Cómo desarrollar y probar aplicaciones mainframe sin un mainframe

Alejandro León
Especialista en Soluciones de Modernización de la Empresa

Cuatro barreras clave impiden un óptimo retorno de la inversión

Décadas de inversiones en aplicaciones

"No entendemos el esfuerzo, el riesgo y el impacto de modernizar nuestras aplicaciones 'legacy'"

Islas de skills, lenguajes y plataformas

"Nuestros skills gaps siguen creciendo. ¿Cómo podemos permanecer actualizados de todos los cambios en los lenguajes y en las tecnologías?"

Equipos mal integrados

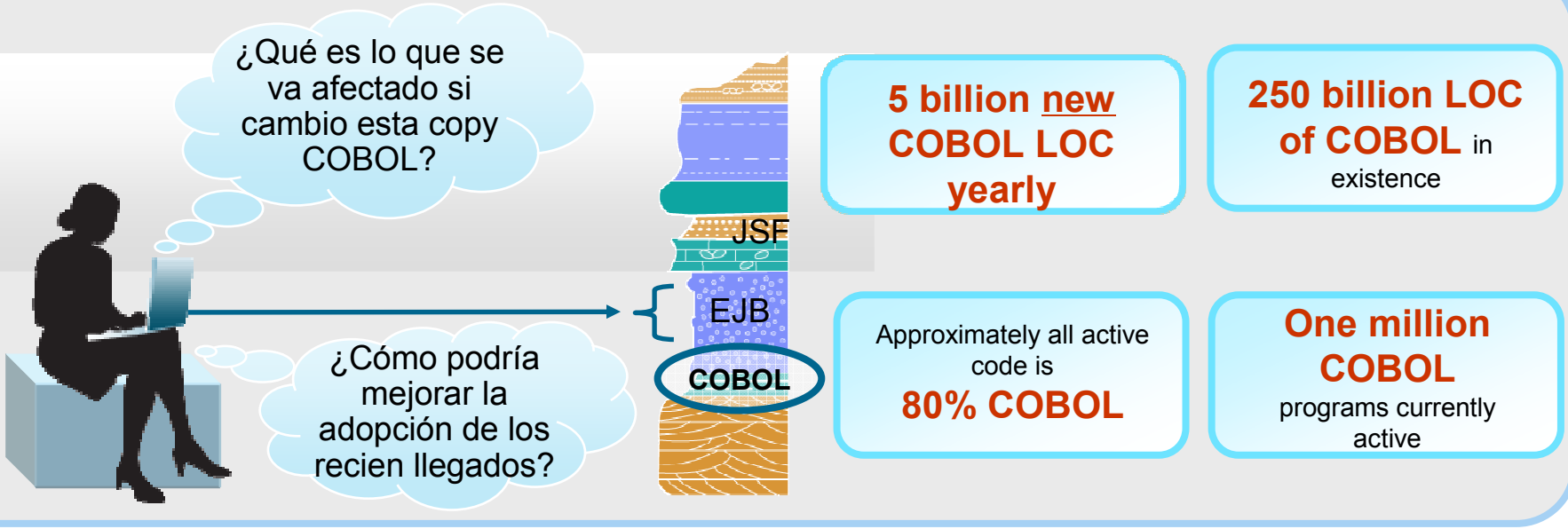
"Necesitamos que nuestros equipos colaboren a través de las distintas plataformas, lenguajes y entornos."

Ineficiencia de la Infraestructura

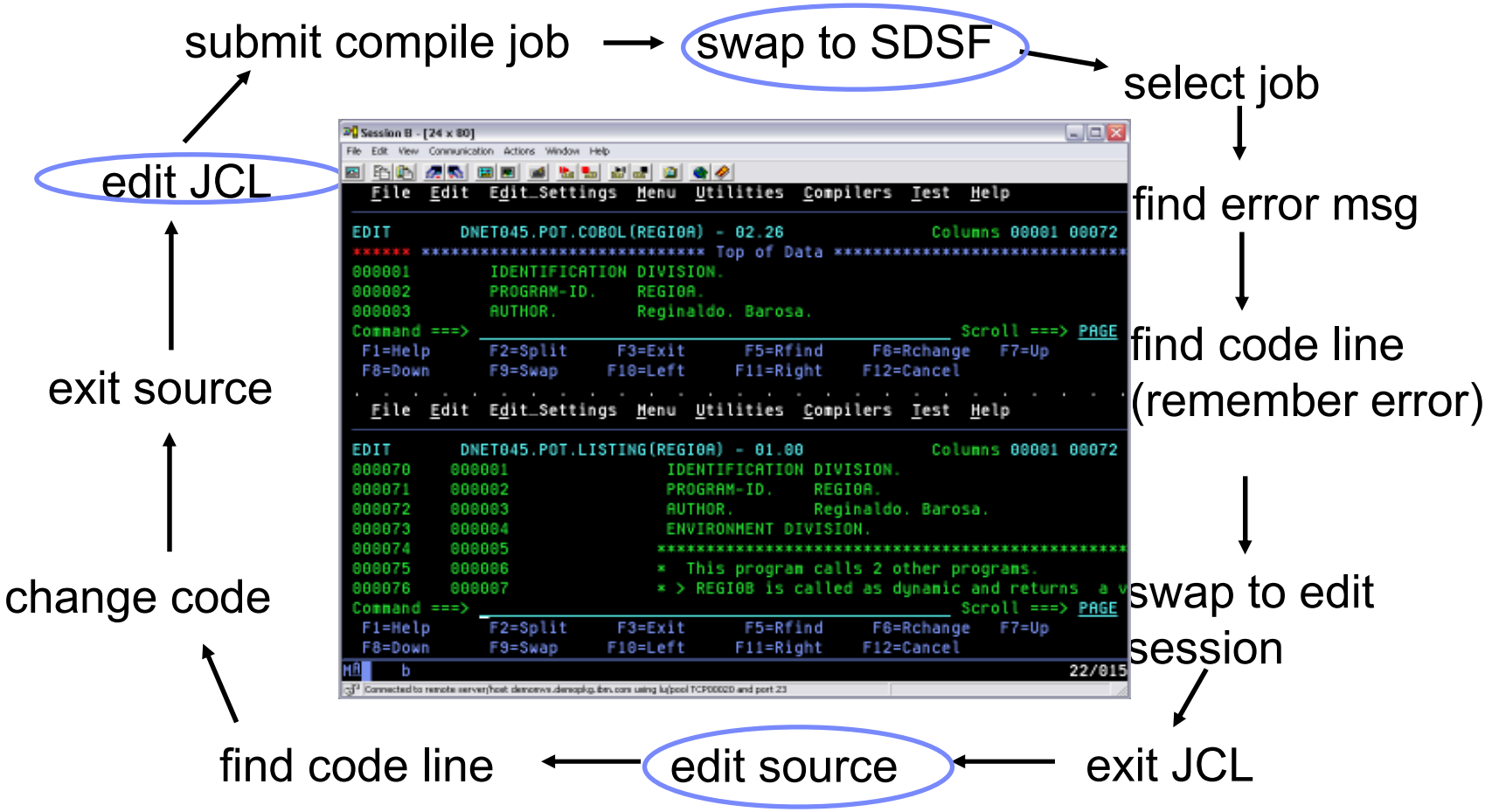
"Necesitamos alguna manera de mejorar nuestra eficiencia y liberar capacidad para poder tener más potencia."

Las compañías quieren…

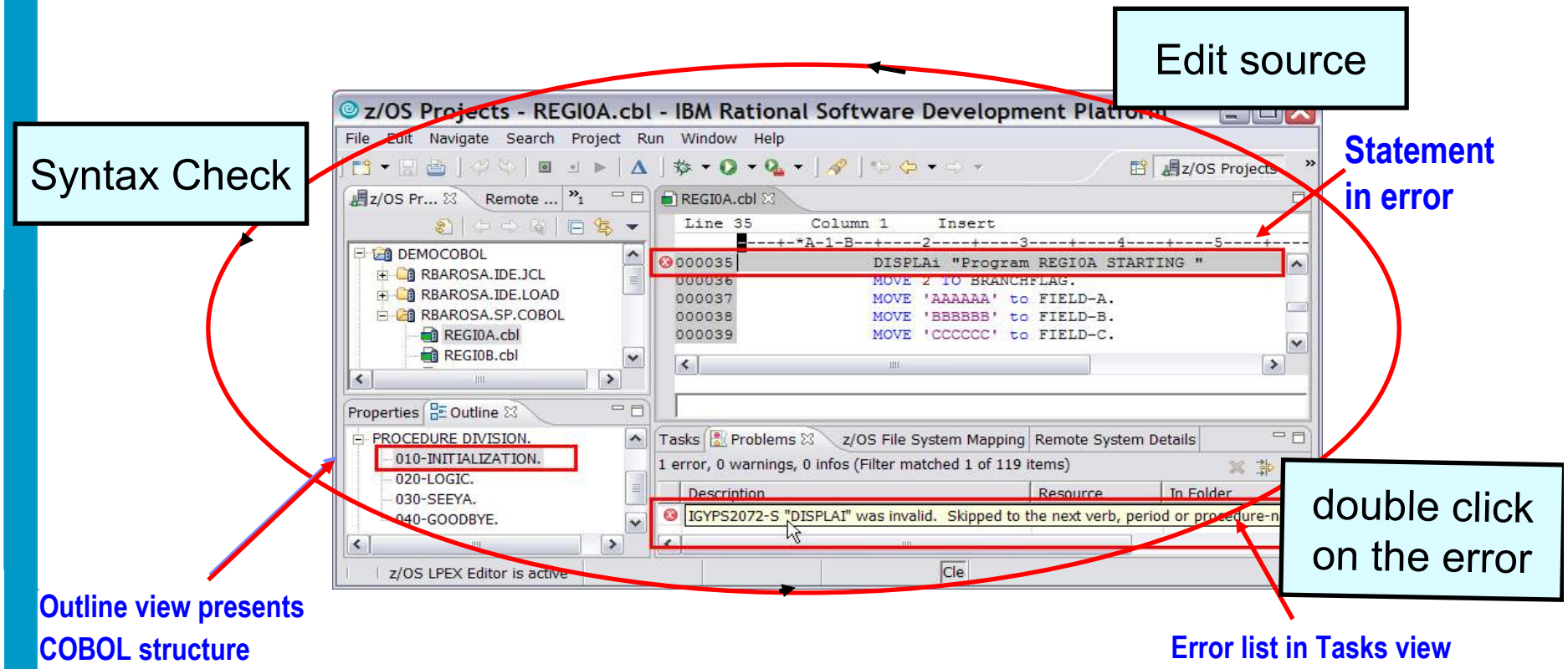
Comprender el impacto y coste de los cambios en sus sistemas de TI.
Unificar, simplificar plataformas y entornos. Mejorar la mantenibilidad.
Hacer que diferentes equipos dispersos geográficamente colaboren.



Desarrollo basado en ISPF



Desarrollo basado en Eclipse



Beneficios: Simplicación del desarrollo en COBOL, PL/I, C/C++ y Java en un mismo entorno de desarrollo.

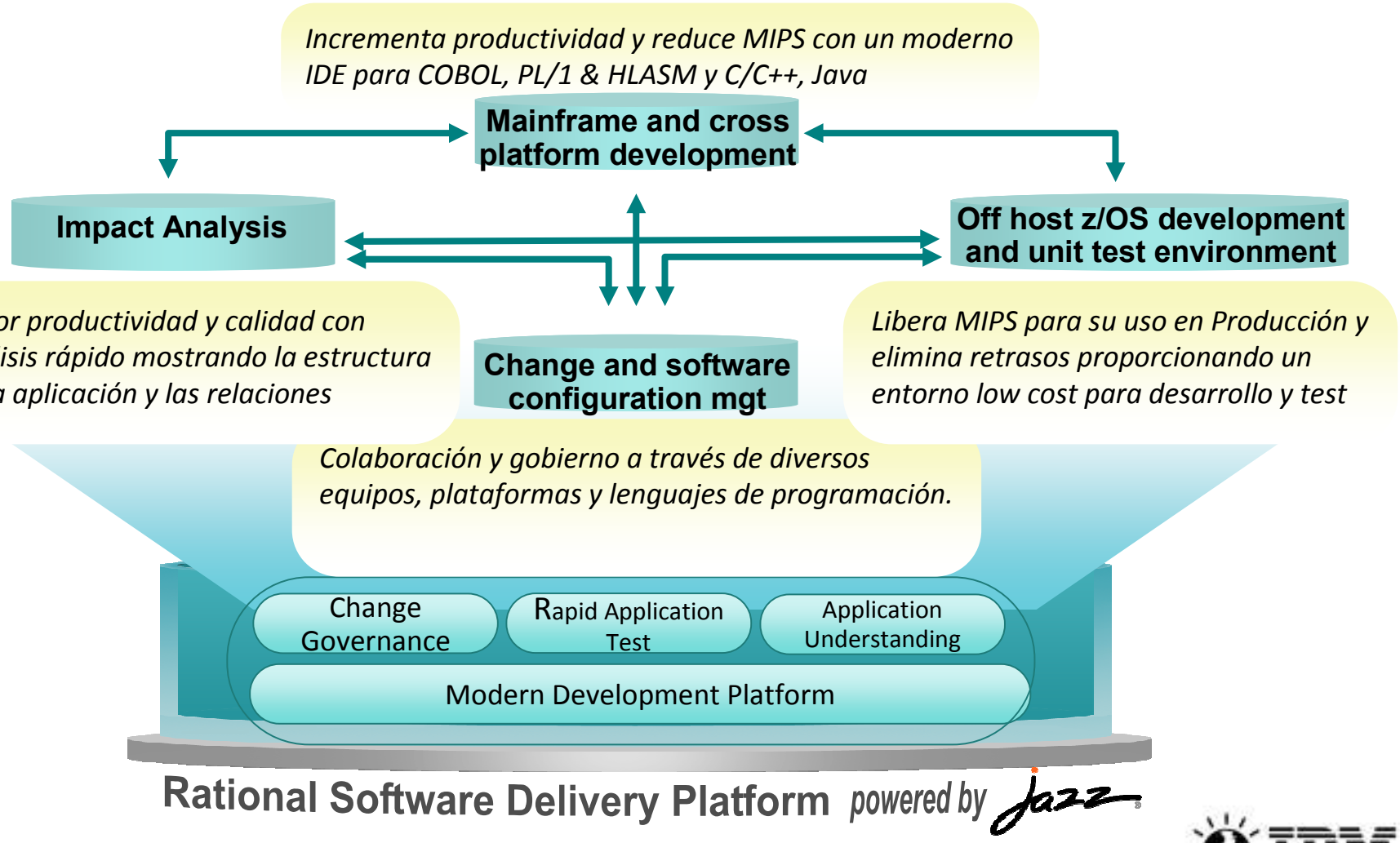
Solución integrada para el desarrollo de System z

Incrementa productividad y reduce MIPS con un moderno IDE para COBOL, PL/1 & HLASM y C/C++, Java

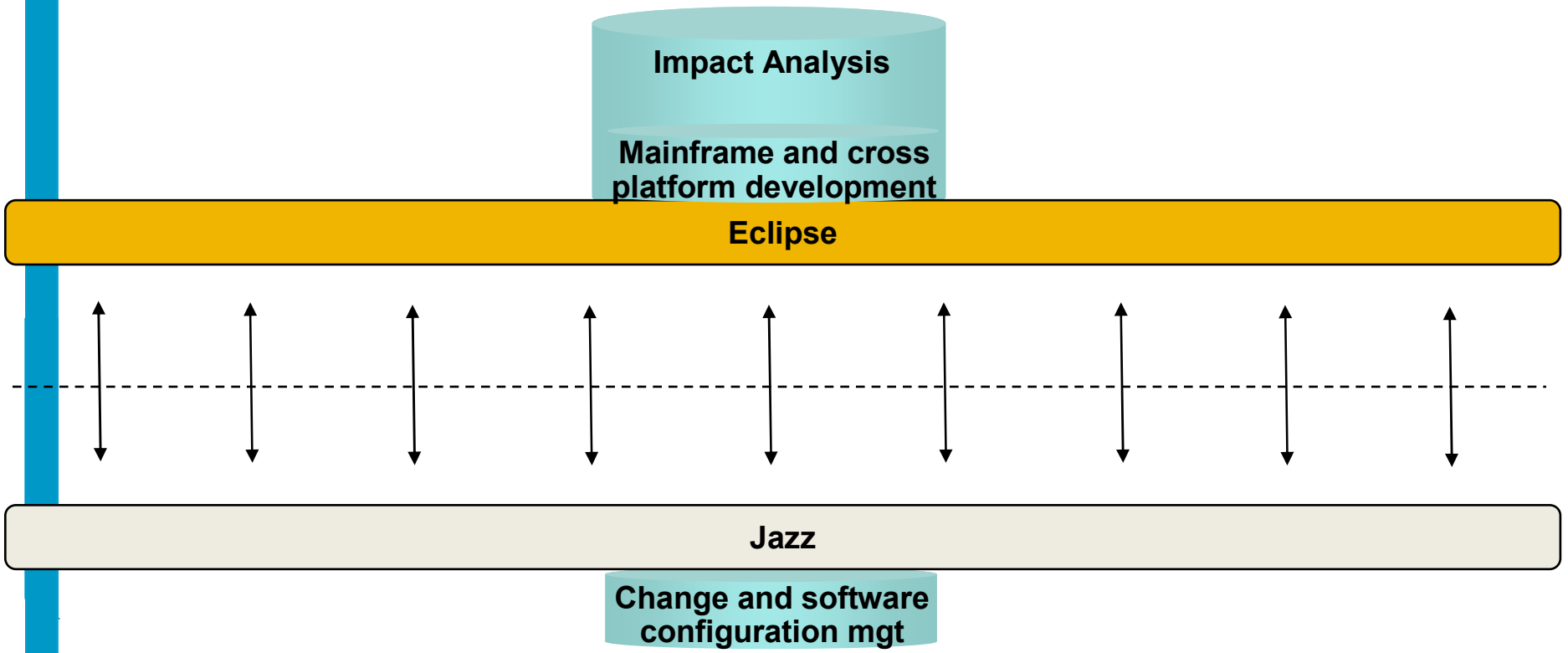
Mejor productividad y calidad con análisis rápido mostrando la estructura de la aplicación y las relaciones

Libera MIPS para su uso en Producción y elimina retrasos proporcionando un entorno low cost para desarrollo y test

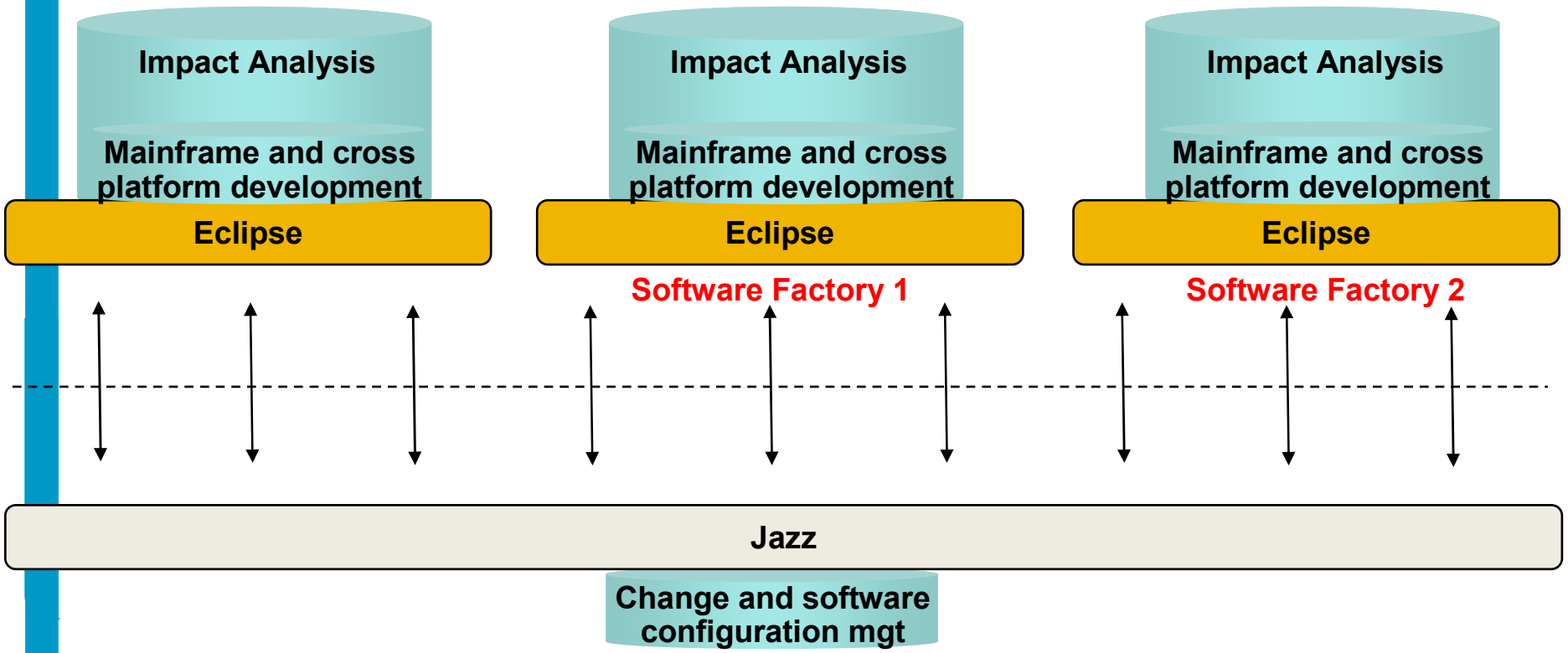
Solución integrada para el desarrollo de System z



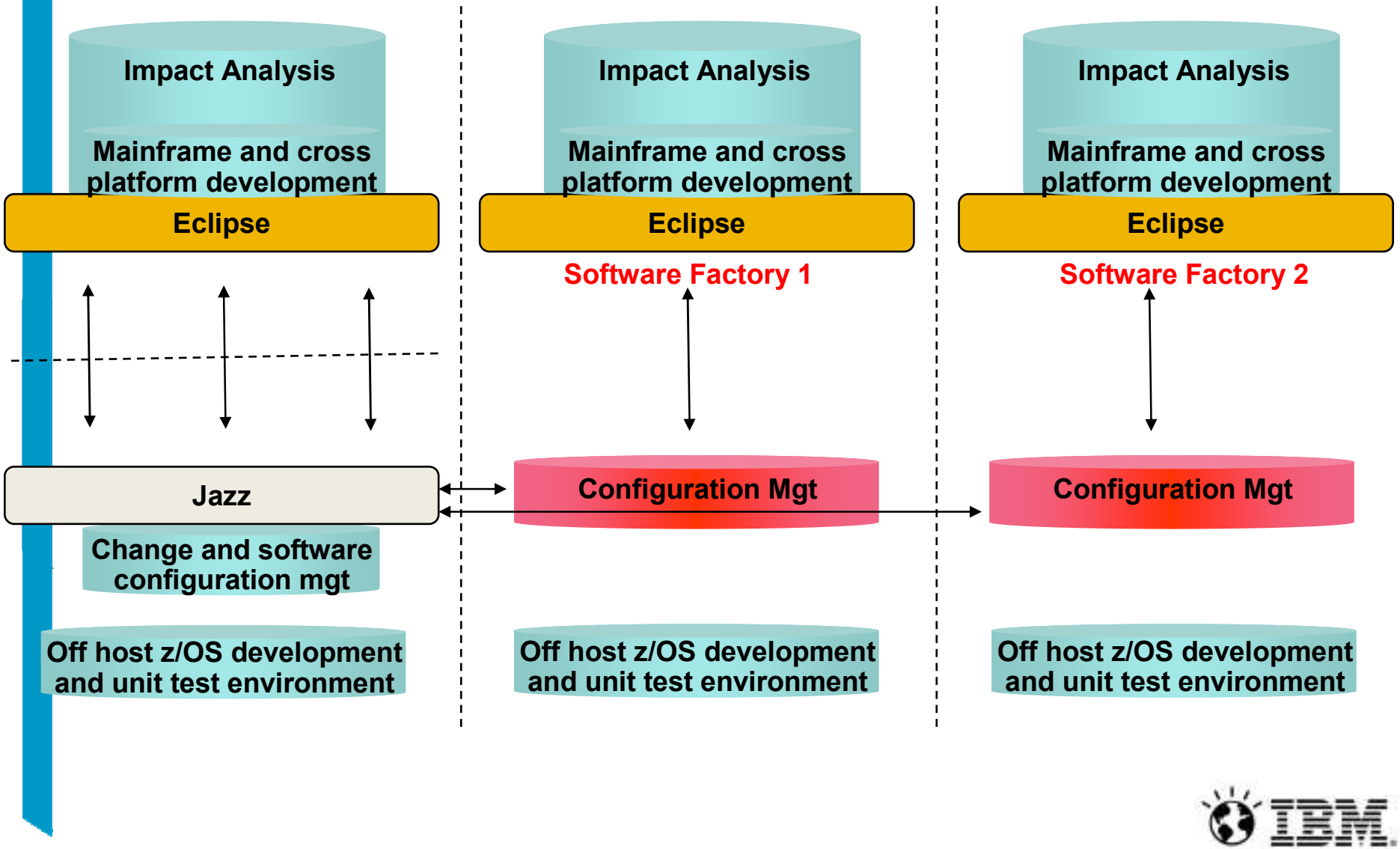
Arquitectura de la Solución (I)



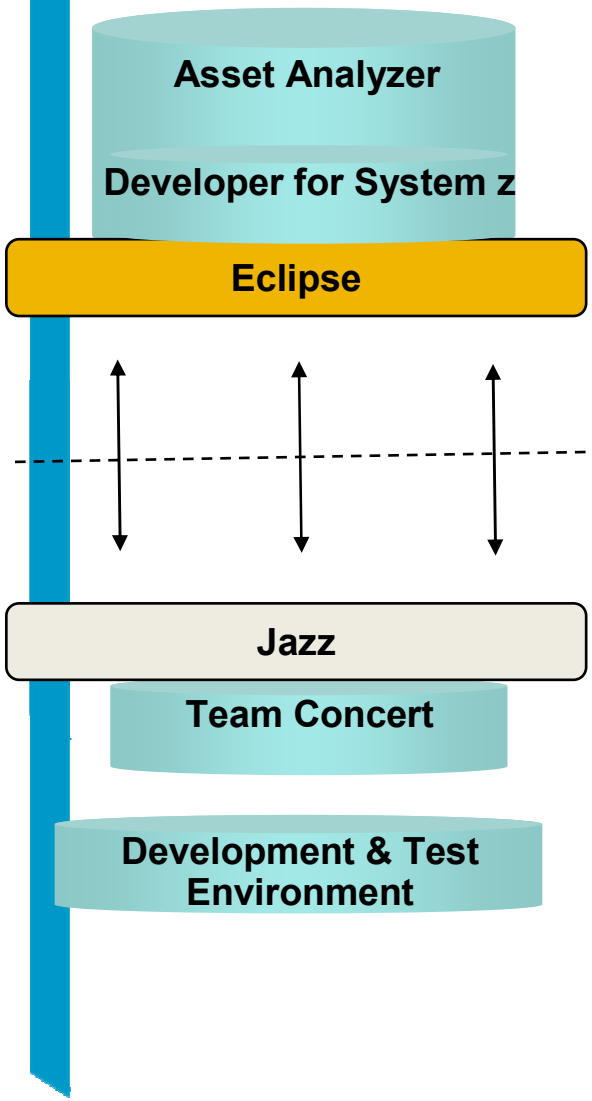
Arquitectura de la Solución (II)



Arquitectura de la Solución (III)



Componentes de la Solución



200x more COBOL transactions/day than **Google searches**

5 billion new COBOL LOC yearly

250 billion LOC of COBOL in existence

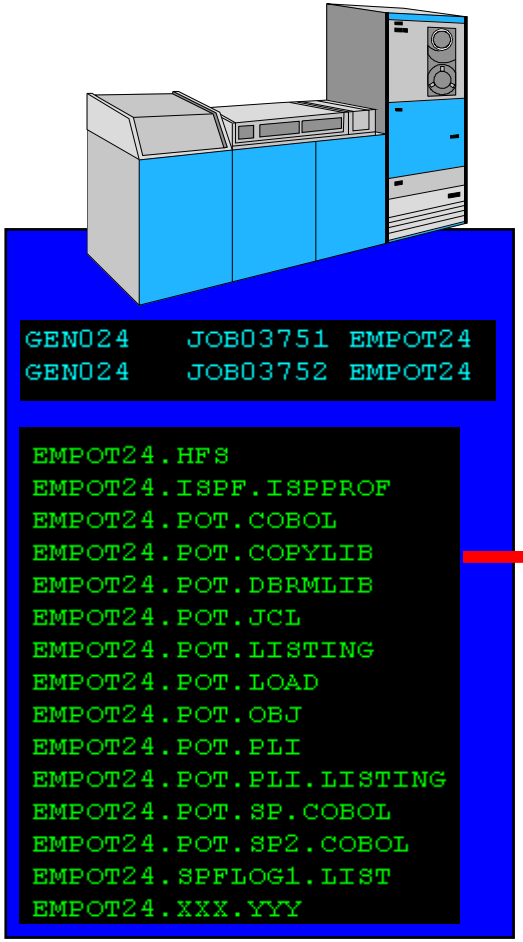
COBOL training **no longer offered** at most universities

Approximately all active code is **80% COBOL**

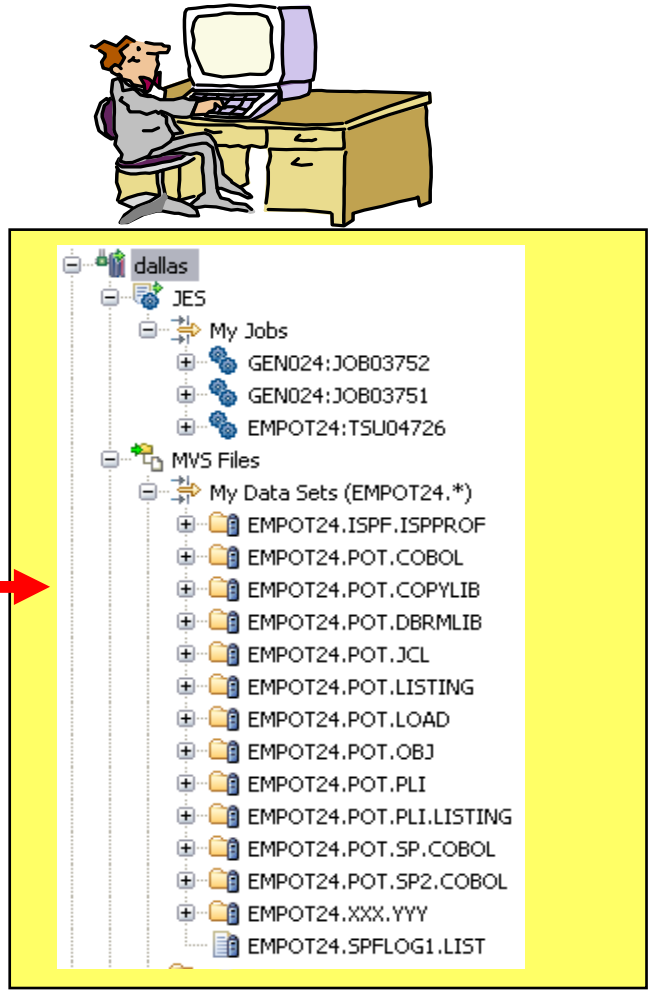
One million COBOL programs currently active

IBM

Developer for System z (RDz)

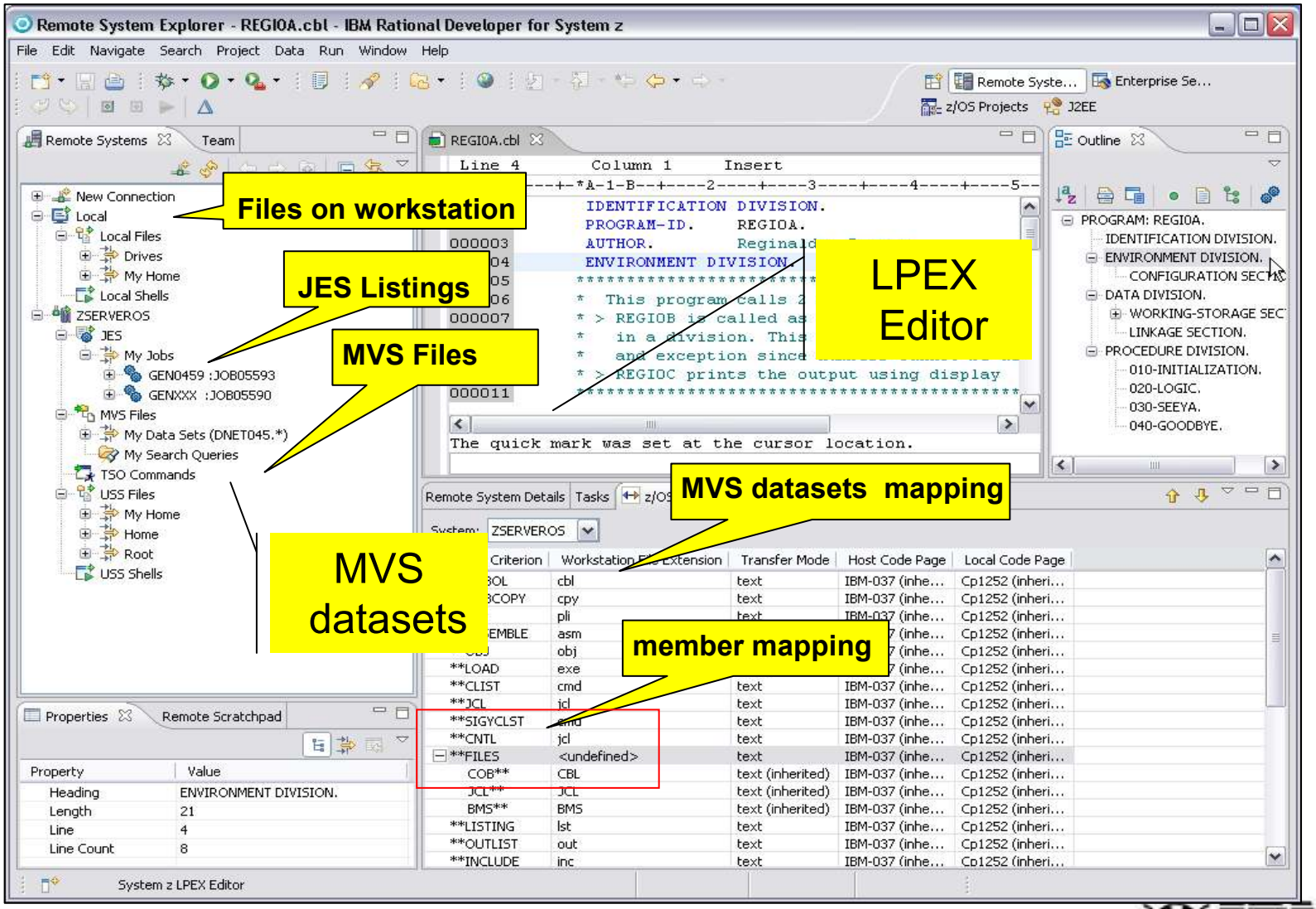


'Magic'

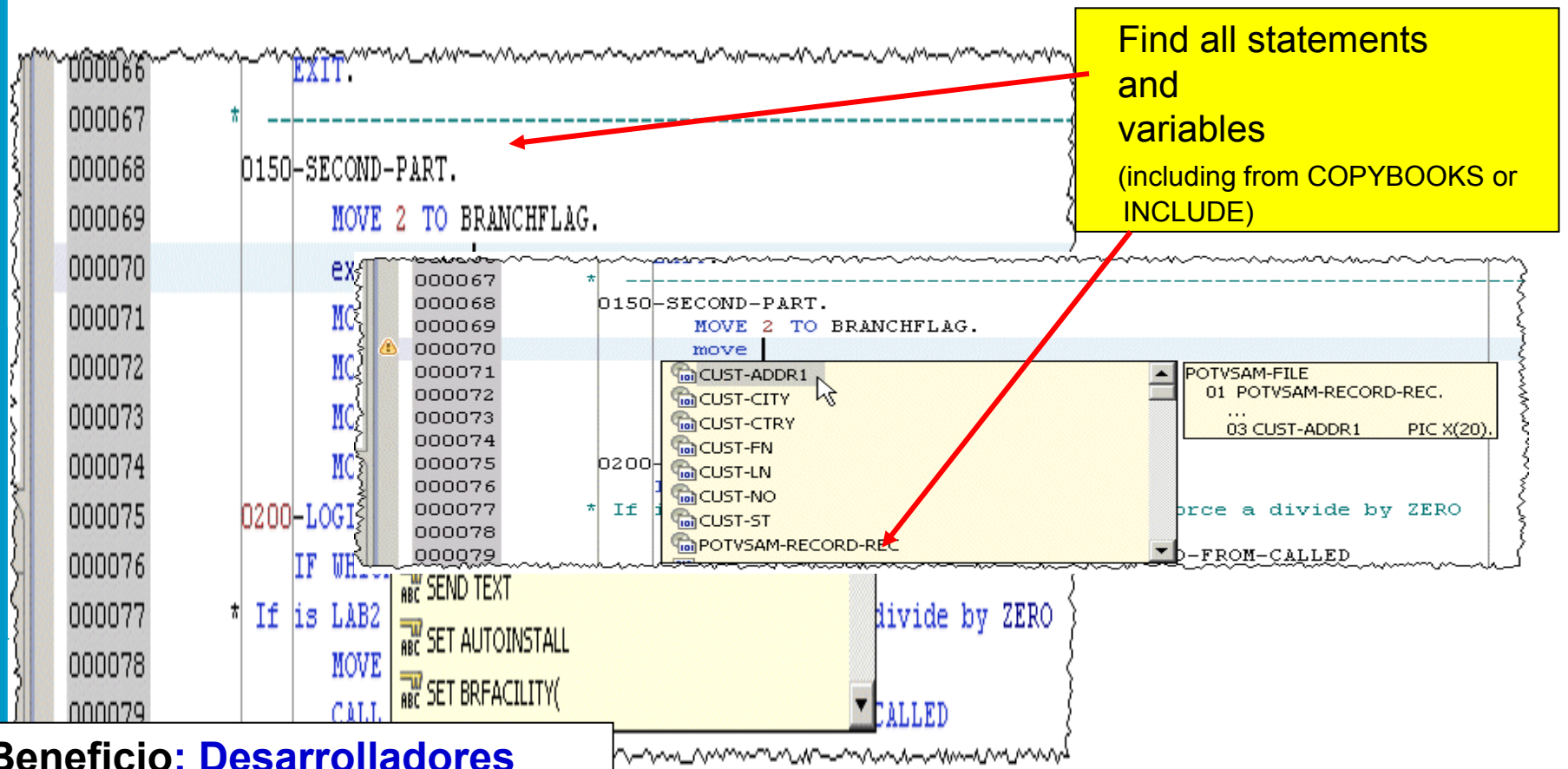


Los archivos en el “host” parecen como si estuvieran en la “workstation”

Acceso interativo en z/OS



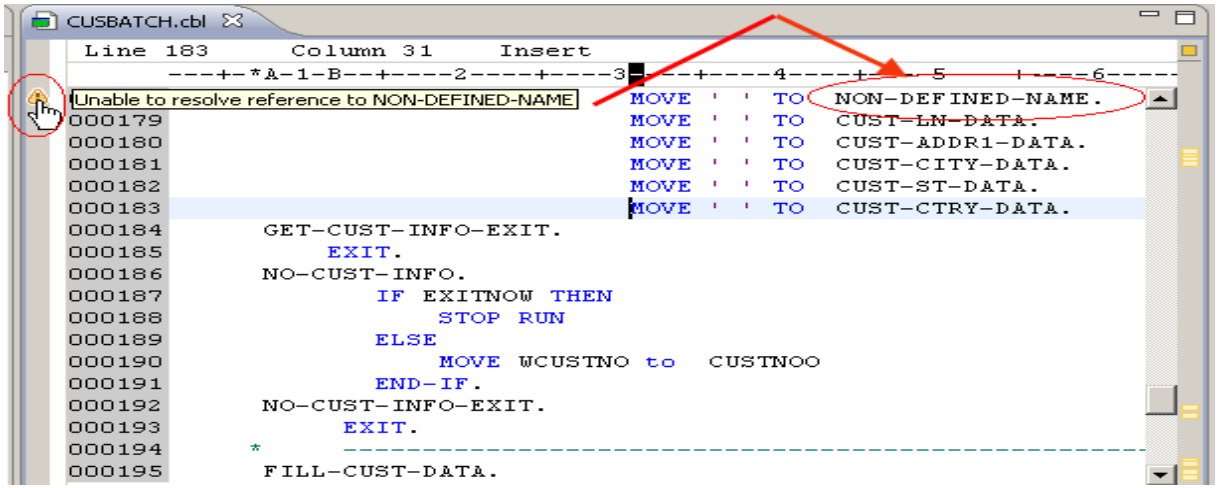
Editor: Asistente de contenido para COBOL - PL/I - C o C++



Beneficio: Desarrolladores finalizan el código de una forma más precisa y eficiente.

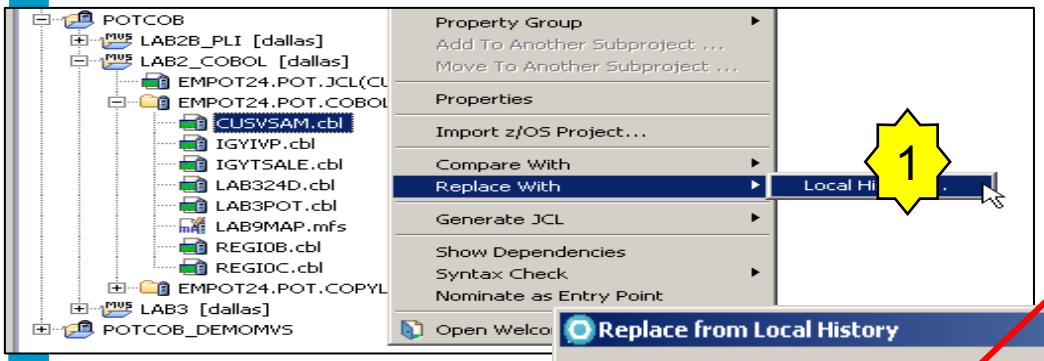
Editor: Comprobación en tiempo real de la sintaxis en COBOL

Real-time syntax check without requiring code compile or save



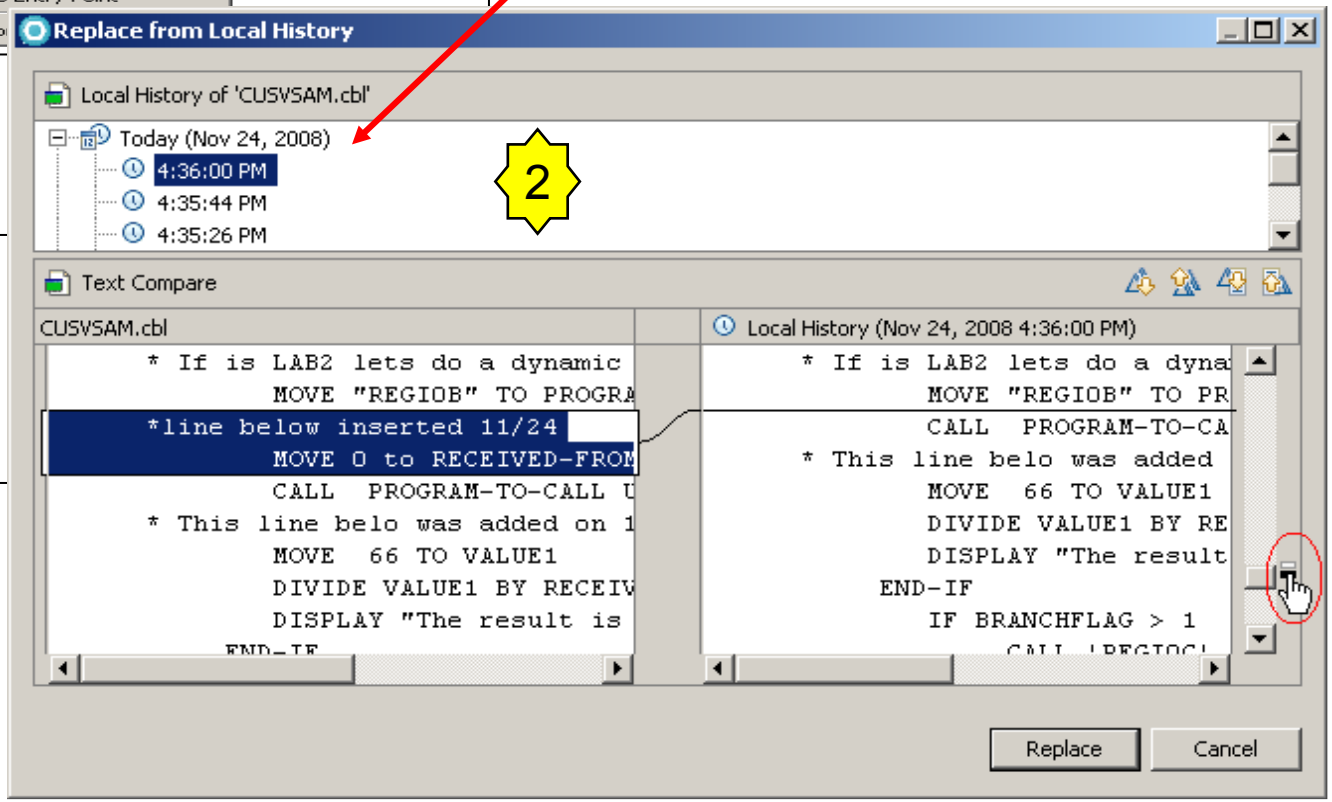
Beneficio: Mejorar la productividad del código.

Funciones de Eclipse útiles disponibles para activos z/OS. Reemplazar con Historia Local.

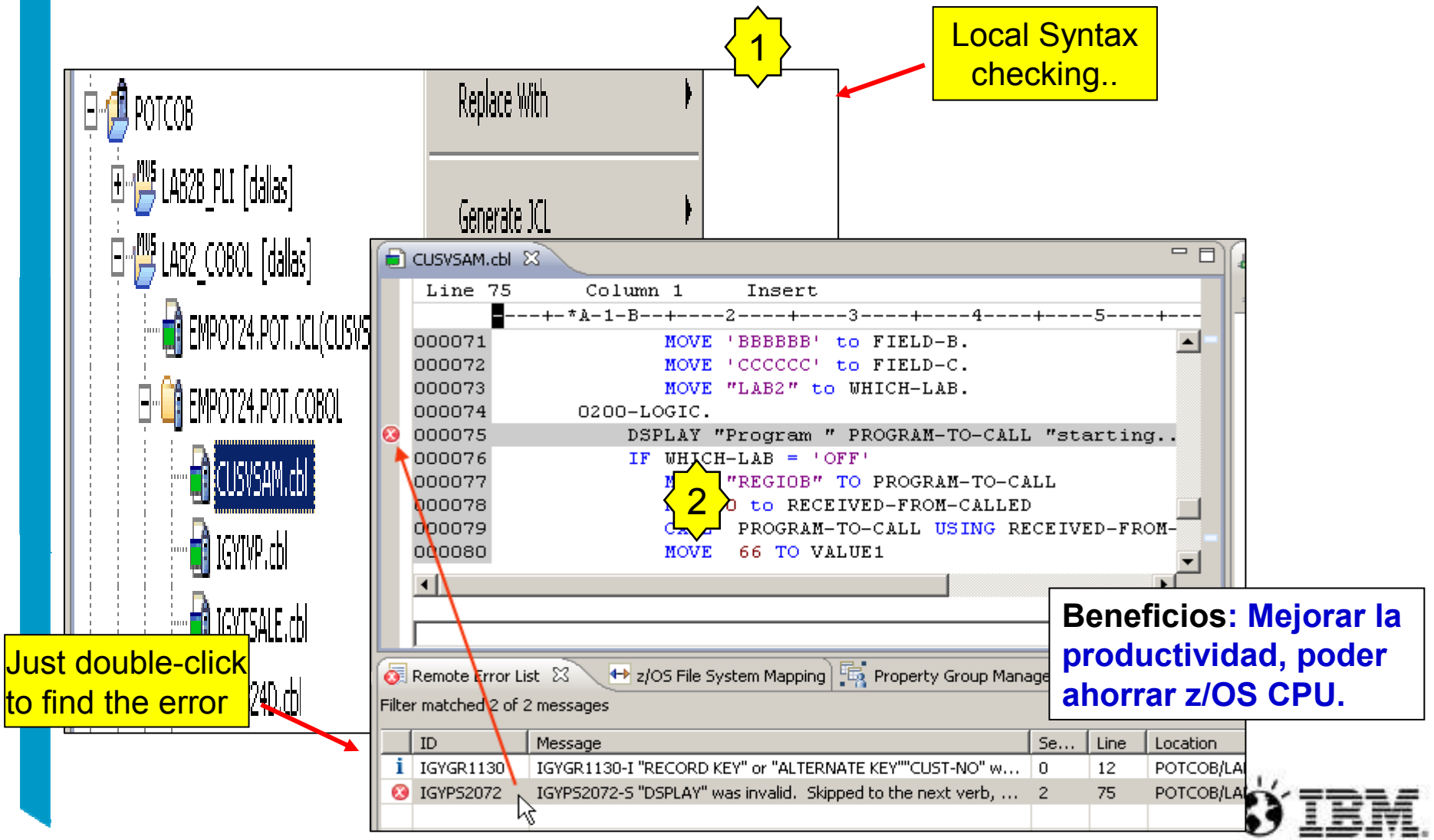


Keep as many local versions as you want and compare with the z/OS current version..

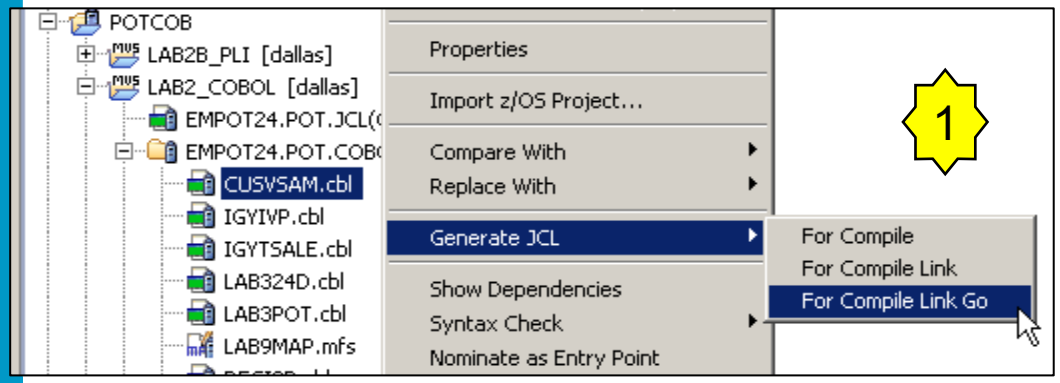
Beneficio: Mejorar la productividad ahorrando en tiempo de recuperación …

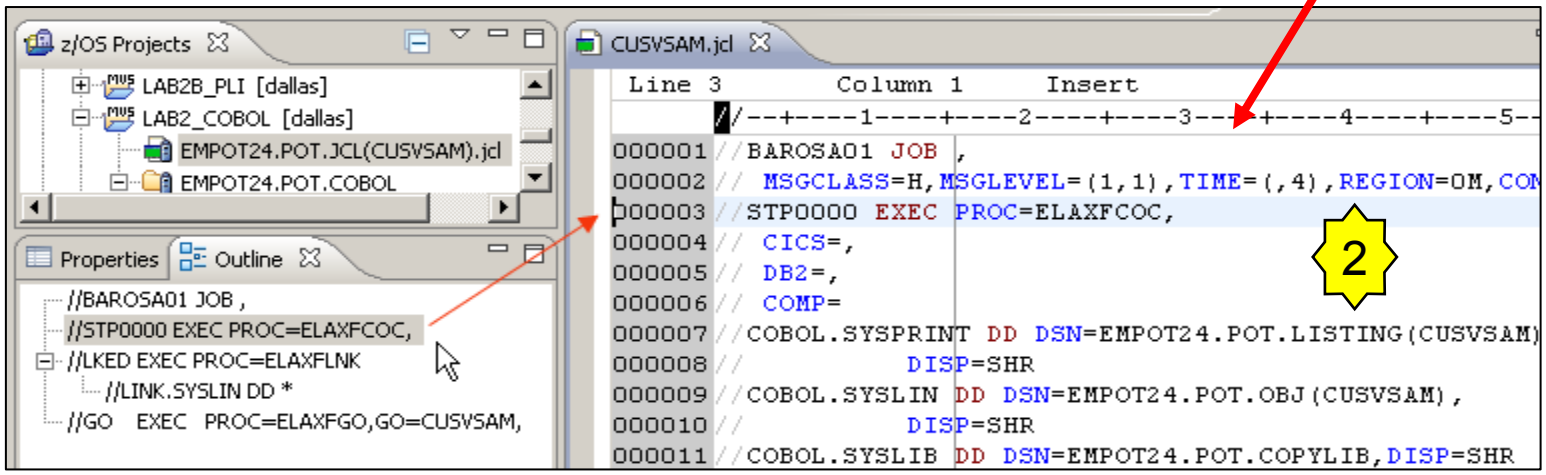


Usar compiladores locales o remotos para verificar la sintaxis



Generación y submit de JCL para su ejecución en z/OS

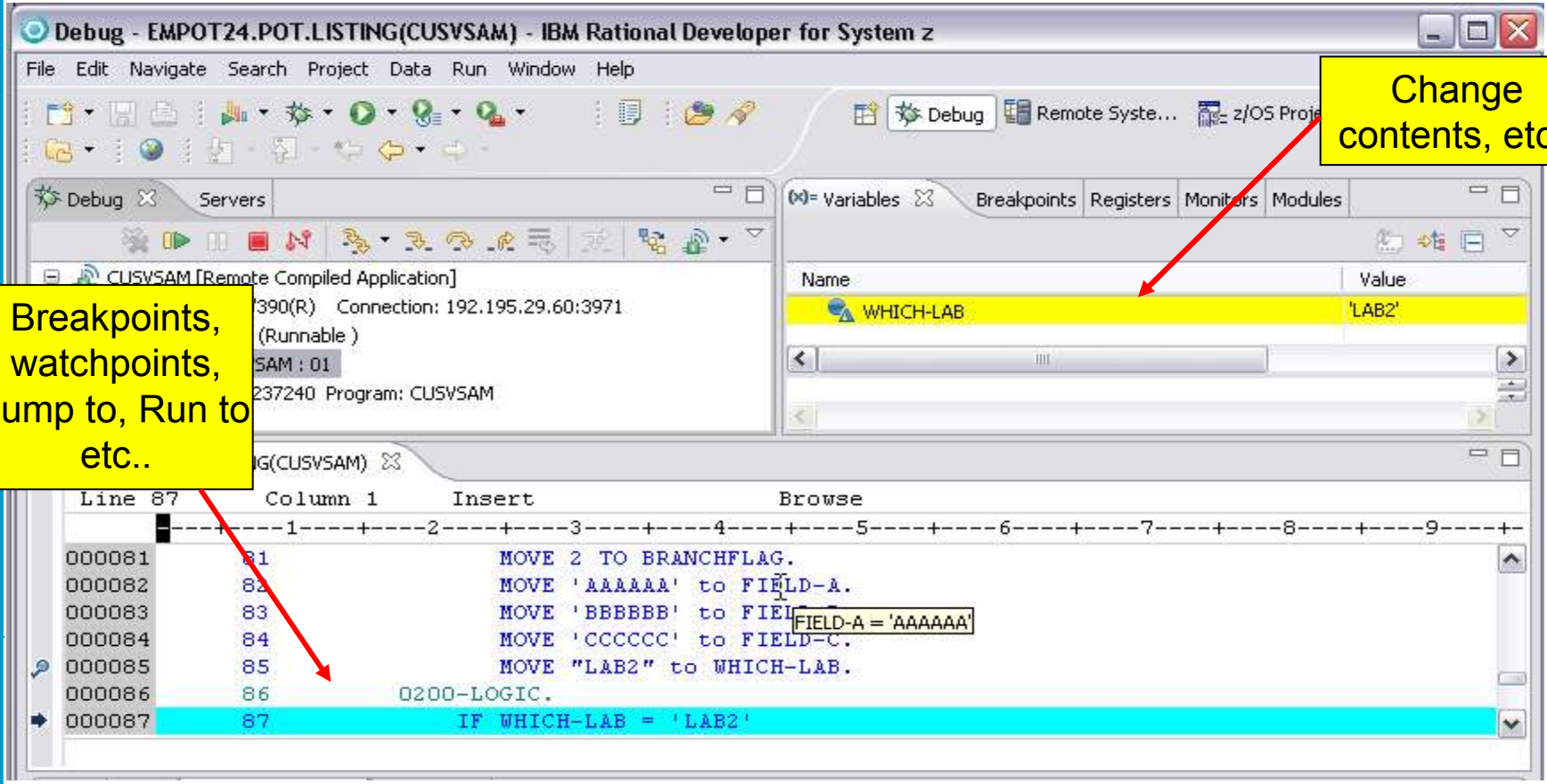




Beneficio: Desarrolladores enfocados en lógica de negocio y no en escribir JCL: JCL smart editor, Outline…

Debug remoto y local

→ Depurar aplicaciones z/OS desde la workstation mientras se ejecutan en un runtime remoto

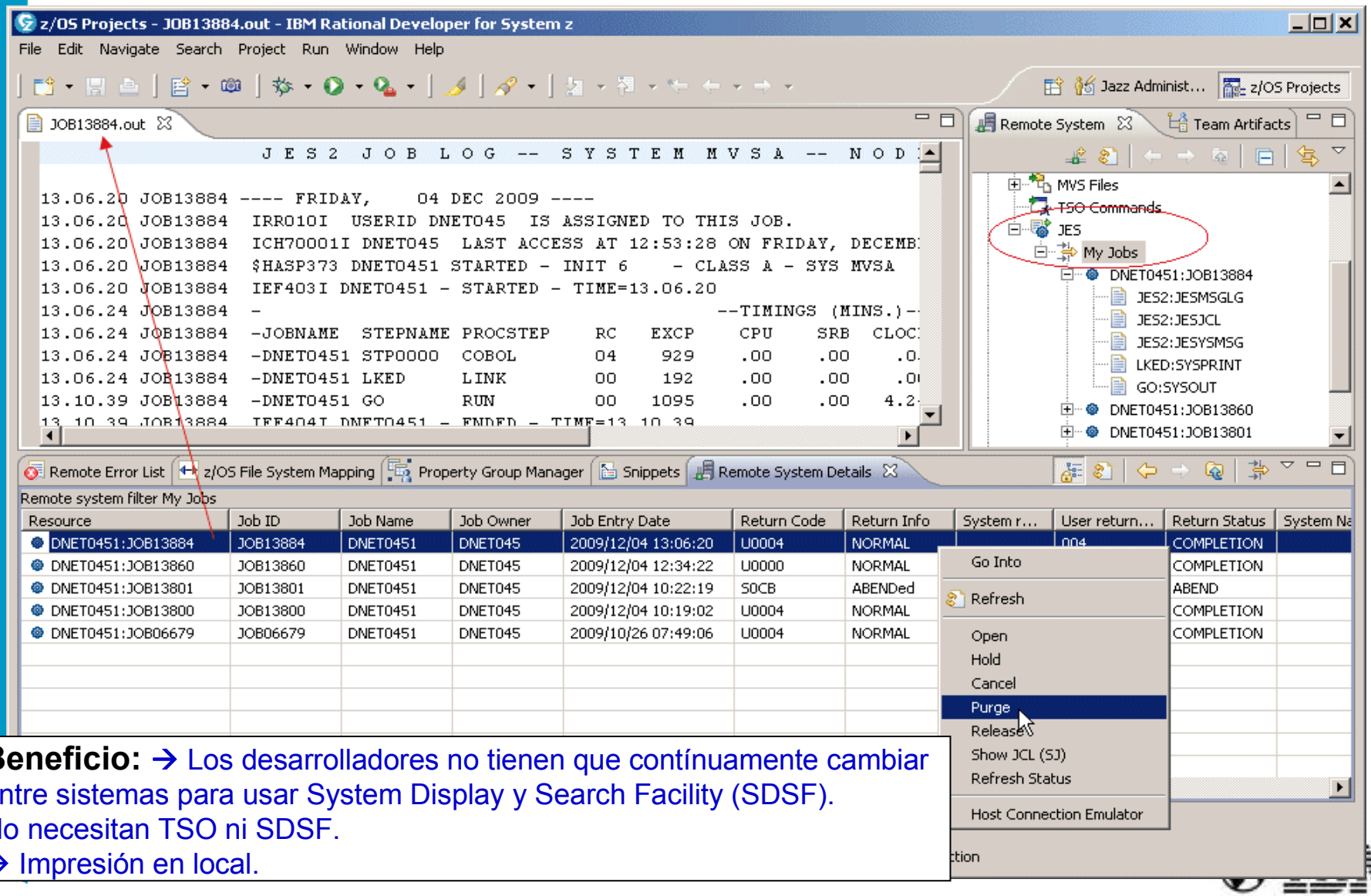


Beneficio: Misma perspectiva Debug para COBOL, PL/I, C, C++, Java, JSP, etc..

→ END to END Debug

Necesita tener instalado el producto z/OS Debug Tool

Monitorizar el Job Output



Beneficio: → Los desarrolladores no tienen que contínuamente cambiar entre sistemas para usar System Display y Search Facility (SDSF).
No necesitan TSO ni SDSF.
→ Impresión en local.

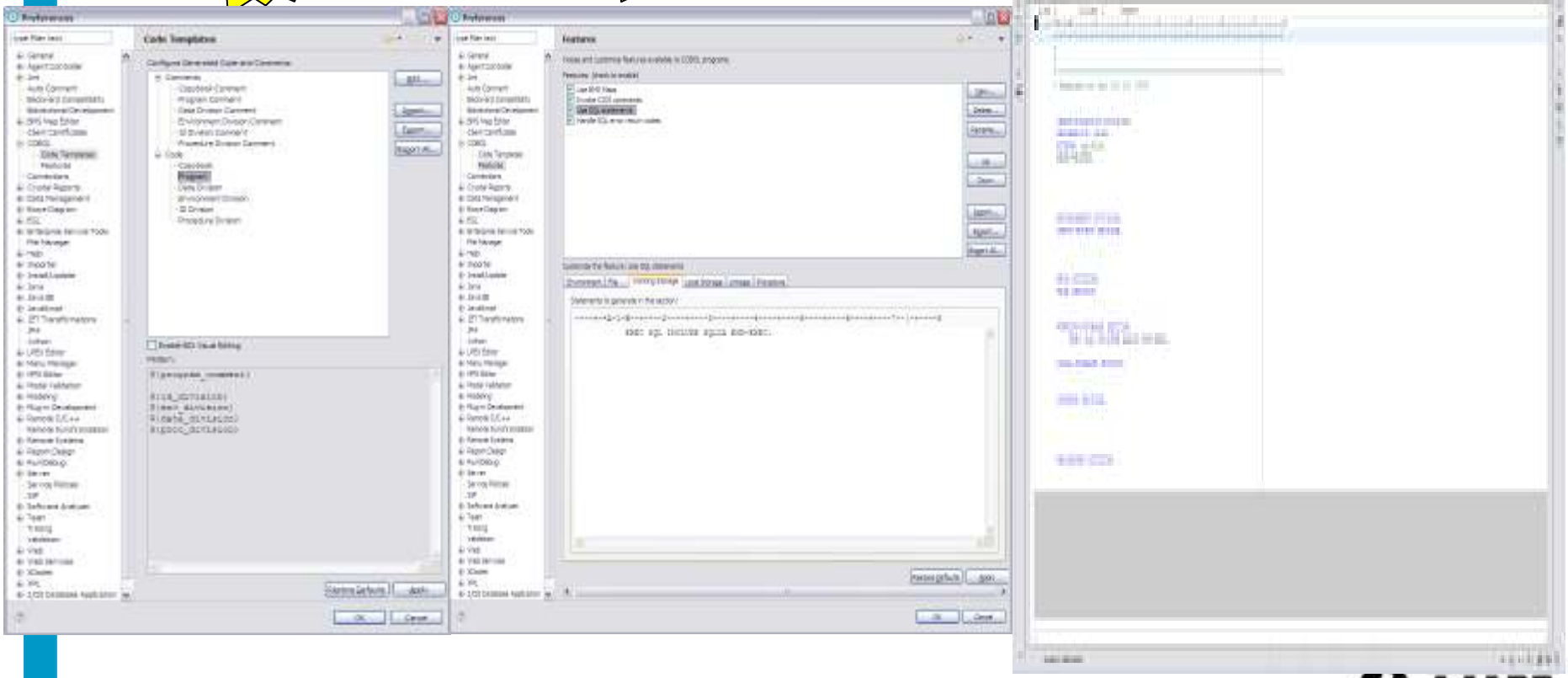
Plantillas para programas

- Ayuda a los programadores a utilizar plantillas pre-definidas.
- Se definen las plantillas y sus características
- El desarrollador las incorpora al crear el programa

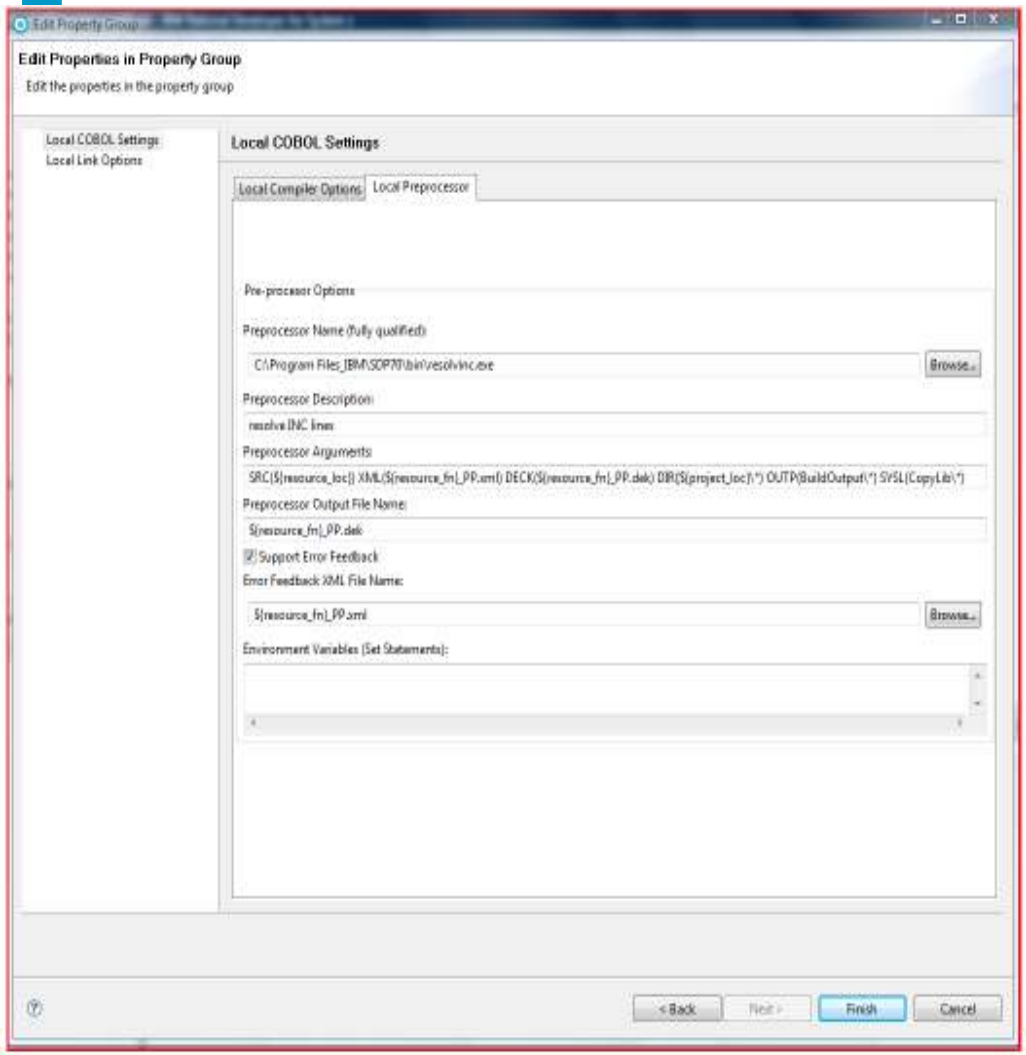
Code templates
1

Features
2

Creacion
3



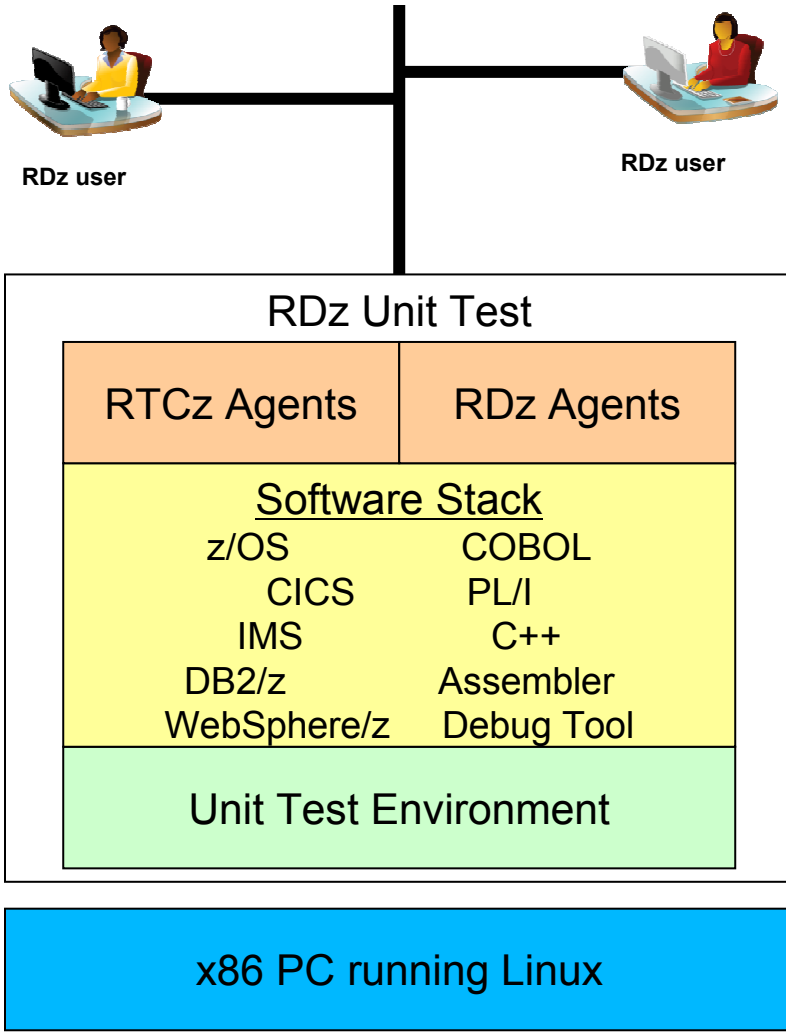
Soporte para preprocesador



- **Parámetros del preprocesador**
  - Local:
    - ✓ El cliente necesita codificar su propio preprocesador
    - ✓ El Preprocesador requiere escribir su fichero de XML y su fichero DEK

      (Botón browse para apuntar a ellos)
    - ✓ Pasar los argumentos necesarios en las opciones
  - Remoto:
    - ✓ Ya codificado por el cliente
    - ✓ Añadir pasos en el JCL
    - ✓ Los Procs de JCL deberan incluirr el paso añadido, y pasar la salida del fichero al compilador

Rational Development & Test Environment (RD&T)

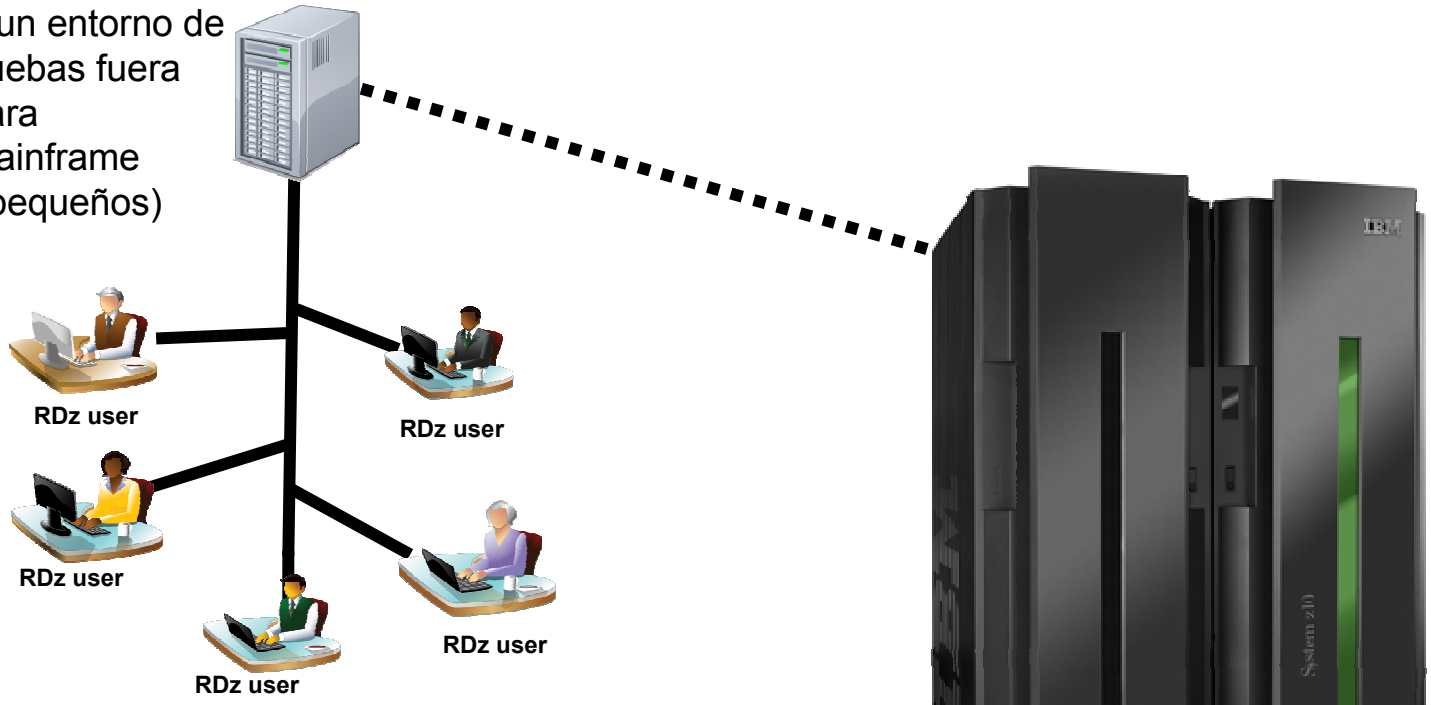


RD&T consiste en lo siguiente:

- Dev&Test – El entorno Unit Test corre sobre un sistema Linux basado en procesador Intel.
  - Puede proporcionar una plataforma de desarrollo de System z y es capaz de ejecutar sistemas operativos z/OS actuales
  - Nota: RD&T,solamente se usa como un sistema de desarrollo y no debe utilizarse (ni está diseñado) para ejecutar workloads de producción.
  - Proporciona gran flexibilidad a la hora de ejecutar un entorno muy configurado
- El stack de software incluído proporciona un middleware de IBM para entorno de prueba
  - Software middleware real (incluyendo z/OS)
  - Compiladores Enterprise reales
  - no simulación de APIs
- Agentes RDz y RTCz
  - Empaquetados para simplificar
  - Necesita la licencia RTC client para activar

Desarrollo actual en Mainframe con RD&T

La RD&T crea un entorno de desarrollo y pruebas fuera de System z para aplicaciones mainframe (para equipos pequeños)



Beneficios:

- Proporciona la capacidad de probar el proceso mainframe en un Linux
- Ayuda a bajar el coste de desarrollo y pruebas unitarias
- Libera capacidad del System z para uso en producción
- Da la flexibilidad de a los desarrollares (equipos) para ejecutar unit tests on-host u off-host

Ventajas de la solución tecnológica (RDz)

- Desarrollo Cobol-PL/1 fuera del Mainframe, resto de tareas (compilación, prueba, …) en Host

Incorporación de IDE moderno (RDz):

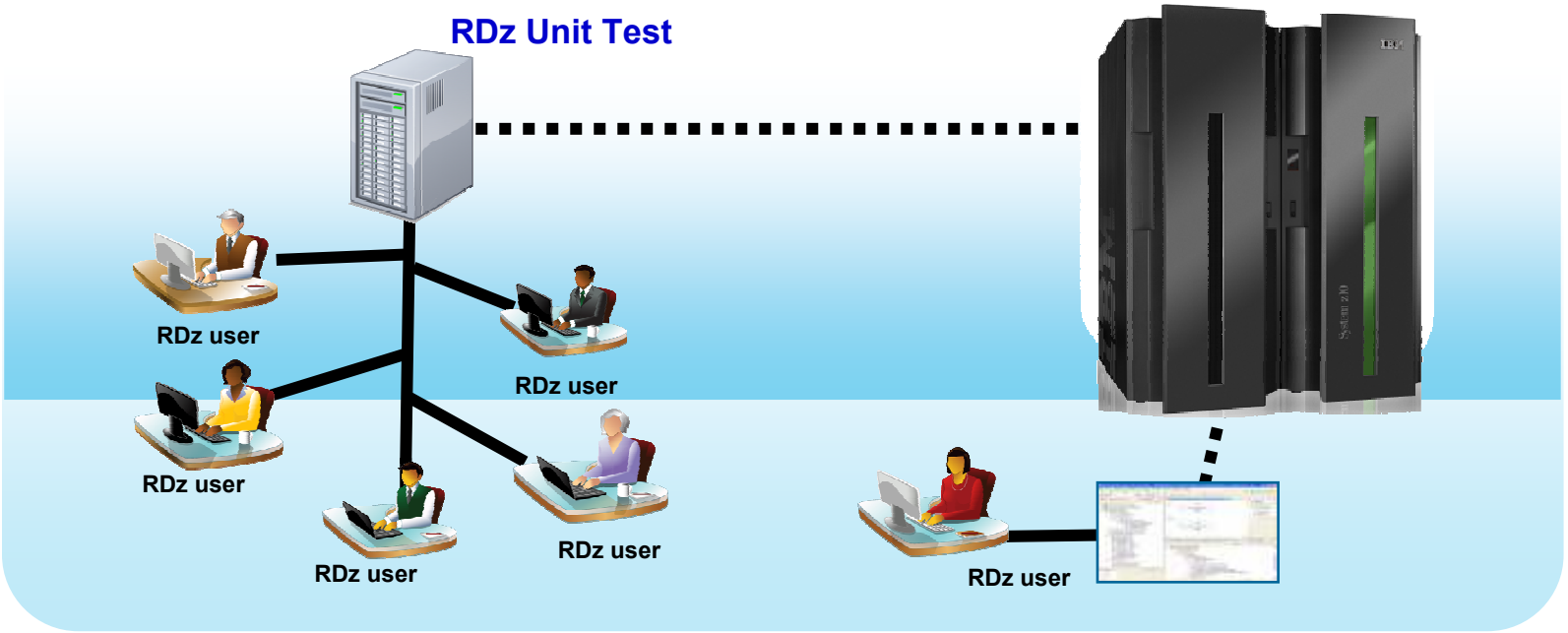
- Mejora de productividad, la herramienta aporta Eficiencia y Seguridad en Desarrollo.
- Reducción de una parte de los MIPS de desarrollo, ya que se trabaja en entorno local
- Integración con otras disciplinas del Ciclo de Vida del Desarrollo

Puntos de mejora:

- MIPS de desarrollo todavía elevados
- Las pruebas generan dependencias y tensiones en operativa de los entornos Mainframe.

Ventajas de la solución tecnológica (RD&T)

- Desarrollo Cobol-PL/1 y Pruebas Unitarias fuera del Mainframe, reducción relevante de consumos en Host.



Incorporación de IDE moderno (RDz):

- Liberación relevante de MIPS en Desarrollo. Reducción Coste, mejora Producción
- Desarrollo y Test de aplicaciones en entorno más flexible y cercano, no productivo
- RD&T simple de mantener y gestionar (menos costes operativos)

Retorno de la Inversión (R.O.I.)

RDz y RD&T en el ciclo de desarrollo

BENEFICIOS ESPERADOS

- **Mejoras en la productividad de desarrollo**
  - ✓ Desarrollo a través de un único interfaz
  - ✓ Simplificación del proceso de desarrollo
    - Uso de editores avanzados
    - Entorno multitarea que permite actividades simultáneas
    - Disponibilidad de herramientas gráficas de debugging
  - ✓ Reducción de la curva de aprendizaje para tecnologías Host
  - ✓ Posibilidad de ampliación futura a otras funcionalidades disponibles
- **Optimización del consumo de CPU y recursos Host**
  - ✓ Reducción muy importante del consumo TSO
  - ✓ Menor número de compilaciones
  - ✓ Compilar fuera del mainframe

IBM

Benchmarks de eficiencia del IDE (RDz)

- 100 tareas comunes (diarias) de ISPF que se ejecutan durante asignaciones de soporte y mantenimiento.
  - Traducción del workflow ISPF (click-for-click) a desarrollo en RDz
  - Los participantes del proyecto creyeron que estaban intentando encontrar gaps entre la funcionalidad de RDz e ISPF
- "Apples-to-Apples" y test scripts
- Mezcla de experimentados (veteranos) programadores de ISPF y nuevos desarrolladores recién incorporados a plantilla
- La primera impresión fue que solamente los recién llegados serían más productivos…
  - …y resultó ser falso

Based on IBM internal productivity study.

All performance data contained in this presentation was obtained in the specific operating environment and under the lab conditions and is presented as an illustration only.

Performance obtained in other operating environments may vary and customers should conduct their own testing.

Analytics	All Participants
Use Case	% Less time to complete tasks with RDz
Source Navigation:	45.00
Program Analysis:	100.90
Primitive Edit Operations:	19.69
COBOL Statement Coding:	35.79
Syntax Check:	57.11
Program Compile/Link Edit:	43.52
DB2 data edit/SQL statement work:	101.30
Total - all use cases:	57.69

Analytics	ISPF Top Guns
Use Case	% Less time to complete tasks with RDz
Source Navigation:	47.21
Program Analysis:	72.60
Primitive Edit Operations:	19.84
COBOL Statement Coding:	36.89
Syntax Check:	68.86
Program Compile/Link Edit:	14.14
DB2 data edit/SQL statement work:	64.68
Total - all use cases:	28.36

12

Mándame un email a alejandro.leon@es.ibm.com y te lo haré llegar

Appendix 1- RDz Productivity Benchmarks

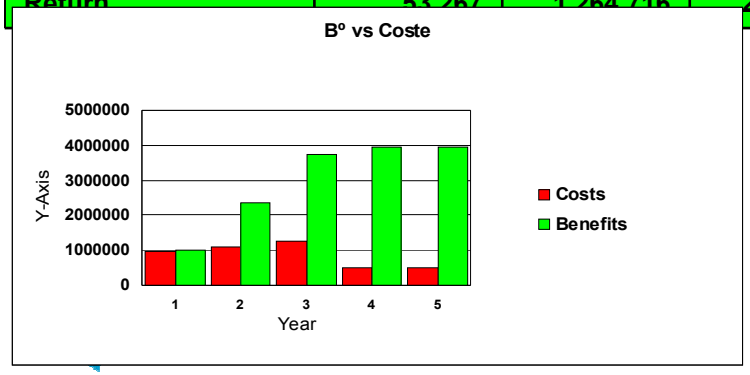
This Appendix documents a Benchmark completed in March 2010 that compared two IBM products: RDz v7.6.1 and ISPF v6.0.

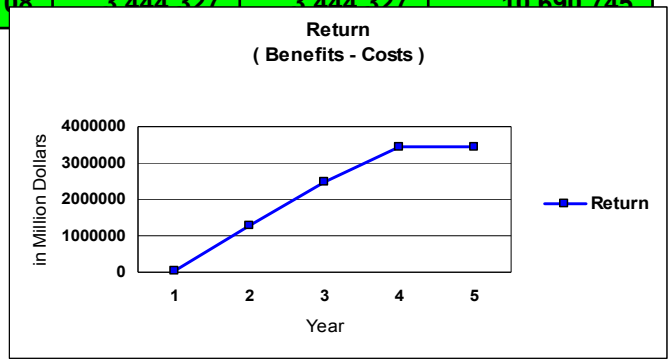
The work performed in the study was a set of tasks normally associated with traditional z/OS COBOL maintenance and production support:

- Code analysis - paragraph flow and program structure analysis
- Navigating and modifying existing COBOL programs
- Adding a small amount of new business logic to an existing program
- Updating fields in copybooks and modifying code that referenced to updated fields
- Doing tradition Data Flow analysis
- Working with SQL and DB2 test data
- Compiling and linking a program

	Year 1	Year 2	Year 3	Year 4	Year 5	Total
	2010	2011	2012	2013	2014	
Rollout RDz	150	175	175			500
Roll Out RD&T	50	50	50			150
Costs						
Software	730.988	823.706	823.706			2.378.400
Subscription	0	146.213	310.950	475.688	475.688	1.408.538
Education/Services	125.000	52.500	52.500			230.000
Setup/Admin Staff	82.500	55.000	55.000	27.500	27.500	247.500
Hardware	27.500	31.250	28.750			87.500
Total Costs	**965.988**	**1.108.669**	**1.270.906**	**503.188**	**503.188**	**4.351.938**
Justification						
Compile CPU Savings	103.726	224.740	345.755	345.755	345.755	1.365.730
TSO CPU Savings	255.528	553.644	851.760	851.760	851.760	3.364.452
Productivity	660.000	1.595.000	2.557.500	2.750.000	2.750.000	10.312.500
Total Justification	**1.019.254**	**2.373.384**	**3.755.015**	**3.947.515**	**3.947.515**	**15.042.682**
Return	**53.267**	**1.264.716**	**2.484.108**	**3.444.327**	**3.444.327**	**10.690.745**

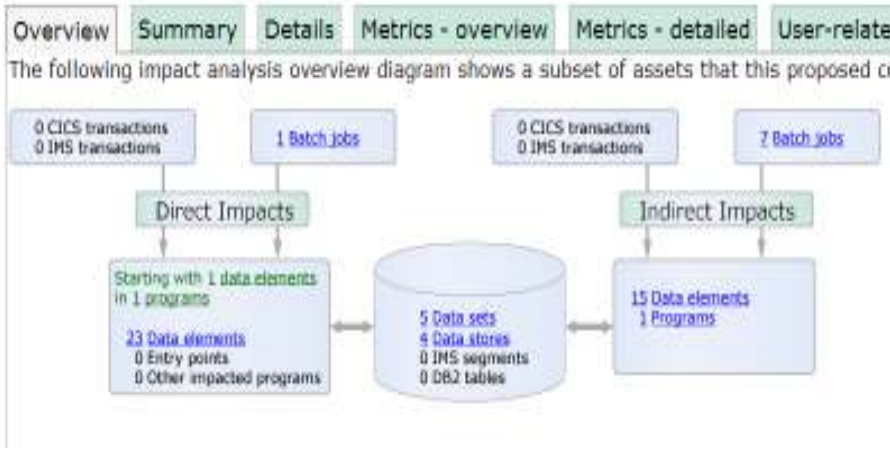
DATOS BASE PARA EL ROI	
Núm Desarrolladores: Año 1 Año 2 Año 3	 150/50 175/50 175/50
Coste anual por Desarrollador	55.000€
Coste 1 día TSO por Desarrollador	11,06 €
Coste medio de 1 compilación	0,62 €





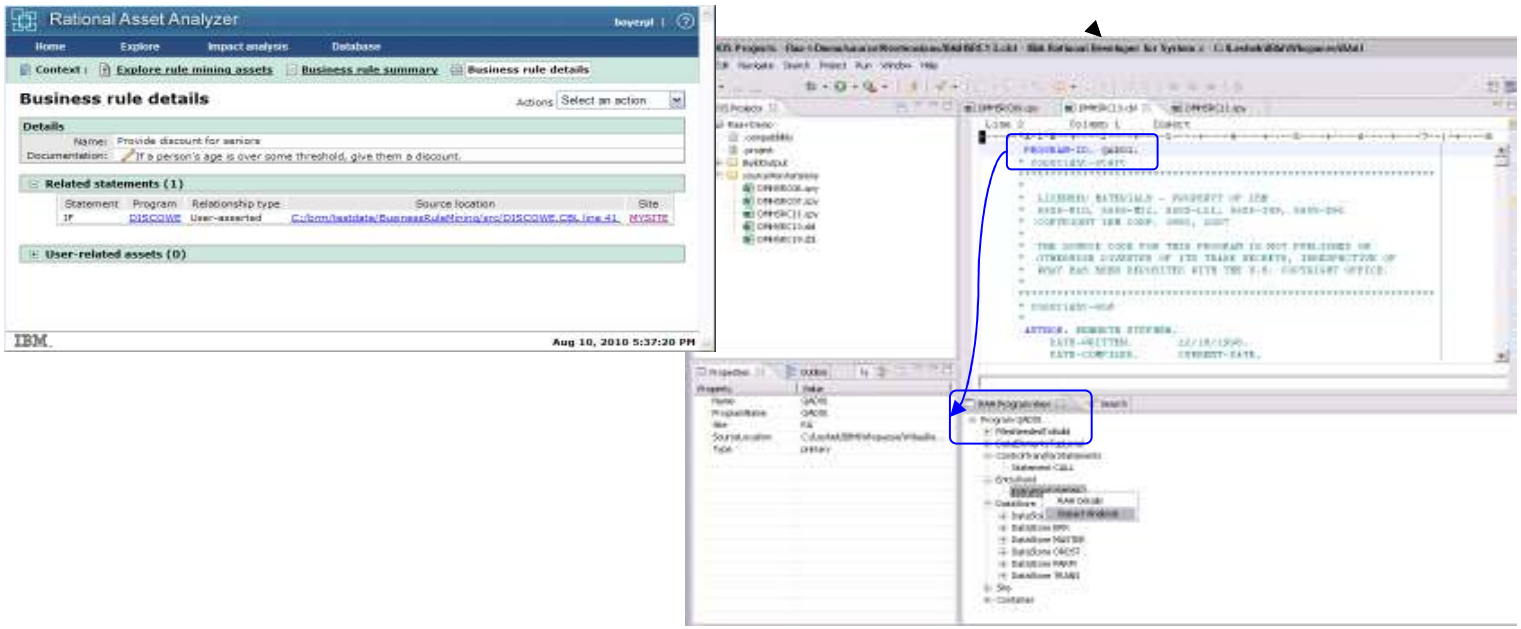
Asset Analyzer (RAA)

- **Acelera la entrega de proyectos respondiendo a las exigencias de negocio**
  - Reduce riesgos, Incrementa la productividad y mejora la calidad de los cambios en las aplicaciones
- Proporciona **control intelectual** de las aplicaciones
  - Identifica las **diferentes tecnologías** que se usan actualmente en la organización – usando dashboards o reports -
  - Comprende **la calidad y la complejidad** de los activos de desarrollo
- Proporciona **transparencia** hacia el desarrollo interno / outsourced
  - Agrupa artefactos en **user-defined groups** llamados Applications para limitar el alcance del área de interés.
  - Usa **varios tipos de diagramas** para entender cómo la aplicación se relaciona (mapa de aplicación)
- **Permite ser configurado e integrado** en los procesos de la organización y entornos de TI



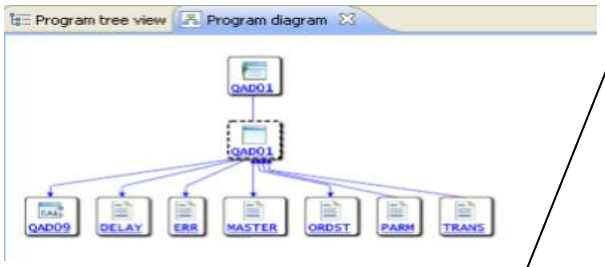
Asset Analyzer (RAA)

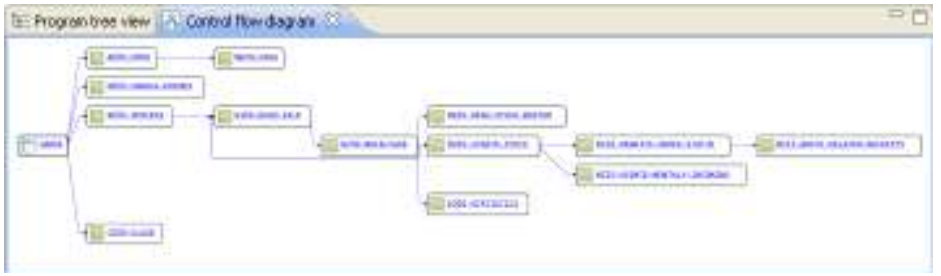
- Parte fundamental de la solución para la toma de decisiones ágil
  - Proporciona identificación de las reglas de negocio
  - Permite la captura y la gestión con WebSphere ILOG JRules BRMS
- Interfaces orientadas al Role
  - Interfaz web fácil de usar, para búsquedas y exploración
  - Interfaz eclipse para el Developer, integrada con RDz

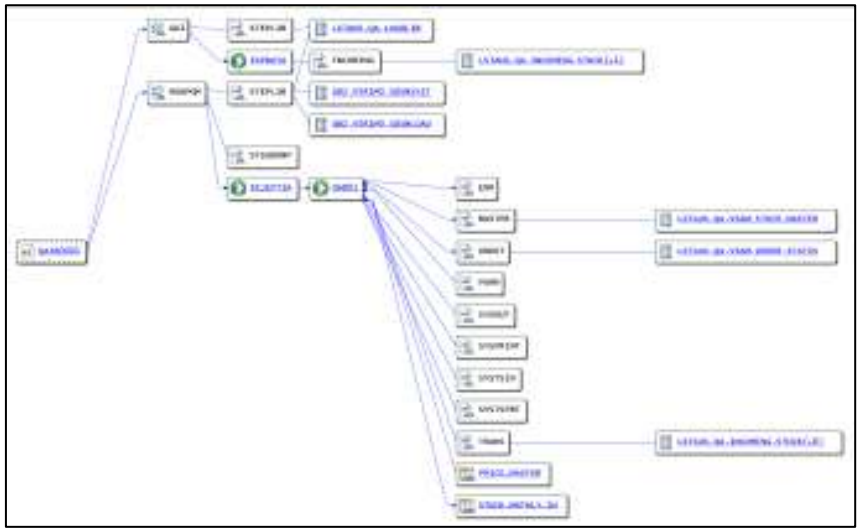


Asset Analyzer (RAA)

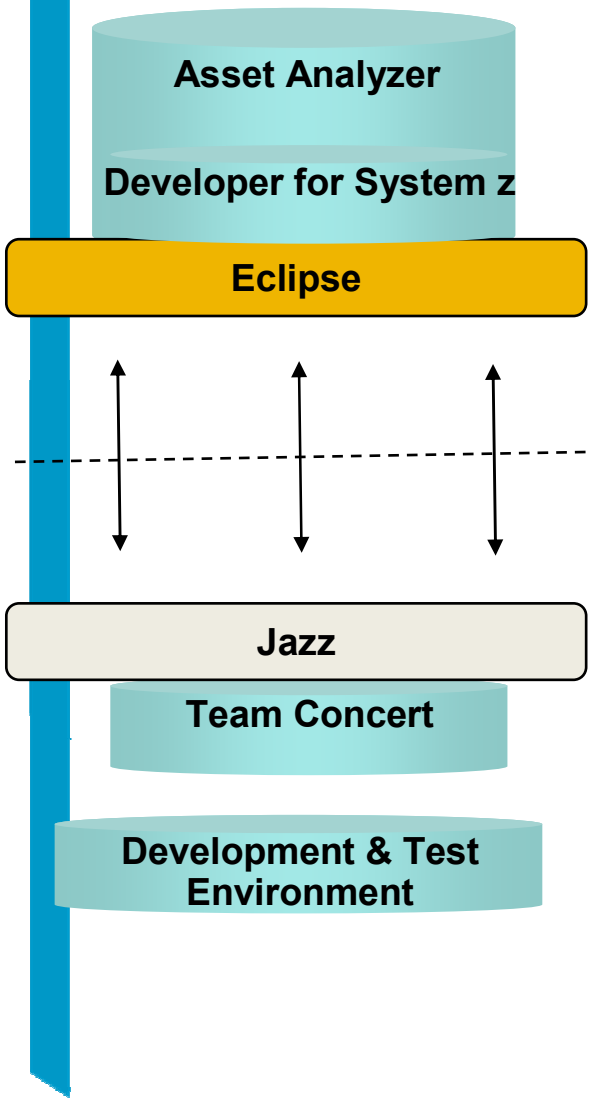
- Proporciona una cartografía o mapa de aplicaciones
  - Permite conocer rápidamente los componentes y sus relaciones







Beneficios de la Solución IBM

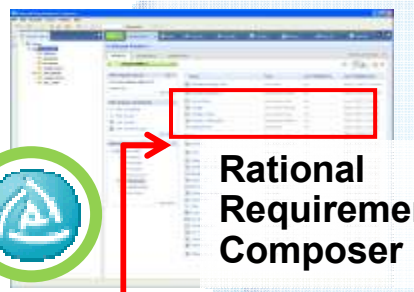


- Arquitectura multiplataforma (Windows, Unix, System i, System z)
- Integración a los colaboradores en el ciclo
  - Colaboración internos/externos en los desarrollos
  - Medida de productividad y visión de capacidad de la subcontrata, con control de acceso
  - Selección del mejor proveedor
- Repositorio Único, Plataforma de desarrollo homogénea,
  - Cobol,PL/I,HLASM,RPG,Java,C,C++,C#… Eclipse, .NET
  - Más fácil administración
- Integración de todas las capacidades de desarrollo y trazabilidad
  - CMMi, Seguridad, Metodologías Agil + Tradicional
  - Requisitos, Modelos, Configuraciones SW, Releases, Pruebas, Documentación
  - Sincronización y mayor confiabilidad en los pases a producción

C/ALM: Alinea las tareas de desarrollo y test con el valor del cliente

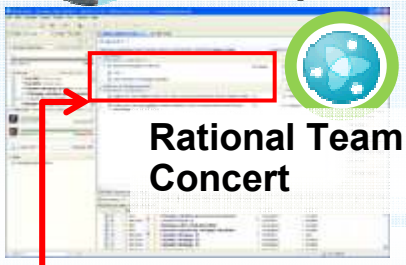
Collaborative Application Lifecycle Management rompe con los silos de información en pos de una mejor ejecución del proyecto

Business analyst



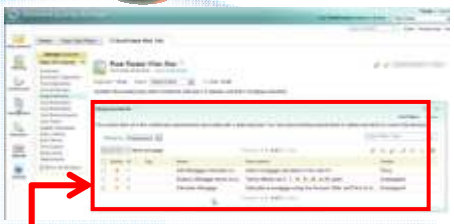
Rational Requirements Composer

Developer



Rational Team Concert

Tester



Rational Quality Manager

Name	Implemented By	Validated By	Type
Agile Planning of Change Requests	Story 61: CQ: this is a sample work item Story 45: CQ: JSON (JavaScript Object No... Story 45: CQ: JSON (JavaScript Object No...	Test Case 49: SCMA4.2.Source.2009.09.06...	Use Case
Architectural Direction	Story 45: CQ: JSON (JavaScript Object No... Story 36: Provide C/ALM compact renderin...	Test Case 33: Architectural Direction Test Case 36: Collaborative ALM Test Case 38: Save Selection As Story Test Case 43: Rich Hover Story	Document
avatar.gif		Test Case 42: avatar.gif	GIF Image
calm build status.doc	Story 70: CQ: Implement rich hover for all ...	Test Case 35: calm build status.doc	Microsoft Wor

Muchas gracias