Revisión del Informe de Gonzalo Marco sobre Deuda Técnica en Genexus

Por Yonathan Benelli

En cuanto al concepto de deuda técnica, el compañero define que "*refiere de acciones o decisiones al momento de desarrollar o mantener un sistema que pueden o no afectar resultados en el futuro"* si por desarrollar se entiende todo el proceso de creación del software entonces es un punto que se comparte, en cambio sí por desarrollar se entiende solo la parte en la que interviene un desarrollador entonces se discrepa, por mi parte se entiende que el concepto deuda técnica abarca todos las etapas del proceso de creación de software desde las decisiones al momento de concepción y definición, como arquitectónicas, técnicas, de codificación, testing, documentación ,etc.

Por otro lado el hecho de que puedan o no afectar resultados en el futuro es un poco ambiguo ¿a que se refiere con resultados? Costos económicos, rendimientos, errores, problemas de otra índole, personalmente interpreto DT como si decisiones que en el futuro podrían llegar a generar problemas, dificultades o mayores esfuerzos en alguna etapa o tarea del proceso de software que son invisibles al usuario final y que de haber tomado otra decisión no se generarían.

En el siguiente párrafo, comenta que *"en principio se desarrollan con la idea de generar una solución fácil y rápida y probablemente para así terminar una versión lo antes posible",* aquí se hace mención solamente a las DTs que son incurridas principalmente a nivel de código y que son adrede, lo cual no hace que necesariamente sean las únicas DTs que se puedan incurrir, pueden ser inconscientes, por omisiones, por desconocimiento y por sobre todo no tiene porque ser en esta etapa del proceso.

Se comparte la dependencia del contexto, se comparte que los problemas adjudicados no surjan en el momento pero se discrepa con *"ni se deban"*

Aunque en lo anterior no lo plasma, en la separación en dos tipos hace referencia a la no intencionalidad de la generación de deuda técnica.

Pero esta segmentación a mi entender es solo una de las tantas posibles, como se pudo apreciar en el Metamodelo y se ha venido discutiendo en clase.

Por el lado de donde se encuentran, se retoma la asociación de *"atajos"* lo cual no sería lo más acertado teniendo en mente la idea que pueden ser intencionales o no, o que incluso sea una decisión o acción que en ese momento de tiempo y lugar de proceso sea considerada el mejor modo de proceder.

En cuanto a la lista donde se puede encontrar, se está de acuerdo que esos ítems figuran dentro de los pasos o áreas del proceso de creación de software aunque no son los únicos.

Cuando se habla de la manifestación, no me queda claro, me da a entender que se siguiera hablando de donde se encuentra, además el énfasis en error o no ejecutar todos los casos de prueba da a entender que siempre hablamos de problemas puntuales detectables, de fallas u omisiones.

De mi parte la manifestación más notoria de las DTs es en cuanto a costo a nivel de tiempo, esfuerzo, planeamiento o directamente imposibilidad de solución cuando se está en un punto o parte del proceso de software en la cual se deba realizar algún tipo de acción, por eso podríamos decir que aquel software que no se va a modificar nunca más en la vida o que no se va a analizar de ningún punto de vista o que está obsoleto, es el único que dadas esas circunstancias de forma permanente no tendría DTS.

En cuanto a la percepción de los desarrolladores, los code smells podrían ser una de las tantas formas de percepción, pero volviendo a lo anterior el hecho que al desarrollador le implique mayor trabajo, rehacer partes del código, etc. es la forma en que un desarrollador pueda detectarla, si es a lo que nos referimos con como la perciben.

Si hablamos de percepción en el sentido de opinión, ahí deberíamos ahondar en si para ellos es algo común, si es algo que en base a su opinión les afecta en el día a día, si siquiera saben lo que es, o si no tiene trascendencia para ellos.

Los ejemplos y tipos de deuda técnica en general me parece correcta la tipificación, claramente no es taxativa y existen otros tipos, lo que si me resultan los ejemplos muy centrados en la calidad interna y esto tendería a hacer pensar que las DTs son básicamente problemas en la calidad interna del software y no es así, las decisiones u acciones tomadas que en futuro puedan desencadenar en deuda técnica no necesariamente se deben a calidad, pueden ser a nivel de procesos, a nivel de limitaciones en el software, puede ser a nivel de forma de trabajo (individual o colaborativa) puede ser sobre casi cualquier aspecto que esté relacionado al proceso de desarrollo, desde personas, a metodología, pasando por diseño, arquitectura, codificación, mantenimiento, instalación, testing, etc. que reitero se puede estar convencido de que se está haciendo el software de la mejor calidad y estar en lo correcto pero de todas formas incurrir en deuda técnica en el futuro, porque esa es la cuestión la DTs se manifiestan en el futuro no en el hoy momento de creación del software.

En cuanto a puntualmente los tipos en Genexus, aportan en mi caso, dado que hace un par de años que no trabajo con él, y me es difícil recordar casos puntuales o diferenciarlos y que sean exclusivos de Genexus.

En lo que refiere a la Gestión de deuda técnica, me parece correcta la incorporación de la tabla, no tanto así el uso de herramientas que se ejecuten sobre el código generado por Genexus en el contexto que creo se dio por entendido de que asumimos las plataformas low code no generarían DTs a nivel de codificación.

Estoy de acuerdo con lo planteado sobre la Wiki, aunque la comunidad de Genexus y la Wiki podrían ser amplias, me he encontrado muchas veces en situaciones en las cuales no fue fácil encontrar soluciones a un problema como puede ser con otras tecnologías, lo que sin duda derivo en workarounds y sus consecuencias.

Sobre el Metamodelo, coincido en que todos los elementos aparecen en el contexto de genexus y esto es lo que lo dificulta, desde mi punto de vista no veo a Genexus como una tecnología separada o aislada del resto a nivel de DTs sino que interviniendo en todos los puntos de forma general pero teniendo alguna particularidad.

Me parecen acertadas las menciones a los elementos sobre Ocurrencia, Factores e Influencias, la particularidad de la designación de roles no sé qué tan cierto sea que se requieran menos roles, tal vez con algún ejemplo pudiera quedar más claro.

En conclusión, comparto mucha de las cosas mencionadas en el informe, solo considero que por momentos deja la sensación de que siempre hay intencionalidad y de que se sabe que se está incurriendo en deuda técnica y que el costo siempre es por lado de los errores, aunque a medida que se avanza en el informe se tratan otros temas. Así como mucha capaz se nota mucha referencia a calidad interna del software.

Por otro lado noto la mención reiterada a la palabra desarrollo a la hora de hablar de Dts y capaz confunde un poco, tal vez se puede sustituir por generación, impacto, implicancia, en mi opinión solo me resulta raro el hablar de desarrollo de deuda técnica me suena demasiado ambiguo.

Hay muy buenos aportes de especificidad en ejemplos de genexus, que como comentaba anteriormente, al estar ya un poco olvidado ayuda a recordar los problemas con los que uno se enfrentaba y de esa forma tratar de descubrir o redescubrir que puede ser deuda técnica en el contexto especifico de Genexus.