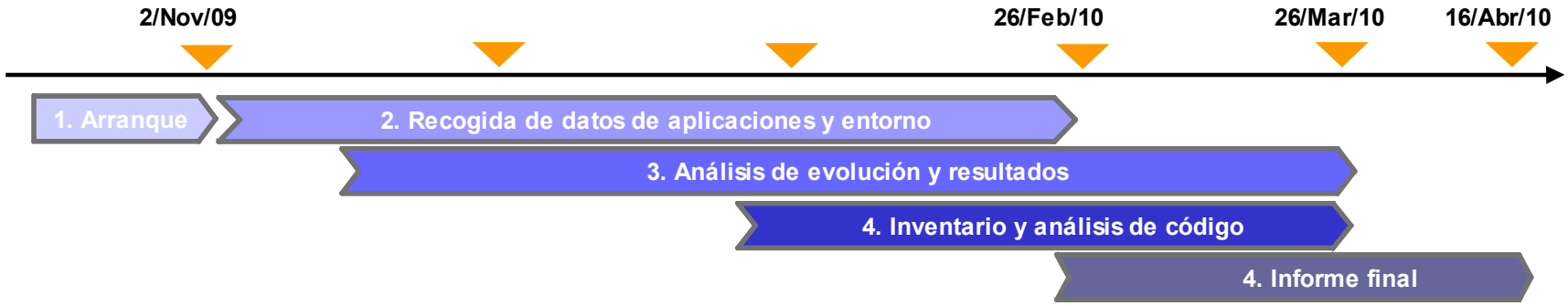
# Estudio para la evolución de los entornos aplicativos Natural/Adabas

***Mayo de 2010***

# Índice

# El estudio se ha estructurado como un trabajo consultivo fundamentado en una amplia recogida de información de situación actual y análisis de código



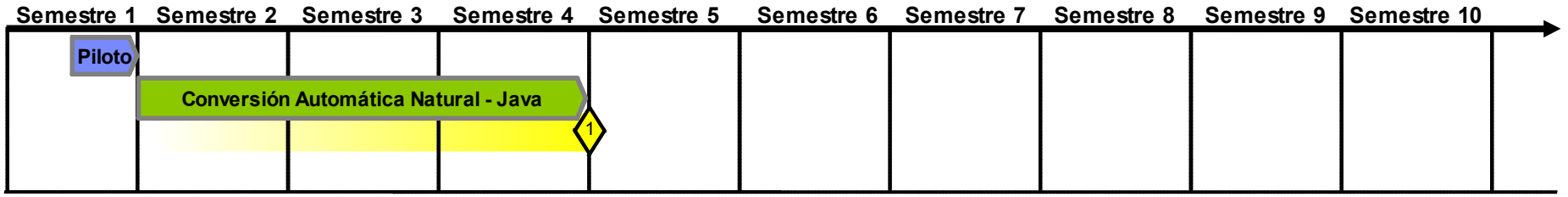

- Realizadas sesiones para determinación de los **requerimientos de evolución del cliente**, con los responsables de Desarrollo de aplicaciones y equipo de Arquitectura.
- La **recogida de datos** de aplicaciones, arquitectura y entornos se realizó en colaboración con los diversos equipos responsables del cliente: Desarrollo, Producción, Tecnología, Auditoría y Seguridad.
- El **análisis de código** se realizó en Febrero y Marzo, a dos niveles:
  - Análisis de código a nivel estadístico de las aplicaciones, con informes de volumetría, estructura, interdependencias e información de base para la estrategia de evolución.
  - Análisis muestral a nivel de sintaxis de código Natural y base de datos Adabas, para una transacción de negocio representativa.
- IBM comenzó tempranamente, y realizó en paralelo, las actividades de **análisis y preparación del informe** del estudio
  - Análisis de situación actual de las aplicaciones y su arquitectura, infraestructura técnica y entornos de desarrollo, preproducción y producción.
  - Determinación de requerimientos y principios para la evolución de la cartera de aplicaciones Natural/Adabas.
  - En base a requerimientos, definición de arquitecturas objetivo y de convivencia y determinación de componentes técnicos requeridos.
  - Análisis de posibles alternativas técnicas de conversión y modernización,
  - Análisis de estrategias de evolución e iniciativas requeridas para un programa de evolución.
  - Preparación de informe final y presentación ejecutiva.

# Índice

# Alternativas de evolución de las aplicaciones Natural/Adabas

| | |
|---|---|
| **1**<br>**Conversión Automática del Código Natural** | ▪ Conversión con herramienta automática del código Natural a Java.<br>▪ Trabajo por fases, atendiendo a criterios de negocio, criticidad e impacto de la conversión.<br>▪ Requiere la implementación de soluciones de convivencia entre componentes Natural y Java.<br>▪ Planeamiento, preparación y ejecución de pruebas de sistema e integración de sistemas; foco en pruebas de regresión e integraciones entre entorno actual y convertido.<br>▪ Realización de pruebas de aceptación funcional y técnica, especialmente con pruebas de rendimiento. |
| **2**<br>**Conversión Automática del código Natural y posterior migración de datos Adabas** | ▪ Incluye la totalidad de las actuaciones de la Alternativa 1.<br>▪ Una vez que el proceso de conversión de código a Java ha finalizado, y las aplicaciones convertidas corren en Producción, se realiza por fases la migración de los datos al Gestor Relacional (creación de objetos en DBMS, procesos de descarga y carga de datos, direccionamiento de las sentencias de acceso a datos hacia el nuevo Gestor).<br>▪ Realización de pruebas y puesta a punto de acceso a bases de datos, fundamentalmente por razones de rendimiento. |
| **3**<br>**Conversión Automática simultánea de Código Natural y datos Adabas** | ▪ La conversión de código y la migración de código se realizan de forma simultánea sobre un conjunto de aplicaciones vinculadas funcionalmente.<br>▪ Requiere la implementación de soluciones de convivencia entre componentes Natural y Java, y entre Adabas y el Gestor Relacional.<br>▪ Las pruebas requeridas son las de las Alternativas 1 y 2 ejecutadas simultáneamente. |
| **4**<br>**Modernización de Aplicaciones** | ▪ Rediseño de aplicaciones apoyado en herramientas que automatizan parcialmente el proceso.<br>▪ Los componentes de aplicación sujetos a modernización se determinan de acuerdo a las necesidades de implementación de servicios de negocio reutilizables identificadas en la definición de los procedimientos del cliente.<br>▪ La herramienta de modernización incluye plantillas adaptables para la generación de código Java.<br>▪ Los componentes no sujetos a modernización se convierten mediante herramientas de conversión automática. |

# *Alternativa 1:*
# Conversión automática del código Natural (1/2)




## Descripción

- Conversión automática de Natural a Java, manteniendo el gestor de base de datos Adabas.

## Consideraciones de planeamiento

- En esta alternativa se utiliza una herramienta de conversión automática Natural-Java.

## Riesgos a estudiar en el piloto

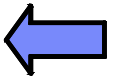
- Incertidumbre sobre la capacidad de evolución hacia un Gestor Relacional y sobre los esfuerzos requeridos para ello.
- No se identifican los potenciales errores de aplicación y los impactos sobre el rendimiento como consecuencia de un futuro cambio de Gestor de Base de Datos.
- La mantenibilidad del código no es adecuada para los equipos de Desarrollo del cliente.
- Potencial degradación del rendimiento de las aplicaciones, especialmente de los procesos batch.
- Rendimiento de la solución de convivencia a nivel de aplicaciones (invocaciones cruzadas).

## Riesgos de la alternativa que no se mitigan en el piloto

- Obsolescencia tecnológica de Adabas.

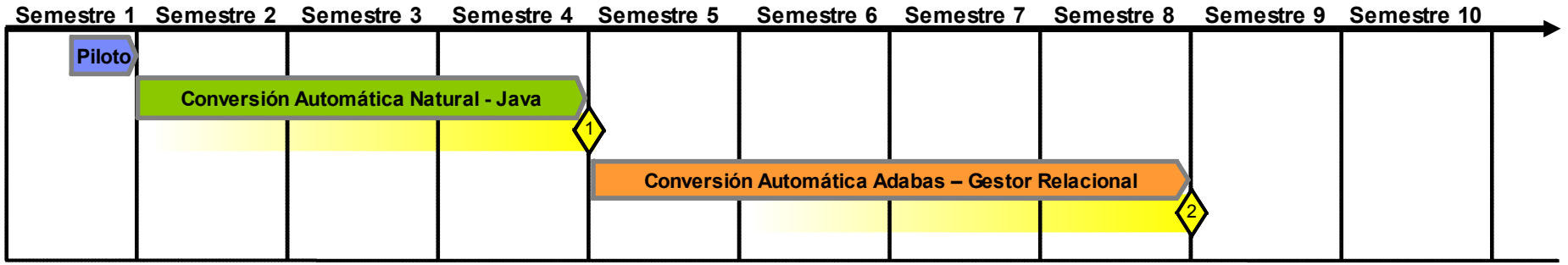
# *Alternativa 1:*
# Conversión automática del código Natural (2/2)

## Posicionamiento de Valor



- Ventajas
  - Máxima velocidad de evolución a entorno Java, con eliminación del riesgo tecnológico de la actual plataforma Natural
  - La conversión de aplicaciones se realiza con completa independencia respecto a los datos, lo que proporciona la máxima flexibilidad para establecer y modificar la secuencia y granularidad de migración de componentes de aplicación.
  - No son necesarias soluciones de convivencia entre distintos Gestores de Base de Datos, aunque si a nivel de aplicación.
  - Se eliminan costes de licencia Natural.
  - Se reducen los requerimientos de core processor en plataforma Z por desvío de las cargas Java a procesadores zAAP.
  - Al no realizar migración de datos las pruebas de funcionalidad y rendimiento requeridas son más ligeras.
  - Requerimientos de convivencia limitados a la interoperabilidad Natural/Java.
- Inconvenientes
  - Se impacta en la mantenibilidad de las aplicaciones frente a situación actual debido a las características del código migrado de forma automática mediante herramientas.
  - No se contempla una estrategia para eliminar el riesgo cierto de obsolescencia tecnológica del Adabas.
  - No se aborda la reestructuración de las aplicaciones ni de los datos: se mantiene la estructura actual de las aplicaciones, y no se adopta el framework corporativo sino el de la herramienta de conversión.
  - El modelo de ejecución del código migrado no es el habitual web Java EE, con impacto en el consumo de recursos.
  - De cara a una orientación a servicios y procesos, estimamos que solo el 45% del código convertido sería invocable desde las aplicaciones construidas sobre el framework corporativo, siempre que se ajustase a las necesidades de los procesos de negocio del cliente.
  - No se eliminan los costes de licencias de Adabas.

# *Alternativa 2 :*
# Conversión automática de código Natural y posterior de datos Adabas (1/3)




## Descripción

- Conversión automática de Natural a Java, seguida de la creación del modelo físico de datos en el Gestor Relacional, y migración de datos desde Adabas a Gestor Relacional.
- Durante la fase de migración de datos se realizan acciones de optimización necesarias para conseguir los siguientes objetivos:
  - Resolver los problemas de rendimiento causados por las diferencias entre el tipo de navegación de datos propio del gestor Adabas y el correspondiente a un Gestor Relacional.
  - Resolver los problemas de contención de accesos causados por las diferencias entre el sistema de bloqueo de datos propio del gestor Adabas y el sistema de bloqueo utilizado en un Gestor Relacional.
  - Aprovechar ventajas del Gestor Relacional cuando la ratio esfuerzo/beneficio así lo aconseje: capacidades de prefetching para el proceso batch, minimización de comunicaciones entre aplicaciones y Gestor BD usando capacidades avanzadas del SQL (acceso simultáneo a múltiples filas, acceso combinado a varias tablas, sentencias combinadas de lectura y actualización, etc.)

# *Alternativa 2 :*
# Conversión automática del código Natural y posterior de datos Adabas (2/3)

## Consideraciones de planeamiento

- Tanto la conversión de código como la migración de datos se realizan en varias fases, con despliegue en Producción al final de cada fase.
- En esta alternativa se utiliza una herramienta de conversión automática Natural-Java.

## Riesgos a estudiar en el piloto

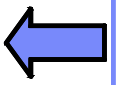
- Incertidumbre sobre la capacidad de evolución hacia un Gestor Relacional y sobre los esfuerzos requeridos para ello.
- No se identifican los potenciales errores de aplicación y los impactos sobre el rendimiento como consecuencia de un futuro cambio de Gestor de Base de Datos.
- La mantenibilidad del código no es adecuada para los equipos de Desarrollo del cliente.
- Potencial degradación del rendimiento de las aplicaciones, especialmente de los procesos batch.
- Rendimiento de la solución de convivencia a nivel de aplicaciones (invocaciones cruzadas).

## Riesgos de la alternativa que no se estudiarían en el piloto

- Obsolescencia tecnológica de Adabas.

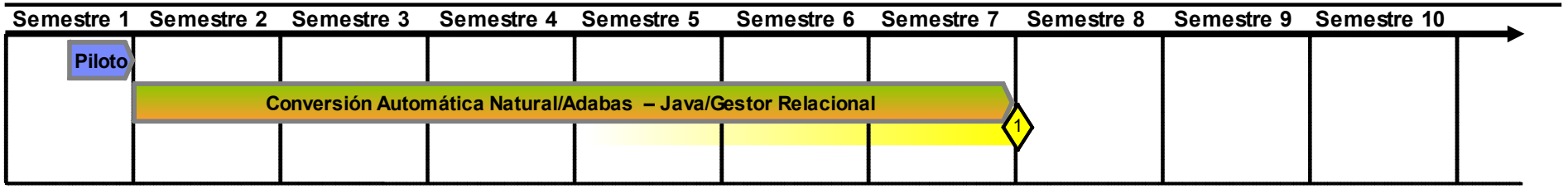
# Alternativa 2 :
# Conversión automática del código Natural y posterior de datos Adabas (3/3)

## Posicionamiento de Valor



- Ventajas
  - Máxima velocidad de evolución a entorno Java, con eliminación del riesgo tecnológico de la actual plataforma Natural y de la prima de coste de recursos de desarrollo.
  - La conversión del código de aplicación se realiza con completa independencia respecto a los datos, lo que proporciona la máxima flexibilidad para establecer y modificar la secuencia de migración de componentes de aplicación.
  - No son necesarias soluciones de convivencia entre distintos Gestores de Base de Datos, aunque si a nivel de aplicación.
  - Se eliminan costes de licencia Natural y Adabas.
  - Se reducen los requerimientos de core processor en plataforma Z por desvío de las cargas Java a procesadores zAAP.
  - Requerimientos de convivencia limitados a la interoperabilidad Natural/Java.
  - Completa eliminación del riesgo tecnológico de la actual plataforma Natural/Adabas.
- Inconvenientes
  - Se impacta en la mantenibilidad de las aplicaciones frente a situación actual debido a las características del código migrado de forma automática mediante herramientas.
  - Mayor duración del programa de evolución, frente a una migración conjunta de código y datos.
  - Las pruebas funcionales y de rendimiento realizadas en la primera fase, deberán repetirse en la segunda fase con una mayor cobertura.
  - No se aborda la reestructuración de las aplicaciones ni de los datos: se mantiene la estructura actual de las aplicaciones, y no se adopta el framework corporativo sino el de la herramienta de conversión.
  - El modelo de ejecución del código migrado no es el habitual web Java EE, con impacto en el consumo de recursos.
  - En a una orientación a servicios y procesos, estimamos que solo el 45% del código convertido sería invocable desde las aplicaciones construidas sobre el framework corporativo, siempre que se ajustase a las necesidades de los procesos de negocio del cliente.

# Alternativa 3 :
# Conversión automática simultánea del código Natural y datos Adabas (1/2)




## Descripción

- La conversión de lenguaje Natural a Java y la migración de base de datos Adabas a Gestor Relacional se realizan simultáneamente.

## Consideraciones de planeamiento

- Conversión de código y migración de datos se realizan en varias fases, con despliegue en Producción al final de cada fase.
- Coexistencia del gestor de base de datos Adabas y el Gestor Relacional durante la duración del proyecto.
- En esta alternativa es recomendable la utilización de herramientas de migración específicamente orientadas a la conversión simultánea de código y datos (Rational Migration Extension).

## Riesgos a estudiar en el piloto

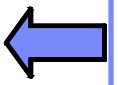
- La mantenibilidad del código no es adecuada para los equipos de Desarrollo del cliente.
- El rendimiento de las aplicaciones, especialmente de los procesos batch, se degrada.
- Rendimiento de la solución de convivencia a nivel de aplicaciones (invocaciones cruzadas) y de datos.

## Riesgos de la alternativa que no se estudiarían en el piloto

- Riesgo operativo de mantenimiento de las réplicas de datos durante la conversión.

# Alternativa 3 :
# Conversión automática simultánea del código Natural y datos Adabas (2/2)

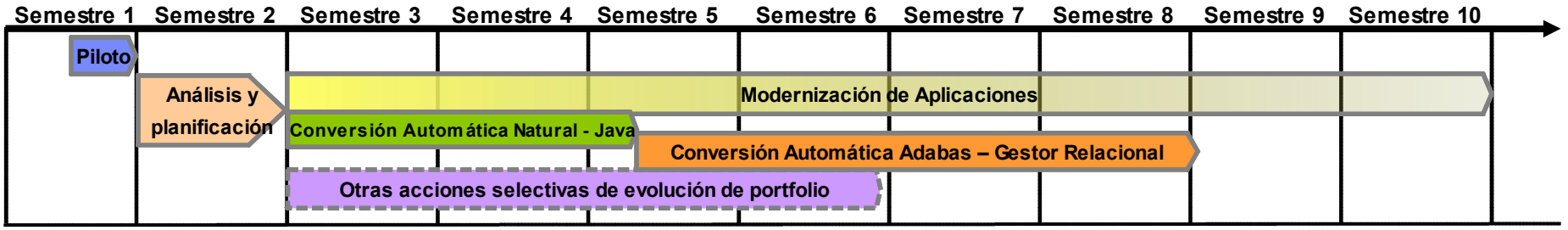
## Posicionamiento de Valor



- Ventajas
  - Máxima velocidad de evolución al entorno destino Java/Gestor Relacional.
  - Se requiere un único ciclo de pruebas para la validación de la migración.
  - Completa eliminación del riesgo tecnológico de la actual plataforma Natural/Adabas.
  - Se eliminan costes de licencias Natural y Adabas.
  - Se reducen los requerimientos de core processor en plataforma Z por desvío de las cargas Java a procesadores zAAP.
  - Al final del proyecto se elimina el riesgo de obsolescencia tecnológica.

- Inconvenientes
  - Se requiere una solución para la convivencia entre distintos Gestores de Base de Datos.
  - Fuerte interdependencia entre la conversión de código y la migración de los datos.
  - En función de la herramienta de conversión elegida, se impacta en la mantenibilidad de las aplicaciones frente a situación actual debido a las características del código migrado de forma automática.
  - No se aborda la reestructuración de las aplicaciones ni de los datos: se mantiene la estructura actual de las aplicaciones, y no se adopta el framework corporativo sino el de la herramienta de conversión.
  - El modelo de ejecución del código migrado no es el habitual web Java EE, con impacto en el consumo de recursos.
  - En a una orientación a servicios y procesos, estimamos que solo el 45% del código convertido sería invocable desde las aplicaciones construidas sobre el framework corporativo, siempre que se ajustase a las necesidades de los procesos de negocio del cliente.

# *Alternativa 4:*
# Modernización de Aplicaciones (1/3)




## Descripción

- Rediseño selectivo en Java/DB2 de aplicaciones Natural/Adabas, orientado a facilitar la implementación en el framework corporativo de los procedimientos y servicios de negocio requeridos.
- Este rediseño se complementa con la migración automática de aplicaciones y componentes no requeridos para la implementación de los procedimientos de negocio y con otras acciones de evolución selectiva de objetos Natural (p.ej. rediseño de los procesos batch críticos para la optimización de su implementación en Java).

## Consideraciones de planeamiento

- El alcance y acciones de modernización se determinará a partir del análisis completo de los procesos de negocio y los procedimientos del cliente y las necesidades de implementación de Servicios de Negocio reutilizables.
- Modernización selectiva de aplicaciones:
  - Modernización selectiva de aplicaciones on line para adecuarlas a la implementación de nuevos procedimientos de negocio.
  - Modernización del batch crítico medio / pesado, orientando su rediseño a la maximización de la paralelización de su ejecución.
- Rediseño apoyado en herramienta con capacidad de análisis semántico de las aplicaciones Natural, y con capacidades de soporte al diseño y construcción de la aplicación modernizada.
- La conversión automática de código Natural a Java se aplica sólo a los componentes que no son objeto de rediseño.
- Incorporación de acciones selectivas complementarias de evolución de portfolio,
- En esta alternativa son aplicables las herramientas de conversión automática, así como las de TLM de Make Technbologies.

# *Alternativa 4:*
# Modernización de Aplicaciones (2/3)

## Riesgos a estudiar en el piloto

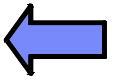
- La mantenibilidad del código de las aplicaciones convertidas automáticamente no es adecuado para los equipos de Desarrollo del cliente. Riesgo minimizado por afectar solo a parte de las aplicaciones.
- El rendimiento de las aplicaciones convertidas automáticamente, especialmente de los procesos batch, se degrada. Riesgo minimizado por afectar solo a parte de las aplicaciones.
- Hay que gestionar la convivencia a nivel de aplicaciones (invocaciones cruzadas) y de datos.

## Riesgos de la alternativa que no se estudiarían en el piloto

- Cambios introducidos en las aplicaciones y datos origen durante el transcurso del proceso de modernización pueden tener que ser trasladados manualmente a la aplicación y datos rediseñados.
- Cambios en la definición de los procedimientos y sus actividades obligan a modificar los rediseños en curso.

# *Alternativa 4:*
# Modernización de Aplicaciones (3/3)

## Posicionamiento de Valor



- Ventajas
  - Máxima velocidad de evolución a la arquitectura objetivo.
  - De cara a una orientación a servicios y procesos, se podrían exponer como servicios y/o invocar desde procesos todo el código que se haya identificado como candidato a tal fin, estando este, además, ajustado a las necesidades de los procedimientos del cliente.
  - Completa eliminación del riesgo tecnológico de la actual plataforma Natural/Adabas.
  - Se eliminan costes de licencias Natural y Adabas.
  - Se reducen los requerimientos de core processor en plataforma Z por desvío de las cargas Java a procesadores zAAP.

- Inconvenientes
  - Máxima complejidad del proyecto.
  - El personal del cliente que participe en el proyecto deberá formarse en las herramientas de modernización.
  - Se requiere una solución para la convivencia entre distintos Gestores de Base de Datos.
  - Para la parte convertida, en función de la herramienta de conversión elegida, se impacta en la mantenibilidad de las aplicaciones frente a situación actual debido a las características del código migrado de forma automática.
  - Para la parte convertida, el modelo de ejecución del código migrado no es el habitual web Java EE, con impacto en el consumo de recursos.

# Análisis de estrategias para el programa de evolución

| Id | Denominación | Menor duración del programa de evolución | Menor tiempo a primeros beneficios | Menor riesgo de obsolescencia | Menor riesgo operacional del programa | Mejor mantenibilidad | Mejor eficiencia | Mejor aproximación a la arquitectura objetivo | Mejor eficiencia en costes de desarrollo |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Conversión automática de código Natural | 4 | 4 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | Conversión completa de aplicaciones Natural/ Adabas | 2 | 4 | 3 | 4 | 1 | 2 | 2 | 1 |
| 3 | Ejecución simultánea de Conversión de Código y Migración de B.D. | 3 | 3 | 4 | 3 | 1 | 2 | 2 | 1 |
| 4 | Modernización de Aplicaciones | 1 | 2 | 2 | 2 | 4 | 4 | 4 | 3 |

Leyenda: 0 1 2 3 4 *Valoración*

Aunque las alternativas de evolución tienen importantes diferencias en cuanto al nivel de transformación de las aplicaciones, y por tanto implica comparar resultados muy diversos, a efectos de posicionamiento relativo se han establecido los **criterios** siguientes

- ***Menor duración del programa de evolución****:* tiempo requerido para concluir los trabajos definidos en la alternativa
- ***Menor tiempo a primeros beneficios***: tiempo requerido para puesta en producción de la solución convertida
- ***Menor riesgo de obsolescencia****:* potencial impacto de posible obsolescencia tecnológica de la plataforma actual en el periodo planeado
- ***Menor riesgo operacional del programa****:* potencial impacto de las actuaciones de evolución sobre el servicio actual y futuro
- ***Mejor mantenibilidad****:* facilidad para mantenimiento del código convertido, para incorporación de cambios evolutivos, normativos o correctivos
- ***Mejor eficiencia****:* según rendimiento y nivel de servicio en ejecución del código convertido
- ***Mejor aproximación a la Arquitectura objetivo****:* nivel de cercanía a la arquitectura objetivo del código convertido
- ***Mejor eficiencia en costes del desarrollo****:* costes involucrados en nuevos desarrollos para las aplicaciones convertidas o relacionados con ellas

***Fin del documento***