized servers have facilitated the distribution of information among employees primarily as a configuration management and process asset library (PAL).

The company originally thought that this

In developing a plan for DSCS, our goal was to meet corporate needs rather than to strictly adhere to CMM precepts.

change would solve their development problems as well. We helped them see that their problems included the two additional areas.

Software architecture. Most DSCS programs have a similar function: converting textual data. We saw that redesigning their software to use a common structure (for example, using Yacc to parse input data) should help minimize their architecture and let programmers handle multiple programs more easily. They had already realized that some of their output formats were in XML and that error-checking all data and converting it first to XML would help create a single set of programs to produce their output formats.

Developing a new hardware profile had little impact on the DSCS software architecture. They initially believed that Fraunhofer would help develop a common architecture as part of the process improvement plan. Only after we discussed some initial changes and started to develop the plan did they see the difference between software architectural design and a software development improvement plan. They now see how software requirements specification, a software design document, and a software test plan can help them develop new software and how configuration management practices can help with version control, software changes, and file backup and archiving. They see redesigning their systems as separate from their process improvement work and plan to redesign them later.

Development process. The company needed a set of development practices that would make their activities more tractable, visible, and understandable. They were right in looking to the CMM as a mechanism to achieve this, but probably for the wrong reason. The CMM should formalize this process because it requires companies to create documents that track development artifacts: designs, source code, test plans, and so on. However, it wouldn't address the chief programmer's problem of specifying which framework would be the best software architecture to process their data. The CMM would address basic project management processes to more efficiently and effectively track costs, schedule, and functionality—necessary for any business. Our goal was to institute a process improvement plan that was based on the CMM and applicable to a small company.

Key implementation factors

In developing a plan for DSCS, our goal was to meet corporate needs rather than to strictly adhere to CMM precepts. The first two concepts that seemed most critical were formal reviews and a configuration management plan. We addressed several key implementation factors.

Recognition of key aspects. The management team understood the differences among system architecture, software architecture, and supporting organizational and development processes. Once the chief programmer understood these differences, DSCS made real progress. (It's difficult to implement process improvement by limiting yourself to the development process. You need to make organizational changes such as committing resources to complete the project, creating a PAL, and establishing process definition standards and terminology.)

Commitment of resources. The company made progress on developing practices and software process improvement after assigning a dedicated person to this effort who was neither the chief programmer nor the president.

Process Asset Library (experience base). DSCS spent time and effort to create a process assets repository for organizational and project information that the process improvement efforts generated to communicate the process assets to the organization. A Web interface on the library makes it easier for end users to locate information. Also, library links are included in work products.

Periodic reviews. DSCS instituted monthly meetings to review the status of projects and changes. These reviews provide a forum for key personnel to spread the word about process improvement and to implement the defined practices. Participants use the PAL to access data at these meetings (action items, agenda, minutes, work products), which lends credibility to this information's value. Management can review, approve, and baseline the defined practices during these reviews. Through participation in periodic reviews, the company president demonstrates support of the process improvement initiative and communicates the effort's status and importance. We should have started these reviews earlier in our activities with DSCS. Ideally, this group would be more like a senior management steering committee; however, with a small organization, this one meeting can serve several purposes.

Terminology and process definition standards. Key personnel are beginning to understand the need to use consistent terms and definitions and document them in a glossary. It took some time for staff to understand the basic components of the process information that they needed before implementing a best practice, including basic process definition, process flow, templates, guidelines and criteria, and roles. The key now is getting that understanding to the rest of the organization.

Other factors

In addition to implementation factors, DSCS addressed two process and development activities that provide overall visibility into the activities. One is managing workflow. The company is implementing a workflow management tool (Workpoint) that will make workflows more visible. As DSCS receives, error-checks, processes, and converts data, the workflow manager will keep track of the status. This will let the organization understand where they are in the process and where any bottlenecks are.

DSCS also needs to track unplanned work. They plan to use PR-Tracker to capture problems, action items, risks, and changes as work progresses and to manage those through to completion. One goal is to identify costs associated with this unplanned work to ensure that data is available to understand a change's cost and to assist in deciding which changes to approve.

Success factors

After one year of this project, major cultural changes are under way. The company is replacing the vision that achieving CMM Level 3 will solve both their marketing and development problems with more realistic goals. They now understand that they need to address system architecture, software architecture, and development processes if they are to grow successfully. Their revised goal is to have new development processes in place so that they can fully realize gains next year. They're beginning to rewrite existing programs to use a common software architecture, but it's no longer an immediate goal and hasn't made any measurable impact so far. They realize that they first need to understand and manage their development process and understand their workflow to track data through their organization before they address this common software architecture.

In looking at the changes in the company over the past year, we can divide success factors into two categories: process related (activities based on specific KPAs) and organizational (activities outside the CMM that DSCS needed to address to achieve success).

Process-related

To help a small organization achieve CMM Level 3, we had to develop a strategy. Resources (time, personnel, and money) were limited. Rather than just addressing KPAs, we first developed an overall set of principles:

- Process isn't just for the sake of process; rather, it should address the organization's business goals and not just CMM compliance goals. For DSCS, we tried to implement a set of software engineering best practices.
- Order your process development and implementation activities on the basis of benefits and process groupings that support your particular needs, rather than being strictly KPA focused.
- You can't sustain process improvement without dedicated resources (even if only part-time).
- First test newly developed processes on selected projects, then refine these processes on the basis of the lessons you learned and expand them to other projects in the organization.
- Plan the introduction of new processes; consideration for ongoing activities is a determining factor.

While achieving CMM Level 3 was one driving force behind our activity, we tried to focus our work on the practices that would promote the company's success.

Organizational

We identified four goals we wanted the company to achieve and 10 areas from the CMM that would best help meet those goals. We plan to track these over the next year and see how they contribute to the company's success.

The first goal was to improve the company's ability to accurately forecast software develop-

To help a small organization achieve CMM Level 3, we had to develop a strategy.

DSCS needs a software technology gatekeeper to understand new technology. This small company has limited resources but now understands this need.

ment projects' costs and schedules. To accomplish this, DSCS will use project plans that include work breakdown structure, schedule, and estimates for initial projects and implement a change control process to manage scope, schedule, and cost changes to existing projects.

The second goal was to improve the company's ability to reduce software development time to market. DSCS will define software cost, schedule, and progress metrics and monitor them to see if they're consistent with plans.

The third goal was to improve the company's ability to achieve competitive-edge quality. DSCS will define industry best practices to support and manage the rapid growth of the company's operations and to position the company for future procurements. They also plan to identify roles and responsibilities to allow for quicker, better decision making.

And finally, the fourth goal was to achieve CMM Level 3 on a fast track through templates, train-the-trainer materials, access to experts, and other resources. DSCS will regularly conduct senior management and project reviews, assign responsibility for the process improvement effort, and track and manage an improvement plan. They'll also establish, manage, and improve a process improvement infrastructure containing process artifacts such as training material, templates, meeting minutes, and other resources. Finally, they will perform Software Capability Evaluation to support and manage the rapid growth of the company's operations, which positions the company for future procurements.

Although CMM Level 3 was goal 4, not all of our efforts were directed toward Level-2 KPAs (for example, goal 1 is often considered a higher-maturity practice).

Results and conclusions

After 18 months of working with DSCS, we can measure cultural changes in the company. We can also see our impact on the company, and we've learned what we can do better.

Cultural changes

The company's leaders now realize that because the company is too large for one manager to do it all, they must delegate tasks. All three separate, distinct areas (system architecture, software architecture, and development processes) must work in unison to support corporate goals. The DSCS president finally realized that he was behind the eight ball in trying to accomplish his work with processes as usual. The company's executives committed to providing the resources needed to position the company for further growth. Risk at the company has been reduced. The chief programmer no longer makes all software decisions. New hardware is in place, and backups of critical data files occur regularly.

Providing process structure and the ability to understand, manage, and measure development risks has a cost, and DSCS's leaders are willing to spend the funds to achieve those goals. Guesswork has been removed. Frequent reviews reduce stress. Management, technical staff, and business development now share a common view, with CMM activities not pigeonholed only for software development.

DSCS needs a software technology gatekeeper to understand new technology. This small company has limited resources but now understands this need. Also, DSCS has refined decision making and responsibilities and made them more explicit. The company president provides corporate objectives for the overall business. The chief programmer provides software vision. Previously he did everything, so defining his role was difficult. Being unable to clearly articulate his job, he was unable to delegate portions of it. Additionally, the company hired a software change agent to provide software development and project management vision. Outside experts (University of Maryland, Fraunhofer Center, and others) provide knowledge of the technology landscape. Small companies can't afford CMM by themselves, and most DSCS staff aren't software professionals.

The company deals with a process to perform a service for their customers. So, addressing processes for software development was a natural extension of their usual work activities.

Employee morale is higher. Some have been able to get more responsibility, while others, such as the chief programmer, were able to offload some of theirs.

Impact on DSCS

Because of our efforts, more than one developer can work on developing software. DSCS has established repeatable processes that more than one person knows. All employees clearly understand what they need to do their jobs. DSCS is able to create software more quickly and cost effectively.

Process improvement takes time. DSCS won't meet the unrealistic goal of achieving CMM Level 3 in one year. However, they now realize that such an achievement takes time and isn't their ultimate goal. The company has instituted a workflow management process, reviews, and configuration management. DSCS now recognizes the value of process improvement, making them realistically able to sustain further growth.

Lessons learned

Not everything worked as well as we'd planned. Lessons we've learned include these:

- Divide and conquer. Organizations could improve more quickly if they assigned process improvement leads or owners who are accountable for focusing on specific improvement areas. DSCS didn't have such leads, which delayed progress on the Configuration Management KPA. We initially targeted this area, but improvements didn't come until the end of the first year.
- Ensure process improvement resources are committed. At the project's beginning, we depended on the chief programmer to serve as the process improvement implementer. Unfortunately, as the primary expert in the organization, he was already resource constrained. So, we made little progress.
- Start formal reviews immediately. We should have instituted key review meetings from the beginning to keep management not only informed but engaged. Better communication at this level would have ensured earlier buy-in that our approach was beneficial to DSCS.
- Assess staff process improvement experience. The individual assigned to support the process improvement effort didn't have much process improvement experience. We had to do much more training and hand-holding for the process improvement staff than we expected.

DSCS is well on its way to taking a more structured approach toward software development. Both the company president and chief programmer have seen the improvement in project tracking and oversight. DSCS is implementing many CMM-related key practices, and while the overall goal of reaching CMM Level 3 isn't immediate, the organization should be able to achieve it in the near future. We've secured funding for a second year, and DSCS's president wants to extend our activities across all company activities.

Acknowledgments

Maryland Industrial Partnerships project 3225 at the University of Maryland sponsored this work. We greatly appreciate the help of Ed Schulke and Mark Anstey of DataStream Content Solutions in implementing the process improvement program this article describes.

References

1. M.C. Paulk et al., *Capability Maturity Model for Software*, tech. report CMU/SEI-93-TR-24, DTIC number ADA263403, Software Eng. Inst., Carnegie Mellon Univ., 1993; www.sei.cmu.edu/pub/documents/93.reports/pdf/tr24.93.pdf.
2. T.J. Allen, *Managing the Flow of Technology*, MIT Press, 1977.
3. M.C. Paulk, "Using the Software CMM in Small Organizations," *Proc. 16th Ann. Pacific Northwest Software Quality Conf. and 8th Int'l Conf. Software Quality*, American Soc. for Quality, 1998, pp. 350–361.
4. J.G. Brodman and D.L. Johnson, "What Small Businesses and Small Organizations Say about the CMM," *Proc. 16th Int'l. Conf. Software Eng.* (ICSE 16), IEEE Press, 1994, pp. 331–340.

About the Authors

Kathleen Coleman Dangle is a scientist at the Fraunhofer Center for Experimental Software Engineering, Maryland, a not-for-profit affiliate of the Computer Science Department of the University of Maryland, College Park. Her research interests are in software measurement, process improvement, and software acquisition. She received her MBA from the University of Maryland, College Park. Contact her at the Fraunhofer Center for Experimental Software Eng., 4321 Hartwick Rd., #500, College Park, MD 20740; dangle@fc-md.umd.edu.

Patricia Larsen is the program manager for the Acquisition Improvement Program in the Office of Procurement at US Customs and Border Protection. A former scientist at the Fraunhofer Center for Experimental Software Engineering, Maryland, she was responsible for designing, implementing, and evaluating software process improvement programs. She has a BA in psychology and sociology. Contact her at US Customs and Border Protection, Office of Procurement, 1331 Pennsylvania Ave. NW, Suite 1310, Washington, D.C. 20004; patricia.larsen@associates.dhs.gov.

Michele Shaw supports process improvement clients at the Fraunhofer Center for Experimental Software Engineering, Maryland. She has over 23 years of experience in information technology. Her research interests include process improvement in small organizations and knowledge management. She received her master's in applied behavioral science from Johns Hopkins University. Contact her at the Fraunhofer Center for Experimental Software Eng., 4321 Hartwick Rd., #500, College Park, MD 20740; mshaw@fc-md.umd.edu.

Marvin V. Zelkowitz is a professor in the University of Maryland's Computer Science Department and Institute for Advanced Computer Studies and the chief scientist at the Fraunhofer Center for Experimental Software Engineering, Maryland. His research interests are in technology transfer and experimental software engineering. He received his PhD in computer science from Cornell University. He is an IEEE fellow and a past chair of ACM SIGSOFT and the IEEE Computer Society Technical Council on Software Engineering. Contact him at the Computer Science Dept., Univ. of Maryland, College Park, MD 20742; marv@zelkowitz.com.