cognizant

Streamlining application modernization with cloud tech currency analysis

By Muthuramalingam Anbarasu,
Associate Director, AWS Business Group,
AWS MAP Delivery

In an era of relentless economic flux, organizations across various sectors are grappling with mounting cost pressures. This has led to a strategic reevaluation of investments, and a focus on adapting to a new normal shaped by evolving market needs and consumer expectations. Consequently, most businesses today are trying to curtail the total cost of ownership (TCO) associated with operations and maintenance.

The digital landscape is rapidly changing and businesses are evolving their application frameworks to stay resilient and responsive. This involves embracing digital transformation to drive predictable outcomes and spur innovation. However, this journey is fraught with challenges, including the proliferation of complex applications and outdated legacy systems that resist integration with modern architectures, leading to inflexibility, security vulnerabilities and compliance hurdles.

This condensed guide outlines a systematic approach to technology upgrade analysis and application modernization, providing actionable insights for navigating the digital transformation landscape effectively. The most common triggers for technology upgrades are listed in Figure 1.

- **End of support/end of life announced (EOS/EOL)** for standard/tech/ domain software
- **Infra complexities—hardware dependent** (e.g., 32-bit vs. 64 bit)
- **Client priorities:** cloud migration, architecture change, product replacement, etc.
- **Security and compliance, scalability and supportability**
- **Bug fixes, performance tuning, new features (digital strategy)**
- **Total cost of ownership** Operation and maintenance cost overhead

Figure 1

The tech currency analysis imperative

Modernizing applications is not just about keeping pace with technological advancements; it is also about unlocking future business value. By upgrading technologies and revamping legacy systems, organizations can scale operations and transition smoothly to cloud-based infrastructures. The key lies in maintaining technology currency by ensuring that software and hardware versions are up-to-date and supported by vendors, which minimizes technical debt and mitigates security risks.

Methodology and benefits

Our approach to application modernization hinges on a robust methodology that delivers substantial improvements in code quality (50%–70%) and optimization of modernization expenditures (10%–25%). By upgrading the technology currency, we aim to slash technical debts by 5%–10%.

Phased process for tech upgrade analysis

Many of the challenges within an application portfolio stem from technical and operational debts. These debts compromise both business agility and operational resilience, leading to an increased TCO. Consequently, technology upgrades are a critical prerequisite for any digital transformation initiative. Software versions that have reached their end of life (EOL) or end of support (EOS) are inefficient, often operating with obsolete security patches. Therefore, upgrading to the latest technology—referred to as tech currency upgrading—is essential for maintaining a risk-free application portfolio. Such upgrades can reduce technical debt by 10%–20%.

During the process of analyzing technology upgrades, we evaluate the tech currency (version of each application) and identify outdated versions (those that have reached their EOL or EOS) based on an analysis of each application within the tech stack. Subsequently, we provide recommendations—either to upgrade or modernize—specifically for applications deemed critical to the business or those with significant business potential, as shown in Figure 2.

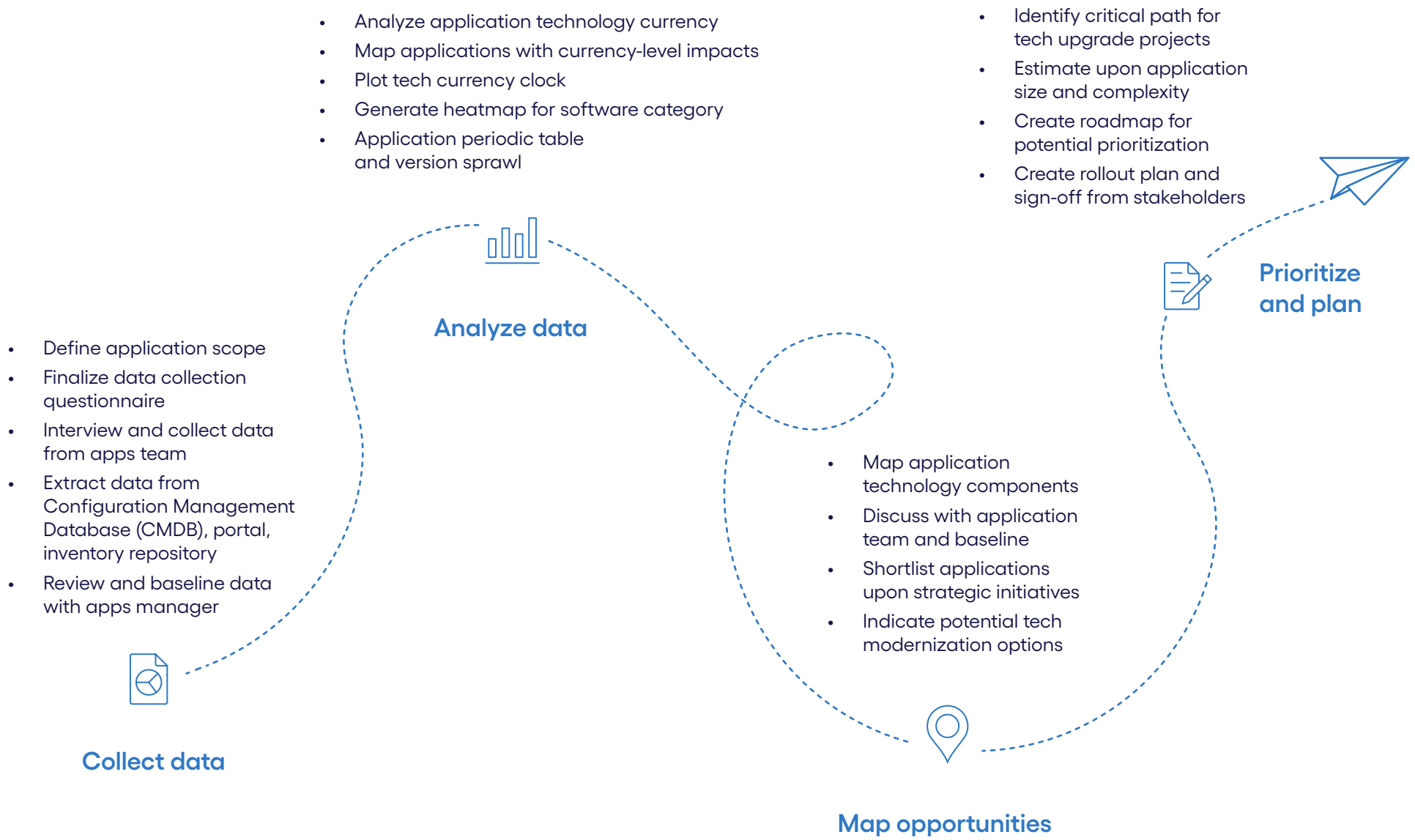

Analyze application technology currency
Map applications with currency-level impacts
Plot tech currency clock
Generate heatmap for software category
Application periodic table and version sprawl
Identify critical path for tech upgrade projects
Estimate upon application size and complexity
Create roadmap for potential prioritization
Create rollout plan and sign-off from stakeholders
Prioritize and plan
Analyze data
Define application scope
Finalize data collection questionnaire
Interview and collect data from apps team
Extract data from Configuration Management Database (CMDB), portal, inventory repository
Review and baseline data with apps manager
Map application technology components
Discuss with application team and baseline
Shortlist applications upon strategic initiatives
Indicate potential tech modernization options
Collect data
Map opportunities

Figure 2

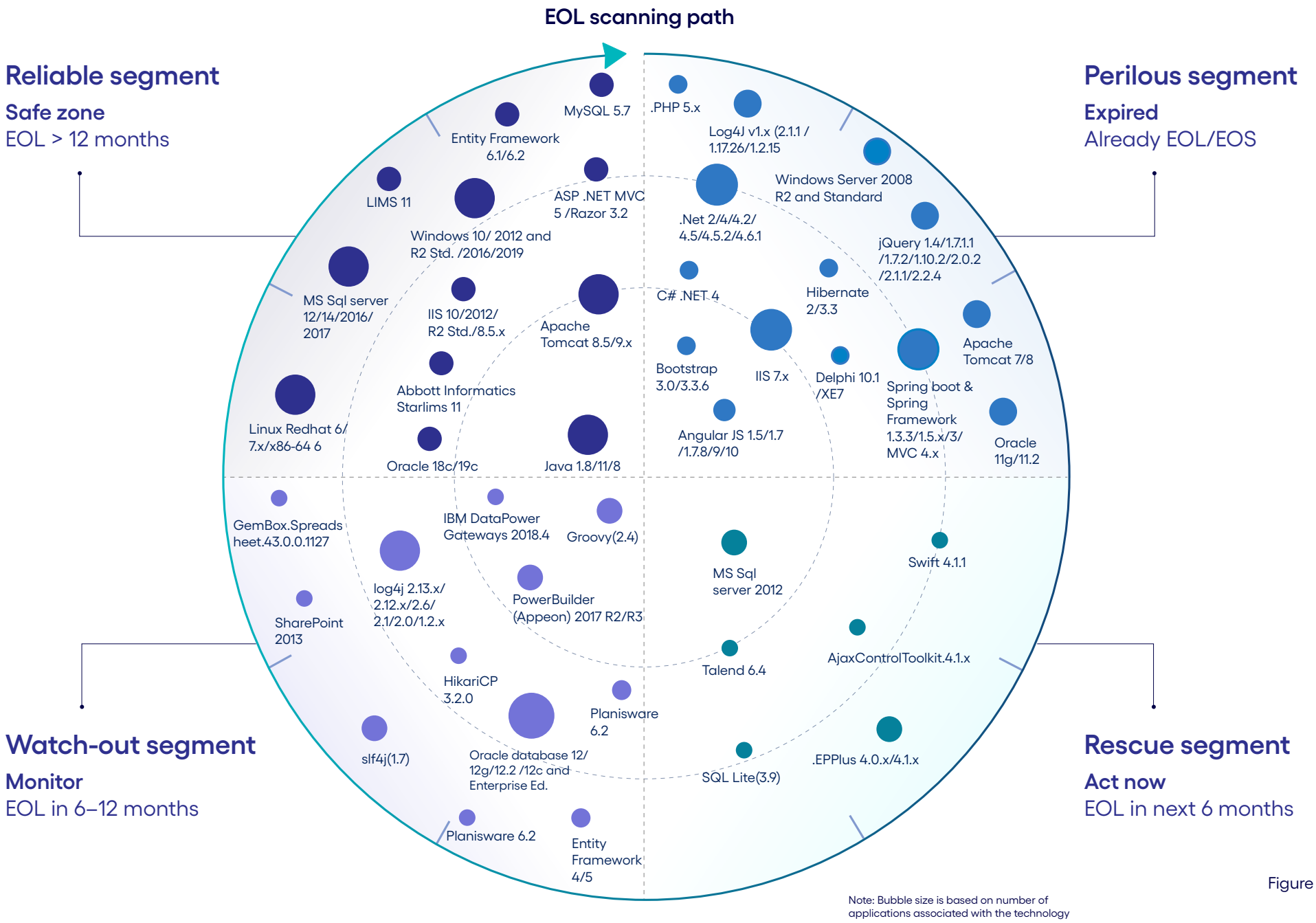
Data collection

Establish a baseline for the application scope and gather comprehensive tech stack data.

Data analysis

Conduct a detailed tech currency analysis to identify outdated versions and recommend upgrades or modernization for critical applications.

- Innovative techniques
  - **Tech currency clock/radar:** A visual tool that segments technology lifecycles into four categories, prompting timely upgrades, as shown in Figure 3.
  - **Technology heat map:** A strategic visualization that categorizes software lifecycles, highlighting areas that need immediate attention, as shown in Figure 4.
  - **Technology periodic table:** A snapshot of technological sprawl, aiding in decision-making for upgrades and rearchitecting efforts, as shown in Figures 5 and 6.



Note: Bubble size is based on number of applications associated with the technology

Figure 3

Technology life cycle	Continue to maintain	Declining	Attempt to eliminate
App. development (language/ framework)	**84 apps** • Java 1.8/11/8 • LIMS 11 • ASP .NET MVC 5/Razor 3.2 • Entity framework 6.1/6.2 • Log4j (2.17) • Maven 3.8 • Node.js 16.9 • Spring Boot (2.x) • Telerik.UI.for.AspNet. Mvc5.2018.1.131 • Unity.Mvc5.1.2.3 • AngularJS 13	**16 apps** • Oracle database 12/ 12g/12.2/12c and Enterprise Ed. • Log4j 2.13.x/2.12.x/2.6/2.1 /2.0/1.2.x • Groovy (2.4) • SLF4(1.7) • PowerBuilder (Appeon) 2017 R2/R3 • Entity Framework 4/5 • Planisware 6.2 • AjaxControlToolkit.4.1.x • Swift 4.1.1 • ClosedXML.0.76.0.0 • DocumentFormat.OpenXml 2.5 • Inspinia framework-2.5	**121 apps** • .Net 2/4/4.2/4.5/4.5.2/4.6.1 • Spring and Spring Boot (framework) 1.3.3/1.5.x/3/MVC 4.x • jQuery 1.4/1.7.1.1/1.7.2/1.10.2/2.0.2/2.1.1/2.2.4 • C# .NET 4 • Log4J v1.x (2.1.1 / 1.17.26 / 1.2.15 • Angular JS 1.5/1.7/1.7.8/9/10 • PHP 5.x • Bootstrap 3.0/3.3.6 • Delphi 10.1/XE7 • Hibernate 2/3.3 • ASP .NET MVC 4 • Html 5 • Oracle Apex(4)
Database	**76 apps** • MS Sql server 12/14/2016/2017 • Oracle 18c/19c • MySQL 5.7 • PostgreSQL 11.5 • SQL Anywhere 17	**46 apps** • MS Sql server 2012 • SQL Lite(3.9)	**13 apps** • Oracle 11g / 11.2 • MS Sql server 2008 • MySQL 5.6
Operating system and servers	**52 apps** • Windows 10/2012 & R2 Std./2016/2019 • Linux Redhat 6/7.x/x86-64 6 • Apache TomCat 8.5 /9.x • IIS 10/2012/R2 Std./8.5.x • Apache server 2.4 • Android 11 • iOS (15) • SharePoint 2019	**13 apps** • IBM DataPower Gateways 2018.4 • SharePoint 2013 • Tableau (2020.2.5)	**23 apps** • IIS 7.x • Windows Server 2008 R2 and Standard • Apache Tomcat 7 / 8 • Jasper report(5.6.0)
Third-party products (COTS)/vendor apps	**32 apps** • Abbott Informatics Starlims 11 • Molecular Devices MetaXpress 6.7 • Elysia-Raytest GINA 10.4 • GDM ARM 2021 • IDBS Activity Base 9.7 • KNIME KNIME 4.14.1 • Thermofisher Chromeleon 7.2 SR5 • Waters SDMS 9 • Xam.Plugin.DeviceInfo 4.0.0.13 • Xam.Plugin.Geolocator 4.5.0.6 • Xamarin.Forms.GoogleMaps 3.0.3	**8 apps** • EPPlus 4.0.x/4.1.x • Talend 6.4 • GemBox.Spreadsheet.43.0.0.1127 • Planisware 6.2 • HikariCP3.2.0 • json 20170516 • Super-CSV2.1.0 • Xam.Plugin.Media 4.0.1.5 • google-maps-services 0.2.6	**3 apps** • VM Workstation Pro (15)

Impacts

- Security vulnerabilities
- No service and support (EOL and EOS)
- Higher cost (extended licensing and support)
- Availability of skilled resources
- Incompatible and heterogeneous landscape
- Slower in time-to-market

Figure 4

Periodic table—before tech upgrade

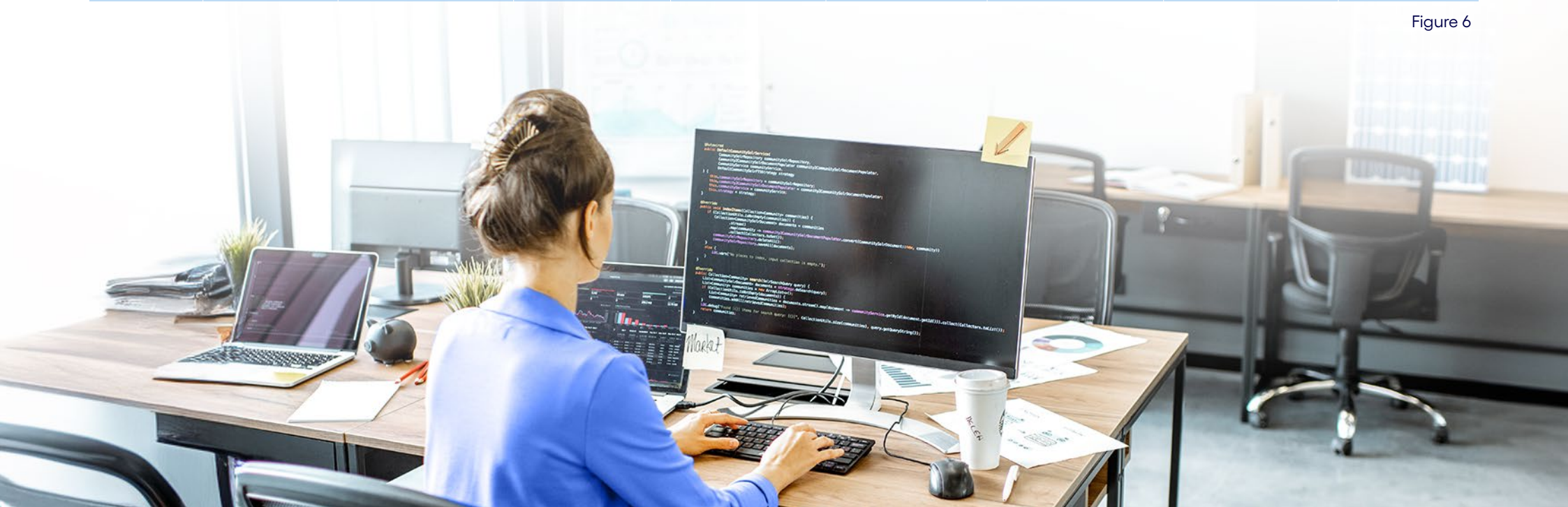
Primary tech/ programming language	Secondary tech/web/ scripting	Frameworks/ libraries	Logging frameworks/ utilities	Middleware/ peripheral tech/ platform	App and web servers	App and DB operating system	COTS product	Database
		ASP .NET MVC 4						MS Sql server 2008
		Hybernate 2						MySQL 5.6
	AngularJS v1.5.0	Hybernate 3.3						Oracle 11g
	AngularJS 1.7	ASP .NET MVC 4.0.3						Oracle DB 11.2.0.4
	AngularJS 1.7.8	Spring MVC 4						MS Sql server 2012
	Angular 9	Spring MVC 4.1.4						SQL Lite(3.9)
.NET 2	Angular 10	Spring 4						RHEL Oracle DB 12g
.NET 4	Bootstrap 3.0.0	Spring 3						Oracle 12
.NET 4.2	Bootstrap v3.3.6	Spring Boot 1.3.3			Apache TomCat 7			Oracle 12.2
.NET 4.5	jQuery 1.4	Spring Boot-1.5.8	Log4J v1.x		Apache TomCat 8			Oracle 12c
.NET 4.5.2	jQuery 1.7.1.1	Entity Framework 4	log4j(1.17.26)		IIS 7		Planisware 6.2	MS SQL server 2014
.NET 4.6.1	jQuery 1.7.2	Entity Framework 5	log4j(1.2.15)		Apache TomCat 8.5	Windows Server 2008 R2	Abbott Informatics Starlims 11	MS SQL server 2016
Delphi 10.1	jQuery 1.10.2	ASP .NET MVC 5	log4j(2.12.1)		Apache TomCat 9	Android 11	Elysia-Raytest GINA 10.4	MS SQL server 2017
C# .NET 4	jQuery 2.0.2	ASP .NET Razor 3.2	log4j(2.13.3)	Jasper report(5.6.0)	Apache TomCat 9.0.7	iOS 15	GDM ARM 2021	MySQL 5.7
PHP 5.5	jQuery v2.1.1	Entity Framework 6.1	log4j(2.6/2.1/1.2)	Delphi XE7	Apache Server 2.4	Red Hat Linux 6	IDBS Activity Base 9.7	MySQL 5.7.22
PHP 5.5.38	jQuery 2.2.4	Entity Framework 6.2	log4net(1.2.10.0)	PowerBuilder (Appeon) 2017 R2	IIS 10	Redhat Linux 7.x	KNIME KNIME 4.14.1	Oracle 18c
Java SWING	HTML 5	Maven 3.8	log4net(1.2.13)	PowerBuilder (Appeon) 2017 R3	IIS 10.0	Windows 10	Molecular Devices metaXpress 6.7	Oracle 19c
Swift 4.1.1	Groovy 2.4		log4net(1.2.15.0)	Tableu(2020.2.5)	IIS 2012	Windows Server 2012 and R2	Thermofisher Chromeleon 7.2 SR5	PostgresSQL 11.5
Java 1.8 / 8	AngularJS 13	Springboot(2.x)	log4net(2.0.5)	SharePoint 2013	IIS 2012 R2 Standard	Windows Server 2016	Waters SDMS 9	SQL Anywhere 17
Java 11	Node.js 16.9	Unity.Mvc5.1.2.3	log4j(2.17)	SharePoint 2019	IIS 8.5.9600	Windows Server 2019	LIMS 11	

Figure 5

Periodic table—after tech upgrade

							Planisware 7.x	
							Abbott Informatics Starlims 11	
							Elysia-Raytest GINA 10.4	
							GDM ARM 2021	
		ASP .NET MVC 5.2.7					IDBS Activity Base 9.7	SQL Lite 3.38.2
.NET 4.8	Bootstrap 5.x	Hybernate 5.6.0				Android 12	KNIME KNIME 4.14.1	MS SQL server 2019
PHP 8.1	jQuery 3.6.0	SpringBoot 2.6.6				iOS 15	Molecular Devices MetaXpress 6.7	MySQL 8
Swift 5.6	Groovy 4.0	Entity Framework 6.4.4		Jasper Report 8.0.x	Apache TomCat 10.x	Redhat Linux 7.x/8.x	Thermofisher Chromeleon 7.2 SR5	Oracle 19c
Java 8	AngularJS 13	Maven 3.8	log4j 2.x	Tableu 2022.1	Apache Server 2.4	Windows 11	Waters SDMS 9	PostgresSQL 14
Java 11	Nod.js 16.14	Unity.Mvc5.1.2.3	log4net 2.0.14	SharePoint 2019	IIS 10.0	Windows Server 2019	LIMS 11	SQL Anywhere 17
Primary tech/ programming language	Secondary tech/web/ scripting	Frameworks/ libraries	Logging frameworks/ utilities	Middleware/ peripheral tech/ platform	App and web servers	App and DB operating system	COTS product	Database

Figure 6



Mapping opportunities and prioritization

Identifying obsolete components and their impact on application health is crucial. By mapping technology components, we can pinpoint applications that require immediate attention for upgrades or replacements. This process helps prioritize applications based on their criticality, size and complexity to ensure focused actions for technology upgrades, as shown in Figure 7.

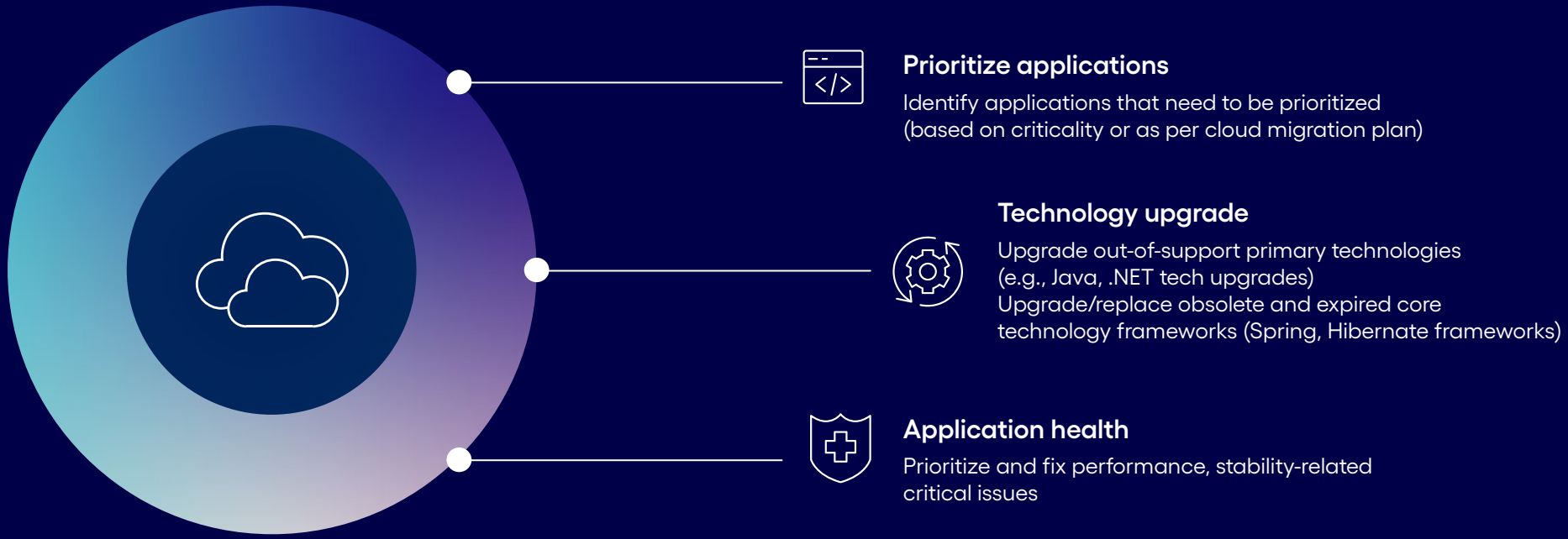


Figure 7

The business could also leverage other technology modernization options such as the examples listed:

- .NET apps—upgrade to .NET framework 4.8 and migrate to PaaS
- COTS apps—transform to SaaS or look for an alternate cloud hosting option
- Containerize/PaaSify applications with relevant app/web server
- Desktop apps—rearchitect through microservices with cloud-native features
- Oracle/SQL servers—migrate to managed database services on cloud such as PaaS DB services, etc.

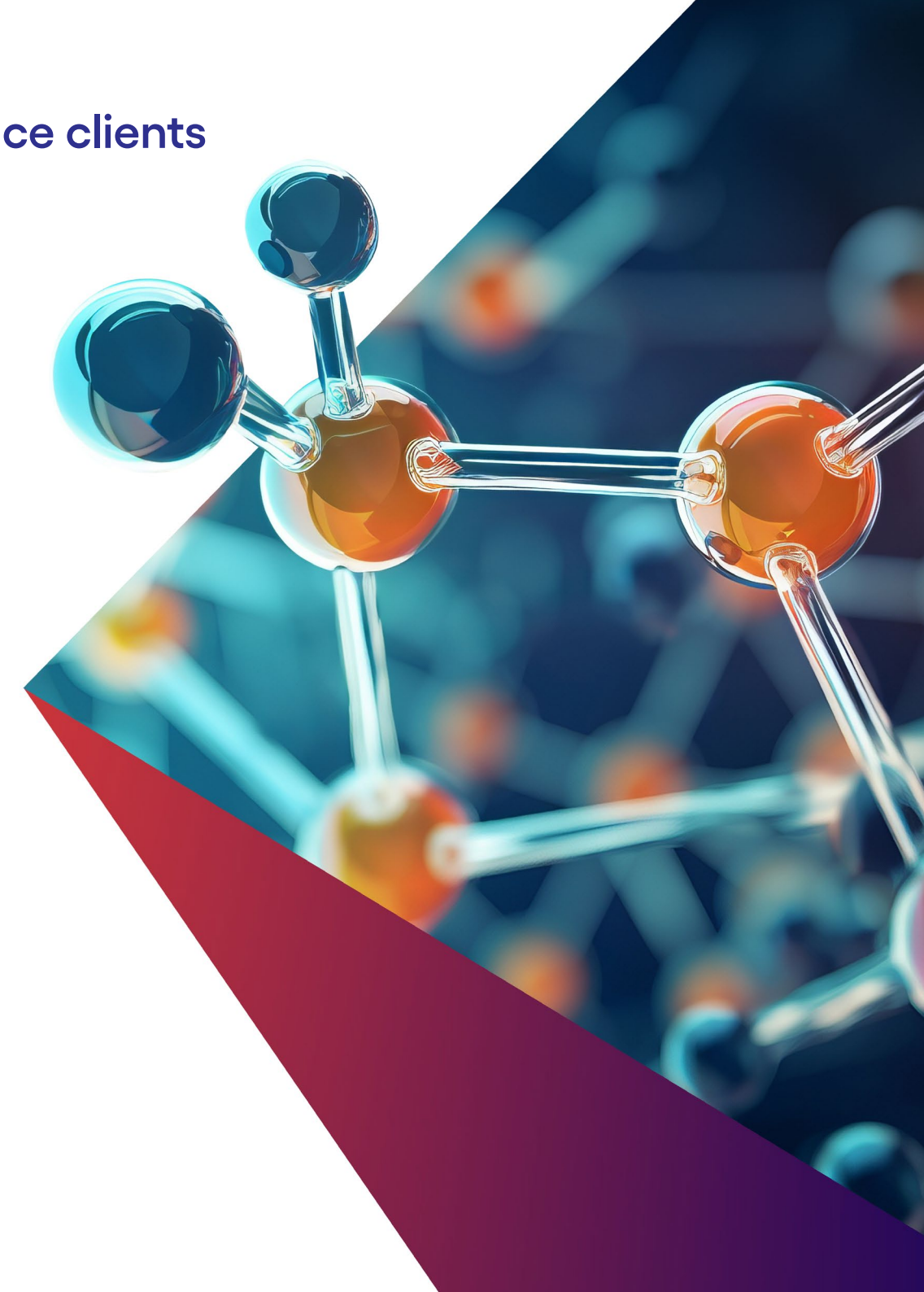
Success story for one of our life science clients

A leading German multinational company in the pharmaceutical sector faced significant challenges due to a complex IT landscape that made it difficult for them to respond quickly to dynamic business changes. Their objective was to:

- Eliminate/minimize the business service impact due to an outdated technology stack
- Simplify the technology stack and identify opportunities for app modernization
- Optimize IT costs through automation, rationalization and cloud adoption
- Improve time-to-market based on delivery

Overall, applications in scope totaled 95 within R&D, load balancer, local apps and portfolios. The application landscape consisted primarily of self-made (custom-built) web applications with 65% of applications (> 60) as .NET, Java and three apps built on PHP technology.

25% of applications (**24**), running on Citrix server	**66%** of applications (**63**) are hosted on Windows servers	**54%** of applications (**51**) connect to Oracle database
20% of application (**19**) are leveraging Log4j and Log4net logging frameworks	IIS and Tomcat are mostly used for web/ app servers	



The tech currency analysis discovered many hindrances using different techniques:

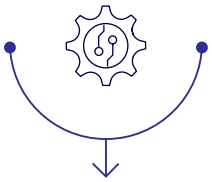
- 67% of applications are impacted by “expired” technologies
- There are legacy tech stack apps—Power Builder, Delphi, Java SWING, Applet apps, etc.
- Too many versions of technology components are used, which impact code maintainability, make it difficult to upgrade tech and cause tight coupling/dependency with other tech, as shown in the following:

.Net 2, 4, 4.2, 4.5, 4.5.2, 4.6.1
jQuery 1.4, 1.7.1.1, 1.7.2, 1.10.2, 2.0.2, 2.1.1, 2.2.4
Log4J v1.x (2.1.1, 1.17.26, 1.2.15)
AngularJS 1.5, 1.7, 1.7.8, 9, 10

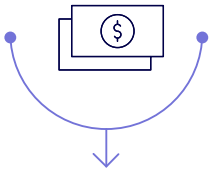
Cognizant’s tech currency analysis methodology offers organizations a significant advantage in identifying opportunities to replatform through tech upgrades that are in line with the strategic enterprise roadmap for cloud migration and modernization.

Our team’s intervention streamlined the company’s R&D portfolio and proposed to reduce the application tech stack components by 40%, thereby enhancing operational agility and reducing time-to-market.

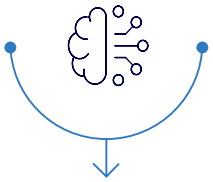
Benefits to the business



10–25% reduced modernization spend through tech upgrades



5–10% reduced technical debts through currency upgrades



50–70% improvement in code quality

Conclusion

As organizations pivot towards a future driven by advanced technologies, the need for a methodical approach to application modernization becomes paramount. By adopting the strategies outlined in this guide, businesses can ensure operational resilience, compliance and the ability to remain competitive in an ever-evolving digital arena.

About Cognizant

Cognizant (Nasdaq-100: CTSH) engineers modern businesses. We help our clients modernize technology, reimagine processes and transform experiences so they can stay ahead in our fast-changing world. Together, we're improving everyday life. See how at **www.cognizant.com**

World Headquarters

300 Frank W. Burr Blvd.
Suite 36, 6th Floor
Teaneck, NJ 07666 USA
Phone: +1 201 801 0233
Toll Free: +1 888 937 3277

European/UK Headquarters

280 Bishopsgate
London
EC2M 4RB
United Kingdom
Tel: +44 (0) 20 7297 7600

India Operations Headquarters

5/535, Okkiam Thoraipakkam,
Old Mahabalipuram Road,
Chennai 600 096
Tel: 1-800-208-6999
Fax: +91 (01) 44 4209 6060

APAC Headquarters

1 Fusionopolis Link,
Level 5 NEXUS@One-North,
North Tower,
Singapore 138542
Phone: +65 6812 4000

© Copyright 2024, Cognizant. All rights reserved. No part of this document may be reproduced, stored in a retrieval system, transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the express written permission of Cognizant. The information contained herein is subject to change without notice. All other trademarks mentioned here in are the property of their respective owners.

WF2799850