REFERENCES CONT.

O'Neill, D. (May/June 2015.) "Software 2015: Situation Dire." Defense Advanced Technology and Logistics (DAT&L) Magazine. http://www.dau.mil/publications/DefenseATL/DATLFiles/May-Jun2015/O'Neill.pdf.
Royce, Winston W. (August 25–28, 1970.) "Managing the Development of Large Software Systems." Technical Papers of Western Electronic Show and Convention (WesCon). Los Angeles, Calif., USA.
Sheard, Sarah. (2015.) "Chapter 5: Complexity, Systems, and Software, 'Software Engineering in the Systems Context.'" Edited by Ivar Jacobson and Harold "Bud" Lawson. College Publications, King's College, London. ISBN 978-1-84890-76-6. 578 pages;
O'Neill, D. (1983.) "Integration Engineering Perspective." The Journal of Systems and Software, 3. 77-83. http://www.sciencedirect.com/science/article/pii/0164121283900067.
O'Donnell, Bob. (June 22, 2016.) "The Internet of Things is facing challenges with scale." Recode. http://www.recode.net/2016/6/22/11991414/internet-of-things-iot-challenges-scale.
O'Neill, D. (September/October 2011.) "Cyber Strategy, Analytics, and Tradeoffs: A Cyber Tactics Study." CrossTalk, The Journal of Defense Software Engineering. http://www.crosstalkonline.org/storage/issue-archives/2011/201109/201109-ONeill.pdf.
O'Neill, D. (March/April 2014.),"Software and Supply Chain Risk Management Assurance Framework." CrossTalk, The Journal of Defense Software Engineering. http://www.crosstalkonline.org/storage/issue-archives/2014/201403/201403-ONeill.pdf.
Koppel, T. (2015.) "Lights Out." Crown Publishing Group. ISBN 978-0-553-41996-2. 277 pages.
O'Neill, D. (1983.) "Integration Engineering Perspective." The Journal of Systems and Software, 3, 77-83. http://www.sciencedirect.com/science/article/pii/0164121283900067.
Larman, C. (2004.) "Agile & Iterative Development: A Manager's Guide." Pearson Education, Inc. ISBN 0-13-111155-8, 82-85.
O'Neill, D., Linger, R.C., Dyer, M. & Quinnan, R.E. (1980.) "The Management of Software Engineering." IBM Systems Journal, Vol. 19, Number 4, 414-477. http://www.research.ibm.com/journal/sj/.
Boehm, Barry. (2015.) "Chapter 6: Principles and Rationale for Successful Systems and Software Processes, 'Software Engineering in the Systems Context.'" Edited by Ivar Jacobson and Harold "Bud" Lawson. College Publications, King's College, London. ISBN 978-1-84890-76-6. 578 pages.
O'Neill, D. (March/April 2013.) "Technical Debt in the Code: Cost to Software Planning." Defense Advanced Technology and Logistics (DAT&L) Magazine. http://www.dau.mil/pubscats/ATL%20Docs/Mar_Apr_2013/O%27Neill.pdf.
O'Neill, D. (June 2009.) "Preparing the Ground for Next Generation Software Engineering." IEEE Reliability Society, Annual Technology Report 2008, 148-151.
O'Neill, D. (November/December 2014.) "Avoiding Proprietary Problems: A Software Clean-Room Method." Defense AT&L Magazine. http://www.dau.mil/publications/DefenseATL/DATLFiles/Nov-Dec2014/O'Neill.pdf.
O'Neill, D. (April 14, 2016.) "Integration Engineering in the Pursuit of Critical Infrastructure Resilience: A Unified Theory." White House Cyber Commission on Enhancing National Cybersecurity, Kickoff Meeting. http://www.nist.gov/cybercommission/upload/Meeting_Minutes_April_14.pdf.
Jacobson, I. & Lawson, H.B. (2015.) "Software Engineering in the Systems Context." Edited by Ivar Jacobson and Harold "Bud" Lawson. College Publications, King's College, London. ISBN 978-1-84890-76-6. 578 pages.
Lewis, Michael. (2014.) "Flash Boys: A Wall Street Revolt." W.W. Norton and Company, Ltd. ISBN 978-0-393-24466-3. 274 pages.

The Software Deployment Process and Automation

Dr. Nary Subramanian, Associate Professor of Computer Science, University of Texas at Tyler

Abstract. Software deployment is the last step in the software development life cycle. During deployment, control of the software transfers from the development team to the customer. After deployment, people in the customer organization will use the software as part of their jobs and derive economic benefits from the software. Any defects found in software post-deployment are resolved as part of the maintenance phase. The first step in mitigating user problems is the proper deployment of software. Software deployment is anything but trivial. Some enterprise software may take months, if not years, to completely deploy. Therefore, efficient software deployment will considerably shorten the deployment phase and save resources in terms of cost and labor. In this article, we explore typical models for software deployment. Based on these models, we develop a generic software deployment model, then identify deployment processes that lend themselves to further automation and may lead to an overall reduction in the deployment effort.

1. Introduction

Software deployment or installation represents the final handover of software from the development team to the customer. After successful deployment, the software system is finally operational so that the customer can benefit economically from its use. At the end of this deployment effort, the software development organization receives payment from the customer and the project is considered successful from both the developer's and the customer's viewpoints. However, software deployment is anything but trivial, depending on the scale of implementation. While a nontechnical person can install a desktop application by either installing a downloaded file or installing from a disk, a large-scale enterprise resource planning (ERP) system such as SAP may take several months — if not years — to be fully configured and ready to use [1, 2, 3].

A question one might have is why certain software deployments take a long time. Is it possible to shorten all deployments to the time it takes to install a desktop application? In this article we examine typical deployment models and discuss some answers to these questions. To answer these questions, we develop a generic deployment model based on typical deployment models, and this generic model will help us rationalize our answers. We also explore opportunities to automate some or all deployment activities.

What happens when, after successful software deployment, users notice defects (or bugs) during normal software operation? The customer reports these bugs to the software development organization's help desk. From there, the software enters its maintenance phase. But a prerequisite for a normal operational state is a successful deployment effort from the software deployment team. In this article, "software developer" refers to the organization that developed the software, while "customer" refers to the organization that has procured the software and will deploy it.

What happens when the deployment effort fails? The failure may have been due to misunderstood system configuration requirements on the customer's end. This means that improper or insufficient hardware resources (including CPU, memory and network bandwidth) were allocated, or that required software (including databases, servers and operating systems) was not provisioned. If the customer is not willing to accept incomplete software — that is, software that has defects or does not satisfy requirements — then the software deployment may need to be scrapped, the developer may not be paid any pending invoices, or (in extreme cases) delayed or failed deployment may lead to litigation[4]. In all cases, failed deployment leads to increased cost for both the software developer and the customer and to unhappy users at the customer organization. An example of this is the FBI's Sentinel project that incurred extensive time and budget overruns and still did not satisfy its users; [5] also, it has been reported that only 7 percent of ERP projects are expected to be deployed on time. [6] Proper understanding of the deployment effort and its success are essential before the software is useful to the customer.

2. Software Deployment Models

Figure 1 depicts the typical model for software deployment for small-scale software, such as software for home use or for a small company. The first step in this deployment is to verify that software meets the user's requirements, including functional requirements like features as well as nonfunctional requirements like performance and reliability. For custom-developed software, this is typically done by demonstrations and references. For COTS (commercial off-the-shelf) software, this is usually done by running trial freeware versions.

At this stage the customer also verifies the constraints imposed on the software, such as the hardware requirements and the operating system. If the software satisfies all requirements, then the customer compares its cost and features to similar software from other vendors and decides on a specific software and vendor. The customer then purchases the software and either downloads it or inserts a disc containing the software in a computer. The software executable is then run either by double-clicking, by issuing a command on the terminal, or by using a GUI-driven setup program. During this process, any activation keys issued by the software manufacturer need to be entered, and the customer must agree to the end-user license agreement. The software is then installed, which is followed by the almost inevitable reboot or restart of the computer. Upon restarting, the software asks for configuration information such as language, time zones, user information and so on. If the software requires access to other systems, like email or the Internet, then this integration information is also provided. For example, when an app is installed on a smart phone, integration authorization is often required. After this, the software is ready to use.

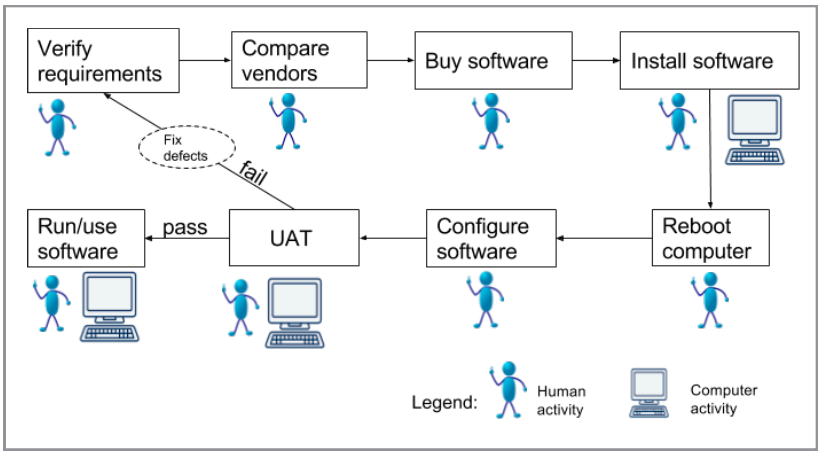

Figure 1. Deployment Process for Small-Scale Software

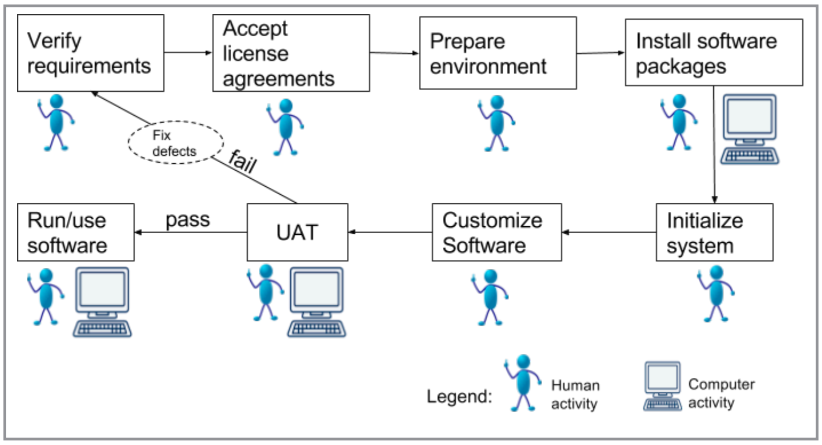

Figure 2. Deployment Process for an IBM Machine

However, the customer has to perform the user acceptance test (UAT) to ensure that the purchased software actually satisfies the requirements in the customer's computer. If UAT fails, the customer can contact the developer to fix defects or get a refund. However, if the UAT is successful, the customer can use the software.

Many of us have followed this deployment process when installing shrink-wrapped software on our computers for many years. Today, apps for smartphones are also installed by following most of the steps in Figure 1. As can be seen, the deployment process has several manual steps, and the computer-based activities are limited to the installation, testing, and execution of the software. Therefore, even though we tend to think of software installation as a one-click process, there is usually significant time spent waiting for the computer to complete executing the code based on our inputs. We also must perform several manual steps, which we tend not to include in the time it takes to install software for personal use or small office use. However, the whole process does not last more than an hour for most normal deployments.

Now let's consider the process for installing software on an IBM machine. [7] As mentioned in this reference, the whole process can take several days, if not weeks. The deployment process for the IBM machine may be abstracted, as shown in Figure 2.

For the IBM deployment process, the first step is to verify the hardware and system requirements for software. Typical system requirements include the ability to restart the system, ability to log in and out, and the ability to install fixes. Subsequently, the deployment team obtains the hardware and software resources needed for software deployment. Any license agreements that need to be signed before the software is installed must be completed, and the server must be prepared for installation, including prepared to accept the media used for software distribution. The distribution media used for software is then attached to or inserted into the system, and the OS and any licensed code for running the software are first installed. The software package is then installed, and the entire system is initialized so that the software package installation is complete. Another aspect covered during initialization is any integration with other systems, including collaboration systems (email, calendar) or authentication (for single login) or payment gateways. Software is then configured as required and tested by users. Any data migration or addition of critical mass of data also happen at this step. If defects are found during UAT, the developer is required to fix them. The cycle then repeats with any changes during defect-fixing incorporated in the process. If the UAT passes, then the software can be used normally. In the IBM deployment process and small-scale deployment, the installation, UAT, and normal use are automated to a large extent. Remaining steps require substantial human intervention.

Figure 3 shows the deployment process for a cloud-based system. For example, documentation for OpenStack, an open-source cloud platform, deployment may be seen in [8]. The typical deployment process starts again with understanding the hardware, software, and networking requirements for the software. These include considerations of average and peak CPU, RAM, disk, and network input and output requirements. To get a clear picture of these requirements, the deployment team needs input from the sales and marketing teams as well. For example, the sales team can provide the total number of expected users, while the marketing team can predict peak loads to expect during sales promotion activities. Using this information, the deployment team can compare the offerings of different cloud service providers. For example, the rates and standard flavors supplied by different cloud vendors need to be matched with the requirements. It is possible that none of the offerings are good enough. If so, the deployment team will have to install the software on a bare-metal server. Once the vendor and solution have been decided, the system will need to be procured and appropriate service-level agreements signed. Subsequently, either the provider or the deployment team will need to spin up the virtual machines, networks, and external gateways based on system requirements. After that, the software packages required are installed in the VMs. The system is then initialized, which includes connecting the application with its front and back ends. Again, during initialization, integration with any external systems (such as email servers, authentication servers or payment gateways) are also performed. Then comes the configuration phase, when the software is customized to be able to add users or customers. Also during configuration, any licenses for software to be used are also activated. UAT is then done on the software, and the developer fixes any defects that arise. The cycle then repeats from the requirements phase with the modified software. Finally, upon successful UAT, the software enters the production phase. In the cloud deployment, the installation, UAT, and normal use are also automated; the remaining steps are largely manual. Typical cloud deployment can take anywhere from a few hours to a few weeks [9] depending on the manual parts of the process and defect fixing.

3. The Generic Deployment Model

After analyzing the typical processes for deploying systems at three different scales, we can create a generic model for software deployment and discuss the reasons for frequent unsatisfactory deployments. We can also recommend ways to improve the situation. The generic model is shown in Figure 4.

The generic model has eight steps. The first step is the "verification" process, when the software requirements are established in terms of the hardware and software required for deploying the software. These include networking requirements, CPU, RAM, disk, backups, recovery processes, security appliances, and so on. In addition, a developer must identify the operating system, databases, servers, and other software requirements at this point. The next step in the deployment process is the "negotiation" phase, when the deployment team negotiates the best offer for both hardware and software from vendors or the IT team (if in-house). Any service-level agreements are also negotiated at this time. The third step is to "procure" the best possible solution. This may include ordering the items and getting them shipped and delivered for on-premise deployments or getting the appropriate third-party provider to set up the solution for hosted or cloud deployments. The fourth step is "installation," when the software environment (including the operating system, databases, servers, and the like) are installed on hardware or appropriate virtual machines spun off for cloud deployments. All required software packages are installed once the environment is installed. The fifth step is "initialization," when the software and hardware systems are started up and global settings for administration credentials, licenses, database schema, and the like are established. Any external network access for emails, the payment gateway or the Internet are also established. The sixth step is the "configuration" step, when the number of users, their access credentials, their authorizations, their memory restrictions, and so on are configured in the software. Any data migration from a legacy system, adding of sufficient data for users, and vendor licenses are also established in this step.
The seventh step is the "User Acceptance Testing (UAT)," when users test the software to ensure it satisfies its requirements including performance, security and reliability. If any defects are found during this step, then it is sent to the developer for fixing and the cycle restarts with the modified software. In the eighth and final step, the UAT is passed, the software is used for production, and economic benefits are derived.

As shown in Figure 4, most of the steps in the deployment model are manual. The only steps that are mostly automated are the installation, UAT, and use of the software. All the remaining steps require extensive human involvement. The step "Fixing Defects" is usually the responsibility of the developer, and therefore is outside the scope of the deployment organization. However, if defects are due to improper deployment in the steps so far, then the deployment organization must step in to fix them.

Table 1 gives the activities performed during each step of the generic deployment model and approximate timeframes. Each of the steps in this table is discussed in section 4.

3.1 Applying the Generic Deployment Model to Common Software Deployment

In this subsection we will apply the generic deployment model to the deployment of three common types of software: web applications, SaaS, and mobile applications. This discussion will allow us to understand the extent of this model's applicability.

A. Deploying Web Applications

There are several steps involved in the deployment of web applications[10]. Three layers characterize a typical web application: the "database layer" or back end, the "application layer" or business logic, and the web server that serves the web pages. Therefore, all three layers need to be understood during the verification step, including the product's brand name, its operating system, and the hardware's sizing requirements. During the negotiation phase, the software and hardware required for deployment are procured from either internal resources or external vendors, and any agreements for this procurement are signed. During the procurement phase, the software and hardware are obtained. For open source software such as LAMP (Linux, Apache, MariaDB or MySQL, and PHP) the software is downloaded into the hardware and installed. The operating system, database software, application software, and web server are first installed during installation phase, then the web application is also installed. During initialization, the empty database tables are created using scripts, super user is created, initial users for the system are registered, and the application is set to a known initial state. During configuration, access rights for users are set, authorizations are configured, any needed database migration is done, and any needed connections to payment gateways are established. During UAT, a small initial set of users test the system. If no defects are found, the system is open to all users. If defects are found, the process repeats from the stage where the defect originated.

B. Deploying SaaS Applications

The first step when deploying a SaaS application is to verify its requirements — the operating system, the back end, the front end, the hardware requirements, backup and disaster recovery procedures and so on. Then negotiation takes place with different cloud service vendors who can provide the software and hardware to satisfy the requirements and the service level agreements. In fact, there is usually a checklist [11] that must be completed before SaaS can be deployed. Negotiation is also required if the software has to interface with third-party application providers or if any agreements need to be signed. During the procurement phase, the required virtual environment for hosting the SaaS is deployed. The software is installed in the virtual environment during the installation phase. SaaS is then initialized with the data necessary for its operation, including the pointers to the databases, networks, and backup devices. During configuration users are created, the access rights matrix is established, user constraints on memory and processor usage are established, payment gateway interconnections are established, and connections with other external APIs (Application Programming Interfaces) are also established. During UAT, the initial set of users test the system and if no defects are found, the system is open to all users. If defects are found, then the process is repeated from the stage where the defect originated.

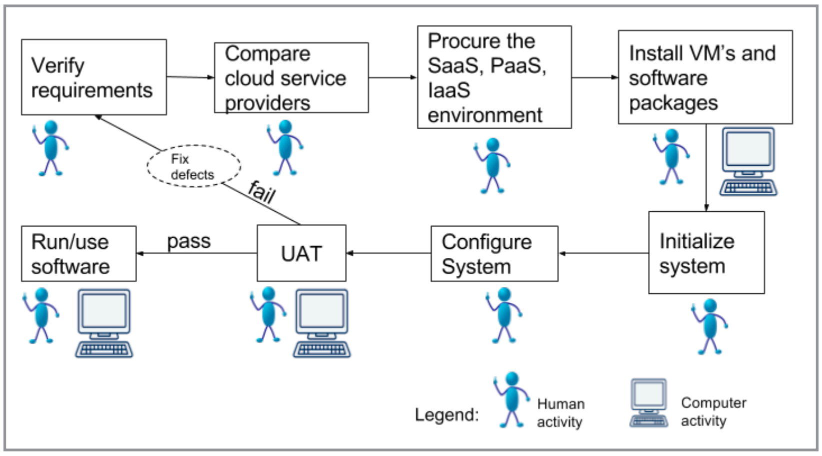

Figure 3. Deployment Process for Software in the Cloud

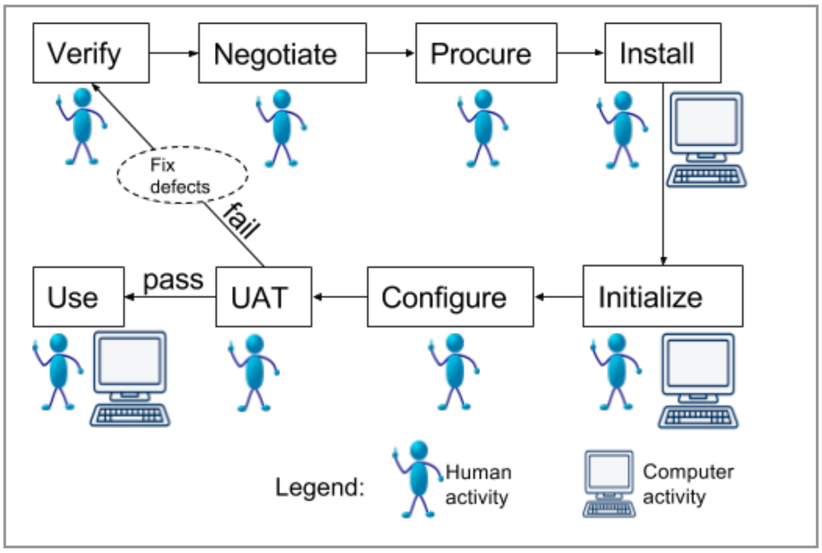

Figure 4. The Generic Deployment Model

C. Deploying Mobile Applications

When we think of deploying a mobile app, we usually think of downloading and using it. However, even a trivial app must go through several steps. We first verify the need for the app, even if the need is simple curiosity. We then decide where to download it from — the official store for the device (AppStore for iOS devices and Google Play Store for Android) or a third party store (such as Amazon.com). We then purchase the software if it is not free. After that, we download the app during the install phase. During initialization, we give permissions for the app to access directories, devices, and logs. During the configuration phase, if needed, we set up permissions to use the device, location settings, the look and feel of the user interface, and so on. UAT is the actual trial run of the software to see if it works as we expected. If we like the app we keep it; otherwise, we uninstall or disable it. Optionally, we may choose to report bugs to the vendor or, if we are happy, give the vendor a good review.

However, for an enterprise app, there are more stringent procedures at each step since such apps are usually frontends for a web application, a server application, or a SaaS application. These apps are approved for download by the enterprise IT team. Therefore, there are two flavors — those installed by the enterprise IT on enterprise devices and those installed by the employee on his or her device following a BYOD (Bring Your Own Device) policy. In either case, the steps are mostly similar. First, verify that the app is indeed the frontend for the enterprise application and will perform most of the functions on the mobile device. Next, negotiate with the vendor on the price or compatibility with different platforms and sign any agreements for support and upgrades. Third, procure the software if it is distributed by an enterprise device manager [12] or else download the app from the appropriate app store during the install phase. During initialization, give permissions for the app to access personal information on the device and sign any end-user agreements. During configuration, enter the authentication credentials into the device as well as the path to the backend systems and the frequency of synchronization of data with the backend. During UAT, use the device and see if it meets the established expectations. If it does, we continue to use it. Otherwise, we report issues to the IT (if downloaded by the employee) or to the vendor (if downloaded by the IT).

For enterprise apps there are also associated issues of mobile device management, mobile device enrollment, and configuring mobile device settings. Device management [12] distributes applications, configuration settings, and security tools (for example, some organizations encrypt devices for using their apps). Device management can be part of the negotiation, installation, initialization, and configuration steps. It usually becomes a big part of normal mobile device operation after deployment in case devices get lost or become part of a botnet (the device manager can either locate the device or remotely wipe it). Mobile device enrollment [13] allows the organization to enroll many devices at the same time and to monitor and manage their use. Enrollment can be part of the verification, initialization, and configuration steps. Mobile device configuration [14] manages mobile device settings and is usually part of the configuration step.

As can be seen, we are not focusing on deploying a mobile app to the app store, [15] which is part of the app development process.

4. Analyzing the Performance of the Generic Deployment Model

As can be seen from Table 1, software deployment takes a long time, even when automated tools are used. This is mainly because of the extensive human involvement in the process. Even though actual software usage occurs over years, it can take several days or months before the benefits from software usage can begin to accrue. We will now analyze each step and see if further automation of the step can help reduce deployment time.

In the first step, the deployment team needs to understand software requirements and obtain sizing estimates. The deployment team often requires a detailed specification sheet so that any deviations from the organization's IT procurement policies may be identified early and budgeted for. Organization compliance teams, including audit and risk management divisions, may also be involved to identify any risks to the organization's information security that may be caused by the deployment of the software. Other professionals, like consultants (who may help with the integration effort later), lawyers (who may help clarify the requirements terms), networking specialists, database experts, and others may also be involved. Thus, the typical timeframe can be anywhere from a few days to months. However, for smartphone apps and small-scale software for home or small business use, a team of professionals is rarely used even though there are reports that privacy policies for some apps may invite legal attention[16].

However, there is very little scope for automating this effort, even if requirements are written using a standard such as BPEL (Business Process Execution Language) [17] because cross-domain human expertise may be required for this step.

In the second step, "negotiation," the agreements are signed with the software developer and the negotiation of cost and delivery with hardware and support software vendors takes place. Therefore, several divisions, including legal, procurement, and finance, could be involved in this process. The legal team is involved to ensure the customer organization is using standard clauses in its agreements and no unreasonable terms are involved. Purchasing is involved to make sure contracted standard vendors are used and any deviations are approved. Finance is needed for payment. Again, the cross-domain expertise required for this step makes automation difficult. Therefore, time needed for this step can be anywhere from a few days to months, especially if the software will be used in several countries and legal ramifications of multi-country use need to be explored. Software for negotiation is still in the research stage [18] and does not seem to have been adopted in mainstream legal practice, [19] so there is little scope for automation in this step.

The third step is procurement, when IT infrastructure is ordered for deployment. This ordering process can be largely automated using an ERP like SAP, [20] and software can be downloaded. However, hardware has to be physically delivered, and this takes time — especially if specialized made-to-order hardware is involved in deployment. Therefore, this step is mostly manual.

The fourth step is installation of software, and this step is usually automatic. However, strict checklists used by organizations during this process and any consultant input will add time to this step. Sometimes corporate audit and review teams need to certify that the installation was successful; this can also delay this step.

The fifth step is the initialization step and involves several manual activities. During initialization disk partitions are created; accesses to network, databases, network attached storages (NAS), and authentication and authorization systems (for example, active directory integration) are provided; and user keys for encryption are provided. Integration with external systems, such as emails, payment gateways, and the Internet, also occur during this step. The company account will need to be used for integrating with such external systems for proper financial accounting purposes. Use of initialization scripts can automate parts of this process, but subsequent validation of all initialization steps is usually done manually.

In the sixth step, the configuration step, the software is configured for use. This includes assigning users to the software, updating database schemas, and any other configuration needed for proper use of the software. Data migration also takes place at this point so the system is ready for use, especially if the system is replacing a legacy system. For new systems, critical mass of data need to be entered into the system before it can be used. For example, in a Document Management System, [21] scans of existing physical documents need to be input into the system before the system becomes useful. Any vendor licenses required for using the software are also activated during this step. Backup procedures are also implemented at this stage so that disaster recovery procedures are ready once the system becomes operational. This configuration step is usually performed manually, though several automated configuration management tools [22] that run configuration scripts are now appearing in the market to automate parts of this process.

The seventh step is UAT. During this step, the software is tested to ensure that it is ready for use. Users are provided software training, then use the software as expected. Any defects found during this step are fixed. These defects may be in the software, which means the software developer fixes them. Or, if the defects are due to improper implementation of the deployment steps, the deployment organization fixes them. Once the software or the deployment activities are corrected, the UAT continues until users consider the software acceptable. Several automated tools exist for testing, [23] but this step is still largely manual [24] and can take several weeks to complete.

The final step of the deployment process is "going live" with the new software, which means the software is put to normal use. There are several issues to consider during this step, including the cut-over process, the establishment of a help desk for maintenance, a provision for regular user training, proper backups, and processes for ensuring business continuity and disaster recovery. The cut-over process can be parallel or abrupt — if it's parallel, the old and new versions of the software must be run together before all users are migrated to the new version. However, this step usually keeps the software running for many years. During this process, all stakeholders — including users — interact with the software. The developer maintains the software under a maintenance contract, which may need approval from other departments. Therefore, there is significant manual interaction during this step.

Deployment Step	Typical Activities	Extenuating Issues	Timeframe	Automated or Manual
Verification	Understand the software requirements document; get hardware and software sizing estimates; complete organization IT procurement datasheet	Involvement of consultants, lawyers, and other professionals; involvement of corporate IT audit and risk management team	Few days to months (depending on the scope of software deployment)	Manual
Negotiation	Legal agreement with software developer; cost and delivery with hardware and support software vendor (the software to be deployed has already been negotiated by the customer); service-level agreements	Involvement of consultants, lawyers, corporate procurement, IT audit, risk management, and finance; multiple-country jurisdiction	Few days to months (depending on scope of agreements and cost)	Manual
Procurement	Actual shipment and delivery of all software and hardware	Shipment of specialized hardware or import from other countries	Few days to weeks	Manual
Installation	Of hardware and all software	Corporate checklists, external experts, corporate audit and review teams	Hours to days	Automated
Initialization	Hardware and software are initialized with data in the global settings including backup system settings; connect application with front-end (web and application servers) and backends (databases)	Multiple administrators for specialized domains (such as networking, database, NAS, security, etc.); integration with external systems such as emails and payment gateways	Hours to days	Manual
Configuration	Software is configured for use with user information and sufficient mass of data is populated in databases	Data migration, license activation, backup procedures implementation	Hours to days	Manual
User Acceptance Test (UAT)	Software is tested by users; automated testing tools may also be used for stress testing	Any defects found need to be fixed and depending on the severity the cycle of deployment process need to be revisited; user training	Days to months	Automated
Use	Actual production use of software; "going live"	Cutover - parallel or abrupt, helpdesk procedures, periodic user training, backups, business continuity, disaster recovery, maintenance contracts	years	Automated

Table 1. Typical Activities and Timeframes for the Generic Deployment Process

5. Conclusion

Software deployment is one of the most important phases in the software development life cycle, though the last one to occur before the software enters its maintenance phase. However, software deployment is anything but trivial and can last for months or even years depending on the complexity of the software. Improper deployment can lead to rejection of software by users, financial loss, or even litigation.

The software deployment process can be broken down into eight steps: verification, negotiation, procurement, installation, initialization, configuration, User Acceptance Testing (UAT), and production use. During verification, the software's deployment requirements regarding hardware and support software are understood. During negotiation, developers negotiate agreements and cost and delivery for deployment hardware and software. During procurement, the hardware and software needed for deploying the original software are procured. All software is installed during the installation phase. During initialization, the software is prepared for subsequent steps. During configuration, user information is entered in the system and any integration with external systems are taken care of. During UAT, users test the software to ensure that it is ready for normal use. Any defects during this step are sent to the software developer for fixing. Finally, once the software passes the UAT, it is put to normal use.

Almost all of these steps require human involvement to a large extent. Only the installation, UAT, and actual use steps are mostly computerized. While automation can help reduce human effort in all eight of the steps, we are still a long way from completely automating the deployment process.

ABOUT THE AUTHOR

Nary Subramanian is an associate professor of computer science at the University of Texas at Tyler in Tyler, Texas. Dr. Subramanian received his Ph.D. in computer science from the University of Texas at Dallas. He is a Fellow of Service Learning at UT Tyler's Center for Excellence in Teaching and Learning. He has over 15 years' industry experience in engineering, sales and management. His research interests include software engineering, system engineering and security engineering.

Department of Computer Science
College of Business and Technology
University of Texas at Tyler
Tyler, Texas 75799
nsubramanian@uttyler.edu

REFERENCES

1. "2015 ERP Report." (2015.) Panorama Consulting Solutions. Available at http://go.panorama-consulting.com/rs/panoramaconsulting/images/2015%20ERP%20Report.pdf
2. http://searchsap.techtarget.com/definition/SAP-Rapid-Deployment-Solutions-SAP-RDS
3. Amin, P. (August 2012.) "What is the SAP Rapid Deployment Solution Implementation Methodology? Does It Work?" Available at http://scn.sap.com/community/rapid-deployment/blog/2012/08/06/what-is-the-sap-rapid-deployment-solution-implementation-methodology-does-it-work
4. Baumann, W. (April 2016.) "Courtroom lessons from a failed SAP ERP implementation." TechTarget. www.techtarget.com
5. Stein, J. (September 2014.) "FBI's Expensive Sentienl Computer System Still Isn't Working, Despite Report." Newsweek. http://www.newsweek.com/fbis-expensive-sentinel-computer-system-still-isnt-working-despite-report-272855
6. Wailgum, T. (September 2009.) "Why ERP Is Still So Hard." CIO. Available at http://www.cio.com/article/2424944/enterprise-software/why-erp-is-still-so-hard.html
7. Software Installation Process, IBM Knowledge Center. Available at http://www.ibm.com/support/knowledgecenter/ssw_i5_54/rzahc/rzahcswinstallprocess.htm
8. http://docs.openstack.org/developer/heat/template_guide/software_deployment.html#software-deployment-resources
9. http://www.tsg.com/resource-centre/articles/how-long-does-it-take-deploy-crm-system. Accessed Oct. 8, 2016.
10. https://docs.oracle.com/cd/E13222_01/wls/docs60/adminguide/config_web_app.html. Accessed Nov. 14, 2016.
11. http://www.oracle.com/us/products/applications/checklist-saas-2193759.pdf. Accessed Nov. 16, 2016.
12. http://searchmobilecomputing.techtarget.com/definition/mobile-device-management. Accessed Nov. 17, 2016.
13. https://docs.microsoft.com/en-us/intune/deploy-use/enroll-corporate-owned-devices-with-the-device-enrollment-manager-in-microsoft-intune. Accessed Nov. 17, 2016.
14. "General settings for Mobile Devices in Configuration Manager." (June 2015.) https://technet.microsoft.com/en-us/library/dn376523.aspx. Accessed Nov. 17, 2016.
15. https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html. Accessed Nov. 17, 2016.
16. http://www.iubenda.com/blog/the-need-for-privacy-policies-in-mobile-apps-an-overview/. Accessed Oct. 8, 2016.
17. Vemulapalli, A. & Subramanian, N. (November 2009.) "Transforming Functional Requirements from UML into BPEL to Efficiently Develop SOA-Based Systems." Lecture Notes in Computer Science. No. 5872, 337-349. Proceedings of On The Move 2009 Conference, edited by Meersman, R., Herrero, P. & Dillon, T.
18. Reeves, D. M., Wellman, M. P.,Grosof, B. N. & Chan, H. Y. (2000.) "Automated Negotiation from Declarative Contract Descriptions." American Association for Artificial Intelligence. http://www.mit.edu/~bgrosof/paps/kbem00.pdf
19. http://www.capterra.com/law-practice-management-software/. Accessed Oct. 9, 2016.
20. http://go.sap.com/product/srm/supplier-relationship-management.html. Accessed Oct. 8, 2016.
21. Krishnan, S. & Subramanian, N. (April 2015.) "Evaluating Carbon-Reducing Impact of Document Management Systems." IEEE Green Technologies (GreenTech) Conference. New Orleans, Louisiana.
22. Jacobs, D. "How to choose and implement automated configuration management tools." http://searchnetworking.techtarget.com/How-to-choose-and-implement-automated-configuration-management-tools. Accessed Oct. 8, 2016.
23. Vingrys, K. (July 2010.) "Acceptance Test Automation." https://www.thoughtworks.com/insights/blog/acceptance-test-automation. Accessed Oct. 8, 2016.
24. Carman, C. (November 2013.) "How to Manage User Acceptance Testing." http://insights.dice.com/2013/11/11/how-to-manage-user-acceptance-testing/. Accessed Oct. 9, 2016.