Estudio de Simulación para la Determinación Óptima de Desarrolladores en un Proyecto con Tareas de Diferente Criticidad

Bibé, Delfina
Caggiano, Juan Cruz
Pessina, Mariano Andrés
Sanchez, Tomas Agustín
Universidad Tecnológica Nacional, Facultad Regional Buenos Aires

Abstract
Este estudio presenta una simulación que tiene como objetivo determinar la cantidad óptima de desarrolladores en un proyecto donde las tareas se separan en criticidad alta y baja. Se ha observado que el 41% de los tickets corresponden a tareas de alta criticidad, mientras que el 59% restante son de baja. Para gestionar estos tickets, se implementaron dos colas según su importancia y se utilizó un equipo compuesto de desarrolladores senior (NPS) y junior (NPJ). Los juniors se encargan exclusivamente de atender tickets de baja criticidad, mientras que los seniors priorizan los de mayor importancia, pero pueden tomar tareas menos críticas si su carga de trabajo lo permite. La simulación busca minimizar la cantidad de desarrolladores, en especial los seniors, optimizando cuatro métricas principales: porcentaje de permanencia en el sistema, tiempo ocioso de los desarrolladores, tiempo de espera en las colas y porcentaje de tickets de baja prioridad tomados por seniors.

Palabras Clave
Simulación, optimización, desarrolladores, criticidad, proyectos de software.

Introducción

La asignación eficiente de recursos humanos en un proyecto de software es crucial para su éxito. A menudo, los proyectos contienen tareas de diferente criticidad, lo que demanda un uso diferenciado de desarrolladores según sus habilidades y capacidades. En este estudio, se busca encontrar la cantidad óptima de programadores seniors y juniors que minimicen los costos y mantengan un balance adecuado en el trabajo, evitando sobrecargar a los seniors con tareas de baja criticidad. La simulación desarrollada aborda este problema utilizando metodologías de evento a evento, simulando un entorno con colas separadas para tickets de alta y baja criticidad.

Elementos del Trabajo y metodología

La metodología se basa en la implementación de una simulación de eventos discretos. El sistema tiene dos colas, una para tickets de criticidad alta (NPH) y otra para tickets de baja criticidad (NSL). Los desarrolladores se dividen en seniors y juniors, donde los seniors priorizan los tickets críticos pero pueden resolver los de baja criticidad si están ociosos. Las variables exógenas son el tiempo de atención de tickets (TAS y TAJ) y el intervalo entre la llegada de nuevos tickets (IA). Las variables de control son la cantidad de seniors (NPS) y juniors (NPJ). El objetivo es encontrar la combinación óptima que minimice los tiempos de espera en las colas y maximice el tiempo ocioso permitido, para evitar el burnout. A continuación, se presenta una tabla con las variables identificadas:

Clasificación de variables	
Datos	IA: Intervalo entre arribos de tickets en minutos TAS: Tiempo de atención del ticket de seniors en minutos TAJ: Tiempo de atención del ticket de juniors en minutos
Control	NPS: Número de seniors resolviendo tickets

	NPJ: Número de juniors resolviendo tickets
Estado	NSH: Cantidad de elementos en la cola de tickets criticidad alta NSJ: Cantidad de elementos en la cola de tickets criticidad baja
Resultado	PPS: Promedio de permanencia en el sistema por semana PTOS(i): Porcentaje de tiempo ocioso senior PTOJ(j): Porcentaje de tiempo ocioso junior PECS: Promedio de espera en cola seniors PECJ: Promedio de espera en cola juniors PTTS: Porcentaje de tickets bajos tomados por seniors

Tablas 1. Clasificación de variables

Una vez finalizado el análisis de las variables se identificaron los eventos que actúan sobre el sistema. En esta metodología, un evento es un hecho o acontecimiento que se produce en el sistema y modifica las variables de estado del mismo. Para presentar los eventos identificados se muestran a continuación la Tabla de Eventos Independientes (TEI) y la Tabla de Eventos Futuros (TEF).

Evento	EFnC	EFC	Condición
Llegada	Llegada	Salida Senior(i)	Tickets de criticidad alta <= Número de seniors atendiendo tickets || (Tickets de criticidad baja > Número de juniors atendiendo tickets && Tickets de criticidad alta <= Número de seniors atendiendo tickets)
		Salida Junior(j)	Tickets de criticidad baja <= Número de juniors atendiendo tickets
Salida Senior(i)	-	Salida Senior(i)	Tickets de criticidad alta > Número de seniors atendiendo tickets || (Tickets de criticidad baja > Número de juniors atendiendo tickets && Tickets de criticidad alta > Número de seniors atendiendo tickets)
Salida Junior(j)	-	Salida Junior(j)	Tickets de criticidad baja > Número de juniors atendiendo tickets

Tablas 2. Tabla de eventos independientes

En las Tablas 2 y 3, tanto “i” como “j” toman valores entre 1 (inclusive) y la cantidad de desarrolladores (inclusive)

Tabla de eventos futuros
TPLL: Tiempo de próxima llegada: Tiempo de próxima llegada de un ticket al backlog (minutos).
TPSSenior(i): Tiempo de próxima salida senior(i): Tiempo de próxima salida de un ticket resuelto por un desarrollador senior (minutos).
TPSJunior(j): Tiempo de próxima salida junior(j): Tiempo de próxima salida de un ticket resuelto por un desarrollador junior (minutos).

Tablas 3. Tabla de eventos futuros

Luego, para obtener los datos, usamos Kaggle, del cual obtuvimos un dataset con datos de tickets de Jira de un equipo de desarrollo.

Luego de obtener los datos del sistema, se realizó un análisis del comportamiento para poder generar las funciones de densidad de

probabilidad (fdps). Estas corresponden a las variables de datos mencionadas anteriormente. Las mismas fueron obtenidas mediante funciones de bibliotecas de Python (Pandas, Numpy, SciPy/Stats, Fitter). Usando Fitter pudimos obtener la distribución que se ajusta mejor a los datos que tenemos para poder obtener las funciones de densidad de
probabilidad (fdps). A su vez realizamos una limpieza de datos utilizando las técnicas de probabilidades "z-score" para eliminar aquellos registros alejados tres veces de la desviación estándar, tanto para intervalo entre arribos como para los tiempos de atención.

En la figura 1 se puede observar la
tendencia de los intervalos entre arribos de los tickets, la cual se ajusta con menor error a la distribución "halflogistic"

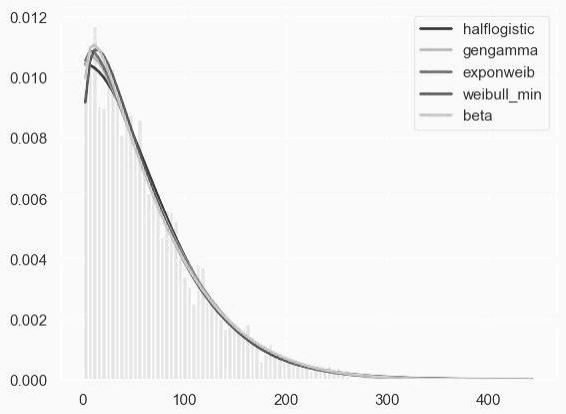


Figura 1. FDP del intervalo entre arribos de tickets

De la misma forma se pueden observar las FDPs del tiempo de resolución de tickets para juniors y para seniors. Las distribuciones que más se adaptaban a nuestros datos eran "gennorm" y "jf_skew_t" respectivamente, tal y como se puede ver en las figuras 2 y 3

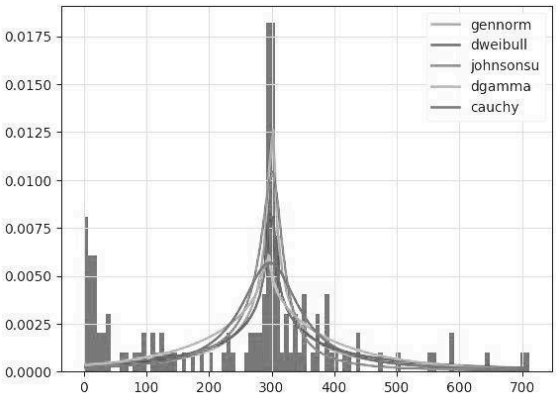


Figura 2. FDP del tiempo de resolución de tickets de juniors

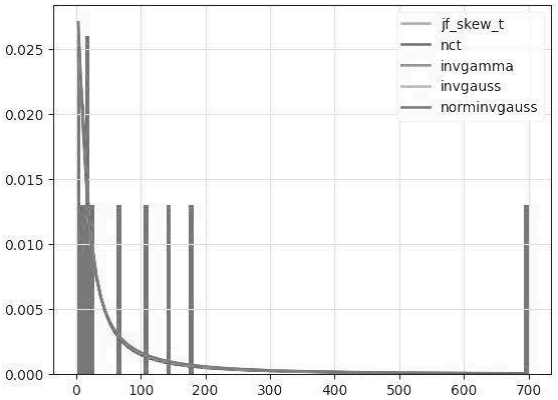


Figura 3. FDP del tiempo de resolución de tickets de seniors

Luego de obtener las FDPs pasamos a construir el programa para realizar la simulación. El trabajo fue realizado en su totalidad con Python utilizando Jupyter Notebooks

Resultados

La simulación arrojó diferentes resultados según las combinaciones de desarrolladores senior y junior. A continuación, se presenta un análisis comparativo entre cada escenario:

	N°1	N°2	N°3	N°4
Cantidad de programadores seniors	2	2	3	4
Cantidad de programadores juniors	2	3	2	1
Resultados				
Promedio de permanencia en el sistema	28.80 días	24.15 días	15.04 días	9.9 días

Promedio de espera en cola de seniors	3.02 días	2.61 días	0.9 días	0.35 días
Promedio de espera en cola de juniors	3.16 días	1.80 días	1.6 días	0.63 días
Porcentaje de tiempo ocioso seniors	0.0%	0.11%	3.81%	30.45 %
Porcentaje de tiempo ocioso juniors	0.01%	9.69%	4.48%	21.13 %
Porcentaje de tickets de criticidad baja tomados por seniors	19.61%	1.56%	34.34%	40.45 %

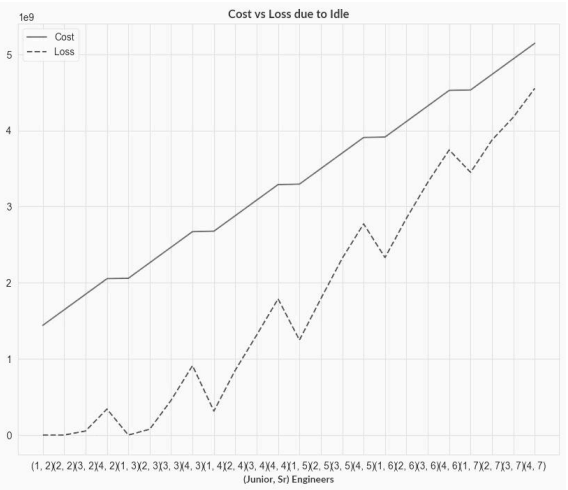
Tablas 4. Escenarios planteados y resultados (parte 1)

	N°5	N°6
Cantidad de programadores seniors	5	6
Cantidad de programadores juniors	1	4
Resultados		
Promedio de permanencia en el sistema	5.43 días	5.04 días
Promedio de espera en cola de seniors	0.09 días	0.22 días
Promedio de espera en cola de juniors	1.3 días	0.24 días
Porcentaje de tiempo ocioso seniors	39.65%	88.36%
Porcentaje de tiempo ocioso juniors	17%	55.86%
Porcentaje de tickets de criticidad baja tomados por seniors	50.96%	21.72%

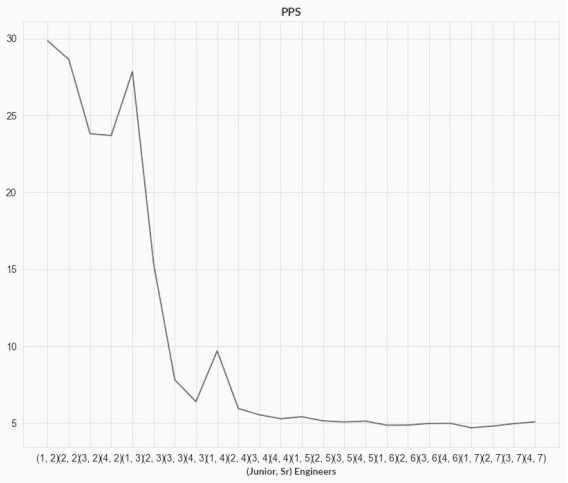
Tablas 5. Escenarios planteados y resultados (parte 2)

Discusión

La simulación demostró que aumentar el número de desarrolladores no siempre conduce a una mayor eficiencia. Aunque la lógica inicial podría sugerir que más desarrolladores resolverán tickets más rápidamente, el escenario con 6 desarrolladores seniors y 4 desarrolladores juniors, nos dice que el porcentaje de tiempo ocioso fue de 88.36% y 55.86% respectivamente. Esto se debe a que un alto número de desarrolladores provocarán mucho tiempo ocioso, resultando más costosos que los beneficios que trae si observamos el promedio de permanencia en el sistema y la espera en cola. Este escenario extremo sirve para enfatizar la importancia de un equipo adecuadamente dimensionado.



Podemos ver cómo a medida que tenemos más desarrolladores ociosos se refleja en pérdida de dinero.



Gráficamente, se observa como la permanencia en el sistema deja de disminuir notablemente, pero el costo y pérdida aumentan.

Estos resultados sugieren que un número superior a 3 seniors no es necesario para optimizar el sistema, ya que el tiempo ocioso y el costo asociado a los desarrolladores seniors es demasiado alto para justificar su inclusión.

Conclusión

Tomando en cuenta los costos promedio anuales de un desarrollador junior y senior, el escenario "2 seniors y 3 juniors" parece ser el más adecuado en términos de balance entre eficiencia y costo. Este escenario minimiza el tiempo en el sistema, evita la sobrecarga de tickets para los seniors y mantiene un tiempo ocioso aceptable para los juniors, lo que asegura su bienestar laboral y productividad.

Agradecimientos

Agradecemos a la profesora encargada del curso Silvia Mónica Quiroga y al ayudante Hernán Darío Martel por darnos el soporte teórico y práctico para realizar este trabajo práctico.

Referencias

Apuntes teóricos de la cátedra de Simulación UTN-FRBA.

Datos de Contacto:

Bibé, Delfina. UTN FRBA.
dbibe@frba.utn.edu.ar
Caggiano, Juan Cruz. UTN FRBA.
jcaggiano@frba.utn.edu.ar
Pessina, Mariano Andres. UTN FRBA.
mpessina@frba.utn.edu.ar
Sanchez, Tomas Agustin. UTN FRBA.
tosanchez@frba.utn.edu.ar