Software Development Performance Index (SDPI) measurement specifications

Summary

The Software Development Performance Index framework codifies a balanced set of outcome measures that, when used within Rally® Unlimited Edition, can give you feedback on your own teams and organization. This document explains the SDPI and how these metrics are calculated. To learn more, please visit www.rallydev.com.

Time buckets

Each metric is calculated for a particular time bucket. The summary SDPI charts are most commonly shown in quarters. The drill down charts are most commonly shown in months.

Real teams from projects

The "Project" entity in Rally is the team container but its hierarchical nature means that some "Projects" represent other organizational entities (meta-teams, divisions, departments, etc.). Some may even represent projects. To determine which "Project" entities are actually teams, we use a Bayesian classifier that looks at how much work is contained in the "Project", how close to the leaf of the hierarchy it sits, and a number of other characteristics.

Team size

We heuristically extract team membership by looking at who is working on what items and who is the owner of those work items. We then determine what fraction of the time each person is working on each team. The team size is the sum of these fractions.

Percentile scoring

The units for each raw metric are different. For some metrics higher is better whereas lower is better for others. To make it easier to interpret the metric and enable the aggregation of dissimilar units into a single index, raw metrics are converted into a percentile score across the entire distribution of all similar metrics. Higher is always better for percentiles.

Calculating the index

The SDPI is made up of several dimensions. Each raw metric is percentile scored and one or

more of those are averaged to make up a particular dimension (e.g. Quality dimension is the percentile score of defect density for defects found in production averaged with the percentile score of defect density for defects found in test). To calculate the overall SDPI, we take the average of the contributing dimensions' scores. If there are four dimensions, then the max contribution of any one will be 25 to this final SDPI score.

Responsiveness score from Time in Process (TiP)

Time in Process (TiP) is the amount of time (in fractional days) that a work item spends in a particular "state". Weekends, holidays, non-work hours are not counted. We take the median TiP of all the work items that completed in a particular time bucket (say January, 2013) and record that as the TiP for that time bucket. While other parameters are possible, we primarily look at the TiP of User Stories and we define "in Process" as ScheduleState equals “In-Progress” or “Completed”.

Quality score from Defect Density

Defect density is the count of defects divided by man days, where man days is team size times the number of workdays in that time bucket. This results in a metric that represents the number of defects per team member per workday.

We look at both the defects found in production as well as those found in test and other areas as indicated by the "Environment" field in Rally. We sense whether or not defects are typically being recorded in Rally for each of these types for each team over a time period and only use it if it passes this test. We'll take either as the Quality Score or the average of the two if both are reliably recorded.

Productivity score from Throughput / team size

Throughput is simply the count of User Stories, Defects, and Features completed in a given time period. The productivity score is the percentile scoring of this Throughput normalized by the team size. While Defects and Features are shown in the drill down charts, currently only User Stories contribute to the Productivity Score of built in scorecards.

Predictability score from Throughput variability

Throughput variability is the standard deviation of Throughput for a given team over 3 monthly periods divided by the average of the Throughput for those same 3 months. This is referred to as the Coefficient of Variation (CoV) of Throughput. Again, we only look at User Stories for this Predictability Score.

Decision versus outcome measurements

The measurements below are generally targeted at characterizing a decision or an outcome. An organization either decides to split people across many projects or they dedicate them to one. The Percent Dedicated Work measurement extracts this decision. Defect Density is an example of an outcome measurement.

Although not strictly accurate, they can be thought of as input and output variables in a correlation analysis.

Scores

Raw outcome measures are translated into a "score" so they can be easily interpreted as indicators of performance. Measures closer to 100 are good, measures closer to 0 are bad. The raw measure and the score are both available for analysis.

Time box granularity

Unless otherwise specified, each metric specified below is calculated for each of the following time boxes:

- Month
- Quarter (Calendar)
- 3-month (sliding)
- 6-month (sliding)
- 12-month (sliding)
- Iteration (not yet as-of 2013-09-01)

The sliding window measurements are useful when trying to identify a correlation where the impact of a decision measurement for a given month might correlate with the outcome measurement over the course of several following months. For instance, field-reported defects will trickle in over time. So, logically, we would expect a change in this measurement to be evident for several months after the impacting decision. The empirical evidence supports this trailing effect because bad decision metrics (non-dedicated-ness) correlate best with the 6-month trailing defect density metric.

Snapshots and the temporal data model

We do not directly measure things like Percent Dedicated Work. It and the other measurements specified in this document are built from snapshots of changes representing transactions of users working with artifacts in their project management, source code management, build, or bug tracking systems. A detailed discussion of this data model including its data structures, constraints, and operations can be found here. Many of the details of calculating these metrics cannot be understood without at least a basic understanding of this underlying snapshot data structure and temporal data model.

"Real" teams

In addition to being associated with a time box, every measurement in the data set is also associated with a team. Our data set does not have a strict definition of a team. Rather, it includes the concept of a team/project hierarchy, where higher level entries might represent divisions or teams of teams and lower level entries represent the team itself. It is also fairly common for a team to break their work down into project streams. This is a typical team/project tree:

- Division ABC
  - Meta-team I
    - Team A
      - Team A - project 1
      - Team A - project 2
    - Team B
  - Meta-team II
    - Team C
    - Team D
- Division XYZ
  - ...

Since the data is non-attributable and huge (25,000 projects) we have no way of asking which entries in this tree represents a "real" team. So, we heuristically extract this using a Bayesian classifier. The features that the classifier keys off of include:

- The number of levels from the leaf nodes of the current branch of the project tree. "Real teams" tend to be at the leaf nodes, which is 0, or one level up, which is 1.
- The number of work items in-progress in the node.
- The full-time equivalent value for the node. “Real teams” tend to have between 5 and 8 members, and outside of this range, the probability of being a “real team” decreases.

Measurements

Percent Dedicated Work

This measurement indicates how much of the work for a given team is done by folks "dedicated" to that team.

Type: decision

Formula

1. Find all transactions (snapshots) for stories, defects, and tasks that are in progress, have no children, have an owner (user), and are not blocked.

2. Sum all transactions by user U_{total}, project P_{total}, and user contribution to a project $U_{project}$ where $U_{total} > 5$ (i.e. user's with a total trans action count less than or equal to 5 are not counted towards $U_{project}$ or P_{total}).
3. Find the percent of a user's total work each project represents: $U_{percent} = \frac{U_{project}}{U_{total}} \cdot 100$
4. Count as "dedicated" for a given project, the users whose $U_{percent}$ is greater than 70% for that project. This threshold was determined by experimentation with a training set of data from teams with known "dedicated" members.
5. For each project sum the dedicated user transactions: $P_{dedicated} = \Sigma\ U_{project}$, for all dedicated members.
6. Find the percent of dedicated work for each project: $P_{percent\ dedicated} = \frac{P_{dedicated}}{P_{total}} \cdot 100$

Data cleaning

The transactions of any user with 5 of less transactions in a given timebox/project pair are ignored when calculating $U_{project}$ or P_{total}. This removed a lot of noise associated with folks who are not true team members (managers, admins, etc.).

Full-Time Equivalent

This measurement is an indicator of team size including contributions from part-time contributors to the team.

Type: decision

Formula

1. Find all transactions (snapshots) for stories, defects, and tasks that are in progress, have no children, have an owner (user), and are not blocked.
2. Sum all transactions by user U_{total}, project P_{total}, and user contribution to a project $U_{project}$ where $U_{total} > 5$ (i.e. user's with a total transaction count less than or equal to 5 are not counted towards $U_{project}$ or P_{total}).
3. Find the fraction of a user's total work each project represents: $U_{fte} = \frac{U_{project}}{U_{total}}$
4. Sum the full-time equivalent for each project: $P_{fte} = \Sigma\ U_{fte}$.

Team Stability

This is an indication of the team's stability. For example, given:

Month n:
George: 90% dedicated
Joe: 50%
Jen: 80%

Month n + 1:
George: 75% (-15% delta)
Jen: 100% (+20%)
Jeff: 25% (new) (+25%)
Joe: missing (-50%)

The TeamGrowth metric for the team would be .2 + .25 = .45 divided by the current team size (2) or 22.5%.

The TeamShrinkage metric for the team would be |-.15| + |-.5| = .65 divided by the old team size (2.2) or 29.54%.

The total volatility would be the sum of the two prior metrics or roughly 52% and Team Stability would be 100 - 52/2 = 74

Type: decision

Formula

1. Find all transactions (snapshots) for stories, defects, and tasks that are in progress, have no children, have an owner (user), and are not blocked.
2. Sum all transactions by user U_{total}, project P_{total}, and user contribution to a project $U_{project}$ where $U_{total} > 5$ (i.e. user's with a total transaction count less than or equal to 5 are not counted towards $U_{project}$ or P_{total}).
3. Find the fraction of a user's total work each project represents for all time periods: $U_{fte} = \frac{U_{project}}{U_{total}}$
4. Sum the full-time equivalent for each project for all time periods: $P_{fte} = \Sigma\, U_{fte}$.
5. For each project and each pair of adjacent time periods (t and $t-1$) compute:
    a. Team growth by $P_{growth} = \frac{\Sigma\, max(0,\, U_{fte,t} - U_{fte,t-1})}{P_{fte,t}} \cdot 100$
    b. Team shrinkage by $P_{shrinkage} = \frac{\Sigma\, max(0,\, U_{fte,t-1} - U_{fte,t})}{P_{fte,t-1}} \cdot 100$
    c. Team stability by $P_{stability} = 100 - \frac{(P_{growth} + P_{shrinkage})}{2}$

Process Type

This measurement is an indicator of what flavor of agile process a team is using.

Type: decision

Formula

1. Find all snapshots for stories whose ScheduleState >= "In-Progress" and have no children.

2. Sum the total number of unique stories S_{total} for each project in each time period.
3. Sum the total number of unique stories that have a non-null field S_{field} for each project in each time period where $field$ is each of c_KanbanState, Iteration, TaskActualTotal, TaskRemainingTotal, TaskEstimateTotal, and PlanEstimate.
4. For each project in each time period, divide the sum for each field by the total number of unique stories and multiply by 100 to get the percent of stories with the field: $P_{field} = \frac{S_{field}}{S_{total}} \cdot 100$
5. After calculating the percent of stories with each field, the project is assigned a value for process type $T_{process}$ as specified in the following table:

$T_{process}$	$if...$
Kanban,ScrumBan	$P_{kanbanState} \geq 90 \ \wedge P_{iterations} \geq 90$
Kanban,No Iterations	$P_{kanbanState} \geq 90 \wedge P_{iterations} < 90$
Iterative,Scrum,Full	$P_{kanbanState} < 90 \wedge P_{iterations} \geq 90 \ \wedge$ $P_{planEstimate} \geq 50 \wedge P_{taskEstimateTotal} \geq 50$
Iterative,Scrum,Story points only	$P_{kanbanState} < 90 \wedge P_{iterations} \geq 90 \ \wedge$ $P_{planEstimate} \geq 50 \wedge P_{taskEstimateTotal} < 50$
Iterative,Scrum,Tasks only	$P_{kanbanState} < 90 \wedge P_{iterations} \geq 90 \ \wedge$ $P_{planEstimate} < 50 \wedge P_{taskEstimateTotal} \geq 50$
Iterative,Other	$P_{kanbanState} < 90 \wedge P_{iterations} \geq 90 \ \wedge$ $P_{planEstimate} < 50 \wedge P_{taskEstimateTotal} < 50$
Other,Estimates	$P_{kanbanState} < 90 \wedge P_{iterations} < 90 \ \wedge$ $(P_{planEstimate} \geq 50 \vee P_{taskEstimateTotal} \geq 50)$
Other,No estimates	$P_{kanbanState} < 90 \wedge P_{iterations} < 90 \ \wedge$ $P_{planEstimate} < 50 \wedge P_{taskEstimateTotal} < 50$

Time in Process (TiP) and Responsiveness score

Time in process (TiP) is a measure for an individual work item (story, defect, feature) indicating how much work-day time (excluding non-work hours, weekends, and holidays) it spent "in process". For stories and defects, "in process" is defined by the ScheduleState field being either "In-progress" or "Completed" (often means "In-test"). For features, "in process" is when ActualStartDate is set and PercentDoneByStoryCount is less than 100%. Although not calculated exactly the same, it is analogous to the common definition of cycle-time or lead-time. For a given project/time-box pair, an aggregation (median, a.k.a. p50) of the TiP of the work items that completed during that time box for that project is

computed. The responsiveness score is based on the percentile of the median. Higher values will result in lower scores, and vice versa.

The median or p50 is used rather than the arithmetic mean as the aggregation because the distribution of TiP measurements for individual work items is far from normal and frequently includes outliers. Median deals well with the non-normal distribution and does not allow a single outlier to greatly impact the measurement like an arithmetic mean would. The data set also includes p75, p85, p95, p99 representing the 75th, 85th, 95th, and 99th percentile coverage levels for the set of completed work items but we currently only use the p50 (median) to calculate the score.

Type: outcome

Variations:

- Stories, Defects, and Features

Formula

1. Find all Stories, Defects, and Features that were in progress, then moved to completed within the time frame under consideration.
    a. Stories and Defects are considered completed when $ScheduleState \geq "Accepted"$.
    b. Features are considered completed when $PercentDoneByStoryCount \rightarrow 100\%$.
2. Calculate a TiP value for each of those Stories, Defects, and Features.
    a. Story and Defect TiP is the duration where $"In Progress" \leq ScheduleState < "Accepted"$.
    b. Feature TiP is the duration between $ActualStartDate$ and when $PercentDoneByStoryCount \rightarrow 100\%$.
3. The Responsiveness score is the percentile rank of the p50 TiP value for Stories.

Defect Density and Quality score

Defect density is merely the count of defects over some normalizing size measurement. In our case we use the team's man-days (FTE * the number of working days in the period) as a proxy for size.

Type: outcome

Variations:

- All defects ("Defect") or just defects found in production ("ReleasedDefect")

Formula

1. Count all defects D_{all} and defects released to production $D_{released}$ for each project.
2. Calculate defect density E for each project by:

$E_{all} = \frac{D_{all}}{P_{fte} \cdot W}$

$E_{released} = \frac{D_{released}}{P_{fte} \cdot W}$

where P_{fte} is the project's full-time equivalent and W is the number of working days in the time period under consideration.

3. For each project, determine if either defects or released defects are being tracked by checking if the defect count is greater than zero for the year granularity that ends at the same time as the granularity under consideration. So for example, if the granularity is a quarter ending on 2013-01-01, we check the full year ending on 2013-01-01 to see if the defect count for the year is non-zero.
4. Compute defects per 1000 man days by:
   $S_{all} = 1000 \cdot E_{all}$
   $S_{released} = 1000 \cdot E_{released}$
5. For each project where defect data is tracked, compute the quality score. Defect density is scored based on percentiles. If a project has the highest measured value for defect density, it is in the 99th percentile, therefore its score is $99 - 99 = 0$. If a project has the lowest measured value for defect density, it is in the 0th percentile, therefore its score is $99 - 0 = 99$.
   $Q_{all} = percentile(S_{all})$
   $Q_{released} = percentile(S_{released})$
6. The total quality score is the quality score for all defects: $Q_{total} = Q_{all}$. Projects not tracking defects will have no quality score.

Throughput and Productivity score

Throughput is a measure of how much work is completed in a given time period. Within a single team, throughput can be compared over time. However, the size of a work item can vary greatly by context so it's hard to compare this across teams. It can also be compared across teams when the size of a work item is controlled. For instance, some organizations will require that each story should be between 0.5 and 3 man days of work. We do not know this information however, so when calculating the "score" we simply look at number of completed stories normalized by the team size (FTE). Throughput per team member is scored based on percentiles. Higher values result in higher scores, and vice versa.

Type: outcome

Variations:

- Defects, Stories, or Features
- Counts or Story Points - The formula below describes the computation by counts of these items. However, we also compute "throughput" (or "velocity" if you prefer) for stories and defects using the sum of the story points of all work items that make the appropriate transition. We do not yet have a good mechanism to identify which teams consistently use story points so the counts are the preferred variation at this time. The

development of iteration-based measures is underway and includes research to explore better use of story points.

Formula

1. For each project, compute throughput T as the sum of
    a. the count of all stories and defects that transitioned forward into the accepted state minus the count of all stories that transitioned backwards out of the accepted state.
    b. the count of all features that transitioned forward to 100% complete by story count minus the count of all features that were 100% complete by story count but transitioned backward into < 100% complete by story count.
2. Compute the throughput per team member by dividing throughput by full-time equivalent: $T_{fte} = \frac{T}{P_{fte}}$
3. Score T_{fte} based on its percentile. If a project has the highest measured value for T_{fte}, it is in the 99th percentile, and 99 is its score. If a project has the lowest measured value for T_{fte}, it is in the 0th percentile, and 0 is its score.

Throughput Variation and Predictability score

Having a stable throughput can be as important as having a high throughput. The coefficient of variation of throughput across several time periods is calculated and translated into a score.

Formula

1. For each project, compute throughput for each month T_i as the count of all stories that transitioned forward into the accepted state minus the count of all stories that transitioned backwards out of the accepted state.
2. For each group of 3 and 6 adjacent months T, compute the:
    a. average $avg(T)$
    b. standard deviation $std(T)$
    c. coefficient of variation $CoV = \frac{std(T)}{avg(T)}$
3. Score CoV based on its percentile. If a project has the highest measured value for CoV, it is in the 99th percentile, therefore its score is $99 - 99 = 0$. If a project has the lowest measured value for CoV, it is in the 0th percentile, therefore its score is $99 - 0 = 99$.