5 Stages of Software Development Maturity
Introduction

Today, every company is investing heavily in software. Whether you’re a bank, a healthcare provider, or a car manufacturer, technology has quietly become your most important asset. For example, Goldman Sachs now has more software developers than traders. Tesla’s market cap exceeds that of Ford or General Motors. And even 145-year-old manufacturing companies like Continental Tire have built up huge software organizations with thousands of developers. Traditional corporations have no choice but to undergo major software transformations. Ten years ago, their IT teams were focused on running tactical systems like SAP, while today this shift to software fuels their business and requires significant cultural change.

Needless to say, in this environment, there is tremendous pressure to attract technical talent and accelerate the pace of innovation. CIOs and engineering leaders are all trying improve efficiency, reliability, and velocity. Unlike long-established functions such as sales or manufacturing, which have proven systems for measuring success, most technology executives have to rely heavily on layers of trust.

Why is that? It is because software is the most complex thing humankind has ever built. The typical software development organization is managing legacy and new systems written in multiple flavors of programing languages. Their portfolio might include web sites, mobile applications, cloud integrations, strategic custom software, billing systems, third-party software, and more. What rests inside these applications is unknown amounts of complexity and intricate detail. Compounding this challenge is the fact that each developer and development team have different skillsets and use various methodologies.

Forward-looking businesses know that they need to move beyond bugs, releases, and PowerPoint slides as their primary tools for managing the development process. They require real-time, fact-driven data to guide their decisions and measure performance. In exploring the current state of software development, we’ve spoken with a wide range of experts and thought leaders about how to drive operational excellence. Through these discussions, we identified four key steps towards world-class software development, which we’ll explore in the chapters that follow.

5 Stages of Software Development Success

<table>
<thead>
<tr>
<th>Status Quo</th>
<th>Transparency</th>
<th>Efficiency</th>
<th>Reliability</th>
<th>Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Escalation-driven investments</td>
<td>Fact-based decisions</td>
<td>Smart improvements</td>
<td>Real-time steering</td>
<td>Faster time-to-market</td>
</tr>
</tbody>
</table>

- **Internal and external development teams managed reactively**
- **Legacy and new systems written in multiple flavors of programing languages**
- **Unknown amounts of complexity across applications and teams**
- **Clear comparison of risks, efforts and opportunities**
- **Fast assessment and decisions on investments**
- **Automated continuous reporting on applications, projects and initiatives**
- **Quick wins through focus on development hotspots**
- **Proactive notifications on potential risks**
- **Better, more accurate, project timeline and cost estimates**
- **KPI monitoring throughout development**
- **Early warnings to resolve issues before the fact**
- **Performance assessments and comparisons to set goals**
- **Holistic view on development process to remove inefficiencies**
- **Continuous benchmarking of software development**
- **Ability to track change initiatives**
The large majority of companies have only an opaque view into their application portfolio and development efforts. Development managers typically provide updates to middle managers, and eventually up to the executive team, but the challenge with this process is that it is primarily people-driven. Since software development is inherently complex, it’s extremely difficult to get a complete and consistent view across business units, teams, service providers, projects, and applications. And when the people you trust leave, things might fall apart.

When each semi-autonomous team or business unit uses its own KPIs, it’s impossible to compare results across the organization. And so, the further you are from the code, the less oversight you have. Rather than proactively steering the organization, many CIOs have no choice but to manage through escalations. The status quo in most organizations tends to be reactive software management when things go wrong. As a result, it difficult for the management team to identify risks to the business ahead of time, or spot otherwise obvious opportunities for growth.

That said, there are some advantages for developers in today’s IT world. Engineering teams can be highly flexible and onboard new developers quickly when they have freedom to develop ad hoc, using their preferred processes. Many groups track specific team-level or developer-level metrics to monitor their progress. For example, you might measure things like the number of bugs, test coverage, release milestones, or system performance.

While empowering teams and giving them some amount of autonomy is an important factor in talent retention, having a set of holistic software metrics to guide the company is equally critical. Thankfully, you don’t have to choose between developer-level or company-level metrics. And you don’t have to change how developers work or the development tools they’re comfortable with. By analyzing the data you already have and looking at footprints in the code, you can distill actionable KPIs that are applicable to developers, mid-level managers, and even the c-suite.

To do this, the first step is connecting your various systems of record to a system of intelligence. This will help you move from the status quo to transparency, the second stage of software development success.

“No one in the boardroom wants to feel like they’re flying blind. But that’s too often what’s happening when I’m sitting next to our CFO or CRO. Unlike them, I don’t have powerful metrics to show when giving an update on our double to triple digit million dollar software investment. In software, there aren’t simple KPIs like those used by finance, HR, sales or marketing.”
In this era where business success depends on software, a company’s value is increasingly tied to its codebase and engineering capacity. Today software and people make up 70% of IT spend – i.e. millions or billions in budget – but executives cannot accurately answer how this cost is distributed across the application portfolio they manage. It is a big challenge to measure fluctuations in development capacity, productivity, and focus. Companies need a macro picture with the ability to drill down into specific areas that require attention.

Because it is difficult to manage to more than two or three metrics, it is best if your KPIs work across applications, projects, teams, and service providers. They should be metrics that make sense at the executive level, but are also intuitive for people who are touching code, so the whole organization is aligned. To get to these elusive, yet extremely powerful, metrics, you’ll need a system of intelligence that can gather data across version control systems, code checkers, issue tracking systems, and test coverage suites.

Once you’ve started defining processes and key insights are flowing from various applications into your central repository, its job is to sort the “fruit,” so you can make apples to apples comparisons -- across applications, programing languages, and development teams.

In terms of specific KPIs to monitor, you can look at things through a fairly simple formula, like:

\[ \text{Software Innovation} = \text{# of Developers} \times \text{Avg. Productivity} \]
Development Capacity

The first component of the above equation is about understanding your development capacity. Today you might measure this through a headcount plan, or a contract you have in place with a service provider.

This is a good starting point, but in many ways it only gives you a rough swag at measurement. It doesn’t account for slips in your hiring plan, time for ramping developers or their vacations, team churn, bug fixing distractions, or shifts in priorities.

To get a real-time view, it is best to look for the footprints hidden in your code and monitor development capacity based on the number of person days spent touching the code.

This metric works at a macro level, while you can also drill down into specific business units, geographies, projects, applications, teams, and service providers to understand fluctuations over time. As an executive, you can ensure you’re investing in the company’s strategic priorities, and as a development team or a project manager, you could use this insight to advocate for resources.

Overall development capacity has been relatively flat, yet resources have shifted from Business Unit #1 to Business Unit #2.
Average Productivity

The second part of the equation is about understanding developer productivity. To be clear, there are many variables that go into productivity, and many would argue that measuring and managing it is as much an art as it is a science. That said, a single metric that is directionally accurate can be a source of truth for all kinds of things -- e.g. talent development, vendor performance, training effectiveness, impact of team churn, documentation completeness, development methodologies, and developer tool usage.

One approach to calculating productivity is to measure new logic added divided by the number of person days spent touching your code.

It is important to note that while this is a very good approximation, there are edge cases you’ll want to check for. Here are some tips:

- The more macro view you take, the more anomalies will wash out.
- When comparing projects or teams, it is best to compare those with similar attributes.
- It’s important to be aware of the other factors that influence productivity. For example, productivity might increase, but quality slips. Or conversely, a developer might appear less productive because they are reducing business logic to refactor complex code.

Another approach to calculating productivity is to use an algorithm that incorporates what your team produced, the factors impacting their productivity, and how much technical debt was left behind.

After working with many progressive CIOs, and continuing to test and tune this KPI approach, we’ve come up with a Seerene Score that evaluates these elements to compute a simple productivity metric.

By looking at impediments, rework, technical debt, and productivity side-by-side, you’ll begin to see the impact of these different dials. In the following chapters, we’ll explore many of these efficiency drivers in more depth.

What goes into the the Seerene Score?

**Developer Activity**

[+] McCabe
[+] File Changes
[+] Revisions
[+] RLOC

**Adjusted for time sinks**

[x] Time spend in complex and undocumented code

**Deductions for code quality**

[-] Complex or undocumented code added to the codebase
Visualizing your codebase

When you have transparency into your software, you can use software maps to get a solid understanding of your technology landscape. These metrics can be viewed at the business unit, project, application, team, or provider level. For a full list of KPI definitions visit the resources tab on Seerene.com.

<table>
<thead>
<tr>
<th>Size</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Code Size</strong></td>
<td><strong>Branch Coverage</strong></td>
</tr>
<tr>
<td>[Lines of code]</td>
<td>[Number of branches]</td>
</tr>
<tr>
<td><strong>Logic Size</strong></td>
<td><strong>Coding Violations</strong></td>
</tr>
<tr>
<td>[Logic points]</td>
<td>[Number of violations]</td>
</tr>
</tbody>
</table>

**Investment**

- **Resources** [Number of developers]
- **Effort** [Person weeks]
- **Effort Defect Fixing** [Person weeks]
- **Code Changes** [Number of changes]

**Risk**

- **Complexity** [Lines of code]
- **Complexity Density** [Percent of lines of code]
- **Deactivated Code** [Lines of code]
- **Missing Documentation** [Logic points]
- **% Undocumented Logic** [Percent of logic points]
- **Knowledge Monopoly** [Lines of code]
- **Line Coverage** [Lines of code]
- **Team Churn** [Percent of number of developers]
Why do developers love Seerene?

While somedeveloper might be reluctant to the idea of creating more transparency, there are lots of benefits to fostering an open engineering culture. In the technology community, it’s common for A+ talent to benchmark their work against others – if only for bragging rights. That’s why software leaders who focus too much on specific metrics often find that developers end up gaming the system.

But the gamification of work can be a great thing. We all learn from having insight into what we do well, where we fall short, and how we can grow our impact on the business. Anyone who takes pride in producing quality output likes to work in an organization where performance can be quantified, recognized, and celebrated.

Developers, in particular, want to know they’re working in a codebase that’s well-documented and free (or nearly free) of technical debt. Every developer hates old, bad legacy code, so it is important for companies to value improvements in code equally to new ‘output.’ It helps a lot to know that management will notice when you put the effort in to refactor poor code. The higher quality your codebase, the more developers will enjoy working in it, because they all want to be fast and efficient.

When a company is transparent about the KPIs they use to measure quality, efficiency, and success, developers can chart their progress over time, create more accurate estimates for future projects, see where they need to improve, and identify best practices to adopt. Development managers can compare team-level metrics for effort, agility, and quality against other groups within or even outside the organization. As a result, it is easier to make a case for more resources when needed, spot and address problem areas, as well as propagate best practices across the company.

An important consideration when introducing transparency is that you’ll need to decide who sees what. What may be interesting to one audience is often just noise to another, so it’s helpful to filter the results appropriately for each team or initiative. Our recommendation is to start disseminating the insights through bi-weekly digests of your top 5 metrics for targeted Slack channels or email distribution lists, which will ensure that the right teams are seeing the right view into your data at a regular cadence. Of course, you can eventually give stakeholders access to pull the information they want and dig into reports, but at first, sharing more targeted insights is a great way to build adoption.
Chapter 3

Efficiency

Once you’ve gained transparency into your technology organization, a natural question for any software leader is **how can we deliver more, faster, within our current capacity constraints**. There are some great models out there that can be useful as you work to define clear processes and drive more efficiency, such as the Capability Maturity Model Integration (CMMI) developed at Carnegie Mellon University or Gartner’s Innovation Management Model.

As you start to spread these types of smart improvements across your entire IT portfolio, key metrics and dials that will help you improve developer productivity include:

**Time Spent Working in Complex Code:**
In order to understand the impact of complex code on your development organization, you need to look for real lines of code that appear in a nesting level 4 or deeper. **Deeply nested code is hard to understand**, since it is usually not easy to follow the logic paths and tell under which conditions that code is executed. This results in heavily reduced developer productivity, and a high risk of new bugs being introduced. Additionally, the number of control paths to be tested grows exponentially with nesting levels, making your code harder to QA. One key to unlocking productivity is to identify where your developers are spending time in complex code, and prioritize refactoring efforts accordingly.

**Time Spent Working in Undocumented Code:**
This KPI counts the number of McCabe-relevant keywords in your code that are not prepended by a comment. These keywords indicate potential branches in the control flow – another measure for complexity. This insight can help you prioritize your documentation efforts, which can also have a big impact on productivity, since **uncommented complexity makes code even more difficult to understand**. (Some exceptions to this rule are comments at the end of a line, or keywords that occur at the beginning of a function.)

**Bug Fixing Ratio:**
This metric looks at the effort spent creating new lines of business logic compared to the effort spent addressing defects, which detracts from developer focus. **Not only does this type of rework force a developer to frequently regain context, it often touches multiple people** and tends to slow big enterprises down.

Large company CIOs often ask “Why does it feel like we are not fast enough in cranking out new code, even though we have thousands of developers working at it?” The most common issue here is that when a majority of time is spent fixing old stuff, your time for innovation (writing new code) is too limited. This is a key contributor to the difference in speed between established enterprises and nimble startups. Small improvements to your bug fixing ratio can pay big dividends when it comes to overall developer productivity and innovation.
Efficiency Metrics

Team Churn:
Development teams are typically most productive when there is a high level of continuity. A churn metric can be very helpful to monitor, because when team members are rotating on and off a project, or splitting time between several projects, there are often wasted calories due to context-switching and more time spent getting up to speed. This is also very important when it comes to getting better visibility into your outsourced teams. Today, few software leaders know specifically which individuals from third-party vendors are touching their code. It can make a big difference to the relationship and your SLA when you know, “Who are the key personnel at the outsourcer that know our code and need to be on this project?”

As you dig deeper into your organization’s efficiency potential across each of these metrics, this information will help with discovery, root-case analysis, weighting investment decisions, and monitoring specific initiatives. Solutions like Seerene offer KPIs and visual software maps that identify low hanging fruits, and help you move quickly to unlock developer productivity.
Let’s explore a few specific use cases that can take your efficiency to the next level:

**Legacy Modernization**

Every company has legacy applications, some more than others. These are systems that have been extended and modified over the years. Most companies are constantly assessing whether it makes sense to re-write parts of their legacy applications or to port apps to a new platform.

From an efficiency perspective, it is important for leaders to **evaluate these opportunities based on fact-based information about development capacity, productivity, and time spent working in complex and undocumented code**. By looking at these variables, along with others (bugs, system performance, costs), to accurately see where complexity exists in an application, you’ll be able to understand if there are simple improvements that your team can make, or if an application is truly a drain on productivity and needs to be re-written or replaced. With Seerene’s clear KPIs, it is much easier to make that case and make an informed decision.

Once you get going, you’ll of course need to monitor these projects and show wins that propel the business forward. You can use real-time dashboards to track your progress with a specific app or across your entire portfolio – using KPIs like the hotspots, rework, technical debt, and developer productivity. With this level of insight for better, more transparent project management, you’ll be able to **focus your efforts, more quickly minimize the risk that legacy applications present, and reduce the likelihood of a failed modernization project**.

**Application Rationalization**

A related, but different, challenge comes when you have two applications that do roughly the same thing. This is often the case for businesses that have grown through mergers and acquisitions. There’s an opportunity to consolidate overlapping apps in your stack, but first you need analytics to help determine which ones to retire, and which to invest in. Through a better understanding of the underlying technology, complexity, code quality, developer skills required, and availability of application experts, you can make better-informed, objective decisions about which applications to keep around.

**Vendor Management**

Companies that leverage outsourced developers and service providers can use these same core metrics to **evaluate vendors and improve cost:return ratios**. One approach is to identify your top providers, and ask “Why have those engagements been successful?” With accurate metrics, you can ensure that your providers stay on track and meet agreed upon goals.

You’ll spot opportunities to prune away low performers and bring select applications back in house when appropriate. You can stack rank vendors based on productivity, quality, and timeliness, and perform comparisons of teams to see where you’re getting the best value. In addition, by having better baseline information about the applications you’re giving to a vendor or business process outsourcer to manage, you will be able to negotiate these contracts and stay on top of the work product delivered.
Another crucial dimension of best-in-class software development is quality. No matter how much code a highly efficient team writes, they won’t reach the top levels of the software success ladder if it’s not good code. Once you’ve mastered holistic transparency and efficiency, it’s time to take a look at how your organization measures code quality. With a 360-degree view of your entire portfolio’s codebase in a tool like Seerene, you can establish threshold values and automate notifications to ensure that your processes are quantitatively managed and continuously improved. You'll know whether your developers are meeting acceptable quality thresholds or creating technical debt that you’ll need to address down the road.

As a result, teams can get early warnings to head off critical problems and resolve potential issues proactively throughout the entire development process. In order make the most of this insight, it’s crucial to assess the organization’s progress towards KPIs over time. For example, you might want to answer questions like, “Are we appropriately using the 10% of our time that’s been allocated to refactoring code so we can remove unneeded complexity and document knowledge?”

A key use case for these metrics is the real-time steering of both internal and outsourced development teams. Several stakeholders across a software organization – from managers to the c-suite – can benefit from more proactively directing teams and identifying low performers that consistently deliver poor code quality. In order to do this effectively, you’ll need personalized, permissions-based dashboards with data that can be sliced and diced for helpful comparisons across projects, teams, or vendors. They should feature relevant KPIs to help align the organization, including several that we’ve identified in previous chapters like effort, productivity, bug fixing ratio, technical debt, complexity, and documentation.
Chapter 5

Velocity

At this stage, a software team has most of its ducks in a row and can focus on development optimizations that further **accelerate reaction time to new requirements**. Rather than waiting weeks to execute, teams are more flexible and agile in supporting ever-evolving business needs. Every technology organization should strive to reach this “nirvana” step in the success ladder, where they can more effectively drive change programs such as zero waste, zero defects, or even the migration from waterfall to agile development. To execute on these transformations, technology leaders need accurate, timely KPIs that track whether their initiatives are successfully speeding the pace of innovation.

For instance, by reviewing project retrospectives when a project is completed or the quarter closes, you can see how you’ve done across the full range of development success measures and continue to tweak your approach based on lessons learned at the efficiency and reliability stages. Find out how code quality measured up to the goals you set, calculate how much time and money went into delivering a new application, compare the average developer’s productivity against similar initiatives at your company, or benchmark your average development cycle time.

Organizations that reach the top level of software development maturity find that they’ve **changed the entire conversation inside IT, and with partners across the business**. They’re innovating with optimum efficiency and reliability, and are aligning around the right strategic priorities because everyone shares a common vocabulary and a single source of truth.

**Velocity metrics**

**Speed** [Logic points per person week]

**Planned vs. Unplanned Requirements**

**Requirement complexity**

**Defect Reaction Time**

**Code Test Coverage**

**Milestone Adherence**
What do CIOs and other executives want to know?

All of the development metrics we’ve explored up to this point are interrelated, as depicted here:
Final Recommendations

Code+people analytics from Seerene can play an important role in helping you drive software development success, because they tell the story of where you are—and where you are going—in a whole new way. You simply can’t keep dozens and dozens projects running smoothly and producing results unless you have the ability to view your entire application portfolio on one dashboard with fact-based KPIs about your code and your engineering resources. With access to holistic, consistent metrics across all projects, teams, and applications, you can prioritize and make the case for investments with ease, standardize around best practices, and get early trouble warnings.

In summary, proactively managing your technology investment mix requires some key insights across your portfolio. At a high level, these should inform strategies such as:

- **Watching where your money and development calories are going** and making adjustments regularly to align with strategic priorities
- **Setting your organization’s software development “dials”** by translating strategy into measurable targets that teams can report on and monitor over time
- **Finding opportunities to drive faster results** (e.g. by addressing productivity drains and your biggest risks), and reacting to business requirements with agility

As you take a more methodical, forward-looking (vs. reactive, exception-focused) approach to managing software, you’ll reap huge rewards. By comparing KPIs across people and initiatives, you and your teams will be able to make faster, better-informed decisions and even share progress at the board level. Once you have a system where every layer of the organization – from development, to dev ops, to operations and executives – is aligned around a consistent set of metrics (yet metrics that make sense for each group), you’ll rapidly climb the ladder towards becoming a highly efficient, productive, reliable, and innovative software organization.

About Seerene

Seerene provides a platform to manage code+people, helping CIOs and executives measure the value and success of their software development efforts. Just as companies have financial and sales metrics, Seerene provides metrics about the health of the codebase and engineering capacity at an organization. This insight helps many top global and Fortune 500 companies manage their risk, improve cost: return ratios, accelerate time-to-market, and run faster than the competition. The technology is grounded in 10 years of academic research at the Hasso Plattner Institute, and was spun out in 2015 as a stand alone company that now has offices in Potsdam, Berlin, London, New York, San Francisco, and Hong Kong.
Appendix

Seerene’s Software Success Ladder Survey

We reached out to a wide range of customers, experts, and thought leaders about the current state of software and asked them about how they drive operational excellence. Thank you for all the amazing insight that’s been shared so far – many of your ideas are included within this eBook. For the full list of questions or to participate in our ongoing survey, you can visit https://www.getfeedback.com/r/RN2k3N61.