Stage and Software

Similarities in Management of Software Development and Theatrical Design Projects

By Brian Alderman

Carnegie Mellon University

May 2013

TABLE OF CONTENTS

INTRODUCTION

Project managers make things happen, regardless of the circumstances of the project or resources available. The theatrical design and software development industries present unique challenges for the project manager because of the creativity inherent to each of these fields and the eccentricities of each individual project. A basic observation is that theatrical management is much less formalized than software management; however they are both rooted in the same basic theory. In *Managing the Software Process*, Watts Humphrey makes the case that, "While this is a new and unique field, traditional management methods can and should be used".[1] The same is true for theatrical management, as evidenced by experiences in theatre management class work and the connections drawn between other project management experiences. Clearly then, there is worth in examining the similarities between the two fields in particular and considering techniques that project managers can utilize across industries.

I don't mean to imply that project management for software and theatre design is exactly like all project management. Rather, these two fields are somewhat unique to project management because of the inherent creativity required for both project and process. The projects in these industries are always different, and will always have a problem that needs to be solved in a new way. In terms of process, Humphrey puts this best, stating, "Software engineering, however, is not a routine activity that can be structured and regimented like a repetitive manufacturing or clerical procedure. We are

[1] Humphrey, *Managing the Software Process*., 25
[2] Ibid., 247

dealing with an intellectual process that must dynamically adjust to the creative needs of the professionals and their tasks. A trade off is thus required between the individual need for flexibility and the organizational need for standards and consistency."[2] This is just as true for theatrical design, the ultimate goal of which is to serve the play that is being produced. Each of these plays requires a creative and intellectual process that must be flexible enough to fit the needs of that play or specific theater for which it is being produced. All of these theatrical projects will follow similar timelines that best fit the organization, as will software projects, but the specifics must vary as necessary for the project.

Carnegie Mellon University is a unique environment because it contains both theatrical designers, labeled “Artists”, and the supposedly more technically based (and mysterious) “Computer Scientist”. Much insight can be gained by working on projects with people from both of these populations, as I have been fortunate enough to do. Discovering the ways these different groups approach a project, whether a formal project related to their studies or an extracurricular endeavor, has led me to the conclusion that they are very similar in their approaches to a challenge despite different fields of study.

This paper will endeavor to examine the techniques used by project managers to work with theatrical designers and software engineers. First, I will create a basic understanding of the types of projects that designers and software engineers complete. This will include an overview of project timelines and phases, work methods, and highlights of management’s interactions and responsibilities during projects. This

[2] Ibid., 247

baseline understating will then be applied to an examination of the theatrical designers and software engineers who work on those projects. A team of software industry management consultants, DeMarco and Lister, wrote in *Peopleware,* "The major problems of our work are not so much technological as sociological in nature."[3] Previous experience on theatrical projects proves that a similar claim can be made about the theatre industry. Therefore, the project manager in both areas must understand the people involved in the teams and projects they will manage. In this vein, I will undertake an exploration of creativity, problem-solving techniques, the structure of teams who work on these projects, incentivizing those teams, and a phenomenon termed “the Hero Complex”. Throughout, I will attempt to connect the techniques used by managers in each field and suggest additional methods that theatrical design project managers and software engineering managers can learn from each other to increase their effectiveness.

Who Are These People?

Before delving too far in, specification is warranted for the terms theatrical design, software engineering, and project management. I am choosing to examine theatrical designers in a typical regional theatre model of a design team- one that includes Scenery, Lighting, Costume, and Sound Designers working in collaboration with a Director to develop a theatrical product for the stage. Many of the same points can be further expanded to other performing arts design teams such as film and television design, commercial theatre, or event production. These designers come from a variety

[3] DeMarco and Lister, *Peopleware: Productive Projects and Teams*., 4

of backgrounds, and can have both formal training in a collegiate education program and a variety of experiences working on productions of various scopes from very small to nationally significant. Every designer has his or her own unique background that informs his or her interactions with a design team, and understanding those experiences is one of the project manager's most important tasks. Being a theatrical Designer is a permanent occupation for many practitioners, many of whom have the career goal of working on larger productions that offer creative satisfaction or higher profile exposure rather than transitioning into a managerial or oversight role.

The scope of the term software engineer is much wider and varies much more based on organization. Watts describes software engineering as, "The disciplined application of engineering, scientific, and mathematical principles, methods, and tools to the economical production of quality software."[4] For the purposes of this paper, I am considering the job to encompass all aspects of high-level idea specification, technical design, programming, user interaction design, and testing/verification aspects of software development. Other common terms for this include Programmer, Computer Scientist, and Software Developer, which may be used interchangeably throughout this paper. Most software engineers have some formal training, either collegiate or otherwise, in some aspect of the software engineering field. The modern software engineer comes from a formal collegiate program. Historically, many software engineers worked in other technical and engineering fields before retraining to work with computers later in their career. In terms of career path, low-level software engineers and programmers tend to either move up within a company or move to other

[4] Humphrey, *Managing the Software Process*.

related industries or tasks relatively quickly, such that the low-level workhorses of a software project are perpetually inexperienced and new to the industry. This important phenomenon will be examined more fully later in the paper.

A project manager is responsible for shepherding any given project, including those in theatre and software, from inception to completion. At the most basic level, the job is anything that will achieve this goal. This usually includes coordinating communication amongst those involved, developing schedules and budgets, and balancing the priorities involved in all aspects of the project. In *The Art of Project Management*, Scott Berkun spends quite a while explaining what the traits and tasks of a project manager can be. His most basic definition calls the project manager "whoever is involved in project leadership and management activity" by which he means whoever is involved in "leading the team in figuring out what the project is, shepherding the project through design and development work, and driving the project through to completion."[5] Characterizing project management any more specifically is a trap that must be avoided, as it limits the flexibility required to accomplish the above.

Further, one must examine where project managers in theatre and software come from in order to understand why each industry uses the project structure and management techniques that it does. In software engineering, many project managers advance from line-level software programming positions. *Peopleware* explains this phenomenon and some of its implications by saying, "It's a rare firm in which new managers have done anything that specifically indicates an ability or aptitude for

[5] Berkun, *The Art of Project Management*., 8

management. They've got little management experience and no meaningful practice."[6] The authors further point out that project managers often begin their careers with a myriad of specific technical skills, which helps them understand the projects they manage. However, they argue that this technical approach often hampers their management role, as they cannot remove themselves from the nitty-gritty technical problems. This is an inherent practice in the industry and is important to address when working on a project.

Theatrical project managers, often called Production Managers, seem to come from a variety of backgrounds and experiences. The advent of formal production management educational programs is a recent phenomenon, only now reaching a critical mass of people working in the industry. Previously, Production Managers worked in some other specific area of theatre (i.e. as a designer or as a technical department head) under the production manager's purview, and have been promoted into the management role. There is no common path to Production Management, as an assembled study of a group called the Production Management Forum members illustrates.[7] As with software project managers who come from line-level programming backgrounds, this advancement without regard for management acumen can create some interesting challenges for the project management role, which will be a focus of a later section.

---

[6] DeMarco and Lister, *Peopleware: Productive Projects and Teams*., 5
[7] Holcomb, "Where Do Production Managers Come From."

THE PROJECTS

At first glance, a computer program like “The Sims” and a production of the musical “Wicked” are completely different objects and not comparable. However, if you were to begin examining the processes that created both “The Sims” and “Wicked”, they’d begin to look a bit more comparable. The first major realization that needs to be made is that the development of both of these things can be modeled as a large project with a single end goal, namely creating a computer program or putting on a show. Once that goal is accomplished the project is complete. Essentially, both of these projects work to put themselves out of business, and by completing them there is a measurable final product. Starting with that basic similarity, further comparison continues.

In terms of overall scope and total time elapsed from inception to completion, theatre and software projects are easily compared, at least in ratios of time. For example, a theatre might begin working on a production six months prior to opening night, with the design stage taking 3-4 months and the implementation phase filling the balance of the time. In software, the timelines can depending on scope of project, but similarly have a design and requirements development phase that is about as long as the implementation and testing phase. Fewer people are usually involved in the earlier stages of both theatrical and software projects, and people are added to the project as the implementation phase begins until the product is released.

The following section will examine some particular project aspects that are similar across both industries. These areas are:

- Timelines and phases of projects

- Project Management methods
- The Project Manager's role

Project Phases

At the highest level, all projects in every discipline and industry can be thought of in two sections- the initial planning phase, and the implementation phase. The planning phase encompasses figuring out what the plan is, and the implementation phase is executing that plan. Theatrical design and software development projects can universally be divided up further. Within the initial planning phase, both of these disciplines have phases that guide the project from general to specific:

- Requirements phase
- Design phase
- Implementation prep phase

Changing an idea into a more concrete, communicable requirement or goal is the first step, termed the requirements phase of a project. The concrete requirements come from the project's initial ideas and can encompass everything from, for example, "Lets put on a show to make lots of money" to "I want to make a game that simulates a city". The chief difference between ideas and requirements is specificity. By the end of this phase, the goal is to point to a list of statements that discretely, concretely, and concisely describe the goals the project wishes to accomplish. In theatre, this usually comes from both the selection of a script and the director or producer's overall vision of that script. We'll use the following example for our comparison or projects. In this phase, "Produce a version of Romeo and Juliet set in Shakespearean England" is a

requirement, as is “Create a software program that allows for the creation of pie charts from a list of data”. These requirements will ultimately inform every other stage of the project, so it is worth a significant amount of time to make sure that all goals of the project are listed.

Additionally, some organizations prioritize these requirements lists. The effectiveness of this greatly depends on the number, specificity, and interconnectedness of the requirements, and should be used sparingly and cautiously. Really, there shouldn’t be items on the list at this point that are lower priority. The idea of them being a requirement is to complete them all, and if there are lower priority requirements, then the project manager should question if those requirements should be included in the project scope.

There is an argument to be made for not spending much time on requirements definitions to determine the scope of the project. “Don’t Define The Problem”, an article written regarding software projects in the federal government, contests that approaching problem solving by first defining a problem is counterproductive because it limits possible solutions, especially when the project calls for particularly creative methods[8]. In theatre, we spend a lot of time doing the opposite by determining a motivation, goal, and general aesthetic for the production at the outset. At some point, the script does this for us and remains consistent throughout the project as an ultimate guiding force. However, further definition through discussion of the problem is necessary in theatre because of the wildly different expertise of the design team and

[8] Lloyd, “Don’t Define the Problem.”

the need to cohesively intertwine them across all aspects of the show. Limiting solutions in this case is advantageous.

To contrast, in software the problems are defined in the initial requirements stage, but are much more open to change throughout. This ease occurs because those required to change a design are working more closely and any change is more easily integrated across their work. Additionally, there is usually not a firm source material, like a script is in theatre, to back up a software project. An additional reason for flexibility in software requirements is that software problems might not have a solution- there are certain things that computers cannot feasibly do given current methods. In academia, these types of problems are called Open Problems, and are the focus of extensive research. By approaching a project by first defining the problem to be solved, software engineers can back themselves into a corner of trying to solve an unsolvable problem where instead they need to focus on finding a way around such a problem.

Once requirements and project goals are fully determined, additional specificity is added in the design phase. Each individual requirement is transformed into a miniature, easily demonstrated version of the final product. The way this is communicated varies widely, as the below table shows. The goal is that all aspects of the design are captured into these methods by the end of this stage. It is then the team and the project manager's job to check the proposed design against the requirements to ensure that all requirements are being met by the proposed design.

Design Communication Methods

Software	**Theatre**
Feature List	Physical Models
Flowcharts/Diagrams	Photos
Pseudocode	Renderings
User Interface Mock Ups	Draftings

Software typically uses feature lists, which detail each discrete component of a product, or diagrams that describe how components in a program fit together. Theatre, usually utilizes a model of the design, photos, renderings, and draftings.

In *Managing the Software Process*, Humphrey comments, "For creative professionals, the most frustrating part of this environment is that the same problems keep repeating. Plans are ad hoc, schedules are arbitrary, design control is nonexistent, and resources are always inadequate."[9] Humphrey is observing a critical, but frustrating, part of the creativity required to find the best solutions in the software design stage. At this point, there is still a lot of problem solving to be done regarding how to fulfill the requirements, and there is little information about the plan for implementation despite there being a viable design. Creative people, especially in software, sometimes feel that their first answer is the right one, which means they want to leap straight to the implementation phase. In order for the design phase to be most successful, it is the Project Manager's job to rein this presumptuous decision-making in and ensure that the design fine-tuned to best meet the challenges posed by the requirements before moving on.

[9] Humphrey, *Managing the Software Process*.

Continuing with the pie charts and Romeo and Juliet examples, a completed design should probably have figured out, respectively, what the user interface is with the round pie-like objects, what a balcony for the scenery looks like and how corsets for the women's costumes are used. In theatre, another important step occurs once the design has been developed- it is brought to higher levels of artistic oversight for approval. Typically, the artistic director or producer will at this point sign off on a design before allowing it to move forwards. In software, there tends to be less direct oversight of the final product because the requirements in the previous stage are more specified and concrete, so upper management would have little to add once the design is compete as they care little about implementation methods. Instead, it falls to project management to ensure that the requirements are being fulfilled by a design prior to progressing.

In the next stage, Implementation Prep, an approved design is transformed into something that can actually be created. Usually, this means adding even more detail. For software projects, prototyping the most complex elements to prove feasibility can be effective. Breaking the remaining features into tasks for programmers to work on is typical as well. In theatre, this phase marks the transfer of the show from design to technical specifications, which usually means adding many more people to the team. This is an important difference between theatrical design teams and software engineering teams- in all but the smallest theaters, designers will at this point pass their work off to other specialists such as a technical director or master electrician for the creation and implementation process. In software, it is much more common to find those that did the bulk of design work continuing on the project through the

implementation phase. Therefore, in theatre, communication of the design needs to be much more complex and detailed as there are different people working on it than designed it. This is accomplished through many conversations and iterations of review of the documents that were used in the design stage, adding further specificity throughout. It is at this point that, in theory, the team knows the most about the project and what the next steps are.

To continue our example, at this point we have prototyped the code used to make a circle in our program, and are working on engineering and technical drawings for how to build a balcony and are choosing fabric colors for the corsets. By the end of this stage, the project is essentially halfway to completion even though there is very little physical material to demonstrate this.

The implementation phase, where the bulk of a project actually happens, can also be segmented. The first phase is getting started, which sounds a lot easier than it actually is. Next comes the middle game, followed by what Berkun calls “The End Game”[10]. After those steps are complete, many organizations will also do a wrap up or archive of the project work.

Starting the implementation phase is very much connected to the previous step of preparing for implementation, but with a more thought-to-action based itinerary. The project manager should, at this time, ensure all of the steps of implementation are listed, prioritized, and assigned to the appropriate people, and that all questions have been answered regarding the design. Then, the project manager must cut the team loose and allow them to begin their assigned tasks. A project manager at this stage

[10] Berkun, *The Art of Project Management*.

knows to keep an eye out for places where little bits of information have not been communicated, or are incorrectly translated from one source to another. However, it is inevitable that some bit of information will get lost in the transition to this stage, regardless of the project manager's involvement. It seems very simple and straightforward, but can in fact be the most complicated part of a process- there's just very little else specifically to say other than be wary and keep an eye out for indications of miscommunication or unanswered questions.

Moving on, the middle game is the closest a project of this type can come to stasis from a project management perspective. Until something goes wrong, there is very little the project manager is expected actively to do. Instead, checking in on progress is the most appropriate use of time. There are many methods for doing this, which are best described in other sources (see *The Art of Project Management* a number of other texts). I will examine one method of particular interest.

One specific practice to highlight is software engineering's use of regular code reviews for this check in process. The idea is that, at regular intervals, you ensure that all work to date is accurate and useful, and have a chance to see what everyone on the team has completed. Every organization has their its method for code review- whether a specific review department or person, a session with the project manager, or a peer review with other engineers. Theatre does not have something quite like this, as there is less of a physical product to examine at any given point in time. Code reviews are not regular, forward-looking progress check-in meetings or production meetings like those that exist in theatre. Instead, they are a detailed examination of specific work accomplished to date in order to determine efficacy of the product. Further

examination of code reviews and a suggestion of how it could be implemented in the theatre process will take place late in this paper.

As a project nears the published completion date, we move into a segment called the end game, which involves putting all the pieces together and releasing the final product. For software, this is the releasing of the program or the pushing of code to the live environment that users interface with. For theatre, this is opening night. The last few weeks of work become more frantic and intense as this final date approaches, but the frenzy is much more apparent in theatre than in software. In theatre, the practice is that the completed show will go on at opening night. Software does not hold release dates in as high regard, having a reputation for pushing deadlines until the project is actually fully complete.

Consider a company like Pixar, which dabbles heavily in both the creative traditionally artistic work of movie making and the more traditional software-based development as they find innovated computer animation techniques. As they ramp up to complete a movie, they are typically finishing the software and rendering parts of their project, but with a similar time constraint as theatre- the opening for a film as big as Pixar's will not be delayed. In *The Pixar Touch*, the author relates stories from Pixar that highlight the high stress, long work hour environment surrounding a release[11]. It sounds a lot like technical rehearsals that occur in theatre- fine-tuning all of the aspects, with every single member of the team involved.

Overall, the sense is that theatrical projects are pushed through to completion while software projects are allowed to linger a bit more. This is because of the end

[11] Price, *The Pixar Touch: The Making of a Company*.

game process and the way project completeness is defined in each industry. One reason for a high level of productivity during the technical rehearsal process, theatre's end game, is that the entire creative team is in one place, working for long periods of time. In software, as different segments of the project are completed, those team members move on to other things or go home. Theatre's advantage, in this case, is to be able to rapidly make decisions because all those needed are already in the room.

Another key point to make regarding the end game is the difference in how these industries view being done with a project. In theatre, this is much more subjective. As long as there is a show on stage, the level of quality and success can vary wildly and may be uncorrelated. In particular for non-profit theatre, failure and experimentation are often built into the model and are occasionally expected outcomes. One expert theatre innovation blogger, Polly Carl, explains how in theatre, "We produce truths because we are driven by certain moral imperatives…"[12] She goes on further to say that this can lead to uncertain outcomes and flexible definitions of success in the theatrical industry.

The industry for non-profit software does not exist, so software engineering is measured much more critically on the monetary success of the final project. Before a release, software engineers want to ensure that there will be no hidden bugs or problems with the software. This requires much more extensive user testing prior to release, which can take time and push deadlines as issues are discovered and repaired. The fact that theatre is more subjective about final products than software makes the

[12] Carl, "Truthiness in the Politics of Theatre."

end game take a finite amount of time, whereas in software it can stretch on until there is a definitive “complete” status for the project.

One difference between theatre and software during the end game is the number of hours that a company is willing to put into a project. In theatre, the culture is that during a tech week, you work as many hours as necessary, often more than full time, to push through to the end. In software, this tradeoff is less common. In fact, *Peopleware* argues that overtime is not worthwhile for a company that is interested in the long-term quality of its work. They claim that the idea of sprinting to the end of a project will cause burnout amongst team members, and will therefore cause them to ultimately leave the team[13]. Theatre comes with the expectation of sprinting to the end, and accepts the fact that those that can’t do so will leave the industry. This contributes to the elite nature of the design teams- all those who are involved are capable of sprinting to the end. In the long run, however, this may make those in theatre remain in the industry for a shorter amount of time. I believe that this difference would be an interesting field for analytical study, such as looking at retirement rates in theatre vs. software or average age of practitioners, but it is beyond the scope of this paper.

Once a project is released or opened, the project manager usually establishes full records of the process. These records are typically more useful in software than in theatre because of how often they are accessed. For a software project, the archive records would include all of the usual process archive material- schedules, design documents, budget information, and the code itself – but also materials that comment on the code and the way problems are solved, termed documentation. This

[13] DeMarco and Lister, *Peopleware: Productive Projects and Teams*.

documentation is invaluable later on for a future version of the product, which in software is a common occurrence. A project manager will have the team contribute heavily to this documentation. The programmer who writes a piece of code needs to write the documentation for it, as they understand it best. A project manager's role is less to write the documentation than it is to help the team develop the archive materials and assemble them into a cohesive product. Good documentation is an industry standard for software, and there cannot be a piece of software without it.

For most theatrical projects, particularly regional sized and smaller, the probability that a show is going to be recreated or taken on tour is small, so the archive documentation does not need to be as thorough. The information included is for later reference instead of recreation of the same product, something that is rarely going to happen because part of the goal in non-profit theatre is to put on a different show each time, and it is very unlikely that a single theatre would mount the same show in future years. (Note: Opera, ballet, and commercial theatre are distinct exceptions to this, as they regularly mount the same productions. The argument for detailed documentation is much stronger in these fields). Some normal materials to archive are budget information, final drawings and model reference information, and some basic notes about the production's development. Overall, this information is much less detailed than software and there is little effort during the implementation phase to assemble this information. In theatre, the archiving of this information falls more on the production manager to assemble than it does the creative team to help provide information particularly for the archive.

When working on a project, the idea of how the information will be archived should be in the project manager's mind, but should not dictate how information is developed. There are few things as creatively dampening as being required to follow a certain format or template for project documents or ideas generation for the sole purpose of later archiving in a mold that fits the organization. It is part of the project manager's job, ideally, to take the information that was created through the process and fit it into achievable format once the project is complete. This process of reformatting materials for archive is not only helpful for creating an archive, but will help to reflect on the project in order to avoid problems from repeating in the organization's future.

Project Methods

The methods that the designers and engineers utilize during each of these project phases are worth an exploration as well. During the initial planning phases, various solutions are developed through both structured and unstructured thought exercises such as brainstorming. Through this time, the project manager assists in maintaining a common language between all those working on the project. In software, this can sometimes be more straightforward, such as the naming of variables in coding, or more aloof such as the names used to refer to particular features. In theatre, this same issue exists as standard theatrical conventions are somewhat lax and different people come up with different descriptions or names for things. Blogger Eric Holk discusses the effectiveness of consistent variable naming, observing that following variable naming conventions helps experienced programmers understand the code much more quickly when reading it, while breaking these conventions leads to a severe penalty in

comprehension. On the other hand, inexperienced programmers seem to take about as long regardless of how the variables are named.[14] This points to a strong need for common naming conventions in software, and makes me suspect that a similar study could be completed with similar results for theatre.

Another method that is fairly common in the software engineering world is that of "mainstreaming" the software development process. The idea of mainstreaming is to get the entire company involved in every aspect of the development process, so that the product is cohesive. Steve Jobs, the late CEO of Apple, refers to his company in an article by Doug Borwick about mainstreaming for the theatre, saying, "Our method was to develop integrated products, and that meant our process had to be integrated and collaborative". This has become the norm for many software engineering companies such as Apple and Google, a conclusion easily drawn from the explanations of project phases above. Borwick takes this argument and applies it to the audience engagement side of theatre, contesting that, "The model of artistic directors or curators operating in a vacuum to develop programming and then handing it off to the rest of the staff to "sell," whether or not it ever served arts organizations well, is no longer a healthy approach."[15] This argument for mainstreaming in software development and in the audience engagement side of theatre can be effectively applied to the design elements of a theatrical production as well. Instead of a design team working to turn their design over to a group who will implement it, imagine the entire group of designers and technicians working to develop the idea and implement it cohesively. This is a model

[14] Holk, "How Do We Read Code?".
[15] Borwick, "Mainstreaming on My Mind."

used in some particularly collaborative theaters and smaller companies with mixed success, but the software engineering example is good evidence to show that this theory can be effective.

The project manager should be constantly reminding the team that their job is to address a problem or set of requirements, regardless of the elegance or technological prowess of the solution. For software engineering, Berkun observes the anomaly that "Value was implicitly defined as quality of engineering: how reliable and performant they were or how much of the latest technology they took advantage of."[16] It is the project manager's job to counter this measure of value via cutting edge technology and instead focus on addressing the problem at hand. *Peopleware* further claims that programming work is not at all about the technology, as they are typically not making big breakthroughs technologically. Instead they are applying the technology from the high-tech business (the people that really make technological break-throughs) in ways that humans can interact with[17]. This is opposite of most people's views of software- it isn't high tech, it is merely the application of high tech. Theatre is very good at this application. Designers, because they are often removed from the implementation process, design to the problem instead of to how it is going to be created. Therefore, their design solves the goal of the project rather than exists to be cutting edge.

[16] Berkun, *The Art of Project Management*.
[17] DeMarco and Lister, *Peopleware: Productive Projects and Teams*.

Project Management's Role in Project Phases

A project manager can take on many different roles in the project process. Each of these roles requires a particular set of talents, but all of them have a few things in common. One of the commonalities is that the project manager is always explaining what their job is to the rest of the team members. Another commonality to note is that the members of project team seem to view project managers as most useful when they work to eliminate roadblocks in their process rather than manage in the more classical, oversight sense of the word. A project manager is universally expected to maintain the client's viewpoint through the process, and negotiate with that client on any potential changes to their needs. The manager is also responsible for establishing and maintaining the culture of the team in a way that is productive in terms of organizational design and physical space.

The role of the project manager has evolved fairly recently in both of these industries. As it has evolved, the rest of the industry has adjusted to create a place for the project manager in their process. Even today, project managers must establish their specific role in every team and communicate that role to those that they manage. This tends to be easier in software engineering than theatrical engineering because software companies work with the same people over and over again, whereas in theatre this is often not the case. Additionally, at larger organizations (like Google or Microsoft), the project management role is also very concretely defined because multiple layers of them are trained to work in similar ways. To contrast, each new theatrical project means new designers, which therefore means orienting that designer

to management's role in the process at that particular organization (which varies widely).

Having the team understand the project manager's talents and background is also very important. A theatrical project manager might come from a background in management, or will have a stronger experience in one or more of the design areas. Demonstrating the project manager's expertise is important in getting the team's trust, engagement, and effective communication throughout the entire project. Similarly in software, many project managers come from a software engineering background and are therefore able to speak with the team on a more technical, detailed level. Having the team realize that the project manager understands that detail is important for gaining the team's trust and having them include the project manager in their process.

The author of blog post "What Programmers Want", Michael Church, is a young programmer whose idealistic view of management is to have them step in to eliminate roadblocks and allow the engineers to work[18]. He unrealistically expects managers to anticipate problems before they occur, but does not highlight the fact that engineers must communicate with project managers to give them information needed to anticipate upcoming problems. As this post highlights, getting the team to realize that this is a two-way street is important, and should be directly explained to the team members in a framework of "I help you, you help me".

It is also project management's role to ensure that the team members understand, on some level, how the overall company and industry work as that contextualizes their work. They must realize the manager cannot shield the team from all aspects of the

[18] Church, "What Programmers Want."

industry, but instead should strive to understand it. Berkun summarizes this, saying "when engineering teams are unaware of how their business works, many decisions made by management will appear illogical or stupid" (45)[19] This counters what Church says in his blog post, but is a stronger argument as it will connect the programmer more to the final product and overall goals of the organization they are working for. Inspiring team members to learn about the larger organization and industry may be difficult, but is an important factor to consider when explaining decision-making.

A major part of project management's job is to communicate progress and questions about the goals to the client or upper management for which the project is being undertaken. In theatre, this means having a working relationship with the producer in order to represent their interests in the decision-making process. Berkun calls out the importance of similar relationships in software, and laments that, "The customer perspective is weakest in many organizations" (45).[20] Software engineering should take a lesson from theatre about the openness of communication with upper management on all stages of the project. A big difference to be taken into account, however, is the goal of the end product. In software, you are serving a specific client who can measure success against specific metrics or amount of profit. In theatre, especially not-for-profit theatre, the project is merely serving the organization's goals, not necessary the client or audience. In commercial theatre, the goal is to sell tickets and therefore the desires of the audience are more respected, much as the needs of the client in software are more respected. In those environments, the project

[19] Berkun, *The Art of Project Management*.
[20] Ibid.

manager's role in communicating with the client and incorporating his or her feedback increases.

Many software companies are known for and thrive in their unique, creatively stimulating physical and corporate environment. These environments are largely inspired by academic environments, which harbor creativity. Google and Pixar are both good examples of companies that have molded their workplaces to foster creativity. Pixar was originally created with "the atmosphere of an academic department- Utah's - and so the result was a loose knit collection of largely self-directed projects."[21] As they grew, their new environments remained very flexible and allowed the employees to customize them to fit needs and wants. There are stories of Pixar employees creating detailed physical environments to work in, such as fish tanks or cushioned closets, and management encouraging in and partaking in this endeavor. Google and other tech companies are known for their comfortable, flexible, open workspaces that allow for collaboration and customization. This helps attract talent, as well as allows them to draw inspiration from their surroundings. In theatre, this creation of a creative space is less applicable because the company does not dictate the designer's workspace, which is usually the designer's own studio or home. Hopefully, designers take into account similar thought processes as Google and Pixar when designing their personal spaces in order to optimize productivity.

Work cycle flexibility is also important at these large tech organizations. Pixar's early days were characterized by an ignorance of actual time, which allowed for insane productivity. Regarding the beginnings of Pixar, David Price notes that "for most of the

[21] Price, *The Pixar Touch: The Making of a Company*.

group, day and night did not exist there."[22] He also talks about regular 22-hour days purely because the employees got caught up in their work. This was successful for Pixar because it allowed employees to work on whatever schedule best suited their productivity. This type of environment seems more like working with a bunch of independent contractors who control the details of their project rather than employees who you expect to work at prescribed times. Similar patterns are seen with theatrical designers, whose schedules project management does not control and who therefore work at the times best for them. This flexibility, seen across industries, can be understood to increase creative productivity and should be embraced by management even though it may increase the difficulties of regular communication due to differing work cycles.

Besides physical and scheduling aspects, the overall corporate culture and structure greatly affects creativity and effectiveness of employees. A danger of large software companies, especially as the software industry developed, was management looking at software engineers as merely a means to an end. Though this is similar to the way corporations might look at employees on an assembly line, Weinberg contests that it is not effective for creative software positions, saying, "If our experiences are any indication, each of them could be functioning more efficiently, with greater satisfaction if he and his manager would only learn to look upon the programmer as a human being, rather than as another one of the machines."[23] Theatre does a good job

[22] Ibid., 21
[23] Weinberg, *The Psychology of Computer Programming*., vii.

of embracing their designers as creative personalities with significant voices in the process, and software should endeavor to do the same.

An interesting observation about software companies, which is not as true in theatre, is that the actual work of software happens amongst less-experienced engineers across a broad-based low level of the organizational chart. In theatre, the actual implementation work is more spread throughout the chart (i.e. Production Managers hanging lights, designers painting props, etc.), so those implementing projects tend to be more experienced. A major reason for this difference is likely that theatrical practitioners may be more satisfied in a lower level position because they enjoy the work. Theatre will compensate such employees appropriately as they advance in a career, paying lower level employees more as they become more experienced in that position instead of advancing them up the organizational chart. Advancing software engineers to positions that don't involve line programming rather than just paying them more to compensate for experience is more common, resulting in this disparate trend.

Peopleware points out the weaknesses in software's broad based, low level structure, saying, "Not only is the structure wastefully top-heavy, it tends to have very lightweight people at the bottom. This is somewhat true throughout the industry, but strikingly true in high-turnover companies. It's not unusual to see serious, mature companies turning out products that are developed by workers with an average are in their twenties, and average experience of less than two years."[24] Software should explore methods to have higher level, more experienced engineers take part in the

[24] DeMarco and Lister, *Peopleware: Productive Projects and Teams*., 107

implementation phases of projects rather than allowing newer employees to do most of this work. This could improve speed and project success significantly for many of the reasons discussed above.

This characterization of management's role in the project process, as well as definitions of the projects in both industries, merely demonstrates how similar software and theatre are and sets up a basis for comparison. The next section, examining the people involved in these projects, is perhaps more important, as the project manager's chief role is to work with these people to accomplish the project.

THE PEOPLE

Certain people are drawn to software design and theatrical design, and figuring out how to characterize those people is an important step to figuring out how to manage them. This understanding stems from learning about how these designers and engineers approach creativity and idea development. This understanding can then be applied to a broader understanding of the teams of people that work on theatrical and software projects, and how those teams can be incentivized by management. Finally, an examination of some of the particular quirks of these types of people is warranted. This will include examining vulnerability and failure in the eyes of these populations, as well as a foray into a concept termed "The Hero Complex".

The Psychology of Computer Programming lists the traits of a successful computer programmer, as listed below. Through a discussion with peers in a drama production management course, I've developed a similar list for theatrical designers. There is quite a bit of overlap between these two lists, as there was much overlap in the projects that these people work on. The following pages will make a case for why this overlap exists, and should inform project manager's actions when working with these people.

Traits of Successful Practitioners

Theatrical Designers	**Software Engineers**[25]
Adaptability	Adaptability to Rapid Change
Honesty	Modicum of Neatness
Ability to Communicate Effectively	Ability to Tolerate Stressful Situations
Good at following through on ideas and tasks	Assertiveness/Forces of Character
	Sense of Humor

[25] Weinberg, *The Psychology of Computer Programming*.

It is much clearer how theatrical designers are considered creative. They make physical designs that the rest of the world can appreciate as art, and are therefore understood implicitly to be creative. But what really is creative thinking? Blogger Merlin Mann has a good definition: "The truth is that creativity is much more about combining the self-discipline to tolerate ambiguity with the will to transform the results into something meaningful. It's not really contradictory; it's largely an issue of intentionality and attention."[26] So creativity is figuring out how to take the ambiguous and make it meaningful. That is exactly what software engineers do when taking a desired program outcome and making it into a working program, and it is what theatrical designers do when reading a script and developing a physical idea out of it.

Software engineering work is not something that initially comes to mind as creative work, at least to the untrained eye. However, good software is only developed through groundbreaking creativity of the involved engineers. The dean of Carnegie Mellon's School of Computer Science, Jeannette Wing, makes the most convincing case for this, explaining that, "Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction."[27] She further argues that software engineering is not about understanding the programing. Also, it requires a complex, fundamental understanding of the problem attempting to be solved, and then finding creative methods to attack those problems utilizing computation.

[26] Mann, "Attention & Ambiguity."
[27] Wing, "Volume 29, No. 3. Pg. 33-35."

Creative people seem to work in similar ways. Across my personal experience, as well as in many overheard or researched anecdotes, similar behaviors could be observed in creative individuals. In an article for “Psychology Today”, quality of life expert Mihaly Csikszentmihalyi observes, "Perhaps the most difficult thing for creative individuals to bear is the sense of loss and emptiness they experience when, for some reason, they cannot work. This is especially painful when a person feels his or her creativity drying out."[28] This observation points to the repeated behavior that creative people dive into the problem in front of them with a drive that is unparalleled. In *The Pixar Touch*, an anecdote about how a vice president would need to remind programmers to go home to sleep or to collect their paychecks speaks to this as well, as do the stories told about Pixar in the preceding section. The creative people working at Pixar will continue in their work as long as is necessary, as long as they feel the work is creatively stimulating.

This happens in theatre as well. Designers and technicians will continue working on a problem or idea until it comes to fruition or a better idea comes along. A common question is “when is the show done”; the common answer is “when the show closes”, meaning that the creative ideas never really stop because the designers behind them are empty when they are not working, trying to improve on their idea. Any prohibition on this type of work will create negative feelings amongst the designers for the reasons pointed out above regarding creative individuals. This may cause a decrease in their happiness or productivity, or a departure to go work on another creatively stimulating project.

[28] Csikszentmihalyi, “The Creative Personality.”

How designers and software engineers actually work creatively is interesting. A common belief is that every creative person has his or her own methods to do his or her best work. Some can only work late at night, some only in certain types of spaces, some only in a particular physical or mental state. Applying any sort of organization or structure to creativity can be looked at hostilely, but there are good arguments for doing so, especially to the process by which creativity occurs.

One designer, Keith Robinson, enjoys a particular workflow and method to creativity, advocating, "I believe that a good designer can be made and the skills needed to be a genuinely creative person can come through discipline, learning, and practice, not just God-given talent. Working hard and getting things done can lead to a more creative life, I'm sure of it."[29] This points to the fact that any person can learn to do creative work given the correct circumstances and workflow, and also implies that people already doing creative work can be more creative by applying practice and discipline to their work.

One of the most enticing methods I've heard of for this comes from *43 Folders Blog*, which is a site dedicated to a method of personal workflow management called "Getting Things Done"[30]. A big component of "Getting Things Done" is to organize tasks by context, meaning location or tools needed to do the task, rather than by the subject of the task. In his post, Merlin Mann advocates for a creative context as well. He establishes that creativity is a physical location or state of mind that is needed for certain kinds of work, and creating such a space for creative work increases

[29] Robinson, "Getting Design Done."
[30] Mann, "Vox Pop."

productivity. This is a different way from how most people treat the idea of context-based organization within “Getting Things Done", but I believe it to be particularly effective given the case that creativity happens best with certain discipline and practice.

Peopleware also discussed a phenomenon known as the Hawthorne effect, which is easily relatable to theatre design and many aspects of software engineering. The Hawthorne Effect states that people perform better when they are trying something new[31]. In theatre, every single design is something inherently new, albeit based on past experiences, research, and ideas. The idea of working on something new each time is what draws many people to theatre, and is a quality that the best designers tend to have, making their work better and their involvement in the show higher quality. For software engineering, most projects involve some new problem or element of a problem, but there is a much wider repository of previous knowledge to pull from. Most programs do not solve every problem from scratch, as theatrical design often does. The Hawthorne Effect implies that the most productive and happiest software engineers are those working on something they have never done before, which a project manager should take into account when assigning tasks and managing the team.

All of these observations are seen in the methods and problem-solving techniques that designers and engineers utilize in their work. Overall, however, the idea of prototyping is the crux of all methods used to develop creative ideas. Prototyping is essentially a test of the ideas that have been developed in order to expose shortcomings. Theatrical designers, especially those in an academic setting, are very

[31] DeMarco and Lister, *Peopleware: Productive Projects and Teams*. Pp. 119

used to going through multiple ideas and expressing those ideas in models and drawings. Those models and drawings are essentially prototypes of their concepts, and are expected within the design process. Software engineers, however, seem to be less open and used to prototyping. Regarding prototyping, Berkun observes that programmers "…might also say that the process is a waste of time (a claim often made of anything that doesn't involve writing code)."[32] It is the project manager's role in software engineering to enforce prototyping experiments in order to ensure project success later on, as discussed in our examination of projects. Unlike theatrical design, software engineers are more hesitant to do this on their own.

The Teams

Though an understanding of the individuals working on a project is important, perhaps more critical is an understanding of how those people work together in a team to develop a software or theatrical product. In both industries, it is rare that a project takes place with only an individual working on it (that would make a manager unnecessary). Working with these teams is the bulk of a project manager's job. Software and theatrical teams share many commonalities, but are distinctly different in structure and role. Establishing those structures will be the first task of this section. Second, we will examine some aspects and best practices of managing teams in these industries.

A prototypical theatrical design team will be made of a director and designers, usually from sound, lighting, scenic, and costume departments (although there is a

[32] Berkun, *The Art of Project Management*.

wide variety of these specific arrangements). The directors are usually seen as the leader and final decision maker of this group, without including those on a higher level of the organizational chart. Each of the designers is highly specialized in their field, and their responsibilities typically do very little crossing over to other departments. The director, on the other hand, comes from a variety of backgrounds and experiences. He or she could be an expert in some aspect of design, or none at all. The experiences of each team member will dictate their relationships and the process by which they collaborate. Gauging this is an important task of the group's production manager. Further, understanding how much of the other design areas the director and other team members understand is essential in determining what areas of the production need more focus, and what the team is communicating effectively about. The project manager should observe and attempt to fill in these gaps, if any, as soon as possible.

Software engineering teams do not have as universally similar a structure. The level of team member specialization will vary widely, which makes understanding that aspect of a team all the more important. The same skill sets are required to complete a project, no matter how many people are on a team. Without a rounded group of skills, the project will not be as successful- artistic and creative projects require many varying viewpoints. Putting together a full complement of skills is the project manager's first task when assembling a team. For instance, many projects could require data structure developers, interaction designers, line programmers, and graphics artists, but some team members might be skilled in more than one of those areas. Then, because the roles are less defined, the task becomes making each team member understand what the skills and roles of the other team members are. In theatre, this is more defined

because of traditional positions, but in software much attention needs to be paid to this.

As mentioned, theatrical teams are made up of a group of people who each have very specific subject expertise and focuses. In order to create a cohesive product, they must work together very closely to integrate these. In a paper on how theatrical team thinking applies to other fields, Kendra Albert accurately contests that "Interdisciplinary collaboration is expected on every single show a theatre person works on."[33] In software, however, the teams tend to be made of people who think more similarly, in a computational manner, and who have more similar backgrounds and skillsets. Theatrical teams have the benefit of a highly interdisciplinary environment, whereas software sometimes struggles with finding a new way to think about a problem. This is a direct argument for two things. First, people in every industry should learn to program because they will then be able to apply their industry specific knowledge to a common problem more easily. Second, software engineering teams should endeavor to work with people who might have no background in software, if only for their divergent way of thinking. For instance, having a user interface designer who does not program but only thinks about the end user experience is becoming more common for some software projects.

Both of these industries have a tendency to habitually assemble teams of the same people over and over again- a trait that has developed for many reasons. The obvious reason is that a team works better when they think on a similar level, making creative work in particular much easier because ideas are communicated with a special,

[33] Albert, "How to Team Problem Solve Right: Advice from the Theatre."

indescribable understanding and trust. On another level, a team of the same people for each project has learned how to fill in the gaps in each other's knowledge and skill sets, allowing them to seamlessly cover all aspects of a project.

Additionally, both of these industries, but especially theatre, are particularly small and the chances of working with someone repeatedly are very high. Weinberg observes that, "From comparisons between teams composed of strangers and teams composed of co-workers, we do know that short-term group behavior is influenced both by past experience with team members and the expectation of having to work with them in the future.[34] There is always the expectation of working with team members again, especially in theatre where having a steady job is invaluable and the path to that is through a cohesive design team. Further, it can be argued that a theatrical design team, due to the stakes of production, is inherently an elite team. This is defined as a group that is hell-bent on the success of the show because it reflects on them personally; goal-oriented; and unique amongst peers because they are working on a product that no one else is doing[35]. Software engineering teams do not typically have quite this level of teamwork, although there are counterexamples, such as the rare specialty Black Teams that DeMarco and Lister discuss in *Peopleware*.

Although software teams are not inherently elite, they do have some aspects that allow them to work in a more efficient, productive manner than theatrical teams. One clear advantage is that, for many organizations, a software team is all in the same place physically. In theatre, many design teams do not all get into a room together until

[34] Weinberg, *The Psychology of Computer Programming*., 91
[35] DeMarco and Lister, *Peopleware: Productive Projects and Teams*.

the first technical rehearsal, when most of their work is already complete. Therefore, it is the theatrical production manager's role to overcome this challenge by opening and maintaining channels of communication between the remote designers. For the software manager, there is much more day-to-day work to be done overseeing this team. When people are in one place, there is a lot more direct interaction that needs to be managed and understood. Increasingly, technology is allowing software companies to work with teams remotely as well, and as that occurs software managers can look at how theatrical managers work with remote teams during all but the end game phase of a project.

A false belief (and one I had when beginning this project) is that creative personalities such as theatrical designers and software engineers are generally more invested in their projects when they choose their own teams and own projects. For instance, theatrical designers at a certain level might only work on shows that challenge or interest them, and companies like Google encourage their employees to spend a large portion of their time working on a project that interests and excites them[36]. A blog post by a young programmer convincingly described how he only wanted to work on projects that interested and challenged him.[37] Further, many software engineering and theatrical design friends have explained how they enjoy what they are working on when surrounded by intellectual and creative equals that they can collaborate with. Additionally, many designers will choose to work on a project

[36] Shaughnessy, "Google as a New Innovation Model (Or Not the 20% Time)."
[37] Church, "What Programmers Want."

because there is another designer or director already committed to it that they either admire or work well with.

A study explained in *Peopleware* corrected this partially false assumption through a description of an experiment they ran:

> "This scheme (choosing their own team and their own project) gave people two unusual degrees of freedom: They got to choose the projects they worked on and the people they worked with. The surprising finding was that the first of these factors didn't matter very much. Management initially feared that only the glamorous projects would be bid for, but it didn't happen that way. Even the most mundane projects were bid for. What seemed to matter was the chance for people to work with those they wanted to work with" (148). [38]

This experiment demonstrates the power and importance of the project team, and further strengthens the argument that these teams are important structures to manage in order to ensure the success of the project.

The idea that programmers care less about the project than the people they are working with refutes a claim made by Weinberg in *The Psychology of Computer Programming*. He states that "If a programmer is indispensable, get rid of him as quickly as possible"[39], meaning that if programmers get so involved and engrossed in a project that no one could replace them because a replacement would not be up to speed, that will eventually taint the project. His claim is that after being so involved in a

[38] Church, “What Programmers Want.”
[39] Weinberg, *The Psychology of Computer Programming*., 100.

project, the programmer will get bored and want to move on to something else, leaving you in the dark. Clearly, there is some disagreement about the importance of the challenges of a particular project for computer programmers. I tend to lean more towards *Peopleware*'s view that the people matter more than the project because all of these people need to have jobs regardless of how interesting they are, and as long as there are interesting coworkers, a project will not bore a programmer.

Peopleware makes the claim that project fragmentation amongst a team, when a team member is working on more than just the one project, is bad. They claim that a team member working on more than one project cannot pay attention to the personal interactions between teammates on any of those given projects. Therefore, the teams are not as tightly knit and are therefore less productive. While working on only one project at a time is fairly normal for the software engineering industry, it is completely atypical of theatre. In a theatrical world, all of the designers are likely working on multiple shows, and the production manager is working on a number of shows as well. *Peopleware* goes on to say that, in software, the project manager might not even be part of a software engineering team- the project manager is merely an external manager. This claim is backed up by the idea that a typical project manager is working on many projects at once, including the ones the engineering team will be working on next, so his or her attention cannot be completely focused on the personal interactions between that team. In theatre, all of the designers and the production manager can be said to be team members because they are all working on multiple shows at a time. Alternately, none of them are really team members and therefore lack some sort of cohesiveness. The trick of the production manager then is to do the best at personal

interactions as possible, even given the constraints of other projects. This is also an argument for the potential success of a resident design team that works together over and over again, as their attention can be dedicated to the same group of people repeatedly.

The Quirks

The concept of flow kept showing up in materials about software productivity. Flow is a state of mind in which the most productive work is undertaken, characterized by uninterrupted, focused attention. One author lamented the vast amount of interruptions in an office, which led to a lack of productivity. However, the same author observes, "…when a person is working in the area of his of her expertise, worries and cares fall away, replaced by a sense of bliss."[40] This is an example of flow, in which the person puts all of their energy into a project. It is the role of management to create an environment such that the designers and engineers can achieve a state of flow easily and regularly.

Flow is not only a productive occurrence; it is also intellectually good for the project and person working on it. Merlin Mann observes that, "For as long as he or she can stay in that Flow state, a good artist is capable of synthesizing unbelievably disparate material and ideas in a way that's often satisfying *and* productive."[41] His assertion that happy engineers and designers come from a satisfying state of synthesis is important for manages to realize, so that they can leave those they manage to work in a way that

[40] Csikszentmihalyi, “The Creative Personality.”
[41] Mann, “Attention & Ambiguity.”

will keep them happy and useful. Theatrical designers have a version of flow as well. There are instances in which a designer will produce a ridiculous amount of work very quickly if he or she gets into the correct mood or flow. The difference with theatre is that the project manager will likely not have as direct control over the environment that the designer is working in, so cannot create an area conducive to this.

The idea of a “Hero Complex” or similar traits repeated itself across many sources describing the way software engineers work. This phenomenon requires exploration, as it seems common and appears to have many parallels to theatre both in occurrences and overall psyche of the people that work in these industries. While everyone describes this complex slightly differently, I am choosing to examine it as purposeful perpetration (or the lack of action entirely) of an action that will eventually hinder the project for the purpose of being able to come back later and heroically fix the problem. Included in this is the lack of timely preparation for an element that will ultimately be completed regardless of timeline. Project managers will likely be able to think of an instance of this occurring in their work.

One of the strongest reasons for this trait across theatrical designers is the background from which most designers come. Commonly, the first step in a professional career is working at smaller theaters that are not staffed as fully as larger institutions. At these sorts of theaters, there is more emphasis placed on the designer’s role in implementing their design. Some places will have these designers build scenery, paint, and sew. Often, this involvement by the designers is not initially expected or discussed as a duty of employment, but the designers end up filling in anyways in order to get the show they want and the reputation that they need to be hired again

and go on to larger jobs. They designed something unfeasible, and the only way to make it happen is to step in and save the day. A similar occurrence happens in the software engineering environment, as Berkun observes, "The Hero Complex mostly develops in people who started their careers in start-ups for very small (volatile) firms."[42] He cites the same reasons for this development- filling in the areas that are otherwise not their jobs only for the sake of completing the project. An example of this is a software engineer fixing a “bug” in their program that they’ve known about for a while, but which has not manifested itself until recently.

If the Hero Complex is so common in software engineering and theatre, it then raises the question of what management can and should do about it. *Peopleware* accepts the inevitability of this phenomenon, saying, "It’s when the truly Herculean effort is called for that we have to learn to do work less of the time and think about the work more. The more heroic the effort required, the more important it is that the team members learn to interact well and enjoy it."[43] So the project manager must then ensure that the team is interacting well in any way possible and regardless of the task’s difficulty.

[42] Berkun, *The Art of Project Management*.
[43] DeMarco and Lister, *Peopleware: Productive Projects and Teams*., 12.

CONCLUDING LESSONS

After examining the similarities between software and theatrical projects, the question then becomes what project managers can take away from these similarities. Throughout the above analysis, many brief suggestions have been offered. We'll now focus on a few of those in particular, as well as some yet-unmentioned suggestions. These include regular code review's place in theatre, management's interactions with clients and upper management, working with remote or physically distant groups, aspects of the end game and technical rehearsals, and a look at hiring and interview best-practices.

Earlier, we discussed the benefits that the software industry gets from regular, formal code reviews and we noted how this is different from a regular check-in of overall project status. Some of those benefits included an audit of accuracy and effectiveness of the work to date. Weinberg observes "A programmer who truly sees his program as an extension of his own ego is not going to be trying to find all the errors in that program."[44] This means that a review of a programmer's work by an outside source, such as what typically happens during a code review, is essential. The same is true in theatre- a designer or technician that doesn't put their work up for full, detailed review by an external party will not be able to find inherent issues.

It is easier to conduct this type of review in software because specific parts of code can be empirically tested for accuracy and effectiveness. The regular process is for a peer to sit down and attempt to understand the code, or for the project manager to do

[44] Weinberg, *The Psychology of Computer Programming*., 53

the same. In theatre, this could take the form of a detailed review of all work to date for a department or a detailed sampling of a specific element. This doesn't necessarily need to involve the project manager, but could. For example, including the production manager on the occasional costume fitting would be a good way to spot potential issues in a costume design process and how that might effect the sound department. Or maybe a lighting designer visits the scenery shop to peer review and attempt to learn about the details of building a particular piece of scenery. This would increase the understanding of the peer doing the reviewing of the object or field they are reviewing, allowing them to expand their thinking regarding how it might affect their own work. Additionally, the area or person whose work is being reviewed could benefit from the reviewer's thoughts about how departments interact and how their work will ultimately be used. This auditing does not need to take place for every element, just a sampling, and would encourage collaboration amongst team members at a more detailed level.

In theatre, the production manager's job is to interface between the design team and the company's artistic management (in cases where these do not overlap). This close relationship allows for fewer changes later in the process, as the upper management is ideally involved throughout and has a deep understanding of the product being developed. In software, project managers could increase productivity by keeping the client or upper management in the loop more often. Current practices engage upper management more in the development of requirements and problem definition, but less once those are defined. This model does not allow for as much flexibility later in the process because there is the need for the upper management to catch up, taking time from decision-making. Imagine if a software executive had a

greater understanding of the difficulties facing a certain aspect of a program. They would better understand the process and be able to make necessary decisions with a different outlook.

Recently, the software industry has been going through a period of globalization in terms of their project staff. Project teams are spread out across multiple offices, and the project manager's role has become one of more coordination than requirements development, which will eventually lead to lower quality products. This spread out team structure is almost universally inherent to theatrical design processes, so theatrical managers are used to working across distances. One method they use is ensuring that communication not only happens through the project manager, but directly between designers and local staff members. This could be useful for software as well. For example, assigning team members specific partners in the local office so that there is always a contact person would foster peer-to-peer communication and the free sharing of ideas. One important difference to note, however, is that in theatre the designers are often working on their own time schedule with large deadlines to meet whereas in software the engineer is an employee who works at a set time of day dictated by the company. This makes distance management in software easier because of predictability, and is something that might eventually find its way into theatre if a company is working regularly with the same team.

Software is known for not delivering projects on time, whereas theatre will usually open as scheduled. The software endgame, a likely place for this schedule slipping to occur, could be improved by physical to allow for easier group decisions. This is something that the project manager can easily implement without having to be

consistently present- all that is needed is a room to put the key team members in, and some food to keep them happy. Additionally, using prioritization techniques during the end game, both with team members and clients/upper level management in order to explain trade-offs necessary to be on time would be extremely useful. Theatre uses both of these techniques in the technical rehearsal process, which allows for everyone to feel the pressure of an opening and work towards solutions in a much more streamlined fashion.

Finally, the way designers and software engineers are hired deserves some examination. The software industry has slowly been adopting hiring practices that involve more demonstrations of candidate abilities than interview sessions. This can take the form of case studies, programming challenges, problem sets, and exams. *Peopleware* makes the case for such hiring practices in software, claiming, "You need to examine a sample of those products to see the quality of work the candidate does. This may seem obvious, but it's almost always overlooked by development managers."[45] In theatre, standard practice when hiring a designer is to examine their previous work in a portfolio and discussion-based setting. Software has slowly begun utilizing similar methods, but may begin straying too far into the product and previous-work based evaluation, which leaves little room for teamwork evaluations.

There are of course downsides to hiring employees based solely on ability to do quality work or show exemplary results on a programming challenge. The people you hire in those ways might not be as knowledgeable in teamwork or communication as their less technically experienced peers. Therefore, a company should work to strike a

[45] DeMarco and Lister, *Peopleware: Productive Projects and Teams*., 101.

balance between abilities testing and an evaluation of the interviewee as a whole through team exercises or more standard interview practices. For theatre, this could look like an examination of previous work in a portfolio, as normally occurs, as well as an interview regarding team-based philosophies. Peer reviews and references are also very useful when evaluating designers. In software, group case studies could be helpful, as would discussion about previous collaborative efforts and the methods they have utilized to make those successful.

Clearly, software engineering and theatre parallel each other in some fascinating ways. As globalization continues to increase, and the use of technology in all industries becomes more prevalent, project managers in all industries will see their realms combining. An understanding of more than one industry will become essential in order to expand effectiveness and project success. Theatre and software have already successfully begun this combination, and learning from that example will make project managers more effective into the future.

Works Cited

Albert, Kendra. “How to Team Problem Solve Right: Advice from the Theatre.” *Lawspeak for L33t Speakers*, December 9, 2012. http://www.kendraalbert.com/post/37601795761/how-to-team-problem-solve-right-advice-from-the.

Berkun, Scott. *The Art of Project Management*. Sebastopol, CA: O’Reilly Media, 2005.

Borwick, Doug. “Mainstreaming on My Mind.” *Engaging Matters*. Accessed February 11, 2013. http://www.artsjournal.com/engage/2013/01/mainstreaming-on-my-mind/.

Carl, Polly. “Truthiness in the Politics of Theatre.” *HowlRound*. Accessed September 18, 2012. http://www.howlround.com/truthiness-in-the-politics-of-theatre-by-polly-carl/.

Church, Michael O. “What Programmers Want.” *Michael O.Church*, October 30, 2012. http://michaelochurch.wordpress.com/2012/10/30/what-programmers-want/.

Csikszentmihalyi, Mihaly. “The Creative Personality,” July 1, 1996. http://www.psychologytoday.com/articles/199607/the-creative-personality.

DeMarco, Tom, and Timothy Lister. *Peopleware: Productive Projects and Teams*. New York, NY: Dorset House Publishing Co., 1987.

Holcomb, David. “Where Do Production Managers Come From,” n.d.

Holk, Eric. “How Do We Read Code?” *The Incredible Holk*, December 18, 2012. http://blog.theincredibleholk.org/blog/2012/12/18/how-do-we-read-code/.

Humphrey, Watts S. *Managing the Software Process*. Addison-Wesley Publishing Company, Inc., 1989.

Lloyd, Iris. “Don’t Define the Problem.” *Public Administration Review* 38, no. 3 (May 1978): 283. doi:10.2307/975684.

Mann, Merlin. “Attention & Ambiguity: The Non-Paradox of Creative Work.” *43 Folders Blog*, August 20, 2008. http://www.43folders.com/2008/08/20/creative-paradox.

———. “Vox Pop: Implementing GTD for Creative Work?” *43 Folders Blog*, July 27, 2007. http://www.43folders.com/2007/07/27/vox-pop-creative-gtd.

Price, David. *The Pixar Touch: The Making of a Company*. New York: Alfred A. Knopf, 2008.

Robinson, D. Keith. “Getting Design Done.” *CreativePro.com*, July 27, 2007. http://www.creativepro.com/article/getting-design-done.

Shaughnessy, Haydn. “Google as a New Innovation Model (Or Not the 20% Time).” *Forbes*, February 7, 2013. http://www.forbes.com/sites/haydnshaughnessy/2013/02/07/googles-role-as-an-innovation-model/.

Weinberg, Gerald M. *The Psychology of Computer Programming*. Computer Science Series. New York: Van Nostrand Reinhold Company, 1971.

Wing, Jeannette. “Computational Thinking.” *Communications of the ACM*, March 2006. http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf.

Works Consulted

Albert, Kendra. “How to Team Problem Solve Right: Advice from the Theatre.” *Lawspeak for L33t Speakers*, December 9, 2012. http://www.kendraalbert.com/post/37601795761/how-to-team-problem-solve-right-advice-from-the.

Andrews, Robert. “GTD: A New Cult for the Info Age.” *WIRED*. Accessed January 31, 2013. http://www.wired.com/culture/lifestyle/news/2005/07/68103?currentPage=2.

Baer, Drake. “Why Doing Awesome Work Means Making Yourself Vulnerable.” *Fast Company*, September 17, 2012. http://www.fastcompany.com/3001319/why-doing-awesome-work-means-making-yourself-vulnerable.

Berkun, Scott. *The Art of Project Management*. Sebastopol, CA: O’Reilly Media, 2005.

Borwick, Doug. “Mainstreaming on My Mind.” *Engaging Matters*. Accessed February 11, 2013. http://www.artsjournal.com/engage/2013/01/mainstreaming-on-my-mind/.

Carl, Polly. “Truthiness in the Politics of Theatre.” *HowlRound*. Accessed September 18, 2012. http://www.howlround.com/truthiness-in-the-politics-of-theatre-by-polly-carl/.

Carnegie Mellon. “Entertainment Analytics.” *Homepage Stories*. Accessed November 7, 2012. http://www.cmu.edu/homepage/society/2012/fall/entertainment-analytics.shtmlhttp://www.cmu.edu/homepage/society/2012/fall/entertainment-analytics.shtml.

“Carnegie Mellon Analysis Shows Online Songwriters Seek Collaborators with Complementary Skills.” Accessed February 25, 2013.

http://www.sciencecodex.com/carnegie_mellon_analysis_shows_online_songwriters_seek_collaborators_with_complementary_skills-106659.

Church, Michael O. “What Programmers Want.” *Michael O.Church*, October 30, 2012. http://michaelochurch.wordpress.com/2012/10/30/what-programmers-want/.

Csikszentmihalyi, Mihaly. “The Creative Personality,” July 1, 1996. http://www.psychologytoday.com/articles/199607/the-creative-personality.

Dean, Peter. *Production Management: Making Shows Happen*. Ramsburry, Marlborough: The Crowood Press Ltd., 2002.

DeMarco, Tom, and Timothy Lister. *Peopleware: Productive Projects and Teams*. New York, NY: Dorset House Publishing Co., 1987.

Holcomb, David. “Where Do Production Managers Come From,” n.d.

Holk, Eric. “How Do We Read Code?” *The Incredible Holk*, December 18, 2012. http://blog.theincredibleholk.org/blog/2012/12/18/how-do-we-read-code/.

Humphrey, Watts S. *Managing the Software Process*. Addison-Wesley Publishing Company, Inc., 1989.

Legassie, Calvin. “The Better Production Meeting Handbook.” Carnegie Mellon University, 2012.

Lloyd, Iris. “Don’t Define the Problem.” *Public Administration Review* 38, no. 3 (May 1978): 283. doi:10.2307/975684.

Mann, Merlin. “Attention & Ambiguity: The Non-Paradox of Creative Work.” *43 Folders Blog*, August 20, 2008. http://www.43folders.com/2008/08/20/creative-paradox.

———. “Vox Pop: Implementing GTD for Creative Work?” *43 Folders Blog*, July 27, 2007. http://www.43folders.com/2007/07/27/vox-pop-creative-gtd.

Peake, Robert. "'Getting Software Done' (part 1)." *43 Folders Blog*, October 17, 2006. http://www.43folders.com/2006/10/17/robert-peake-part-one.

———. "'Getting Software Done' (part 2)," October 18, 2006. http://www.43folders.com/2006/10/18/robert-peake-part-two.

Price, David. *The Pixar Touch: The Making of a Company*. New York: Alfred A. Knopf, 2008.

Robinson, D. Keith. "Getting Design Done." *CreativePro.com*, July 27, 2007. http://www.creativepro.com/article/getting-design-done.

Shaughnessy, Haydn. "Google as a New Innovation Model (Or Not the 20% Time)." *Forbes*, February 7, 2013. http://www.forbes.com/sites/haydnshaughnessy/2013/02/07/googles-role-as-an-innovation-model/.

Steele, Oliver. "Responsive Interfaces and Effective People." *Oliver Steele*, September 11, 2004. http://osteele.com/posts/2004/09/responsive-interfaces-and-effective-people.

Stross, Randall. "Air Force Stumbles Over Software Modernization Project." *The New York Times*, December 8, 2012, sec. Technology. http://www.nytimes.com/2012/12/09/technology/air-force-stumbles-over-software-modernization-project.html.

Tuutti, Camille. "Will the Digital Strategy Change Management Patterns?" *FCW*, July 12, 2012. http://fcw.com/articles/2012/07/15/home-page-management-watch-digital-strategy.aspx.

Weinberg, Gerald M. *The Psychology of Computer Programming*. Computer Science Series. New York: Van Nostrand Reinhold Company, 1971.

Wing, Jeannette. “Computational Thinking.” *Communications of the ACM*, March 2006. http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf.