HERRAMIENTA DE SOPORTE PARA LA INSPECCIÓN DE LA CALIDAD DEL CÓDIGO DE LAS APLICACIONES HPC.

SUPPORT TOOL FOR CODE QUALITY INSPECTION OF HPC APPLICATIONS.

Autores: [1]**Duanys Miguel Peña López y** [2]**Alberto Carlos Carbonell Marce.**

[1]ORCID ID: https://orcid.org/0000-0003-1054-5517
[2]ORCID ID: https://orcid.org/0000-0002-2159-7999

Articulo recibido: 4 de Mayo del 2021
Articulo revisado: 7 de Junio del 2021
Articulo aprobado: 23 de Junio del 2021

[1]Ingeniero mención Ciencias Informáticas egresado de la Universidad de las Ciencias Informáticas (Cuba). Posee una Maestría en Metodología de la Investigación del Instituto Superior Blas Roca Calderio (Cuba).
[2]Ingeniero mención Ciencias Informáticas egresado de la Universidad de las Ciencias Informáticas (Cuba).

Resumen

La naturaleza del desarrollo de aplicaciones HPC anima al diseño digital ad hoc y su aplicación, en lugar de análisis de requerimientos formales y especificaciones de diseño como es típico en la ingeniería de software. Sin embargo, no se puede esperar simplemente que los desarrolladores de HPC adopten los procesos de ingeniería de software de una manera formal, incluso cuando hay una necesidad de mejorar la estructura y calidad del software mantenimiento la funcionabilidad. Por lo tanto, en este articulo científico se propone el uso de Yworks como herramienta que los desarrolladores de HPC pueden utilizar a su consideración para obtener información sobre la estructura y la calidad de sus códigos. Esta retroalimentación vendría en forma de métricas y análisis de calidad de código, presentado cuando sea necesario en visualizaciones intuitivas e interactivas. Esta investigación resume el análisis de herramientas de este tipo, aplicando como punto de referencia un HPC estándar como prueba de concepto.

Palabras claves: Aplicaciones HPC, Calidad del código, Herramienta de soporte.

Abstract

The nature of HPC application development encourages ad hoc digital design and implementation, rather than formal requirements analysis and design specifications as is typical in software engineering. However, HPC developers cannot simply be expected to adopt software engineering processes in a formal way, even when there is a need to improve software structure and quality while maintaining functionality. Therefore, in this scientific article, the use of Yworks is proposed as a tool that HPC developers can use at their discretion to obtain information about the structure and quality of their codes. This feedback would come in the form of code quality metrics and analysis, presented when needed in intuitive and interactive visualizations. This research summarizes the analysis of tools of this type, applying as a reference point a standard HPC as a proof of concept.

Keywords: HPC applications, Code quality, Support tool.

Sumário

A natureza do desenvolvimento de aplicativos HPC incentiva o projeto e a implementação digital ad hoc, em vez de análise formal de requisitos e especificações de projeto, como é típico na engenharia de software. No entanto, os desenvolvedores de HPC não podem simplesmente adotar processos de engenharia de software de maneira formal, mesmo quando há necessidade de melhorar a estrutura e a qualidade do software, mantendo a funcionalidade. Portanto, neste artigo científico, propõe-se a utilização do Yworks como uma ferramenta que os desenvolvedores de HPC podem utilizar a seu critério para obter informações sobre a estrutura e qualidade de seus códigos. Esse feedback viria na forma de métricas e análises de qualidade de código, apresentadas quando necessário em visualizações intuitivas e interativas. Esta pesquisa resume a análise de ferramentas deste

tipo, utilizando como ponto de referência um HPC padrão como prova de conceito.

Palavras-chave: aplicativos HPC, qualidade de código, ferramenta de suporte.

Introducción

En los procesos de ingeniería de software el uso de aplicaciones HPC no son comúnmente utilizados por los desarrolladores por varias razones (Carver, J., Hochstein, L., Kendall, P., Nakamura, T., Zelkowitz, V., Basili, R., & Post, E., 2006). En primer lugar, los desarrolladores pueden centrarse en el rendimiento con respecto a otros aspectos del código, como los que facilitaría el mantenimiento futuro. En segundo lugar, el desarrollador podría ser el único cliente de la aplicación, al menos inicialmente; por lo tanto, puede haber poca motivación para pensar acerca de cómo usar, mantener, adaptar o extender una aplicación más allá de su uso inicial. En tercer lugar, muchos desarrolladores de HPC no están capacitados formalmente en los procesos de ingeniería de software; que permitan un avance del desarrollo de una manera ad hoc, posiblemente con los requisitos, planes de diseño, o la documentación. En conjunto, estos aspectos del software de HPC implican altos costos de desarrollo y mantenimiento durante la vida útil de un sistema de software.

Sin embargo, es poco realista esperar que ocurran cambios generalizados repentinos en el proceso de desarrollo de aplicaciones HPC (De Giusti, E., Tinetti, G., Naiouf, M., Chichizola, F., De Giusti, C., Villagarcía Wanza, H. & Eguren, S., 2016). En lugar de ello, se prevé la creación de herramientas para ayudar a comunicar el mantenimiento de los softwares, mediante la reutilización de componentes y aspectos de calidad de código de software de HPC durante el desarrollo del código. Yworks como herramienta podrían proporcionar dicha retroalimentación discreta, es decir, de una manera que permite al desarrollador logrando un equilibrio entre el rendimiento y la capacidad de mantenimiento del software. Además, al proporcionar el tipo correcto de análisis y visualizaciones, esta herramienta podría incluso ayudar en la reestructuración y la mejora del código durante la programación de este.

Esta investigación resume la implementación continua de Yworks como herramienta para aplicaciones C y C ++ a gran escala. Mostrando comentarios sobre la calidad del código, que se implementan combinando una serie de herramientas como se describe en esta investigación, no impone formalismos o procesos particulares de diseño o desarrollo (Larman, C., & Basili, R., 2003); más bien, proporciona información y orientación en forma de métricas de calidad de código específicos. Además, estas herramientas pueden proporcionar esta información en forma de visualizaciones interactivas cuando sea apropiado, por ejemplo, al tratar de comprender si la arquitectura general del software coincide con el modelo mental del desarrollador, o si un subconjunto de archivos o funciones podría extraerse fácilmente como un componente reutilizable.

Proporcionado ejemplos concretos de los análisis de esta herramienta. Estos ejemplos provienen de la aplicación de una herramienta específica como un punto de referencia HPC real, (Panas, T., Quinlan, D., & Vuduc, R., 2007). siendo necesario la realización de esta investigación ya que mediante ella se busca estimular la discusión sobre el papel de la retroalimentación interactiva sobre la calidad del código durante el desarrollo de aplicaciones HPC, con herramientas que sirve como un ejemplo constructivo de lo que es posible.

Desarrollo

Código de la métrica de calidad y análisis

Yworks presenta al usuario una variedad de métricas y análisis de calidad de código, de los cuales esta sección proporciona ejemplos concretos. Salvo que se indique lo contrario, generamos estos ejemplos aplicando a las herramientas el código de referencia SMG2000, el cual posee 50 mil líneas de códigos en paralelo que implementa el método de semi referencia de multi redes (Karavanic, L., May, J., Mohror, K., Miller, B., Huck, K., Knapp, R., & Pugh, B., 2005). Para la realización de esta investigación los autores analizaran el SMG2000 en base a 3 categorías: en orden creciente de contexto sobre el código que cada uno requiere. En términos generales, cuanto más contexto se necesitará para comprender una métrica o análisis, mejor se adaptará a la visualización estática o incluso interactiva las categorías de análisis son:

- Estadísticas simples
- Informe de coherencia
- Visualizaciones interactivas

Estadísticas simples

Yworks proporciona muchas métricas y análisis que se pueden representar simplemente como un número único o una lista textual corta, como las líneas de código, el número de variables globales o el número de llamadas a funciones (medidas estáticamente), entre otras. Las tablas 1–3 muestran varios ejemplos.

Las estadísticas calculadas en la Tabla 3 se encuentran entre las estadísticas simples más interesantes. Por ejemplo, las herramientas de análisis pueden verificar ciertos tipos de violaciones de seguridad relacionadas con el uso indebido de API e informa la complejidad ciclomática de McCabe. (Harriette-Cabrera, D., & Sanamé-Álvarez, D, 2014). En su caso, el usuario puede seleccionar umbrales particulares en estas estadísticas, y la herramienta informará las infracciones en consecuencia.

Tabla 1. *Estadísticas globales simples*

Propiedad	Valor	Observaciones
Número de líneas de código fuente.	46.000	Basado en el tamaño del árbol de síntesis abstracta, sin incluir las bibliotecas externas
Número de líneas de código fuente. (wc)	28.301	Basado en el tamaño del árbol de síntesis abstracta, sin incluir las bibliotecas externas
Número de archivos	71	Código fuente y encabezamiento
Número de archivos (Código Fuente)	53	
Número de archivos (Encabezamiento)	18	
Número de nodos del árbol de síntesis abstracta	515.030	
Número de nodos del árbol de síntesis abstracta	858.660	Incluyendo las extensiones a las bibliotecas externas
Número de Clases	0	
Número de Estructuras	33	
Número de Uniones	1	
Número de Funciones	394	Solo SMG2000
Número de Llamadas de funciones	1.268	

Fuente: Los autores

Tabla 2. *Variables estadísticas*

Propiedad	Valor	Observaciones
Número de variables locales	3.726	
Número de variables de clase	0	
Número de variables de clase Público	0	
Número de variables de clase Estática	0	
Número de variables de clase Estructuras	245	
Número de variables de clase Unión	2	
Número de variables de clase No miembro	2	Globales
Número de variables de clase No miembro estática	2	Acceso a archivos

Fuente: Los autores

Tabla 3. *Resultados de análisis simples*

Análisis	Limite	Números de violaciones
Seguridad	0	4
Complejidad ciclomática de McCabe por función	<20	20
Complejidad ciclomática de McCabe por archivo	<100	4
Variable global	0	2
Líneas de código fuente por función	<200	19
Dependencia cíclica (funciones)	0	3
Dependencia cíclica (archivos)	0	1
Convención de nombres	<3 char	53
Total		106

Fuente: Los autores

Informe de coherencia

Además de las estadísticas simples, es necesario utilizar herramientas que puede informar sobre la existencia de incoherencias. Destacándose dentro de las que existen Yworks ya que presenta un informe de coherencia en forma de imágenes estáticas.

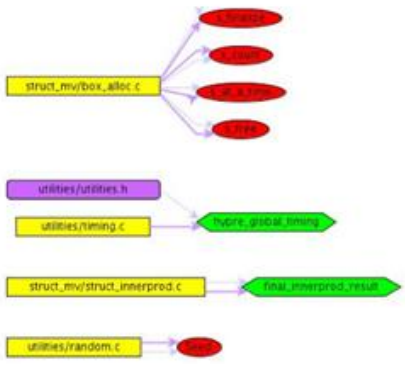

Figura 1 *Gráfico de acceso variable.*
Fuente los autores https://www.yworks.com/products/yed/download#download

Por ejemplo, considere el gráfico de acceso variable en la Figura 1, que muestra las coherencias entre las variables globales (óvalos y hexágonos) y los archivos de origen que los hacen referencia (cuadros) como bordes. La Figura 1 revela la existencia de dos variables globales (hexágonos) y cinco variables con alcance de archivo (es decir, variables declaradas como estáticas en el alcance global de un archivo). Las dos variables globales verdaderas se usan cada una solo dentro de un único archivo fuente, Una de las variables, hypre global timing, también aparece en un archivo de encabezado, pero solo cuando se compila condicionalmente para su inclusión en el archivo fuente correspondiente que se muestra. y, por lo tanto, podrían eliminarse convirtiéndolas, por ejemplo, en variables con ámbito de archivo. Sin embargo, si hacerlo o no depende del desarrollador; esta herramienta solo revela esta oportunidad.

Como otro ejemplo, se debe considerar el gráfico de inclusión de archivos para SMG2000 que se muestra en la Figura 2, que muestra las coherencias entre los archivos de origen (cuadros amarillos) y los archivos de encabezado, donde se distingue entre los archivos de encabezado del programa (cuadros verdes) y los encabezados del sistema (cuadros grises). Un desarrollador podría usar esta imagen para determinar, por ejemplo, si existe un intercambio no deseado de archivos de encabezado entre componentes lógicamente distintos. (No se muestran tales casos en esta figura)

.

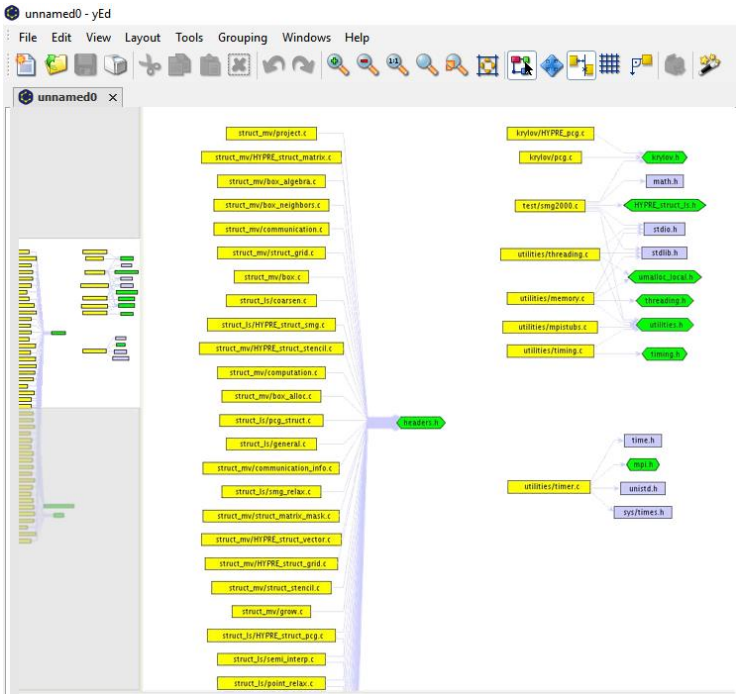

Figura 1 *Grafico de inclusión de archivo.*
Fuente los autores https://www.yworks.com/products/yed/download#download

Visualizaciones interactivas

En algunos casos, los volcados de texto e incluso las imágenes estáticas pueden no iluminar los problemas para el desarrollador. Yworks como herramienta puede proporcionar visualizaciones 3D interactivas para dicha información, y utilizar metáforas visuales familiares del mundo real para ayudar a la comprensión y comprensión del programa en estas visualizaciones.

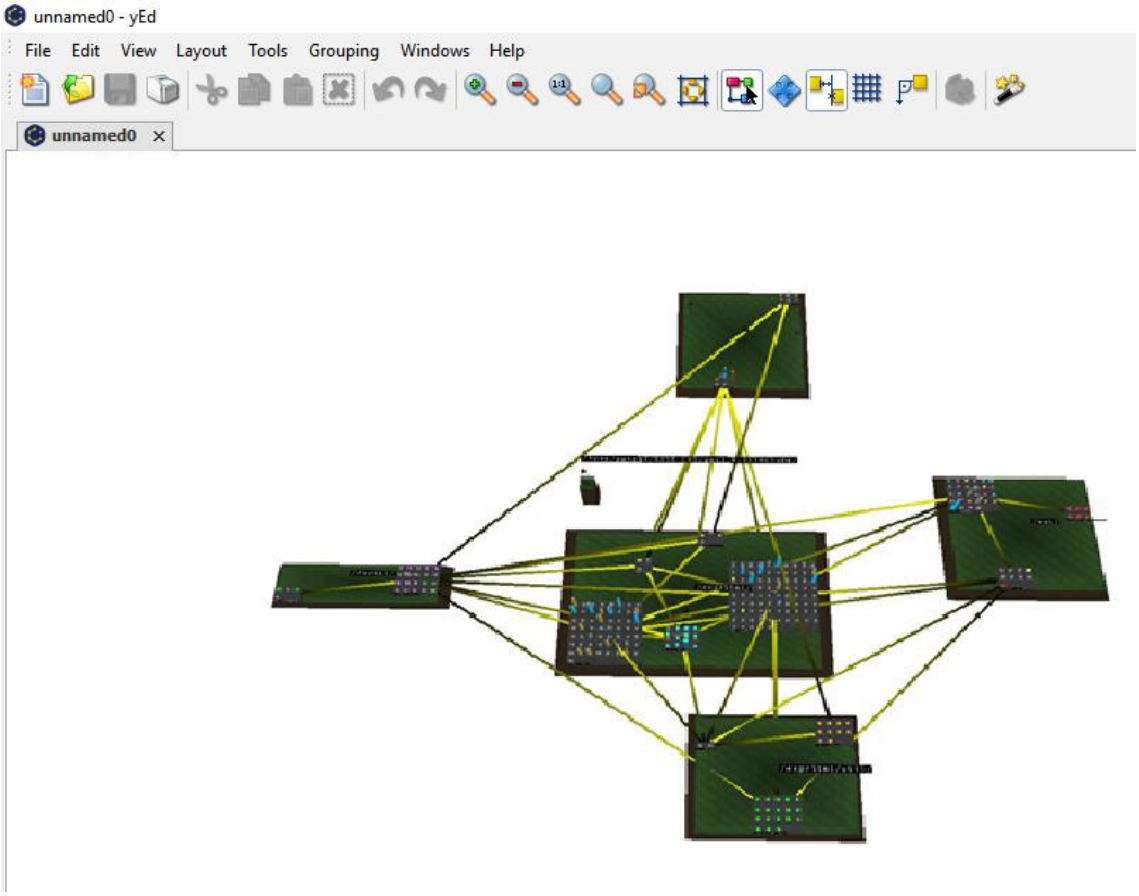

Figura 3 *Visualización interactiva arquitectura de software.*
Fuente los autores https://www.yworks.com/products/yed/download#download

Tal imagen le da al desarrollador una idea de la arquitectura general del software. Los cinco directorios de este submódulo aparecen como cinco islas, archivos individuales como ciudades dentro de la isla y las definiciones de funciones individuales como edificios. Además, los bordes sombreados agregados entre ciudades indican que algunas funciones en un archivo (extremo amarillo) llama a alguna función en otro archivo (extremo oscuro). Otras métricas seleccionadas por el usuario y análisis estáticos o dinámicos se pueden representar como texturas, colores e íconos en esta vista (no se muestra; estos elementos aparecen al hacer zoom para ver las estructuras más de cerca de lo que se muestra en la Figura 3).

De la Figura 3, un desarrollador puede ver rápidamente si su concepto lógico de componentes en el sistema se mantiene. Por ejemplo, la isla más a la izquierda solo tiene bordes entrantes, lo que implica que, de hecho, es un componente reutilizable. Si el desarrollador decide hacer cambios en el sistema para mejorar la calidad o la estructura de los componentes, puede comparar visualizaciones antiguas y nuevas para hacer un juicio rápido.

Implementación de herramientas

El sistema de visualización de la herramienta Yworks combina las herramientas que se muestran en la Figura 4. Utilizando ROSE como la interfaz en un lenguaje de programación C++. (Schordan, M., & Quinlan, D., 2003) ROSEVA extiende la API ROSE para su uso dentro del marco VizzAnalyzer (Löwe, W., Ericsson, M., Lundberg, J., Panas, T., & Pettersson, N., 2003). ROSEVA tiene dos

interfaces principales, una para el análisis de código C / C ++ (recuperación de fuente y construcción AST) y otra para análisis de alto nivel. Dado que ROSE y ROSEVA se desarrollan en C / C ++ y VizzAnalyzer en Java, los programadores utilizaron Babel para conectar estos mundos (Naquid, Q., Jiménez, M., & Compeán, G., 2014). Finalmente, usan Vizz3D (Löwe, W., & Panas, T., 2005) para visualizar los resultados.

ROSE

ROSE es una infraestructura abierta para la construcción de fuente a fuente de análisis y transformación de las herramientas basadas en compilador. Para C y C ++, Rose es totalmente compatible con todas las características del lenguaje, conserva toda la información de la fuente para su uso en el análisis, y permite la traducción arbitrariamente compleja a nivel de fuente a través de su sistema de reescritura. Aunque la investigación en el proyecto ROSA hace hincapié en la optimización del rendimiento, ROSE contiene muchos de los componentes comunes a cualquier infraestructura de compilador, y por lo tanto apoya el desarrollo de herramientas de análisis y transformación basada en la fuente general.

ROSEVA

ROSEVA ha sido diseñado como una biblioteca mediante la API de ROSE. ROSEVA tiene dos interfaces que VizzAnalyzer accede a través de Babel: recuperación y análisis. La recuperación toma los archivos para ser analizados por ROSE como entrada y construye internamente una representación AST de los archivos de origen C/C++. Para la interacción del usuario, los gráficos seleccionados son devueltos a VizzAnalyzer. Estos resultados pueden ser visualizados directamente o inspeccionados adicionalmente con análisis adicionales.

Por lo tanto, ROSEVA proporciona una segunda interfaz que permite a VizzAnalyzer acceder a una variedad de análisis en cualquier gráfico producido por el frontend. Analiza los resultados, tales como la complejidad ciclomática de McCabe o el número de líneas de códigos fuente, se alimentan de nuevo directamente como propiedades al VizzAnalyzer. Estas propiedades son utilizadas por el VizzAnalyzer de visualizar de forma flexible cualquier tipo de información del programa.

Babel

Babel es una herramienta para la mezcla de C/C++, Fortran77, Fortran90, Python, y Java en una sola aplicación. Babel es la base para un marco componente científico en varios idiomas. Babel aborda el problema del idioma interoperabilidad utilizando técnicas de lenguaje de definición de interfaz (IDL). En particular, Babel utiliza un lenguaje de definición de interfaz de la Ciencia (del SIDL) que responda a las necesidades únicas de la computación científica en paralelo.

VizzAnalyzer

El marco VizzAnalyzer es utilizado en Yworks como sistema de composición para la ingeniería inversa, el apoyo a la composición rápida de las herramientas de ingeniería inversa de software individuales mediante la reutilización de componentes de ingeniería inversa arbitrarias. VizzAnalyzer distingue dos dominios: recuperación y análisis. Los programas son parte del dominio de análisis de software, así como la visualización del programa es parte del dominio de la visualización de información

(Löwe, W., Ericsson, M., Lundberg, J., Panas, T., & Pettersson, N., 2003).

Vizz3D

Vizz3D es un sistema de visualización de la información 3D. Se presenta la estructura del sistema y la información de calidad a un usuario de una manera comprensible y aprovecha la comprensión de ese sistema. Vizz3D es altamente flexible y permite a los usuarios definir y volver a asignar algoritmos de diseño y metáforas en tiempo de ejecución. Por lo tanto, las visualizaciones pueden ser configurados en línea. Esto también permite un análisis de software interactivo e iterativo, donde las vistas apropiadas se crean bajo demanda.

Conclusiones

En la presente investigación, se busca extender y evaluar la herramienta de retroalimentación de calidad del código mediante la colaboración con los equipos de desarrollo. El trabajo en esta última zona es, por supuesto, fundamental para determinar cuán útil podría ser este enfoque, en particular para las prácticas comunes de desarrollo de aplicaciones HPC.

La versión actual de los informes de la herramienta en una amplia colección de métricas y análisis. Por otra parte, estos informes (texto, imagen estática, y visualización interactiva) pueden incluir una variedad de métricas de interés específico para los desarrolladores de HPC. Por ejemplo, se puede informar sobre recuentos estáticos de operaciones de punto flotante dentro de cada función, o cualquier recuento de procesadores de hardware recogidos en tiempo de ejecución a través de una interfaz en ROSE a la HPCToolkit, aunque no se muestra estas en la investigación. También se está implementando un análisis para extraer información del repositorio, por ejemplo, desde el CVS ampliando activamente las métricas y análisis.

Referencias Bibliográficas

Carver, J., Hochstein, L., Kendall, P., Nakamura, T., Zelkowitz, V., Basili, R., & Post, E. (2006). Observations about software development for high end computing. CTWatch Quarterly, 2(4A), 33-37.

De Giusti, E., Tinetti, G., Naiouf, M., Chichizola, F., De Giusti, C., Villagarcía Wanza, H. & Eguren, S. (2016). Arquitecturas multiprocesador en HPC: software, métricas y aplicaciones. In XVIII Workshop de Investigadores en Ciencias de la Computación (WICC 2016, Entre Ríos, Argentina).

Harriette-Cabrera, D., & Sanamé-Álvarez, D. . (2014). Aplicación informática para obtener la Complejidad Ciclomática de algoritmos que utilicen estructura if y while. . Ciencia & Futuro, 4(2), , 64-75.

Karavanic, L., May, J., Mohror, K., Miller, B., Huck, K., Knapp, R., & Pugh, B. (2005). Integrating database technology with comparison-based parallel performance diagnosis: the PerfTrack performance experiment management tool. In SC'05: Proc. In SC'05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing (págs. p.39-39). IEEE.

Larman, C., & Basili, R. (2003). Iterative and incremental developments. a brief history. Computer, 36(6),, 47-56.

Löwe, W., & Panas, T. (2005). Rapid construction of software comprehension tools. International Journal of Software Engineering and Knowledge Engineering, 15(06),, 995-1025.

Löwe, W., Ericsson, M., Lundberg, J., Panas, T., & Pettersson, N. (2003). Vizzanalyzer-a software comprehension framework. In Third Conference on Software Engineering Research and Practise in Sweden,. Sweden.: Lund University.

Naquid, Q., Jiménez, M., & Compeán, G. (2014). The babel file system. In 2014 IEEE

International Congress on Big Data (págs. p.234-241). IEEE.

Panas, T., Quinlan, D., & Vuduc, R. (2007). Tool support for inspecting the code quality of hpc applications. In Proceedings of the 3rd International Workshop on Software Engineering for High Performance Computing Applications (pág. p.2). IEEE Computer Society.

Schordan, M., & Quinlan, D. (2003). (2003, August). A source-to-source architecture for user-defined optimizations. In Joint Modular Languages Conference (págs. p. 214-223). Berlin, Heidelberg.: Springer.