Quick Answer: When to Use Semantic Code Transformation to Modernize Applications

Published 29 November 2021 - ID G00760986 - 4 min read
By Analyst(s): Howard Dodd
Initiatives: Applications and Software Engineering Leaders

Software engineering leaders often struggle to decide when to use semantic code transformation to modernize applications. Here, we identify three modernization use cases that are best suited for semantic code transformation.

Quick Answer

When should we use semantic code transformation to modernize applications?

- When understanding and maintaining the application codebase is essential
- When changing the application structure to meet new requirements
- When addressing a decline in application knowledge and legacy development skills

More Detail

Use Case 1 – Understanding and Maintaining Critical Application Code

Some critical applications have functionality that cannot be easily replaced or bypassed. Software engineering leaders often find that an application’s original language, structure or supporting technologies create barriers to improving its functionality.

Semantic code transformation enables software engineering leaders to extend the life of critical applications by translating an existing codebase into another language. Semantic code transformation not only changes the language but also its structure, resulting in fundamental changes to the original source code. The translated code may require manual inspection to validate functional equivalence, but it will be readable, well-structured and maintainable. When used as a rearchitecting approach to modernization, semantic code transformation accelerates incremental improvement of specific components.
By contrast, syntactic code transformation produces a line-by-line translation between language syntaxes. This approach is fast and produces functionally equivalent code, but it often produces unmaintainable target code when transforming to a different language. Thus, software engineering leaders should only choose syntactic transformation when they do not intend to maintain the resulting transformed code. This approach can also be useful when supporting rapid migration to a new platform with no code change, or when updating the source codebase to support a newer version of a compiler or library.

Simply transforming poorly structured and documented legacy code to another language can transfer existing constraints, while also harboring new technical debt and increased complexity. To remediate these challenges, software engineering leaders must identify the right modernization opportunities where they can apply the right type of code transformations.

Figure 1 shows the different application outcomes produced by using syntactic versus semantic code transformation.

**Figure 1: Different Outcomes From Syntactic Versus Semantic Code Transformation**

<table>
<thead>
<tr>
<th>Syntactic Code Transformation</th>
<th>Semantic Code Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Application to New Language" /></td>
<td><img src="image2.png" alt="Application to New Structure" /></td>
</tr>
</tbody>
</table>

Source: Gartner

Use Case 2 — Changing the Application to Meet New Requirements

Application modernization should be a continuous, business-driven process, where software engineering leaders incrementally address constraints.

Semantic code transformation allows software engineering leaders to respond to business changes by:
Use Case 3 — Addressing a Decline in Application Knowledge and Legacy Development Skills

Applications are often built over a long period of time using incremental fixes and extensions, making them difficult to maintain and modernize — even under ideal conditions. As developers with legacy development skills leave the workforce, software engineering leaders will lose the developers’ knowledge of these core applications and technologies. This knowledge is nearly irreplaceable, as software engineering leaders face an immense shortage of talented people who have the skills and desire to work with aging technology.

In concert with a comprehensive talent strategy, semantic code transformation can help reduce the impact of losing legacy development expertise. Use semantic code transformation to:

- **Attract new developers who are motivated to learn, create and maintain modern code.** Use semantic code transformation to create structured code in a modern and popular language that developers are willing and motivated to learn and maintain.
- **Adopt modern software practices and tools.** The 3Q21 Gartner Global Labor Market Survey indicates that technology level — the extent to which the organization invests in modern technology and equipment — is a top-five attraction driver for IT talent. Use semantic code transformation to drive the adoption of modern software tools and incremental delivery processes.

- **Discover or document existing application knowledge.** When applied incrementally and with tempered expectations, semantic code transformation can expose and provide insight into dependencies and integration points. Use this new codebase to create important documentation and build knowledge of the application business logic.

**Recommended by the Authors**

- Choose the Right Approach to Modernize Your Legacy Systems
- Application Modernization Should Be Business-Centric, Continuous and Multiplatform
- How to Assess the Fitness of Your Application Portfolio
- Using TIME for Application and Product Portfolio Triage: Data From the Field

**Evidence**

1 Gartner’s 3Q21 Global Labor Market Survey was based on responses from 18,001 employees globally, including 1,905 employees in IT function, 352 of which were software engineering employees. Responses were collected monthly across 40 different countries in 15 languages and were then aggregated to generate quarterly findings. There are no statistically significant differences in the sample composition across the three months.