The Total Economic Impact™ Of BMC Compuware Topaz For Total Test

Cost Savings And Business Benefits Enabled By Topaz For Total Test

AUGUST 2021
Table Of Contents

Executive Summary .................................................. 1
The BMC Compuware Topaz For Total Test
Customer Journey ................................................... 5
  Key Challenges .................................................. 5
  Investment Objectives ....................................... 6
  Composite Organization ..................................... 7
Analysis Of Benefits ................................................ 8
  Incremental Output Per Developer ......................... 8
  Direct Cost Avoidance: Bug Remediation ............. 10
  Direct Cost Avoidance: Testing Automation .......... 12
  Direct Cost Avoidance: Risk Avoidance ............... 14
  Flexibility ...................................................... 16
Analysis Of Costs .................................................. 18
  Estimated Topaz For Total Test License Fees ....... 18
  Estimated Setup And Training Costs .................... 19
  Estimated Implementation Costs ......................... 20
Financial Summary .................................................. 22
Appendix A: Total Economic Impact ....................... 23
Appendix B: Supplemental Material ......................... 24
Appendix C: Example Calculations For Waterfall
  Development .................................................. 24
Appendix D: Endnotes ............................................. 25

ABOUT FORRESTER CONSULTING

Forrester Consulting provides independent and objective research-based consulting to help leaders succeed in their organizations. For more information, visit forrester.com/consulting.

© Forrester Research, Inc. All rights reserved. Unauthorized reproduction is strictly prohibited. Information is based on the best available resources. Opinions reflect judgment at the time and are subject to change. Forrester®, Technographics®, Forrester Wave, RoleView, TechRadar, and Total Economic Impact are trademarks of Forrester Research, Inc. All other trademarks are the property of their respective companies.
Executive Summary

Businesses can modernize mainframe software development and deliver business applications faster by automating testing with BMC Compuware Topaz for Total Test. Forrester interviewed four customers and found that their developers spent less time manually testing and more time writing code supporting new business logic. Based on the customers’ experiences, Forrester developed a composite financial model that readers can apply to their own businesses.

When writing mainframe software, developers typically test code manually. For example, consider the typical process for functional testing: The developer specifies inputs, runs the program, and then verifies outputs. Although the process is conceptually simple, the time required to perform robust testing increases exponentially as programs become more and more complex. Furthermore, such a process is unreliable — developers can easily forget to run a test case or even misinterpret results.

Many solutions exist for developers writing code off-mainframe, but solutions for mainframe developers are relatively rare. Moreover, mainframe developers may have an even greater need for testing solutions because of the scale, complexity, and importance of their work.

Using BMC Compuware Topaz for Total Test, mainframe developers can automate testing.¹ BMC Software — which acquired Compuware in 2020 — commissioned Forrester Consulting to conduct a Total Economic Impact™ (TEI) study and examine the potential return on investment (ROI) enterprises may realize by adopting the platform.² This study provides a framework to evaluate the potential financial impact of adopting Topaz for Total Test within their own organizations.

To better understand the benefits, costs, and risks associated with this investment, Forrester interviewed seven decision-makers at four customers with experience using the tool. Forrester then aggregated the experiences of the interviewees and combined the results into a single composite organization.

Before using the testing platform, all of the interviewed customers tested their mainframe code manually. This was time-consuming: Developers spent 30% of their time testing. Despite their best efforts, issues still slipped through. The interviewees all wanted to increase developer productivity as well as improve code quality. If manual testing were not so laborious, developers could spend more time writing code supporting new business logic.

Readers can calculate the potential financial impact of Topaz for Total Test at their own organizations using Forrester’s companion Business Case Builder.
After deploying the solution, the interviewed organizations began automating their testing. The IT teams invested time into writing test scripts for the platform. Writing a test script took about as much time as performing a manual test. However, unlike a manual test, the test scripts were reusable and could be run automatically. Any subsequent time developers changed a program covered by automated tests, they saved significant amounts of time — about 90% of the time they had previously spent manually testing. The developers reinvested that time into building new features, fixing issues, and testing more. Development velocity and code quality increased, and the interviewed organizations saw the benefits on their bottom lines.

**KEY FINDINGS**

**Quantified benefits.** Risk-adjusted present value (PV) quantified benefits over three years include:

- **Overall time savings and increased developer output saved $4.5 million.** Developers reported that automated testing saved them about 10% of overall time — in other words, about one day every two weeks. For a developer, that is a significant amount of time. The teams used the time they saved to deliver projects faster and innovate more.

- **Bug reduction of 20% saved $376,000.** Automated testing caught defects that would have otherwise been missed through manual testing alone. Testing automation also caught bugs earlier in the software development process, when they are cheaper to fix.

- **Automation efficiencies saved $2.9 million.** With automation, code changes could be tested in a fraction of the time needed for manual testing. The interviewed organizations ran more tests in less time, ultimately increasing code quality.

- **Reduced risk of major mainframe defects saved $2.6 million.** The interviewees gave examples of Topaz for Total Test catching bugs that would have otherwise escaped and impacted customers. An analysis of public news sources suggests that major mainframe outages can occur as frequently as once every five years until organizations implement modern software development practices. Although the degree to which the interviewed organizations had adopted agile and DevOps practices varied, they had all deployed testing automation software to support their agile and DevOps transformations.

**Unquantified benefits.** A benefit that is not quantified for this study is cultural transformation. Developers writing code supporting business initiatives were happier and more productive. Testing automation software helped modernize development in an area of programming frequently mired in outdated practices.

**Costs.** Risk-adjusted PV costs include:

- **Licensing fees of an estimated $2 million over three years.** BMC Compuware licenses Topaz for Total Test on a per-user basis.

- **System setup and training costs of $235,000.** The interviewees reported that setup and training costs were relatively minor, but organizations considering testing automation should still plan for them.
EXECUTIVE SUMMARY

- Implementation costs of $1.2 million over three years. Organizations invested time into building their test script repositories. Organizations that invested more effort into automating testing realized greater benefits and realized those benefits sooner.

The customer interviews and financial analysis found that a composite organization experiences benefits of $10.34 million over three years versus costs of $3.40 million, adding up to a net present value (NPV) of $6.95 million and an ROI of 205%.

“Code quality has gone up — definitely — and the number of issues or defects we have had has gone down.”
Senior advisor for mainframe development, healthcare

Benefits (Three-Year)

- Incremental output per developer: $4.5M
- Direct cost avoidance: bug remediation: $375.8K
- Direct cost avoidance: testing automation: $2.9M
- Direct cost avoidance: risk avoidance: $2.6M

“Testing manually would have taken us two to three days. [But] this area was frequently changing and highly sensitive, so we had an automated test from a previous project. Instead of taking two to three days to test, we completed testing in 2 to 3 hours.”

— Senior advisor for mainframe development, healthcare
EXECUTIVE SUMMARY

TEI FRAMEWORK AND METHODOLOGY
From the information provided in the interviews, Forrester constructed a Total Economic Impact™ framework for those organizations considering an investment in Topaz for Total Test.

The objective of the framework is to identify the cost, benefit, flexibility, and risk factors that affect the investment decision. Forrester took a multistep approach to evaluate the impact that Topaz for Total Test can have on an organization.

DUE DILIGENCE
Interviewed BMC Compuware stakeholders and Forrester analysts to gather data relative to Topaz for Total Test.

CUSTOMER INTERVIEWS
Interviewed four decision-makers at organizations using Topaz for Total Test to obtain data with respect to costs, benefits, and risks.

COMPOSITE ORGANIZATION
Designed a composite organization based on characteristics of the interviewed organizations.

FINANCIAL MODEL FRAMEWORK
Constructed a financial model representative of the interviews using the TEI methodology and risk-adjusted the financial model based on issues and concerns of the interviewed organizations.

CASE STUDY
Employed four fundamental elements of TEI in modeling the investment impact: benefits, costs, flexibility, and risks. Given the increasing sophistication of ROI analyses related to IT investments, Forrester’s TEI methodology provides a complete picture of the total economic impact of purchase decisions. Please see Appendix A for additional information on the TEI methodology.

DISCLOSURES
Readers should be aware of the following:

This study is commissioned by BMC Compuware and delivered by Forrester Consulting. It is not meant to be used as a competitive analysis.

Forrester makes no assumptions as to the potential ROI that other organizations will receive. Forrester strongly advises that readers use their own estimates within the framework provided in the study to determine the appropriateness of an investment in Topaz for Total Test.

BMC Compuware reviewed and provided feedback to Forrester, but Forrester maintains editorial control over the study and its findings and does not accept changes to the study that contradict Forrester’s findings or obscure the meaning of the study.

BMC Compuware provided the customer names for the interviews but did not participate in the interviews.
The BMC Compuware Topaz For Total Test Customer Journey

Drivers leading to the Topaz for Total Test investment

<table>
<thead>
<tr>
<th>Interviewed Organizations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Industry</strong></td>
</tr>
<tr>
<td>Banking/financial services</td>
</tr>
<tr>
<td>Banking/financial services</td>
</tr>
<tr>
<td>Banking/financial services</td>
</tr>
<tr>
<td>Banking/financial services</td>
</tr>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Banking/financial services</td>
</tr>
</tbody>
</table>

KEY CHALLENGES

Before adopting Topaz for Total Test, the mainframe developers at the interviewed organizations tested code manually. Manual testing was both time-consuming and unreliable, but the development teams had no other option.

- **Developers devoted 30% of their time to testing.** Testing a simple code change could take half a day; testing a more complex change could take two to four days. The interviewed organizations typically budgeted about three days out of every 10-day sprint for testing. Testing was a time-consuming process for several reasons:
  - **Rigorous testing was necessary because mainframe code is mission-critical.** The organizations’ mainframe code supported systems essential to the business. The organizations also operated in highly regulated and highly sensitive environments — a bug that affects customers could easily cost an organization tens of millions of dollars. Code had to be tested rigorously.
  - **Legacy code.** In some cases, the code that the developers changed was decades old. In those instances, any new code would have to go through integration testing.
  - **Scale and complexity.** Similarly, the developers were working on programs that contained millions of lines of code, and these programs often worked together in complex ways. As the senior advisor for mainframe development at the healthcare organization explained, “Figuring out and testing the entire chain of programs relevant to a scenario was very hard and took a lot of time.”
  - **Multiple test cases.** To prevent regressions — that is, new code inadvertently breaking existing working code — the teams had standard tests that they performed for every release. It took days to run all of the test cases and scenarios.
  - **Manual testing was unreliable.** Even with the most rigorous standards and processes, the development teams sometimes failed to catch errors — or even accidentally introduced errors — during testing. Such issues were more likely when developers rushed to meet deadlines.
INVESTMENT OBJECTIVES

The interviewees searched for a solution that could:

- **Modernize mainframe development and accelerate agile and DevOps transformation.** The interviewees had adopted agile and DevOps software development practices to varying degrees. One of the organizations still used the waterfall approach for 70% of their projects; one of the organizations had fully adopted agile and DevOps practices; and two of the organizations were still transitioning. Nevertheless, all of the organizations knew that agile and DevOps could help them release higher-quality software faster, and they searched for tools that would support more modern ways of working. This was especially true for mainframe development; programmers rarely viewed the greenscreen interfaces as exciting environments in which to work.

- **Improve code quality and save developers’ time.** Bugs are expensive. Fixing a typical defect could take a developer anywhere from a few hours to one or two days. (One organization estimated that most defects took half a day to fix, but two organizations estimated that they fixed most issues within one to two days.) In addition, the time to fix a defect depended on when the bug was caught: An issue caught during QA could be fixed within days, but a defect that escaped into the production environment might take weeks to fix. Developer time is valuable, but a defect in production was live and visible to users — it could also result in reputational damage for the company or even regulatory fines.

**Deployment.** After completing initial proofs of concept (POCs), the interviewed organizations chose Topaz for Total Test and began deployment.

- **Optional but encouraged.** The interviewed organizations preurchased a certain number of user seats. Developers were encouraged to automate testing — at the most successful organizations, managers were very supportive of agile and DevOps initiatives — but ultimately, the individual developers decided whether or not to integrate testing automation into their regular workflows.

- **Increasing adoption.** Usage increased organically: Developers started automating testing once they saw their colleagues do so and realize benefits. However, at all of the interviewed organizations, only a subset of mainframe developers used testing automation. Adoption ranged from around 10% (e.g., 50 out of 500 developers) to around 60% (e.g., 300 out of 500 developers). The number of developers testing code automatically depended on several factors: 1) length of time since deploying Topaz for Total Test (one of the organizations saw adoption increase from 10% to 60% over a year and a half); 2) support from management; 3) inertia (cultural change is hard, and sometimes developers clung to old ways of working); and 4) the software compatibility with the codebase (one organization found that parts of its codebase were not able to be tested automatically).

- **Increasing code coverage.** To reap the benefits of testing automation, the organizations had to write test scripts. These test scripts are reusable, so writing them is a one-time investment. Most of the organizations incrementally built up their repository of test scripts: Instead of testing a single code change manually, they used the time to write test scripts, which took about the same amount of time. That meant that any subsequent times that code was changed, it could be tested automatically. The organizations thus increased the percentage of their codebase that they could test automatically over time.
COMPOSITE ORGANIZATION

Based on the interviews, Forrester constructed a TEI framework, a composite company, and an ROI analysis that illustrates the areas financially affected. The composite organization is representative of the four companies that Forrester interviewed and is used to present the aggregate financial analysis in the next section. The composite organization has the following characteristics:

**Description of composite.** The composite organization is a large, multibillion-dollar financial services firm with global operations and millions of customers. The composite organization’s mainframe codebase contains thousands of programs and millions of lines of code, some of which are decades old. The organization has several hundred mainframe developers and completes several million dollars’ worth of mainframe development work each year.

**Deployment characteristics.** The composite organization creates an implementation team of 10 developers who deploy the testing automation software on the organization’s mainframes and then spend 30% of each year—time that they would have otherwise spent testing manually—writing test scripts for Topaz for Total Test. By the end of Year 1, the composite organization’s repository of test scripts covers 10% of the organization’s mainframe codebase; by the end of Year 2, 13% of the codebase; and by the end of Year 3, 17% of the codebase.

In the beginning of Year 1, 200 developers use Topaz for Total Test. The number of developers using the software grows at a rate of 20% per year due to a combination of organic adoption as well as encouragement from management. In Year 2, 240 developers use the platform, and in Year 3, 288 developers use the platform. Also, developer salaries grow by 3% per year.

Ordinarily — without testing automation software — the mainframe developers spend 30% of their time testing code manually. The organization uses agile and releases code in two-week sprints. However, Forrester’s analysis is agnostic of the software development methodology an organization uses, and Appendix C contains an equivalent analysis for an organization using waterfall development practices.
Analysis Of Benefits

Quantified benefit data as applied to the composite

### Total Benefits

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Benefit</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
<th>Total</th>
<th>Present Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atr</td>
<td>Incremental output per developer</td>
<td>$1,018,368</td>
<td>$1,766,016</td>
<td>$2,791,895</td>
<td>$5,576,279</td>
<td>$4,482,899</td>
</tr>
<tr>
<td>Btr</td>
<td>Direct cost avoidance: bug remediation</td>
<td>$122,204</td>
<td>$150,958</td>
<td>$186,325</td>
<td>$459,488</td>
<td>$375,842</td>
</tr>
<tr>
<td>Ctr</td>
<td>Direct cost avoidance: testing automation</td>
<td>$687,398</td>
<td>$1,103,881</td>
<td>$1,781,737</td>
<td>$3,573,016</td>
<td>$2,875,851</td>
</tr>
<tr>
<td>Dtr</td>
<td>Direct cost avoidance: risk avoidance</td>
<td>$800,000</td>
<td>$1,040,000</td>
<td>$1,360,000</td>
<td>$3,200,000</td>
<td>$2,608,565</td>
</tr>
<tr>
<td></td>
<td>Total benefits (risk-adjusted)</td>
<td>$2,627,971</td>
<td>$4,060,855</td>
<td>$6,119,957</td>
<td>$12,808,783</td>
<td>$10,343,157</td>
</tr>
</tbody>
</table>

### INCREMENTAL OUTPUT PER DEVELOPER

**Evidence and data.** Testing automation saved developers time, which they then spent writing code supporting new business logic.

Testing automation saved 10% of developers’ time, the equivalent of about 1 day/sprint

- A financial services firm adopted testing automation as well as other software from BMC Compuware as part of its transition to agile and DevOps. After the organization transitioned, the mainframe team’s output increased by 233% — from an average of 300 story points per release to more than 1,000. Forrester attributes about half of the increase to testing automation with Topaz for Total Test.

During their interview with Forrester, the financial services firm mentioned two specific projects that were completed faster after the mainframe development team was “firing on all cylinders.” The first was an e-commerce authentication program that the team released in four months instead of nine — 225% faster. The second project was a financial fraud-prevention program that had resulted in $952,000 worth of benefits at the time of the interview. According to the interviewees, modern software development practices — including testing automation — shaved three to four months off of the development timeline.

- The developers at two other financial services firms separately estimated that automating their testing saved about 10% of their overall time. This was equivalent to about one day every two weeks — a significant gain for the developers.

**Modeling and assumptions.** For the composite organization, Forrester assumes:

- In Year 1, the developers save 5% of their overall time due to testing automation. In Year 2, they save 7%, and in Year 3, they save 9%. The amount of time that developers save increases
as the composite organization builds up its repository of test scripts. This is because testing automation only saves developers time when they already have test scripts written for the code they are changing.

- The developers use 80% of the time they save productively. Realistically, workers are never productive 100% of their time. However, a developer saving 5% of their time saves about 4 hours or half a day every two weeks. That is enough time to complete a significant amount of work.

Although the composite organization practices agile and releases code in two-week sprints, an organization using waterfall practices would see equivalent benefits. Appendix C shows calculations for a composite organization using waterfall.

**Risks.** This benefit can vary from organization to organization due to the following factors:

- **Investment in testing automation (i.e., implementation effort).** Teams must devote time to writing test scripts to reap the benefits of testing automation. Organizations that invest more resources (both time and employees) into testing automation will see greater benefits sooner. Conversely, organizations that invest fewer resources may see lower benefits or realize benefits more slowly. For more information, see Estimated Implementation Costs.

- **Organizational buy-in and developer adoption.** Forrester assumes that the number of developers using Topaz for Total Test grows by 20% each year. Management encourages the use of automated testing, and some growth happens organically. However, the adoption rate may vary depending on how supportive an organization’s management is. Also, testing automation entails a cultural shift to more agile ways of working, which some developers might resist.

- **Compatibility.** One of the interviewees reported that testing automation software was not compatible with all of the code in their mainframe codebase. However, even that organization still saw significant time savings of 10% to 20%. Readers should consult with BMC Compuware regarding the technical details of their specific codebases.

**Results.** To account for these risks, Forrester adjusted this benefit downward by 10%, yielding a three-year, risk-adjusted total PV (discounted at 10%) of nearly $4.5 million.

Incremental output per developer: 43% of total benefits
ANALYSIS OF BENEFITS

DIRECT COST AVOIDANCE: BUG REMEDIATION

Evidence and data. Topaz for Total Test caught defects that would have otherwise escaped. In addition, catching issues sooner in the development process made them easier to fix. Fixing a defect during unit testing was negligible, but fixing the same defect in production could take two to three weeks. An issue caught by Topaz for Total Test was an issue that developers did not have to spend time fixing during the next release. Interviewees reported that testing automation reduced the number of defects by 10% to 30%.

- The financial services firm that adopted testing automation along with other software from BMC Compuware as it transitioned to agile and DevOps observed 83% fewer bugs. Previously, the development team registered about 90 defects for every 300-story-point release; afterward, the team registered only 50 defects per release, and the releases contained 1,000 story points — more than three times more work. Each issue took about 4 hours to fix.

- The healthcare company registered a 20% reduction in defect density, or the number of bugs per 1,000 hours of development time. For this organization, a defect could take one to two days to fix if it were caught during QA but more than two weeks to fix if it were found in the live production environment.

“...the better quality we have.”

Senior advisor for mainframe development, healthcare
Another financial services firm reported that testing automation reduced the number of defects by 15% to 30%. This team measured only the significant issues that took at least one to two days to fix. The developers tested more often, as well earlier in the software development lifecycle. With testing automation, they tested code themselves before checking it in. The organization did not even measure any defects caught at that stage because the developers fixed them right away. Because the issues were caught so early, fixing them was frictionless.

Topaz for Total Test catches 20% of bugs. This means that the issues do not make it into the release and so do not need to be fixed later (e.g., during an upcoming sprint).

**Risks.** This benefit can vary from organization to organization due to the following factors:

- **Organization-specific characteristics.** The average cost to remediate one bug, as well as defect density, may vary from organization to organization. Readers can easily customize Forrester’s calculations by substituting in relevant metrics for their own organizations.

- **Percentage of bugs trapped.** No testing process is foolproof, and readers should not expect testing automation to catch 100% of bugs. Although testing automation can help teams improve code quality, it is not intended to be the only solution teams adopt.

**Results.** To account for these risks, Forrester adjusted this benefit downward by 10%, yielding a three-year, risk-adjusted total PV of more than $375,000.

*First*, fewer defects travel to the test environment because we can run regression testing automatically. Second, 100% of the regression test cases are run, and there is no human error—we don’t have testers missing one or two cases or missing the validation. So fewer bugs are being released to the test environment that have to be fixed later.”  
Senior software engineer, financial services

**Direct cost avoidance: bug remediation:** 4% of total benefits

Modeling and assumptions. For the composite organization, Forrester assumes:

- A defect takes an average of 4 hours to fix.
- Without testing automation, the development team creates six defects for every 1,000 hours of development time.
### Direct Cost Avoidance: Bug Remediation

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Metric</th>
<th>Calculation</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>Average time to remediate 1 bug (hours)</td>
<td>Assumption</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>B2</td>
<td>Bugs per 1,000 hours of development time before Topaz for Total Test</td>
<td>Assumption</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>B3</td>
<td>Estimated percentage of bugs trapped by Topaz for Total Test</td>
<td>Assumption</td>
<td>20%</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td>B4</td>
<td>Developer hourly rate</td>
<td>A2</td>
<td>$68</td>
<td>$70</td>
<td>$72</td>
</tr>
<tr>
<td>B5</td>
<td>Hours per developer per year</td>
<td>Assumption</td>
<td>2,080</td>
<td>2,080</td>
<td>2,080</td>
</tr>
<tr>
<td>B6</td>
<td>Number of developers using Topaz for Total Test</td>
<td>A1</td>
<td>200</td>
<td>240</td>
<td>288</td>
</tr>
<tr>
<td>Bt</td>
<td>Direct cost avoidance: bug remediation</td>
<td>B1*(B2/1,000)<em>B3</em>B4<em>B5</em>B6</td>
<td>$135,782</td>
<td>$167,731</td>
<td>$207,028</td>
</tr>
<tr>
<td></td>
<td>Risk adjustment</td>
<td>↓10%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Btr</td>
<td>Direct cost avoidance: bug remediation (risk-adjusted)</td>
<td></td>
<td>$122,204</td>
<td>$150,958</td>
<td>$186,325</td>
</tr>
</tbody>
</table>

**Three-year total: $459,487**

**Three-year present value: $375,842**

---

**DIRECT COST AVOIDANCE: TESTING AUTOMATION**

**Evidence and data.** Testing automation reduced the amount of time developers needed to spend manually testing code changes. Previously, interviewees had spent roughly 30% of a release on manual testing. The gains from testing automation were significant: Developers saved 80% to 95% of the time they had previously spent manually testing. The teams used the time they saved to do further testing, thereby improving code quality.

- **Reusable tests.** Before a financial services firm had a solution for automated testing, developers wrote tests that were rarely reusable. As the lead software engineer explained: “Before Topaz for Total Test, any unit test was effectively thrown away and had to be repeated because the developers would do amendments. They would write the unit test, [create] custom jobs, create custom data sets, prove the change, check the change, [etc.]. Unless you gave the same developer the same test all the time, it’s disposable.” Testing automation alleviated this problem in a few ways. First, test scripts were stored in a central repository so they could be easily accessed and used by multiple developers — once a developer wrote a test, others could use it, too. Second, the software made managing test data easy and thus eliminated the need for one-off, custom data sets. Before implementing an automated testing tool, the financial services firm had spent 12 hours setting up and...
performing unit tests. Afterward, preparing unit tests took less than 2 hours. One team ran 180 such unit tests per year and saved about 1,800 hours — almost the equivalent of one FTE.

- **Efficiency.** Another financial services firm uses automated testing primarily for regression testing. Previously, manual regression testing had taken three to four days. However, with automated testing, developers completed the same testing regimen in 2 or 3 hours. Although there were still some manual steps — it is difficult to automate the entire process — the team had realized time savings of more than 90%. They used the time saved to perform additional tests that they previously had not had capacity to complete. Similarly, a healthcare firm went from running 10 tests per day manually to running hundreds of tests per hour. The senior advisor for mainframe development explained: “You can put all of your test cases in a spreadsheet and then run every one of them together by clicking a single button. We’re getting more testing done in a shorter amount of time and thereby improving code quality.”

### Regression tests per day before and after Topaz for Total Test

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>800</td>
</tr>
</tbody>
</table>

**Modeling and assumptions.** For the composite organization, Forrester assumes:

- With manual testing only, developers must spend 30% of every release testing code.
- Developers only save time if there are already automated test scripts for the code they are changing. In Year 1, for example, the implementation team writes test scripts for 10% of the organization’s mainframe codebase. Thus, in Year 1, there is only a 10% chance that the developers will save time with testing automation.
- If there are already test scripts written for the code that the developers are changing, then they realize time savings of 90%.

**Risks.** This benefit can vary from organization to organization due to the following factors:

- **Investment in testing automation (i.e., implementation effort).** The lead software engineer at a financial services firm explained: “The first time, there is no saving because the amount of effort required to set up in Topaz for Total Test would be the same as the amount of effort to do the test by hand. But [for subsequent test executions], you’re entering the savings zone because you don’t have to do the manual testing again, because of your reusable test cases.” This means that organizations that invest more resources into testing automation will see greater benefits sooner. Conversely, organizations that invest fewer resources may see lower benefits or realize benefits more slowly. For more information, see [Estimated Implementation Costs](#).

- **Frequency of code changes.** At most organizations, developers change certain mainframe programs (e.g., those under active development) more frequently than others. The senior advisor for mainframe development in healthcare estimated that 40% of his organization’s codebase changed only once every three years. Since testing automation starts saving developers time the second time they change code, organizations that change mainframe code more frequently (e.g., several times per year) may see their investments pay off.
sooner. Conversely, organizations that change their mainframe codebase less frequently (e.g., the healthcare organization noted above) might view testing automation as long-run investment.

- **Time savings.** Time savings may vary depending on the type of test automated as well as on organization-specific characteristics such as testing processes. Automation typically results in time savings of at least 80%. None of the interviewees expected to fully automate testing; all expected testing to always involve some manual steps.

**Results.** To account for these risks, Forrester adjusted this benefit downward by 10%, yielding a three-year, risk-adjusted total PV of nearly $2.9 million.

### Direct Cost Avoidance: Testing Automation

<table>
<thead>
<tr>
<th>Ref</th>
<th>Metric</th>
<th>Calculation</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Number of developers using Topaz for Total Test</td>
<td>A1</td>
<td>200</td>
<td>240</td>
<td>288</td>
</tr>
<tr>
<td>C2</td>
<td>Hours per developer per release</td>
<td>A4</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>C3</td>
<td>Releases per year</td>
<td>A5</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>C4</td>
<td>Percentage of time spent testing per release before Topaz for Total Test</td>
<td>Assumption</td>
<td>30%</td>
<td>30%</td>
<td>30%</td>
</tr>
<tr>
<td>C5</td>
<td>Percentage of codebase covered by Topaz for Total Test automated testing suite</td>
<td>Assumption (including growth)</td>
<td>10%</td>
<td>13%</td>
<td>17%</td>
</tr>
<tr>
<td>C6</td>
<td>Percentage of testing time saved via automation with Topaz for Total Test</td>
<td>Assumption</td>
<td>90%</td>
<td>90%</td>
<td>90%</td>
</tr>
<tr>
<td>C7</td>
<td>Developer hourly rate</td>
<td>A2</td>
<td>$68</td>
<td>$70</td>
<td>$72</td>
</tr>
<tr>
<td>Ct</td>
<td>Direct cost avoidance: testing automation</td>
<td>C1<em>C2</em>C3<em>C4</em>C5<em>C6</em>C7</td>
<td>$763,776</td>
<td>$1,226,534</td>
<td>$1,979,707</td>
</tr>
<tr>
<td></td>
<td>Risk adjustment</td>
<td>↓10%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ctr</td>
<td>Direct cost avoidance: testing automation (risk-adjusted)</td>
<td></td>
<td>$687,398</td>
<td>$1,103,881</td>
<td>$1,781,737</td>
</tr>
</tbody>
</table>

**Three-year total:** $3,573,016  
**Three-year present value:** $2,875,851

### DIRECT COST AVOIDANCE: RISK AVOIDANCE

**Evidence and data.** Topaz for Total Test helped the interviewed organizations avoid risk by either catching customer-facing bugs that would have escaped or by allowing developers to quickly fix defects that had escaped. In some industries — such as finance or healthcare — customer-facing
mainframe issues can easily cost an organization millions of dollars.

- One of the organizations estimated that Topaz for Total Test had caught four significant defects in the past seven months that would have otherwise escaped with their old, manual testing methods.

- Another organization discovered a live bug in a highly sensitive area of its production environment. This bug affected customers. Ordinarily, the organization would have needed three days to manually test its fix. However, the defect was in a program that changed frequently, so the team had automated testing for it. With Topaz for Total Test, the development team tested a fix in less than 3 hours and fixed the issue one week sooner.

For this study, Forrester analyzed primary news sources and found that significant, customer-facing mainframe issues occurred roughly once every five years until organizations adopted agile software development practices. At the interviewed organizations, Topaz for Total Test contributed to the adoption of agile practices and improved overall development speeds. The testing automation software thus helped reduce the likelihood of major, customer-facing mainframe issues.

**Modeling and assumptions.** For the composite organization, Forrester assumes:

- A significant, customer-facing mainframe issue costs the composite organization $50 million. These damages come from both regulatory fines and reputational damage. The composite organization is a financial services company; it operates in a highly regulated, highly sensitive environment. Moreover, for the composite organization, consumer trust is paramount: Brand reputation is crucial to attracting and retaining customers.

- Without testing automation, the composite organization’s probability of exposure is 20% annually (i.e., each year, the composite organization has a one-in-five chance of accidentally releasing a significant, customer-facing bug).

- As the composite organization builds up its repository of test scripts — thereby increasing the percentage of its codebase covered by automated testing — it correspondingly reduces the risk of a significant mainframe issue escaping testing and going live.

**Risks.** This benefit can vary from organization to organization due to the following factors.

- **Exposure.** The potential exposure — i.e., the cost when a major mainframe issue impacts customers — as well as the baseline probability of exposure will vary across industries and organizations. Industries such as finance may have more exposure to regulatory fines and while organizations in industries such as healthcare or insurance might be more exposed to reputational damages. Also, the analysis of historical incidents suggests that the software development practices organizations use could generally influence the risk of major mainframe issues.

- **Investment in testing automation (i.e., implementation effort).** Organizations that invest more resources into testing automation may see greater benefits sooner. Conversely, organizations that invest fewer resources may see lower benefits or realize benefits more slowly. For more information, see [Estimated Implementation Costs](#).

**Results.** To account for these risks, Forrester adjusted this benefit downward by 20%, yielding a three-year, risk-adjusted total PV of $2,608,565.
ANALYSIS OF BENEFITS

Direct cost avoidance: risk avoidance: 25% of total benefits

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Metric</th>
<th>Calculation</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Potential exposure</td>
<td>Assumption</td>
<td>$50,000,000</td>
<td>$50,000,000</td>
<td>$50,000,000</td>
</tr>
<tr>
<td>D2</td>
<td>Probability of exposure before Topaz for Total Test</td>
<td>Assumption</td>
<td>20%</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td>D3</td>
<td>Percentage of codebase covered by Topaz for Total Test automated test suite</td>
<td>C5</td>
<td>10%</td>
<td>13%</td>
<td>17%</td>
</tr>
<tr>
<td>D4</td>
<td>Risk reduction attributed to Topaz for Total Test</td>
<td>D2*D3</td>
<td>2.00%</td>
<td>2.60%</td>
<td>3.40%</td>
</tr>
<tr>
<td>Dt</td>
<td>Direct cost avoidance: risk avoidance</td>
<td>D1*D4</td>
<td>$1,000,000</td>
<td>$1,300,000</td>
<td>$1,700,000</td>
</tr>
<tr>
<td></td>
<td>Risk adjustment</td>
<td>↓20%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dtr</td>
<td>Direct cost avoidance: risk avoidance (risk-adjusted)</td>
<td></td>
<td>$800,000</td>
<td>$1,040,000</td>
<td>$1,360,000</td>
</tr>
</tbody>
</table>

Three-year total: $3,200,000  
Three-year present value: $2,608,565

FLEXIBILITY

In Forrester’s Total Economic Impact™ (TEI) analysis, flexibility captures the strategic value that can be obtained from future additional investments that build on top of an initial investment. During the interviews, the interviewees reported additional benefits that were not quantifiable but were nevertheless important to their experiences. These benefits could provide the customers with additional opportunities in the future.

- **Cultural transformation.** Multiple organizations reported that testing automation changed their mainframe development culture and helped employees become happier, more productive, and more innovative. Before testing automation, some programmers had viewed mainframe development as outdated; however, after the organizations adopted testing automation software, mainframe development became cutting-edge. Testing automation supported faster software development, both at organizations with established agile and DevOps practices as well as at those organizations just beginning their DevOps journeys.

At one organization, automated testing reduced the employee ramp-up time from nine months down to four or five months. New employees became fully productive sooner. Because Topaz
for Total Test has a graphical user interface (GUI), new hires found the platform easy to learn. They quickly integrated into the mainframe team's development practices.

Testing automation also helped the same organization recruit and retain a younger workforce. Topaz for Total Test (along with other software from BMC Compuware) provided capabilities that younger programmers had come to expect in modern software development. The organization certainly did not replace its experienced developers. Rather, with an automated testing suite, the organization positioned itself to recruit and cultivate the next generation of talent.

The precise value of flexibility is unique to each customer. Flexibility is described further in Appendix A.

“Early gains have to do with the culture changes — for example, removing the handoff between dev and test. No longer is a piece of code to be tested like a hand grenade that gets lobbed across the fence. . . . We stopped calling people functional testers. It became banned to described somebody within the team as a tester. Everybody is a dev team member..”

Lead software engineer, financial services
Analysis Of Costs

Quantified cost data as applied to the composite

---

**Total Costs**

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Cost</th>
<th>Initial</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
<th>Total</th>
<th>Present Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Etr</td>
<td>Estimated Topaz for Total Test license fees</td>
<td>$0</td>
<td>$660,000</td>
<td>$792,000</td>
<td>$950,400</td>
<td>$2,402,400</td>
<td>$1,968,595</td>
</tr>
<tr>
<td>Ftr</td>
<td>Estimated setup and training costs</td>
<td>$44,880</td>
<td>$149,600</td>
<td>$30,800</td>
<td>$38,016</td>
<td>$263,296</td>
<td>$234,897</td>
</tr>
<tr>
<td>Gtr</td>
<td>Estimated implementation costs</td>
<td>$0</td>
<td>$466,752</td>
<td>$480,480</td>
<td>$494,208</td>
<td>$1,441,440</td>
<td>$1,192,717</td>
</tr>
<tr>
<td></td>
<td>Total costs (risk-adjusted)</td>
<td>$44,880</td>
<td>$1,276,352</td>
<td>$1,303,280</td>
<td>$1,482,624</td>
<td>$4,107,136</td>
<td>$3,396,209</td>
</tr>
</tbody>
</table>

**ESTIMATED TOPAZ FOR TOTAL TEST LICENSE FEES**

**Modeling and assumptions.** Pricing from BMC varies based on volume and other factors. Installation support and training are included in the licensing costs.

**Risks.** This cost can vary from organization to organization due to the following factors.

- **Organization-specific characteristics.** Enterprise software for mainframe computers is inherently different from, say, cloud software. Pricing could vary based on a variety of organization-specific characteristics, and prospective customers should always contact BMC for detailed quotes specific to their organizations.

- **Developer adoption.** Forrester assumes that the number of users at the composite organization grows by 20% each year due to both encouragement from management as well as viral growth — developers see their colleagues using Topaz for Total Test and decide to try it, too. Of course, the actual adoption rate that an organization experiences may vary from Forrester’s model. The mean growth rate for developer adoption that the interviewees reported was approximately 30%.

**Results.** To account for these risks, Forrester adjusted this cost upward by 10%, yielding a three-year, risk-adjusted total PV (discounted at 10%) of $1,968,595.
ANALYSIS OF COSTS

ESTIMATED SETUP AND TRAINING COSTS

Evidence and data. The interviewed organizations spent time installing Topaz for Total Test on their mainframes and on their developers' local PCs, but this was not a major cost for them.

In addition, the developers spent time learning how to use the software. Mainframe developers are experts in mainframe languages such as COBOL, and some developers had to learn JavaScript.

However, developers typically became proficient with the platform in one or two days. One organization trained its developers in two days via online courses. Another organization used a combination of recorded videos and live lessons for a total of 10 hours of training per developer. BMC Compuware also provided ample support resources and services to ensure customers were successful.

Modeling and assumptions. For the composite organization, Forrester assumes:

- In addition, each new user requires 10 hours of online training to become proficient with the platform.

Results. To account for variation from organization to organization, Forrester adjusted this cost upward by 10%, yielding a three-year, risk-adjusted total PV of $234,897.

### Estimated Topaz for Total Test License Fees

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Metric</th>
<th>Calculation</th>
<th>Calculation</th>
<th>Initial</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Topaz for Total Test per developer license fee</td>
<td>Assumption</td>
<td>$3,000</td>
<td>$3,000</td>
<td>$3,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td>Number of developers using Topaz for Total Test</td>
<td>A1</td>
<td>200</td>
<td>240</td>
<td>288</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Et</td>
<td>Estimated Topaz for Total Test license fees</td>
<td>E1*E2</td>
<td>$0</td>
<td>$600,000</td>
<td>$720,000</td>
<td>$864,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Risk adjustment 10%</td>
<td></td>
<td>$0</td>
<td>$660,000</td>
<td>$792,000</td>
<td>$950,400</td>
<td></td>
</tr>
</tbody>
</table>

Three-year total: $2,402,400

Three-year present value: $1,968,595

$234,897

three-year cost PV

Estimated setup and training costs: 7% of total costs
ANALYSIS OF COSTS

ESTIMATED IMPLEMENTATION COSTS

Evidence and data. To benefit from testing automation, organizations must invest time into writing and building repositories of test scripts to cover their mainframe codebases.

The organizations that Forrester interviewed had decided to build their repositories over time: whenever they changed a certain section of code, they wrote test scripts for it instead of testing the code manually.

Writing test scripts took about the same amount of time as testing code manually, and so developers did not realize any benefits from testing automation the first time that they changed a segment of code. However, because they had invested in writing test scripts, they did see benefits every subsequent time that they changed that code. Testing automation was a long-run investment.

Even those organizations that only written test scripts to cover a minority of their codebase saw returns if the code covered by the test scripts changed frequently.

Modeling and assumptions. For the composite organization, Forrester assumes:

- The implementation team of 10 developers spends 30% of each year writing test scripts. The composite organization decides to concentrate the effort of writing test scripts in the hands of the implementation team. Without testing automation, the implementation team would spend 30% of their time testing code manually anyway. Hypothetically, the implementation team could devote more time during the year to writing test scripts for Topaz for Total Test. Doing so would enable the composite organization to realize higher returns from testing automation sooner. However, the implementation team also has to balance implementing testing automation with its other projects and so it spreads out the effort.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Metric</th>
<th>Calculation</th>
<th>Initial</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Topaz for Total Test implementation team size</td>
<td>Assumption</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>New developers using Topaz for Total Test</td>
<td>Assumption</td>
<td>10</td>
<td>200</td>
<td>40</td>
<td>48</td>
</tr>
<tr>
<td>F3</td>
<td>Hours spent planning and deploying Topaz for Total Test on mainframes</td>
<td>Assumption</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4</td>
<td>Additional hours spent setting up local PCs and training</td>
<td>Assumption</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>F5</td>
<td>Developer hourly rate</td>
<td>A2</td>
<td>$68</td>
<td>$68</td>
<td>$70</td>
<td>$72</td>
</tr>
<tr>
<td>F6</td>
<td>Estimated setup and training costs</td>
<td>(F1+F2)*(F3+F4)*F5</td>
<td>$40,800</td>
<td>$136,000</td>
<td>$28,000</td>
<td>$34,560</td>
</tr>
<tr>
<td>F7</td>
<td>Risk adjustment</td>
<td>↑10%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F8</td>
<td>Estimated setup and training costs (risk-adjusted)</td>
<td></td>
<td>$44,880</td>
<td>$149,600</td>
<td>$30,800</td>
<td>$38,016</td>
</tr>
</tbody>
</table>

Three-year total: $263,296

Three-year present value: $234,897
composite organization’s mainframe codebase. This is a rate consistent with those reported by interviewees. In subsequent years, though, codebase coverage increases by 30% per year, to 13% in Year 2 and then about 17% in Year 3, as shown in row C5 of the calculation table for Benefit C: Testing Automation.

**Risks.** This cost can vary from organization to organization due to the following factors.

- **Normal variation in developer skills and test complexity.** According to the interviewees, the effort to write test scripts can vary depending on developer skills and experience as well as on the complexity and type of test script.\(^3\) However, such variation is typical of all types of programming. In addition, all of the interviewed organizations reported that writing a test script took about as much time as performing a manual test. (So experienced developers both wrote test scripts faster and tested manually faster, but writing test scripts was a far better investment of their time — the reusable test scripts saved them time any subsequent time they changed the same code.)

- **Technical characteristics specific to an organization.** Organizations very different from the composite organization might expect different implementation costs. E.g., organizations with smaller mainframe codebases might expect lower implementation costs. Hypothetically, there could also be exceptional technical reasons why implementation costs differ from what the interviewed organizations experienced. E.g., an organization’s mainframe codebase may be atypically complex; testing automation might not be compatible with certain mainframe programs; etc. Such a detailed technical analysis is beyond the scope of this study.

**Results.** To account for these risks, Forrester adjusted this cost upward by 10%, yielding a three-year, risk-adjusted total PV of $1,192,717.

### Estimated Implementation Costs

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Metric</th>
<th>Calculation</th>
<th>Initial</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>Topaz for Total Test implementation team size</td>
<td>F1</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>G2</td>
<td>Developer hourly rate</td>
<td>F5</td>
<td>$68</td>
<td>$70</td>
<td>$72</td>
<td></td>
</tr>
<tr>
<td>G3</td>
<td>Implementation hours per developer</td>
<td>Assumption: 30%*2,080</td>
<td>624</td>
<td>624</td>
<td>624</td>
<td></td>
</tr>
<tr>
<td>Gt</td>
<td>Estimated implementation costs</td>
<td>G1<em>G2</em>G3</td>
<td>$0</td>
<td>$424,320</td>
<td>$436,800</td>
<td>$449,280</td>
</tr>
<tr>
<td>Gtr</td>
<td>Estimated implementation costs (risk-adjusted)</td>
<td>↑10%</td>
<td>$0</td>
<td>$466,752</td>
<td>$480,480</td>
<td>$494,208</td>
</tr>
</tbody>
</table>

**Three-year total: $1,441,440**  
**Three-year present value: $1,192,717**
Financial Summary

CONSOLIDATED THREE-YEAR RISK-ADJUSTED METRICS

Cash Flow Chart (Risk-Adjusted)

The financial results calculated in the Benefits and Costs sections can be used to determine the ROI and NPV for the composite organization’s investment. Forrester assumes a yearly discount rate of 10% for this analysis.

These risk-adjusted ROI, and NPV values are determined by applying risk-adjustment factors to the unadjusted results in each Benefit and Cost section.

Cash Flow Analysis (Risk-Adjusted Estimates)

<table>
<thead>
<tr>
<th></th>
<th>Initial</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
<th>Total</th>
<th>Present Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total costs</td>
<td>($44,880)</td>
<td>($1,276,352)</td>
<td>($1,303,280)</td>
<td>($1,482,624)</td>
<td>($4,107,136)</td>
<td>($3,396,209)</td>
</tr>
<tr>
<td>Total benefits</td>
<td>$0</td>
<td>$2,627,971</td>
<td>$4,060,855</td>
<td>$6,119,957</td>
<td>$12,808,783</td>
<td>$10,343,157</td>
</tr>
<tr>
<td>Net benefits</td>
<td>($44,880)</td>
<td>$1,351,619</td>
<td>$2,757,575</td>
<td>$4,637,333</td>
<td>$8,701,647</td>
<td>$6,946,948</td>
</tr>
<tr>
<td>ROI</td>
<td>205%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix A: Total Economic Impact

Total Economic Impact is a methodology developed by Forrester Research that enhances a company’s technology decision-making processes and assists vendors in communicating the value proposition of their products and services to clients. The TEI methodology helps companies demonstrate, justify, and realize the tangible value of IT initiatives to both senior management and other key business stakeholders.

TOTAL ECONOMIC IMPACT APPROACH

**Benefits** represent the value delivered to the business by the product. The TEI methodology places equal weight on the measure of benefits and the measure of costs, allowing for a full examination of the effect of the technology on the entire organization.

**Costs** consider all expenses necessary to deliver the proposed value, or benefits, of the product. The cost category within TEI captures incremental costs over the existing environment for ongoing costs associated with the solution.

**Flexibility** represents the strategic value that can be obtained for some future additional investment building on top of the initial investment already made. Having the ability to capture that benefit has a PV that can be estimated.

**Risks** measure the uncertainty of benefit and cost estimates given: 1) the likelihood that estimates will meet original projections and 2) the likelihood that estimates will be tracked over time. TEI risk factors are based on “triangular distribution.”

---

The initial investment column contains costs incurred at “time 0” or at the beginning of Year 1 that are not discounted. All other cash flows are discounted using the discount rate at the end of the year. PV calculations are calculated for each total cost and benefit estimate. NPV calculations in the summary tables are the sum of the initial investment and the discounted cash flows in each year. Sums and present value calculations of the Total Benefits, Total Costs, and Cash Flow tables may not exactly add up, as some rounding may occur.

---

**PRESENT VALUE (PV)**

The present or current value of (discounted) cost and benefit estimates given at an interest rate (the discount rate). The PV of costs and benefits feed into the total NPV of cash flows.

**NET PRESENT VALUE (NPV)**

The present or current value of (discounted) future net cash flows given an interest rate (the discount rate). A positive project NPV normally indicates that the investment should be made, unless other projects have higher NPVs.

**RETURN ON INVESTMENT (ROI)**

A project’s expected return in percentage terms. ROI is calculated by dividing net benefits (benefits less costs) by costs.

**DISCOUNT RATE**

The interest rate used in cash flow analysis to take into account the time value of money. Organizations typically use discount rates between 8% and 16%.

**PAYBACK PERIOD**

The breakeven point for an investment. This is the point in time at which net benefits (benefits minus costs) equal initial investment or cost.
### Appendix B: Supplemental Material

**Related Forrester Research**


### Appendix C: Example Calculations For Waterfall Development

Software development teams following modern agile practices typically plan their work in two-week sprints and release at the end of every sprint. Teams following older waterfall development practices have release cycles that are much longer — usually consisting of several months — and these teams may only release finished code a handful of times per year. IT teams may prefer agile for a variety of reasons: Agile can result in faster software development and applications that better meet their users’ needs.

However, Forrester’s analysis of Topaz for Total Test does not depend on the type of software development methods a team is using. Forrester interviewed organizations with varying degrees of agile adoption — from fully embracing agile to still mostly waterfall — and found developers were consistently spending roughly 30% of their time testing mainframe code until they invested in Topaz for Total Test. After implementing Topaz for Total Test, all of the organizations realized similar benefits.

To help readers construct their own business cases for Topaz for Total Test, Forrester has included example calculation tables for an organization identical to the composite organization except that it uses waterfall rather than agile development practices. In these example calculations, the organization releases code every four months.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Metric</th>
<th>Calculation</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Number of developers using Topaz for Total Test</td>
<td>User input (incl. growth)</td>
<td>200</td>
<td>240</td>
<td>288</td>
</tr>
<tr>
<td>A2</td>
<td>Developer hourly rate</td>
<td>User input (incl. growth)</td>
<td>$68</td>
<td>$70</td>
<td>$72</td>
</tr>
<tr>
<td>A3</td>
<td>Percentage of overall development time saved due to Topaz for Total Test</td>
<td>Assumption (incl. growth)</td>
<td>5.0%</td>
<td>7.0%</td>
<td>9.0%</td>
</tr>
<tr>
<td>A4</td>
<td>Hours per developer per release</td>
<td>Assumption</td>
<td>520</td>
<td>520</td>
<td>520</td>
</tr>
<tr>
<td>A5</td>
<td>Releases per year</td>
<td>Assumption</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>A6</td>
<td>Hours saved annually</td>
<td>A3<em>A4</em>A5</td>
<td>104</td>
<td>146</td>
<td>187</td>
</tr>
<tr>
<td>A7</td>
<td>Percent recaptured</td>
<td>Assumption</td>
<td>80%</td>
<td>80%</td>
<td>80%</td>
</tr>
<tr>
<td>At</td>
<td>Incremental output per developer</td>
<td>A1<em>A2</em>A6*A7</td>
<td>$1,131,520</td>
<td>$1,962,240</td>
<td>$3,102,106</td>
</tr>
<tr>
<td></td>
<td>Risk adjustment</td>
<td>↓10%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atr</td>
<td>Incremental output per developer (risk-adjusted)</td>
<td></td>
<td>$1,018,368</td>
<td>$1,766,016</td>
<td>$2,791,895</td>
</tr>
</tbody>
</table>

**Three-year total:** $5,576,279  
**Three-year present value:** $4,482,899
Only the calculation tables for Incremental Output Per Developer and Direct Cost Avoidance: Testing Automation change. The other tables as well as the final results are the same.

### Appendix D: Endnotes

1. Topaz for Total Test supports common types of testing: unit testing, functional testing, system testing, integration testing, and regression testing. Although the differences between these types of tests are important for software developers, Forrester’s interviews with Topaz for Total Test customers did not reveal significant differences in terms of impacts on the business. All types of testing were equally onerous for developers, and the organizations benefited in similar ways from automating testing regardless of the type(s) of testing that they automated. Thus, the economic analysis in this study generally does not distinguish between testing types, except when those differences are significant.

In addition, prospective Topaz for Total Test customers may encounter the terms “virtualized testing” and “nonvirtualized testing”; these are terms that BMC Compuware uses to describe Topaz for Total Test’s features. Forrester interviewed Topaz for Total Test customers that were using only virtualized testing, only nonvirtualized testing, and both types of testing together. Again, Forrester found similar economic impacts regardless of the feature set(s) in Topaz for Total Test that the organizations used.

2. Total Economic Impact is a methodology developed by Forrester Research that enhances a company’s technology decision-making processes and assists vendors in communicating the value proposition of their
products and services to clients. The TEI methodology helps companies demonstrate, justify, and realize the tangible value of IT initiatives to both senior management and other key business stakeholders.

3 One interviewee reported that experienced developers wrote Topaz for Total Test test scripts twice as fast as newer developers. For example, he estimated that an experienced developer could write a complex script for unit testing in around 3 or 4 hours, whereas a newer developer would need a full day. However, no interviewee reported spending more than one day writing a single test, and all of the interviewees reported that time writing a Topaz for Total Test test script took about as much as time as running a manual test.