International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Predictive Software Engineering: Transform Custom Software Development into Effective Business Solutions

Boris Kontsevoi[1], Sergey Kizyan[2]
[1, 2]Intetics Inc., Naples, FL, USA
Email address: [1]boris@intetics.com, [2]s.kizyan@intetics.com

Abstract**—** *The paper examines the principles of the Predictive Software Engineering (PSE) framework. The authors examine how PSE enables custom software development companies to offer transparent services and products while staying within the intended budget and a guaranteed budget. The paper will cover all 7 principles of PSE: (1) Meaningful Customer Care, (2) Transparent End-to-End Control, (3) Proven Productivity, (4) Efficient Distributed Teams, (5) Disciplined Agile Delivery Process, (6) Measurable Quality Management and Technical Debt Reduction, and (7) Sound Human Development.*

Keywords**—** *Agile: Disciplined Agile Delivery: Distributed Team: Predictive Software Engineering: Measurable Quality Management and Technical Debt Reduction System (MQM&TDR).*

I. Introduction

Predictive Software Engineering (PSE) mitigates the "random" and debatable components of software development, transforming it into a precise and predictable engineering exercise. Through its 7 principles, PSE enables software development teams to standardize their engineering, making their processes controllable and transparent.

In this article, we will examine the core principles of PSE, each of which was formed in accordance with 25+ years of development experience.

II. Meaningful Customer Care

Whether your business is B2B, B2C, or another type, it's focused on the customer. Your success is measured by how many customers you have, their satisfaction, and their brand loyalty.

Until your customers recognize the quality of your products or services and follow through with purchases, the company is failing. Therefore, to understand whether you're moving in the right direction, customer satisfaction and feedback should be the primary KPIs you measure.

If your company isn't moving in the right direction, your next question should be: how do you transform unhappy customers into happy ones? There are 8 strategies for doing so:

1. Implement a seamless onboarding process for all customers.
2. Introduce a customer portal.
3. Set up a clear governance model.
4. Draw up a clear escalation path.
5. Conduct early problem discovery.
6. Carry out deep customer complaint analysis.
7. Managing customer satisfaction.
8. Set realistic expectations.

III. Transparent End-to-End Control

In the business world, transparency indicates trust, openness, and cooperation. Only a small percentage of organizations are transparent, and achieving this status requires plenty of effort. But the rewards are well worth it; clients recognize that working with you is less risky and more reliable.

To ensure transparency outside the organization:

1. Conduct governance meetings.
2. Implement Agile methodologies.
3. Conduct status reporting with every project.
4. Use tools to track teamwork.
5. Use project portals that enable customers to track their projects at all times.
6. Conduct process audits.
7. Use proactive project monitoring rather than reactive.
8. Implement onsite visits when the project begins, when there's a big delivery, and when there are any communication issues.

To ensure transparency inside the organization:

1. Periodically hold C-level meetings.
2. Sync meetings with the department director.
3. Hold regular checks with project managers.
4. Involve the talent management team in teamwork.
5. Hold team-building and company events.

IV. Proven Productivity

Proving the productivity of your software engineers can be a difficult task, but it's relevant to all organizations. There are two tools for measuring productivity and velocity during development: Measurable Quality Management and Technical Debt Reduction System (MQM&TDR, which will be addressed in Section VII) and the Software Development Efficiency KPI.

The latter contains 10 sub-KPIs: (1) static code analysis for the project, (2) static code analysis for the developers' code, (3) planned vs. actual work, (4) bugs per line of code, (5) opened bugs per feature, (6) templates missed, (7) algorithms missed, (8) standard libraries use missed, (9) missed exception/error handling, (10) security problems.

Boris Kontsevoi and Sergey Kizyan, "Predictive Software Engineering: Transform Custom Software Development into Effective Business Solutions," *International Research Journal of Advanced Engineering and Science*, Volume 6, Issue 4, pp. 178-180, 2021.

Some guidelines for proving productivity for these KPIs include:

- 2+ developers should review the same code.
- The entire code review process should be documented.
- The team manager or leader should review and approve the KPI results.
- Use tools that prohibit cheating while also simplifying the review process.
- Try to review less than 500 lines of code at once.

V. Efficient Distributed Teams

Physical locations no longer limit organizations; distributed teams make it possible to build a geographically diverse workforce. Some benefits of using the distributed team model include time effectiveness, cost savings, and talent pool diversification:

- Time Effectiveness: When you have distributed teams, your employees are located across the world — therefore, they can work in multiple time zones. By strategically placing your development centers, you can make sure that at least one team is available at all times. This is really helpful for organizations that need 24/7 support.
- Cost Savings: By hiring team members from other countries, organizations can cut costs on salaries, rentals, and operating fees.
- Talent Pool Diversification: Through the distributed team model, companies can access a practically unlimited talent pool of highly skilled developers.

But as physical distance increases, communication problems and inefficiency may arise. To ensure your distributed teams are as efficient as possible, follow these guidelines:

1. Design tasks in a way that facilitates cooperation, considering time constraints, industry regulations, workloads, etc.
2. Clearly define all members' roles and duties.
3. Use well-defined, appropriate communication tactics, such as closed-loop communication.
4. Loosely couple tasks, so team members' workloads are not too dependent upon each other.
5. Team members should mutually monitor performance while still carrying out their own tasks [1].

There are also social aspects to consider: because your team members are located in various areas of the world, social norms and communication styles might vary drastically. To mitigate this, you should foremost ensure that the team has appropriate, effective leadership that can motivate members, facilitate cohesion, and offer coaching as needed.

Furthermore, it is important to promote common ground between the team members and, when possible, train the teams in a common environment.

VI. Disciplined Agile Delivery Process

Nowadays, nearly every software development team follows Agile methodologies — but there are thousands of adaptations. We've found that Disciplined Agile Delivery (DAD) is one of the most advantageous versions for service organizations.

DAD's lifecycle is different from typical Agile methods; it has three main phases: inception, construction, and transition [2].

A. *Inception*

- Identify the project's vision
- Obtain stakeholder agreement with the project's vision
- Identify the project plan, initial technical strategy, and initial requirements.
- Create the initial team.

B. *Construction*

- Address the stakeholders' changing needs.
- Produce a possibly usable solution.
- Move closer to the deployable release.
- Maintain or improve the existing quality levels.
- Address the biggest risks.

C. *Transition*

- Ensure the solution is ready for production.
- Ensure stakeholders are ready to receive it.
- Deploy the solution to production.

VII. Measurable Quality Management and Technical Debt Reduction

The Measurable Quality Management and Technical Debt Reduction System (MQM&TDR) is used to measure a project's technical debts, assess the quality level of software, and determine the product's business efficiency.

MQM&TDR isn't just for techies; it is relevant to everybody working with a product. Testers and developers get an unbiased assessment of their output, managers get feedback on the overall capacity, and users get a well-performing, reliable final product.

It's also useful to investors, as they can use the results to better define the product's fair market value and assess investment risks.

MQM&TDR evaluates the following components of software:

- Source code quality
- Usability
- Security
- Performance
- Business logic
- Solution architecture
- Data quality
- Open-source software & other 3rd-party code

With this knowledge, MQM&TDR brings product owners, development teams, and investors these benefits:

1. Technical debts can be paid before getting out of hand.
2. Redevelopment and support costs are greatly cut.
3. Progress evaluation is adjusted.
4. The product's business efficiency is predicted.
5. Comprehensive quality analysis is conducted.
6. The product's key features are analyzed in detail.

Boris Kontsevoi and Sergey Kizyan, “Predictive Software Engineering: Transform Custom Software Development into Effective Business Solutions,” *International Research Journal of Advanced Engineering and Science*, Volume 6, Issue 4, pp. 178-180, 2021.

7. A compliance check is carried out.
8. Improvement recommendations are given.

MQM&TDR is recommended for every software development project, and it has received resounding support from clients.

VIII. Sound Human Development

Humans aren't resources, regardless of how the HR department is named — they are agents that make decisions, further their skills, and set goals.

By contributing to human development, you aren't just improving the skills of your workforce; you are also facilitating a sense of fulfillment and bolstering loyalty [3].

A. Benefits

One area to consider is employee benefits; you can include training and certification and English courses in addition to standard benefits (medical insurance, sports compensation, food in the office, etc.)

B. Mentor Program

An internship/mentorship program is incredibly helpful. The interns are learning from real projects rather than getting theoretical knowledge. What's more, they aren't just learning about programming and testing; they're also getting communication and processes knowledge. And quite often, after the internship concludes, they remain on the team, so your people are training a future partner.

C. Performance Review and Career Growth

Create an individual development plan for every employee. Sit with the employee and the talent management team, listen to their goals and career decisions, and create a career path plan. Another strategy is to conduct proprietary performance reviews. Every 6 or 12 months, have the employee conduct self-measurement, and then present your own scores. Review the scores together and decide on the next cycle's steps for improved productivity.

D. Skills Improvement

Always be ready to help your team members take courses, attend training, and get certifications. You can help people grow with these three main strategies: (1) bring in external advice from experts whenever necessary, (2) set up plenty of internal and web training, and (3) implement Centers of Excellence.

IX. Risks That PSE Covers

When a software development team adopts Predictive Software Engineering and implements the principles listed above, they will effectively mitigate many risks, including:

- Executives failing to support the project (Transparent Control and Meaningful Care)
- Ill-defined scope (Transparent Control)
- Inaccurate estimates (Disciplined Agile)
- Resource shortfalls (Distributed Teams)
- Inadequate training (Sound Human Development)
- Ambiguous decisions (Meaningful Care)
- Delays in required infrastructure (Transparent Control)

Acknowledgment

Predictive Software Engineering (PSE) is a framework that addresses bottlenecks of custom software product development. It reconstructs the reliable approach to delivering software development services.

PSE overcomes the “art” component of programming. It makes software engineering exactly what it was meant to be: an engineering exercise that is precise and predictable.

The PSE framework consists of seven specific concepts. Together they ensure transparency, as well as render the process controllable and predictable in its essence. PSE is a proprietary framework developed by Intetics as result of 26+ years in custom development practices.

The work on creation of Predictive Software Engineering is just started and far away from completion! We invite other companies to join us in this work as Predictive Software Engineering is not about Intetics, it is about the industry and programming technology. Of course, a lot of companies are working on improvement of their processes. It would be great to join the forces and cooperate as this really is not about competition but about the profession.

The research was performed under the leadership of Boris Kontsevoi, President & CEO, and Sergei Kizyan, Delivery Director, Sandbox based on global engineering practices.

References

[1] A Larsson, P Törlind, L Karlsson, A Mabogunje, L Leifer, T Larsson, B-O Elfström, “Distributed team innovation – a framework for distributed product development,” in ICED 03, Stockholm, pp. 1-10, 1997.

[2] S. W. Ambler, M. Lines. “Introduction to Disciplined Agile Delivery,” *Crosstalk Journal*, vol. 40, pages 7-11, 2013.

[3] United Nations Development Programme, “Human Development Report 2016,” UNDP, New York, USA, Rep. ISBN: 978-92-1-126413-5, 2016.

[4] B. Kontsevoi, S. Terekhov "TETRA™ Techniques to Assess and Manage the Software Technical Debt", *Advances in Science, Technology and Engineering Systems Journal*, vol. 6, no. 5, pp. 303-309, 2021. DOI: http://dx.doi.org/10.25046/aj060534.

[5] B. Kontsevoi, “Predictive software engineering,” Global Engineering Community, unpublished.