Hewlett Packard Enterprise

Business white paper

Getting your ALM data ready

Predictive analytics for ALM

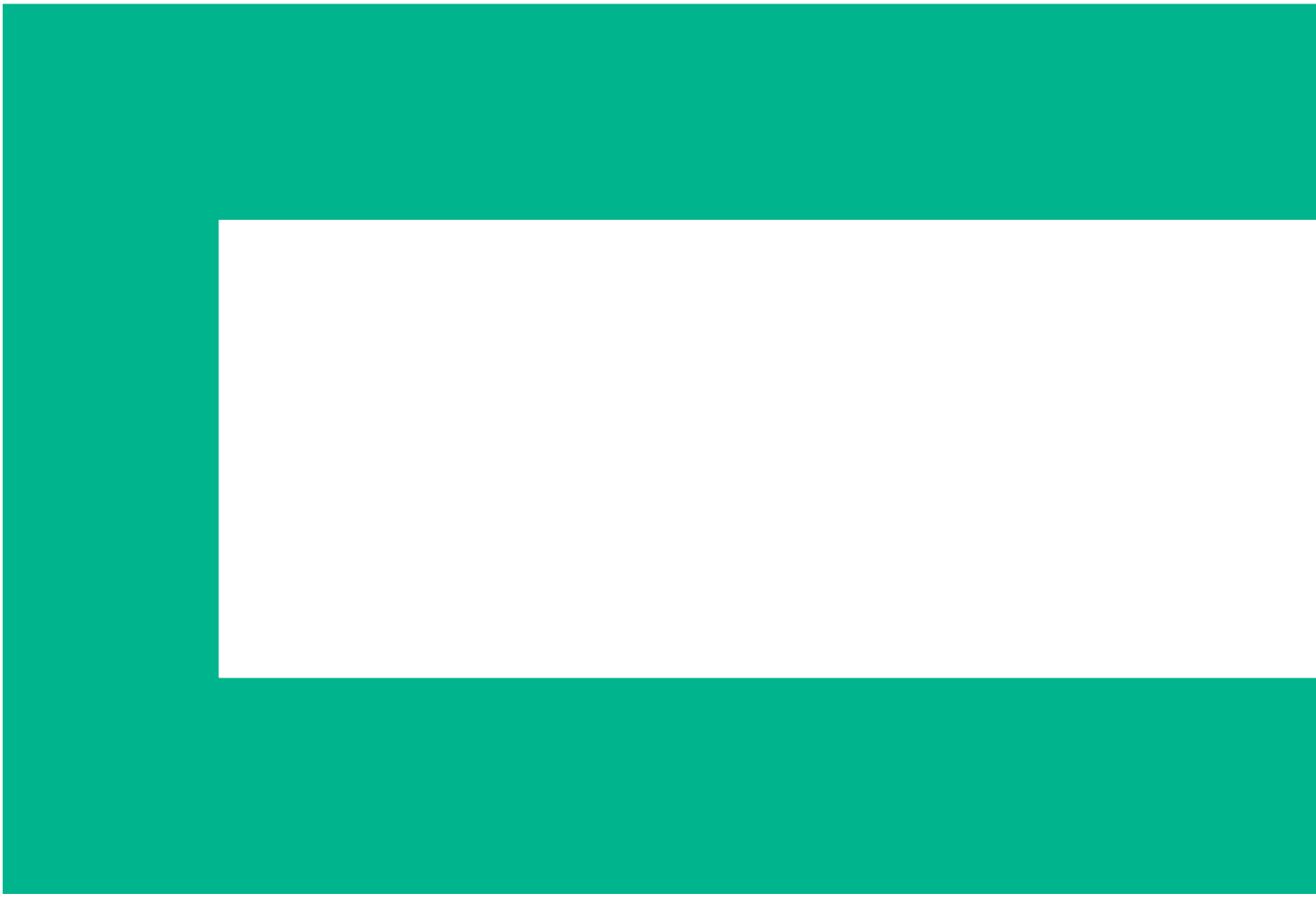

Table of contents

About the Author
Megan Sheehan, Ph.D.
megan.sheehan@hpe.com

Megan Sheehan is a senior product manager in HPE Software Application Delivery Management focused on predictive application lifecycle management (ALM). She has a Ph.D. in Communications Sciences with an emphasis on quantitative research (SEM, CFA, Multivariate Regression, Data Mining, Segmentation, Meta-Analysis, Conjoint Analysis, Interaction Analysis, Experimental Design, Time Series Analysis, and Survey Design). In addition, she has over 15 years of experience in product management, program management, technical product planning, product innovation, and market research on a range of enterprise software products.

Previously, Megan led efforts around the HP (now Hewlett Packard Enterprise) LES Enterprise Innovation research on hardware, software, and services. Before joining HP (now Hewlett Packard Enterprise), Megan was a senior product manager for Infragistics' data visualization and reporting tools. Megan also spent six years at Microsoft® focused on the developer audience while working on Windows® 7, Visual Studio, and Windows Embedded. Furthermore, she served as the owner of Habits & Practices Pillar as well as redesigned Microsoft's approach to capturing, quantifying, and prioritizing user needs and translating them into product features.

Introduction

You may have heard discussions about Big Data and how companies such as Netflix, Amazon, or American Express are using it to make business decisions. Admittedly, there has been a lot of hype around the topic. You may have thought about how it could apply to your business, in general, perhaps in areas such as customer satisfaction or marketing.

As HPE Quality Center (QC) and HPE Application Lifecycle Management customers, you may think that Big Data and advanced analytics are not applicable to you. Actually, QC and ALM are large data repositories and, for most organizations, represent an untapped gold mine of insight. While they are not capturing the same volume of data as highly trafficked sites such as Twitter or Facebook, QC and ALM contain large amounts of very minable data.

When you consider the metadata associated with every requirement, test, and defect across many projects, you quickly achieve a significant amount of data that is ripe for analysis. For most organizations, this is "dark data." Many people even think of it as overhead, i.e., archived projects that must be preserved for audit trail or compliance. Rather, we would like to propose a different perspective.

All of your organization's existing projects, including: requirements, tests, test runs, defects, state updates, code check-ins, and others, are some of your most valuable and unused resources. Until now, we have not heard much about applying advanced analytic techniques to ALM, quality assurance, and operations management. That is about to change. The purpose of this paper is to introduce some key data sciences concepts, explain how they relate to ALM, and offer recommendations for steps you can take now to prepare your data for analysis in the near future.

About the document

This document contains forward-looking statements regarding future operations, product development, and product capabilities. This information is subject to substantial uncertainties and is subject to change at any time without prior notification. Statements contained in this document concerning these matters only reflect Hewlett Packard Enterprise's predictions and expectations as of the date of this communication. Any actual results and future plans of Hewlett Packard Enterprise may differ significantly as a result of, among other things, changes in product strategy resulting from technological, internal corporate, market, and other changes. This is not a commitment to deliver any material, code, or functionality and should not be relied upon in making purchasing decisions.

Context

Organizations are being challenged to accelerate the pace of software delivery. According to IDC, software is becoming the critical component of sustainable competitive advantage for businesses, and IT organizations are under increasing pressure to deliver customer-aligned functionality faster, with increasing levels of quality and security.[1]

Teams are faced with the increased pressure to deliver high-quality software to support their businesses at a blistering pace. Therefore, it is imperative to reduce uncertainty and risk in projects, as well as optimize the outcomes as much as possible. Data sciences is an integral part of the solution.

[1] IDC MaturityScape: DevOps, June 2014

Data sciences

The excitement around data sciences has increased in recent years and while it may seem like magic, data sciences uses computational science approaches to turn data into knowledge and insights. Several types of data analysis are employed, especially machine learning techniques, recommender systems, anomaly detection, and time series analysis. The objective of machine learning is to program systems to learn from historical data and improve over time.[2] These approaches are being applied to problems in many different domains: medical research, robotics, environmental sciences, marketing, and others. With machine learning, the starting point is a theoretical model, in other words, how you expect things to work together, and then how the model is applied to observed data. This is refined as well as improved over time.

Data scientists are researchers who specialize in machine learning, statistics, and other data analysis techniques to answer key business questions. It is difficult to find data scientists who have both the required technical skills and business acumen.[3] Understanding your business makes any data analysis efforts more relevant. Whether your biggest challenge is streamlining production processes, improving product quality levels, retaining customers, improving operations efficiency, or optimizing software development processes, they are all very diverse problems. How you would go about addressing each of those issues and what data sources you would look at is very different. Focusing on the right questions is a critical first step.

Data analysis: Getting down to business

The biggest requirement for advanced analysis is access to existing data. All data science techniques use existing data to validate and refine the models. Before you can predict what is most likely to happen in the future, you have to understand what has previously happened. The strongest predictor of future behavior is past behavior.[4] If you have done something before like—run a six-minute mile, paid your credit card bills on time, closed a defect in three days, or used a specific development tool, you are more likely to do those things again in the future.

The correlations are strongest at points that are close in time, for example, what happened in the most recent past will be a better predictor of the near future than what happened five years ago. One of the most widely used applications for this principal is credit scoring. Credit scores are algorithms that assess an individual's financial risk. The length of credit history, payment patterns, and recency of opening credit accounts, especially in a short period—all factor into the credit score calculation.[5] Another example is the product suggestions that surface in your Amazon account. Through countless hours of analyzing sales data, Amazon's data scientists have developed highly effective algorithms to predict what consumers are likely to purchase.[6]

Predictive analytics and software lifecycle management

Many other industries have made use of Big Data and advanced analytics, but software development has not fully embraced it yet. Like other business challenges, software development, testing, deployment, and maintenance are complex processes where many people across teams and sometimes across geographies do their best to coordinate and deliver high-quality products.

Predictive ALM software from Hewlett Packard Enterprise unlocks the insight hidden in previous ALM projects. We have developed specialized algorithms that mine existing data and can be applied to current and future projects, in order to accelerate development, improve quality, and mitigate risk.

Predictive ALM software intends to deliver functionality over time that maps challenging customer use cases across the software development lifecycle. New algorithms are developed that address different needs, and at times, target different "personas" or users. In each phase, we help optimize customers' processes based on analysis of historical data.

[2] Carnegie Mellon University **ml.cmu.edu/**

[3] How to Take a First Step to Advanced Analytics, Gartner webinar, October 2015

[4] **userwww.sfsu.edu/efc/classes/biol710/timeseries/timeseries1.htm**

[5] **myfico.com/crediteducation/whatsinyourscore.aspx**

[6] **fortune.com/2012/07/30/amazons-recommendation-secret/**

The algorithms can identify anomalies that result in problems such as overestimating capacity, high-defect rates, missing the release timelines, or test coverage gaps. We have used a variety of machine-learning techniques and multivariate analysis to create algorithms, which generate predictions of the most likely outcomes and, more importantly, recommendations for avoiding problems, mitigating risks, or helping optimize deliverables. The specific prediction and recommendation are dependent on the algorithm.

The framework we are using divides the activities into four areas: predictive planning, predictive development, predictive testing, and predictive operations.

Predictive planning—addresses customer needs during the planning stage of a project. Target users include project managers, developer team leaders, test team leaders, business analysts, and architects. Possible uses cases include improved feature size estimates based on story points, identifying and correcting inaccurate estimates, improved requirements prioritization, and identifying under or overcapacity. The algorithms in this module contribute to more accurate planning process. Starting a project with the most accurate estimates can set it up for success.

Predictive development—addresses customer needs during the development stage of a software project and focuses on improved efficiency and accuracy. Target users include developers and testers. Possible uses cases include identifying code check-ins that will break the build before they are checked in, code-completion suggestions, analyzing source code for defects or complexity, and promoting code reuse by identifying existing code that provides desired functionality.

Predictive testing—addresses user needs during the testing phase of a project. The primary target users are developers and testers although IT operations managers may reap benefits from some of the capabilities too. Some possible use cases include predicting the rate at which defects are fixed, predicting the injection rate of defects, identifying defects that are likely to cause escalations in production, determining the root cause of a failed test, and more. They also include making recommendations about which tests should be run based on code changes and identifying existing tests that provide desired functionality and can be reused.

Predictive operations—addresses user needs during the deployment and production phase of a project. The primary target users are IT operations managers, but testers and developers benefit from some of the algorithms too. The information learned from analysis of production data are fed back into the planning, development, and testing algorithms. Possible use cases include identifying gaps between end-user actions and the workflows that tests are covering, reducing the likelihood of escaped defects, and linking customer defects with user requirements.

Predictive ALM framework

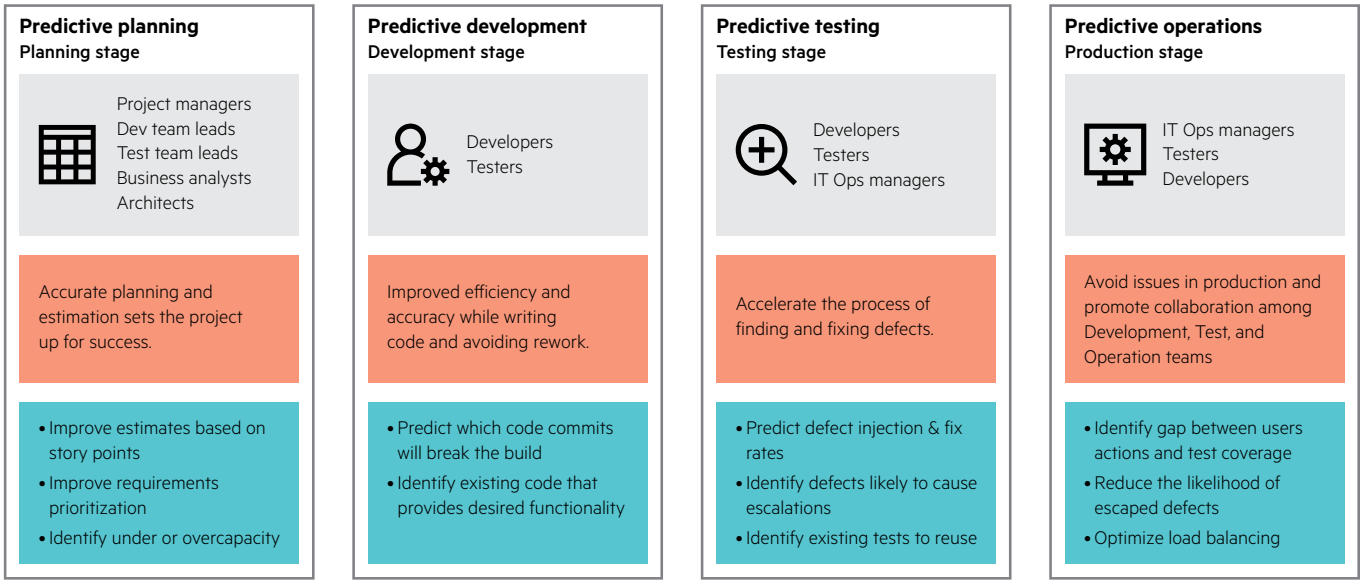

Figure 1. Overview of the HPE Predictive ALM framework

Getting your data ready

To make sure your project data is ready for analysis with Predictive ALM software, first, if you are not already saving your previous projects, you should. Next, we recommend using at least one ALM project per product, across several releases, as opposed to one project per release. If you have more historical data for analysis, the more accurate your results are likely to be. If the data set it too small, the predictions will not be accurate.

The accuracy of the ALM data is crucial. For example, the defect status should reflect its actual status. As soon as a user starts working on a defect, they should change its status from "New" to "Open." Likewise, when a defect is closed or reopened, its status should be updated accordingly. When we analyzed customer project data, we found many inaccuracies for defect status and this negatively affected the validity of our predictions.

One of the biggest challenges with Big Data analysis is dealing with missing, inconsistent, or poor-quality data. Therefore, for new software development projects, we recommend instituting a standard project template. If you are not using one already, a standard project template provides consistency both across projects and teams by ensuring that everyone is using the ALM fields in the same way. Aside from the analytic possibilities that are possible with Predictive ALM, using a standard template provides immediate benefit to your organization because you can institute a common project structure and use ALM's valuable cross-project features.

In addition, here are suggestions for using the ALM modules so your project data will be ready for analysis in Predictive ALM. While the suggestions outlined later focus on standard fields, we also encourage you to employ user-defined fields. There can be a wealth of information captured in these customized variables—just ensure that everyone is using them consistently.

Releases

A well-planned release is the basis for a successful project. Some key tasks in this phase include:

- Setting the release start and end dates.
- Defining the scope of the release in terms of features, themes, change requests, and identifying the requirements, tests, test instances, and defects that have to be managed.
- Defining milestones and associating them with scope to identify the release activities.
- Specifying the planning objectives per milestone by selecting KPIs and thresholds.

During the course of your application development project, things may change, such as customer priorities, resources, business needs, and more. Therefore, release planning is not finished at the beginning of a project; we recommend approaching it as an ongoing process to adjust your estimates as the resources and timelines change.

Remember, all ALM assets related to the release including requirements, tests, test instances, and defects must be in the same ALM software project as the release.

Requirements

ALM software helps you define and maintain a repository of requirements and tests. Requirements help cover business and testing needs. Tests can be automatically generated from requirements so that the most important functionality in an application is being tested. We recommend that you manage your requirements in ALM, for the information such as linkages between requirements and other entities (test, defects, and more) can be mined for data analysis. However, if you manage your requirements outside of ALM, we suggest that you synchronize them into ALM and manually update the relevant links.

Next, create a tree for your requirements and define different groups of requirements within it. The structure of your requirements tree should reflect the structure of the product, i.e., each product or feature area should be represented by its own requirement. Including rich-text descriptions and relevant attachments enhances understanding of your requirements and provides more information for analysis. You can use the requirement types to separate them according to product areas from other types of requirements. Assign requirements to a release and assign a priority level for each of them. This is useful for creating your test plan and for laying the groundwork for predictive analysis for future projects.

We also recommend that you add traceability between your requirements. Traceability reveals related requirements that might be affected by a proposed change. Generating a traceability matrix can also be helpful for you to assess the completeness of the relationships among your requirements. Afterward, we suggest you create coverage between your requirements and tests to ensure that all requirements are implemented in the project. You can do this by converting requirements to tests in the test plan tree. These associations are useful in current projects because they provide a better sense of dependencies and related work items. Also, having these links can enable a greater depth of analysis with Predictive ALM.

Finally, you should link requirements to specific defects. Linking defects to requirements and tests can help you complete requirements and testing. If a requirement changes, you can immediately identify which tests and defects are affected, and who is responsible for fixing and testing them. This also represents a valuable association that can be mined with Predictive ALM.

Test plans

For test plans, you should link each test in the test plan tree with one or more requirements. By specifying the requirements coverage for a test, you will be creating valuable metadata about the relationship between your tests and your requirements. This association proves useful for analysis and prediction of future requirements and the estimated test efforts they will need.

Likewise, link your tests to specific defects. This is useful when a new test is created for a defect. By creating this link, you will be able to identify which tests should be run and re-run for each defect. Furthermore, Predictive ALM will be able to mine this information and provide recommendations in future projects based on the observed associations between these tests and defects.

Test runs

For the duration of test run information to be accurate, we recommend running your tests continuously rather than interrupting them. For example, starting a test run at the end of the day, leaving the tests midway through, and then continuing them on the following day distorts the test duration metrics. In order to provide accurate recommendations in future projects, the Predictive ALM software algorithms need accurate test run metrics to mine. Ensuring the accuracy of your test run data is a great first step to improving your projects.

Defects

The defects module is one of the most-widely used tools in ALM. Many customers have a substantial amount of data about their defects from previous projects. While many customers also have user-defined fields, we recommend, at a minimum using the following:

- **Summary:** A brief summary of the defect, usually about one sentence.
- **Description:** A more detailed text description of the defect that may include reproduction steps, expected result, and actual result.
- **Detector:** The user who identified the defect.
- **Detection date:** The date when the defect was detected.
- **Assignee:** The user assigned to work on fixing the defect.
- **Status:** The status of the defect, chosen from the following options:
  - New: A defect was detected and a record was created in the system.
  - Open: The defect has been assigned to a developer, who has started working on the fix.
  - Fixed: The fix has been committed and is awaiting verification.
  - Closed: The defect fix has been verified and no further work needs to be performed on it.
  - Rejected: The developer, to who it was assigned, rejects the defect as irrelevant.
- **Severity:** A categorical field describing the functional impact that the defect has on the system. This field is frequently a source of contention between testers and development.
  - Critical: The most-serious defect level and must be fixed for a release. There is no workaround.
  - High: The defect impacts major functionality or data. It may have a workaround, but it is not obvious and is difficult to use.
  - Medium: The defect is related to minor product or feature functionality of data.
  - Low: The defect does not impact functionality or data, and does not need a workaround. May be a minor inconvenience to users.

In addition to the standard fields, we also suggest using the following user-defined fields (UDF):

- **Defect type:** Whether the defect is a bug or potential defect.
- **Detection mode:** Captures how the defect was discovered. Some examples include automated testing, bug hunts or bashes, formal testing, regression tests, security reviews, and more.
- **Initial estimated fix time:** You should include information about how long you expect the bug fix to take. This can be useful to compare to actual fix times in order to improve the estimations.

It is also important to note that besides using the fields listed above, you want people to use them consistently. Having accurate data on when a defect was created, assigned to a developer, reclassified as fixed, and more is very important for Predictive ALM. If defects are not being accurately classified, assigned, or updated, your previous project data becomes misleading and does not give you accurate predictions about upcoming projects. While we realize that development processes change over time and require changes to field values such as the list of defect statuses, the more consistent the list values are over time and across projects, the easier it is to apply the Predictive ALM algorithms and generate accurate recommendations.

Integrations with other tools

If you are not using ALM software for incident tracking, we recommend at least making sure that the values in your incident tracking tool match the list values in ALM. Having the same list values makes it much easier to look at links across defects and incidents to gain valuable insights.

In addition, we recommend using HPE Application Lifecycle Intelligence (ALI) tool, which tracks development activities and links them to ALM entities. ALI integrates your source code management (SCM) and build management tools with ALM. Having this connection lets you view activities such as code changes, builds, unit test results, and code coverage analysis from your SCM and build management tools in the context of ALM entities such as releases, requirements, defects, and tests.

Having a more comprehensive view of your SDLC by combining SCM and build information with ALM promotes greater control and consistency across the process. You can also use ALI to create and enforce SCM policies. For example, you can allow code check-ins only for the features that are scheduled for the release or allow check-ins during stabilization only for changes that fix severe defects. You can also require metadata before allowing a check-in or lock a code base for a release. Remember, having accurate metadata is essential for accurate results in the Predictive ALM software algorithms.

Finally, if we have to stress two points about getting your ALM project data ready to use Predictive ALM, consistency and sample size are the two most important points. How much data you have affects the level of accuracy. The more data you have, the more reliable are the Predictive ALM predictions and recommendations. Most development projects have a high rate of churn, so it's critical to have a large data repository to analyze.

Checklist

- Save your previous projects to gather sufficient historical data for analysis. The more data, the better.
- Have at least one ALM project per product, across several releases.
- Ensure the accuracy of the ALM data, e.g., defect status should reflect its actual status.
- Institute a standard project template to have consistency among teams and across projects.

Releases

– Include all ALM assets in the same ALM project as the release: requirements, tests, test instances, and defects.

Requirements

– Use requirements tree.

– Requirement tree structure should represent the product structure.

– Include rich-text descriptions and relevant attachments.

– Add traceability between your requirements.

– Link requirements to defects.

- Test plans
  - Link each test in the test plan tree with one or more requirements.
  - Link your tests to defects.
- Test runs
  - Capture accurate duration of test run metrics.
  - Run tests continuously without interruption.
- Defects
  - Use the following fields: summary, description, detector, detection date, assignee, status, and severity.
  - Employ user-defined fields.
  - Make sure people use the defect fields consistently.
- Use integrations with other tools
  - Incident tracking.
  - HPE ALI tool.

Learn more at
hpe.com/software/alm

4AA6-3470ENN, December 2015

Hewlett Packard Enterprise