Editor: **Rafael Prikladnicki**
Pontifica Universidade Catolica
do Rio Grande do Sul
rafael.prikladnicki@pucrs.br

Software Project Management

Learning from Our Mistakes

Pedro Silva, Ana M. Moreno, and Lawrence Peters

IN THE JANUARY/FEBRUARY 2014 *IEEE Software*, the Voice of Evidence article, "Looking for the Holy Grail of Software Development," reviewed the main practices that software project managers should engage in to make success more likely.[1] A complementary question is, what practices should they avoid to make success more likely? Answering this question will help current and future software project managers prevent, or at least mitigate, problematic scenarios that, if unresolved, will lead to additional project failures.

Antipatterns come into play for formally describing dysfunctional approaches to problem solving and offering refactored solutions for successfully overcoming dysfunctions.[2] In software development, antipatterns are related to different activities including software project management. What are these antipatterns, and to what software project management issues are they related?

Software Project Management Antipatterns

We performed an extensive literature search according to a systematic-mapping-studies protocol,[3] looking for information about software project management antipatterns in journals and conference publications over the last 10 years. We searched five major databases: IEEE Xplore, the ACM Portal, the Web of Knowledge, Google Scholar, and the Directory of Open Access Journals. The search string was

> (anti-pattern OR antipattern OR anti pattern OR malpractice OR bad practice)
>
> AND (software project management OR project management OR management)

We looked for not just simple prose descriptions of errors but also well-reported software project management antipatterns:

> *A properly documented antipattern describes a general form; the primary causes which led to the general form; symptoms describing how to recognize the general form; the consequences of the general form; and a re-factored solution describing how to change the antipattern into a healthier situation.*[4]

Additionally, we looked for antipatterns related to the five software project management activities identified in classical software project management literature:[5,6] planning, scheduling, controlling, staffing, and motivating. Details of the literature review appear elsewhere.[7]

Surprisingly, our search didn't provide significant results. We then searched for books and other sources. We found three books[4,8,9] and a few websites,

0740-7459/15/$31.00 © 2015 IEEE

TABLE 1

A consolidated list of software project management antipatterns.

No.	Antipattern name	Description	Reference
1	Absentee Manager	A manager who engages in avoidance behavior or is invisible for long time periods	9
2	All You Have Is a Hammer	One-dimensional management that uses the same techniques on all subordinates in all situations	9
3	Appointed Team	The false assumption that a management-selected group of people will immediately become a team	10
4	The Brawl	A project manager with no leadership or management experience	8
5	Detailitis Plan	Excessive planning leading to complex schedules with a high level of detail, giving the false perception that the project is fully under control	4
6	The Domino Effect	Moving critical resources between projects, blurring project boundaries	8
7	Dry Waterhole	Specifying stringent requirements for a job when this isn't strictly necessary, resulting in a limited pool of available talent	10
8	Fire Drill	Months of boredom followed by demands for immediate delivery	4
9	Glass Case Plan	Lack of tracking and updating of initial plans, assuming the plan is enough	4
10	Inflexible Plan	Lack of flexible plans and processes	8
11	Irrational Management	Irrational management decisions, habitual indecisiveness, and other negative management practices	4, 9
12	Leader Not Manager	A manager with a vision (a leader) but no plan or management methodology	9
13	Micromanagement	Excessive management involvement in tasks beyond their responsibility	8
14	Mushroom Management	Isolating developers from end users, under the mistaken assumption that the requirements are stable and well understood by both the software team and end users at project inception	4, 9
15	Myopic Delivery	Management insisting on the original delivery date even when reducing staff or funding	8
16	Process Disintegration	Failing processes due to a decline in overall cooperation and morale	8
17	Project Mismanagement	Lack of proper software project monitoring and control	4
18	Proletariat Hero	The false assumption that coercion is an efficient way to increase productivity	9
19	Rising Upstart	Superstars who can't wait their time and want to skip learning phases	9
20	Road to Nowhere	Lack of planning	9
21	Size Isn't Everything	Assuming developers are interchangeable and that the number of people working on a problem is inversely proportional to the development time	4, 8, 9
22	Ultimate Weapon	Relying heavily on a superstar on the team	9

mostly related to the antipatterns detailed in those books. To complement the books' information, we used the Portland Pattern Repository.[10]

Table 1 shows our consolidated list of antipatterns, which resulted from a detailed scrutiny of the literature we found.[7]

Antipattern Categories

We thought it would be interesting to go deeper into the analysis of the previous antipatterns by dealing with the following questions:

- Which of the five software project management activities are the antipatterns related to?

TABLE 2

Antipattern categories.

Category	Criteria	Antipattern no. (see Table 1)	%
Impacted software product management activity	Controlling	1, 4,5, 8, 9, 11, 14, 15, 17	41
	Motivating	1, 2, 4, 11, 13, 16, 18, 19, 22	41
	Planning	10, 12, 20	14
	Scheduling	5, 21	9
	Staffing	3, 7	9
Impacted role	Developer	4, 5, 6, 7, 9, 10, 16, 17, 21, 22	50
	Manager	5, 6, 7, 9, 10, 16, 17, 19, 21, 22	50
	Customer	8, 14, 15, 17	18
Root cause	Ignorance	1, 3, 4, 6, 9, 10, 11, 12, 13, 14, 17, 18, 19, 21	64
	Sloth	1, 10, 16, 17, 20, 22	27
	Pride	13, 15, 22	14
	Avarice	5, 7	9
	Haste	8	5
Solution type	Training	2, 3, 4, 6, 10, 11, 12, 13, 14, 17, 18, 19, 21	59
	Process	5, 7, 8, 9, 10, 15, 20	32
	Role	1, 13, 22	14
	Technology	16	5

- Which general roles in a software project (developers, managers, or customers) do the antipatterns impact?
- Are these antipatterns due to ignorance, sloth, pride, avarice (ambitious behaviors related to not only money but also people, project schedules, and delivery dates), or haste?
- What type of solution does each antipattern imply? Is the solution training-based, process-based, role-based (focusing on assigning responsibility to an individual or group), or technology-based?

We answered these questions on the basis of the literature and our experience and knowledge.[7] Additionally, two senior software project managers at international software consulting companies assessed our categorizations through a detailed discussion with us. The categorizations might vary slightly depending on the actors, but our aim is to bring to practitioners' attention the main factors related to each antipattern.

You can access and work with the categorization results at http://is.ls.fi.upm.es/research/spmantipatterns/home.html. A simple Web tool lets you sort the antipatterns according to the previous questions and filter them according to criteria (for example, viewing all antipatterns due to ignorance). Table 2 summarizes this categorization (in some cases, a particular antipattern might be related to different criteria in the same category, so the percentages for each category might exceed 100 percent).

The software project management activities most impacted by antipatterns were, not surprisingly, those that last the longest throughout a project: controlling and motivating.

Investigating the roles most impacted by antipatterns led to a paradoxical discovery. In half of the antipatterns, the manager experienced the greatest impact—managers who, in many cases and for various reasons, had created the situation they were suffering from. This ironic revelation should motivate project man-

agers to avoid malpractices and engage in continuous improvement and professional development.

As we expected, antipatterns also had a relevant effect on developers because they're the main performers of the work orchestrated by project managers. Additionally, a noteworthy finding is that several antipatterns directly impacted customers. Although all the antipatterns represent undesirable scenarios, project managers should particularly avoid those that significantly affect customers.

Software project managers often identify schedule pressure, a manifestation of haste, as a cause of project troubles. However, we found that haste was the least common root cause of the antipatterns; the most common cause was ignorance (attributable to the project manager's lack of experience or training). This result supports the observation that staff is sometimes assigned to project management without being fully qualified or trained.[11] As Table 2 shows, project management training can solve most of the antipatterns. So, the solution shouldn't be to terminate underperforming software project managers and hire new ones with potentially similar limitations. A much more powerful and long-lasting solution is to invest in these managers, training them to enhance their skills and preparing aspiring managers for the future.

Software project managers are in a unique position to identify and avoid antipatterns. Here's how.

First, identify specific principles and practices to tackle for the antipatterns, according to your development process. For example, in agile project management, creating a general release plan will help prevent Detailitist Plan, and the continuous delivery of working software will help prevent Fire Drill.

Second, at regularly scheduled points during a project, meet with your team and, as a group, identify what is and isn't going well. Honesty and openness are essential. No one should be defensive. Avoid criticism.

Third, try to map the issues you've identified to the antipatterns in Table 1, keeping in mind that you might have identified new antipatterns. The antipatterns in Table 1 might be only the tip of the iceberg.

Fourth, on the next project, avoid the antipatterns you've identified and repeat this process throughout the project and at its conclusion.

Finally, software project managers should receive training before they assume this role and during their tenure. In less than a decade, the knowledge about motivation, productivity, and team development has grown by leaps and bounds.[12] Keeping the software project manager up to date will have benefits today and for years to come.

References

1. P. Ghazi, A.M. Moreno, and L.J. Peters, "Looking for the Holy Grail of Software Development," *IEEE Software*, vol. 31, no. 1, 2014, pp. 92–96.
2. A. Koenig, "Patterns and Antipatterns," *J. Object-Oriented Programming*, vol. 8, no. 1, 1995, pp. 46–48.
3. K. Petersen et al., "Systematic Mapping Studies in Software Engineering," *Proc. 12th Int'l Conf. Evaluation and Assessment in Software Eng.* (EASE 08), 2008, pp. 68–77.
4. W.J. Brown et al., *AntiPatterns: Refactoring Software, Architecture, and Projects in Crisis*, John Wiley & Sons, 1998.
5. L.J. Peters, *Getting Results from Software Development Teams*, Microsoft Press, 2008.
6. H.R. Kerzner, *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, 11th ed., John Wiley & Sons, 2013.
7. P. Silva, "Categorization of Anti-patterns in Software Project Management," master's thesis, Universidad Politécnica de Madrid, 2014; http://oa.upm.es/32705.
8. W.J. Brown, H.W. McCormick, and S.W. Thomas, *AntiPatterns in Project Management*, John Wiley & Sons, 2000.
9. P.A. Laplante and C.J. Neill, *Antipatterns: Identification, Refactoring, and Management*, Taylor & Francis, 2005.
10. "Management Anti Pattern Roadmap," *Portland Pattern Repository*, 2014; http://c2.com/cgi/wiki?ManagementAntiPatternRoadMap.
11. R. Katz, "Motivating Technical Professionals Today," *IEEE Eng. Management Rev.*, vol. 41, no. 1, 2013, pp. 28–38.
12. L.J. Peters, "*Managing Software Projects: On the Edge of Chaos, from Antipatterns to Success*," Kindle ebook, Software Consultants Int'l, 2015.

PEDRO SILVA is an embedded-software engineer and a recent master's graduate at Universidad Politécnica de Madrid. Contact him at pedro.pdesilva@gmail.com

ANA M. MORENO is a full professor at Universidad Politécnica de Madrid. Contact her at ammoreno@fi.upm.es.

LAWRENCE PETERS is a project manager, consultant, and part-time lecturer at Universidad Politécnica de Madrid. Contact him at ljpeters42@gmail.com

IEEE Software

visit us online

computer.org/software