Managing the Academic Computing Infrastructure in the Age of Automation

Shaun M. Lynch, Ph.D.
Department of Mathematics and Computer Science
University of Wisconsin-Superior
Superior, WI 54880
slynch@uwsuper.edu

Abstract

The computing infrastructure of the Department of Mathematics and Computer Science at the University of Wisconsin-Superior has evolved from a system that depended primarily on manual effort, to a system that relies heavily on automation. No single decision mandated automation; instead, automation came about as means to manage complex deployments, provide new capabilities, and utilize scarce resources. The computing infrastructure fulfills an operational need and serves as an educational resource, but the transition has incurred tradeoffs and unintended consequences. This paper presents the author's observations and experiences now that automation plays a predominate role in the computing infrastructure. Relevant factors provide the context in which priorities changed and automation arose to address managerial challenges. The paper highlights abilities, skills, and technologies that drive the move to automate systems. Finally, examples illustrate how automation has shifted the balance between the operational and educational roles the system has traditionally fulfilled.

1 Introduction

The author began thinking about the impact of automation after reading the article "Crash: How Computers are Setting Us Up for Disaster" by Tim Harford [1]. The article describes the crash of Air France 447 and factors that led pilots to misinterpret flight controls and lose control of the aircraft after the aircraft downgraded modes from autopilot to one that required manual intervention in challenging flight conditions. The article refers to problem called "the paradox of automation" where pilots become accustom to working with and overseeing automated systems while essential piloting skills erode as the amount of manual flight time decreases.

As the administrator of the department's computing infrastructure, this article resonated in such a way that it raised a number of questions: Is the department's computing infrastructure in jeopardy due to efforts to automate systems? Is automation eroding or even depriving student lab assistants of the learning experience they need to be successful? Are we setting ourselves up in a way that is unrecoverable should personnel not be available? What happens when there is a systematic problem in the automated systems? These are difficult questions to answer, but the author's observations over the past year suggest there is merit in asking these questions and seeking answers.

Since its inception in the early 2000's, the computing infrastructure of the Department of Mathematics and Computer Science at the University of Wisconsin-Superior has evolved from a system that depended primarily on manual effort to configure and operate, to a system that relies heavily on automation to perform its intended function. There was no single decision mandating the automation of the infrastructure; instead, automation came about as means to manage increasingly complex deployments, provide new capabilities to faculty and students, and utilize ever-scarcer resources and an unpredictable talent pool. The transition has not been benign leading to tradeoffs and unintended consequences that would have been difficult to predict even a few years ago.

The department's computing infrastructure is an academic system that serves dual roles. First, the system fulfills an operational need to provide students and faculty the computing resources required for classroom activities, such as software development, virtualization, productivity, multimedia, *et cetera*. Second, the system is an educational resource that directly involves students in its operations to include development, deployment, and maintenance of relevant technologies. Over time, the forces that encourage automation have swayed the balance between these two roles. Reconciling changes with economic and academic conditions is difficult, but one that faculty and system administrators need to contend with.

This paper presents the author's observations and experiences now that automation plays a predominate role in the department's computing infrastructure. To begin with, the author provides a brief background along with a history of the department's effort to host its own computing infrastructure. Next, relevant factors provide the context in which priorities changed and automation arose as a means to address managerial challenges. Then the paper highlights the abilities, skills, and technologies that seem to drive the move to create automated systems. Finally, examples are provided that illustrate how automation has shifted the balance between the operational and educational roles the system has traditionally fulfilled. Particular attention is given to the positive and negative impacts on managing the infrastructure as well as attracting and involving students in its operations. The paper closes with thoughts on the future of department's hosting their own computing infrastructures and the challenges they face.

2 Background and History

The Department of Mathematics and Computer Science began hosting its own computing facilities around 2002 as a dedicated computer lab for the Computer Science program. Referred to as the Development Lab, the facility contained 20 computer workstations, four servers dedicated to the self-contained network infrastructure, and a pool of five application servers. The lab served both operational and academic purposes within in the department. First, the lab provided specialized computing resources needed for department programs beyond those found in the general computing laboratories available on campus and serviced by the University's Information Technology Department. Second, the lab provided opportunities for students working as lab managers to develop skills deploying and maintaining the hardware, software, and networking infrastructure.

At the onset of the Development Lab, nearly every aspect of the lab's operations were the result of manual effort to deploy, configure, and operate the systems necessary to provide the required functionality. The lab often had two lab managers working between 10-20 hours a week to ensure the lab remained in working order. The start of each semester often brought a surge of activity to install new operating systems, refresh the configuration on individual workstations, add and remove workstation and network applications, and enroll users in the Active Directory environment. Attempting to maintain the system's configuration during the semester was on ongoing challenge since it often meant working with each individual piece of gear. Any automation in the lab was at a rudimentary level and usually involved creating batch files used to perform selected tasks across systems.

Around 2008-2009, the architect of the Development Lab pursued a new employment opportunity resulting in a loss of leadership and vision for the lab. The computing infrastructure languished over the next few years as equipment failures, technological obsolesce, and deteriorating facilities began to take a toll. Several talented students serving as lab managers were able to keep the lab operating during this period despite the challenging conditions. Ironically, the situation created a real learning opportunity for the student managers who found workarounds and improvised solutions to the increasing number of problems occurring in the infrastructure.

Early in 2011 as the department was on the cusp of making a long overdue move to a newly constructed academic building (Swenson Hall), one of the department's most talented computer lab managers proposed a new server and network infrastructure. This one idea spurred an effort to consolidate separate threads of activity into a single cohesive architecture using server virtualization and failover clustering to combine services essential to the system operations. The change brought a critically needed vision and a sizable expansion of the computing infrastructure that fit nicely with the space the new facility provided. Overall, the computing infrastructure nearly tripled in size to serve a broader range of students and classes enabling the department to advance its teaching and research mission.

The author essentially abandoned the Development Lab in the move to Swenson Hall in 2011. Fifty new workstations replaced the hodgepodge of workstations in the old lab and a new server and network infrastructure along with updated operating systems were in place within the year. The change was so extensive and complete that many of the old practices no longer applied after the transition. The virtualization platform that resided at the core of the new computing infrastructure offered much greater flexibility and could easily scale to meet the demands of the department.

3 Changing Priorities and Automation

The computing infrastructure of the department effectively moved from a cottage industry to a full-fledged production environment within the span of a year. The combination of a highly virtualized environment, Active Directory integration of Linux systems, new operating system capabilities and server applications, and automation tools launched a cycle of expanding capabilities along with escalating expectations. The new emphasis on production disrupted the historical balance between the operational and academic priorities.

Any number of factors could have set the automation process in motion; however, the author has identified three reasons that seem most relevant. First, significant budget reductions since 2008 led to funding cuts for lab assistants. In the early years of the Development Lab, resources for up to two students working 10-20 hours per week were budgeted. Federal work-study programs and student assistant funds provided compensation that was competitive with external opportunities. By 2012, student assist funds had all but vanished. This limited the talent pool to only those with federal work-study grants. Unfortunately, many of the grants were not enough to provide competitive compensation and enough hours necessary to support the infrastructure. Automating routine lab activities seemed to be a reasonable way to compensate for the reduction in student labor due to budget cuts.

Second, the new computing infrastructure saw a dramatic increase in complexity. For example, virtualization, fail-over clustering, IPv4-IPv6 dual-stack networking, and Active Directory integration are highly entwined systems that require personnel to have a deep understanding of the interaction between the various subsystems, components, and protocols to ensure proper functionality. Newly hired computer lab assistants face a daunting learning curve in order to become familiar with the technologies used on a daily basis in the infrastructure. Again, automation offered a way to systematize many of these systems to help manage the underlying complexity and reduce the learning curve.

Third, the short-term approach students employ to complete activities does not always lend itself to a production-oriented environment. Developing and sustaining a computing infrastructure used for production is more akin to a marathon than a sprint. Issues related to documentation, standardization, reliability, and consistency require a sense where the system has been, where it is going, and the ability to manage configuration. Given that student lab assistants work a limited number of hours over a period of one to three years, they rarely get enough exposure to grasp the system's configuration in its entirety. In this regard, automating processes helps capture and codify configuration details to ensure activities are documented and consistent.

Using automation to overcome managerial challenges essentially shifts problems from one domain to another. In many ways, automation itself now manifests all the same characteristics it was called upon to resolve. In other words, automation is time intensive, complex, and requires significant effort to manage its configuration. The upside is that when automation works, it does so consistently and requires little if any human intervention. Ironically, the ever-increasing number of automated systems has gradually displaced student participation instead of enhancing it; thus, furthering the imbalance between the academic and operational roles of the computing infrastructure.

4 Essential Automation Abilities, Skills, and Technologies

The author's efforts to automate processes arose organically to solve a series of complex managerial challenges. Automation is not new to the IT industry, which has embraced automation as a means to streamline processes and improve the economics of utilizing information technology. As automation became a normal part of the operations of the department's computing infrastructure, certain abilities, skills, and technologies seemed to arise that differentiated those who automate systems from individuals satisfied with the status quo. In hindsight, the author and certain student lab assistants who routinely automate systems seem to have an inclination and see beyond the immediate tasks and formulate processes that achieve or better current outcomes.

Three steps seem to be critical to this endeavor. First, the individual has an ability to identify activities suited for automation. This ability may stem from intuition or a predisposition to formulate logical sequences that achieve certain results. It is not necessarily a skill learned through rote or instruction, but may be innate and honed through practice and experience. Those that excel in this step seem to naturally express this ability without prompting and pursue automation with zeal and enthusiasm.

Second, the individual is able to identify existing tools or create new technologies that can perform tasks within the logical sequence of activities. Success seems to rely upon a combination of innate ability and learned skill. Innate ability stems from natural curiosity that drives individuals to seek out and find solutions. What to do with the solutions seems to rely upon decision-making skills such as choosing a methodology, establishing criteria, evaluating alternatives, and selecting a course of action.

Third, the individual can utilize and exploit the technology to accomplish the tasks needed to achieve the desired result. This step very much relies on learned skills that enable individuals to implement solutions such as writing scripts, configuring a system, or deploying an application. Although some individuals are better at this than others are, most can learn how to utilize a technology to solve problems through instruction and practice.

Several key technologies surfaced as automation became integral part of the computing infrastructure. Scripting tools are undoubtedly the most important technology needed to automate systems providing a mechanism to codify knowledge, manage configuration, and deploy repetitive settings. Windows PowerShell is the primary scripting language used followed distantly by batch files and Visual Basic Scripts. PowerShell is Microsoft's most recent addition to scripting technologies and draws upon the .NET framework and the Common Information Model (CIM. Unlike other scripting technologies, PowerShell is object-oriented and facilitates access to object properties without having to parse strings returned by commands.

Next, a centralized mechanism to manage settings and configuration is essential to successfully automating processes. Microsoft Active Directory is the department's principle authentication and directory service provider; therefore, Group Policy is the natural and preferred choice for managing site configuration. For automation, Group Policy provides a platform in which to deploy settings and launch scripts across the enterprise with a minimal amount of manual intervention. Several Linux servers rely on the Active Directory infrastructure for authentication but are outside the bounds of Group Policy. However, applications such as Puppet and Chef provide configuration services for Linux-based hosts should the need arise. The department does use the open source application Firewall Builder to manage firewall settings across the Linux-based systems.

Rounding out the list of essential technologies is a centralized system for managing and deploying images. Microsoft Deployment Toolkit (MDT) provides image deployment and application installation services to systems hosting Microsoft Windows as their primary operating system. MDT facilitates the creation and management of Windows images and includes provisions for configuring roles, installing drivers and applications, and deploying customized settings. The platform enables automated, touch-free deployment of images to individual computers based on BIOS or network settings. In addition, MDT integrates with Microsoft System Center Configuration Manager (SCCM) to provide extended functionality for mass deployments in enterprise settings.

5 Automation in Practice

Historically, student lab managers in the Development Lab were actively involved in nearly every aspect of lab operations. The transition to a production-oriented environment in combination with budget cuts, increasing complexity, and operational changes spurred the use of automation to address managerial challenges. Although numerous systems have benefited from automation, several key areas directly affected the role of student lab assistants.

Account Management

Enrolling users is a time-sensitive activity that occurs at the beginning of every semester. Timeliness is important because students and instructors must have access to computing and information resources shortly after classes begin. Account management typically involves creating accounts, configuring account settings, resetting account passwords, and removing expired accounts along with dependent resources. Several different account types are available including guest, student, staff, privileged, and administrator accounts that offer users different services such as roaming profiles and network storage, and elevated privileges. A user may have multiple accounts to access different resources. Most accounts employ expiration dates and all student accounts expire at the end of each semester.

Approximately 12-15 classes use the advanced computing labs per semester. This is nearly double the use compared to the Development Lab. Several classes from the Information Technology and Systems program plus multiple sections of 100-level math and computer science general education courses represented the bulk of the increase. The broad range of students using the labs has dramatically increased the number of accounts needed. In total, students and staff use nearly 250 accounts per semester.

Configuring accounts manually for all practical purposes is infeasible given the resources available. A new system for creating and managing accounts was required that would allow accounts to loaded and configured in bulk. Scripts written in PowerShell performed basic account functions such as importing user lists, reopening existing or creating new accounts, assigning initial passwords, and notifying users. New scripts providing additional functionality were written and added to the library as needed. A second version of the script module was released in 2014 that included a number of refinements and new management capabilities.

Advantages: Student lab assistants can create, manage, and remove accounts by typing in one-line commands. Accounts are created and managed consistently and rules applied uniformly. Account management can be completed in a timely manner.

Disadvantages: Scripts mask backend account management processes and mechanisms. Student lab assistants do not interact with Active Directory objects such as user accounts, groups, and organizational units; group policy settings that enable roaming profiles and redirected folders; and file shares that host redirected folders. Rules are codified in scripts and may be unfamiliar to lab assistants making it difficult for them to make changes or fix problems should they arise.

Server Configuration

Configuring servers requires a considerable amount of planning and attention to detail. Prior to 2012, servers in the Deployment Lab were built directly on bare hardware and configured manually using a graphical user interface. Physical servers often hosted multiple roles and/or applications to provide a variety services needed within the lab infrastructure.

The new computing infrastructure operates eight physical servers, five of which function as virtualization platforms for nearly two dozen virtual servers. Virtualization adds another level of complexity by introducing issues such as provisioning, load balancing, clustered resources, failover, and resource utilization. Host systems generally do not include graphical user interfaces to minimize the attack surface; therefore, configuration requires settings to be entered at the command prompt. PowerShell is the primary tool used to configure servers hosting the Windows Server operating system. All production servers authenticate against Active Directory with the exception of a few Linux-based servers used as network appliances.

Approximately a third of the servers are mission-critical and must be operational for the computing infrastructure to perform its intended function. Mission-critical servers include two domain controllers, a management console, a DHCP server, two fail-over cluster nodes, and a fileserver for redirected folders. While non-mission critical servers can be deployed at any time, deploying mission-critical servers generally occurs during off-peak periods such as the summer months or the break between fall and spring semesters. Unfortunately, students are generally not available during these periods.

In 2012, the author created a PowerShell module containing a collection of scripts that automated the configuration of servers hosting Microsoft Windows Server 2012 (and later 2012 R2) operating system as described in the paper, *Automating Computing Infrastructure Configuration with Emphasis on System-Level Design and Integration* [2]. Settings declared in a XML file guides the configuration process and reduces build times to nearly one-seventh the time it takes to configure a server manually.

Advantages: Bring mission-critical servers online during off-peak periods in an efficient and expedient manner with a minimal number of personnel involved. Configure servers in a consistent manner, and model and test configurations prior to the actual deployment.

Disadvantages: Student lab assistants are not available during the deployment of mission critical servers and have difficulty participating in the underlying configuration due to scheduling issues.

Image Deployment and Workstation Configuration

The Development Lab contained 20 workstations for projects and instruction. Details regarding the deployment of the operating system and applications are unclear although the use of a sector-based imaging application is likely. If so, each workstation required additional configuration to include drivers or applications needed beyond the master image. In 2011, a classroom and lab modernization grant funded the purchase of 50 identical computers to replace the mixed-model computers in the Development Lab. The new computers were divided evenly between two advanced computer labs located in the new academic building.

The process of deploying workstation images effectively started all over after the move. The department adopted another sector-based imaging application called FOG (Free Open-source Ghosting) to clone and deploy images to the lab workstations. There was one important difference however; virtual machines hosted the master images to prevent hardware specific drivers from being loaded into the images. A 64-bit version of Windows 7 Enterprise served as the base operating system for the image along with Microsoft Office, Visual Studio, NetBeans, VMware Workstation, and a collection of utility applications.

Managing the image proved to be challenging; however, it spurred efforts to streamline the deployment of new applications. The lab assistant employed during this time was particularly adept at scripting batch files and finding applications to update the many utility programs installed. In the meantime, the author created a number of PowerShell scripts to automate the installation and removal of applications on the workstations. Fortunately, the image deployed in 2011 managed to survive until 2016 when it came time for a planned replacement of the lab computers.

Automating this portion of the computing infrastructure was long overdue. The effort to manage the piecemeal collection of scripts and the “sneaker-net” consumed the few resources available and the workstation images were beginning to deteriorate. In 2014, the author began evaluating other alternatives and eventually settled on using MDT (Microsoft Deployment Toolkit) to automate the deployment process. MDT is a major departure from traditional sector-based image cloning applications and offers numerous ways to build, capture, and deploy images. However, it is complicated.

Replacement computers for the two advanced computing labs arrived in time for Fall Semester 2016. However, the new deployment system was not yet operational leading to delays preparing the workstations for use in the computer labs. It took nearly two months to work out the initial details between MDT, Windows 10, and a greatly expanded pool of applications. Nonetheless, the lab workstations were imaged and ready for use by November of 2016.

The winter break between Fall Semester 2016 and Spring Semester 2017 offered time to complete a second iteration of the deployment model leading to improvements and expanded capabilities. First, the author modified MDT to deploy specific application combinations to individual workstations using a hierarchy of images embedded in a single Windows Image file. This change extended the deployment process to all 54 workstations located in the both advanced computer labs, hardware lab, and learning lab. Second, the author changed the image build procedure to create a touch-free deployment process. Reimaging a workstation launches at the press of a key without need for further intervention. Third, the author rationalized the image build process to release updated images at the beginning of each semester. This step minimizes the number of changes made to workstations over the course of the semester.

Deploying a pristine image each semester solves many problems, but it does not address the day-to-day abuse workstations receive over course of a semester. In the past, simple things like removing rouge icons from the public desktop, ensuring the designated desktop background theme exists, or updating application configuration files to reflect the latest settings all required manual intervention. This problem was resolved by creating a workstation configuration script that automatically performed basic cleanup tasks and synchronized application configuration files against a managed pool. Creating a Group Policy object that launches the script when the computer starts helps ensure proper workstation configuration.

Advantages: Image deployment and workstation configuration is essentially touch free and eliminates the need for an ongoing “sneaker net” to maintain the systems. Users receive a consistent desktop experience and applications work identically across all workstations regardless of location. Centralized image management and a flexible image build procedure accommodate configurations for the breadth of workstation uses in the department with a minimal amount of intervention.

Disadvantages: Lab assistants rarely interact with individual workstations any longer except in cases where there is a hardware problem or something unusual is going on. When problems do occur, they are often complex and require a lot of trouble shooting.

6 Concluding Thoughts

It is important to keep in mind that the department utilized automation to solve problems that were arising from external factors and changes in operational scope. Automation was not embarked upon for economic gain and there was no decision or advanced planning to use automation as a way to reduce labor or student involvement. To the contrary, automation helped solve difficult managerial problems to in hopes of preserving the academic component of the computing infrastructure. Sometimes hope and reality do not intersect as planned and a new path needs charting.

It is important to empathize with students in the age of automation. Learning new things and pursing interesting work motivates and engages students. Automation seems to have disrupted this balance and begs the following questions: How do you get students up to speed when the enterprise is nearly all automated? How would a student ever figure out how to automate a system if they do not have the opportunity to build one and practice? Is automation an end in itself?

This paper attempts to open a discussion, gathering ideas, and chart a course going forward. It draws on the author’s observations and concerns about a condition that is playing out not only in the department’s computing infrastructure, but also throughout the entire industry and arguably the whole economy. Viewing this problem from different perspectives highlights the complexity of the issue while examples provide evidence for trends and portrays nuances of the situation. Powerful tools like automation have the ability to solve problems as well as instigate disruptions. The challenge is to stay focused on goals and make course corrections. Tom Hartford [1] describes this eloquently in the article cited in the opening:

> *In principle, such technology should not fall victim to the paradox of automation. It should free up humans to do more interesting and varied work...*

References

[1] T. Hartford, “Crash: how computers are setting us up for disaster,” The Guardian, 11 October 2016. [Online]. Available: https://www.theguardian.com/technology/2016/oct/11/crash-how-computers-are-setting-us-up-disaster.

[2] S. M. Lynch, "Automating Computing Infrastructure Configuration with Emphasis on System-Level Design and Integration," in *Proceedings of the 46th Midwest Instruction and Computing Symposium*, La Crosse, WI, 2013.