Estudio para averiguar la cantidad de desarrolladores y la frecuencia óptima de actualización de prioridades, para un proyecto con tareas de prioridad cambiante

Fasciolo, Ornella
Spisso, Gabriel
Deheza, Alejandro
Universidad Tecnológica Nacional, Facultad Regional Buenos Aires

Abstract
Una empresa en desarrollo de software está en proceso de planificación para un nuevo proyecto de gran envergadura. Con el objetivo de optimizar la eficiencia del equipo asignado a este proyecto, se propone realizar un análisis con datos del avance de otros proyectos. Utilizando la simulación con metodología evento a evento, se quiso saber la cantidad optima de desarrolladores para el proyecto, la frecuencia con la que se actualizan las prioridades de los tickets y el valor de cada punto de estimación de un ticket. Tras simular varios escenarios, se llegó a la conclusión de que lo mas optimo son 5 desarrolladores, actualizaciones de prioridades cada 2 días y que 1 punto de estimación equivale a 12 horas. Este resultado óptimo permite reducir el tiempo de permanencia en el sistema de los tickets

Palabras Clave
Simulación, metodología de evento a evento, sistemas de colas, optimización, prioridad cambiante.

Introducción

Para llevar a cabo este análisis, se utilizarán datos históricos recopilados a través de la plataforma de gestión de proyectos Jira. Estos datos proporcionan información detallada sobre el intervalo entre arribos de los tickets, el tiempo de espera en el backlog, el tiempo de resolución de los tickets y el puntaje asociado a cada ticket, que estima su complejidad y tiempo de resolución.

El proyecto contará con 4 colas de prioridad distintas: Highest, High, Medium y Low. Cada ticket será clasificado en una de estas colas según su nivel de prioridad.

El objetivo principal es calcular el tiempo promedio de permanencia de un ticket en el sistema, lo que permitirá evaluar la eficacia del equipo en la gestión de tareas pendientes en cada una de las colas de prioridad. Además, se analizará el promedio de tickets resueltos por semana y el promedio de desfase de ticket, lo que proporcionará una visión clara del rendimiento del equipo en términos de productividad y capacidad de cumplimiento de plazos en cada una de las colas de prioridad.

Evento actualización: cuando ocurre este evento se calcula la prioridad de los tickets pendientes en el backlog en base al estimado de tiempo que van a tardar y a cuánto tiempo llevan en el backlog. Si un ticket se puede resolver en poco tiempo y lleva mucho tiempo en backlog, sube de prioridad. Luego del cálculo se mueven los tickets a la cola de prioridad que corresponda.

Este evento actualización posee una frecuencia que está determinada por la variable de control "Tiempo de actualización de prioridades de tickets"

Esperamos poder observar como una cantidad de desarrolladores determinada sube o baja la cantidad de tickets que se resuelven en un lapso de tiempo. Y a su vez, poder observar cómo influye la frecuencia en la que se actualizan las prioridades de los tickets en la cantidad de tickets resueltos

A través de este análisis detallado, la empresa podrá determinar el número ideal de integrantes del equipo y establecer procesos eficientes de gestión de tickets para maximizar la productividad y eficiencia en el desarrollo del software.

Elementos del Trabajo y metodología

Para analizar este problema, se ha elegido la metodología de simulación evento a evento. Esta metodología es utilizada para realizar un avance del tiempo en el modelo por incrementos variables. Así, a lo largo de la línea de tiempo de la simulación, el modelo se ve modificado por eventos que generan cambios en sus variables y permiten su funcionamiento.

Para la formulación del modelo y simulación del caso, se identificaron y analizaron las variables que en él actúan. A continuación, se presenta una tabla con las variables identificadas:

Clasificación de variables	
Datos	Intervalo entre arribos de tickets (IA): El tiempo entre la llegada de cada ticket al backlog del proyecto. [minutos] Tiempo de resolución de un ticket (TR): Tiempo necesario para resolver cada ticket del backlog del proyecto. [minutos] Puntaje de estimación de un ticket (PT): Estimación realizada por los programadores del tiempo que demorara en ser completado un ticket.
Control	Cantidad de desarrolladores (N): Representará el número total de desarrolladores asignados al proyecto. Tiempo de actualización de prioridades de tickets (TAPT): Cantidad de tiempo en la que ocurrirá un evento de actualización de tickets. [minutos] Valor de cada punto de estimación de tickets (VPET): Representará el valor de tiempo que cada punto de estimación tiene
Estado	Cantidad de tickets pendientes en prioridad HIGHEST (NS1): Representa la cantidad de tickets pendientes en el backlog con prioridad HIGHEST en un momento dado. Cantidad de tickets pendientes en prioridad HIGH (NS2): Representa la cantidad de tickets pendientes en el backlog con prioridad HIGH en un momento dado. Cantidad de tickets pendientes en prioridad MEDIUM (NS3): Representa la cantidad de tickets pendientes en el backlog con prioridad MEDIUM en un momento dado. Cantidad de tickets pendientes en prioridad LOW (NS4): Representa la cantidad de tickets pendientes en el backlog con prioridad LOW en un momento dado.
Resultado	Tiempo promedio de permanencia en el sistema de los tickets (PPS): Esto indicará cuánto tiempo en promedio permanece un ticket desde su creación hasta el momento en el que queda resuelto. Promedio de tickets resueltos por semana (PTS): Medirá la eficiencia del equipo en términos de la cantidad de trabajo completado en un período de una semana. Promedio de desfase de ticket (PDT): tiempo promedio que un ticket demora en ser realizado comparado con su tiempo estimado. 0 indica que no hay desfasaje, un numero positivo significa que se tardo menos de lo estimado, un numero negativo indica que se tardo más de lo estimado

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	Condicion
Llegada	Llegada	Salida(i)	tickets en HIGHEST + tickets en HIGH + tickets en MEDIUM + tickets en LOW <= Cantidad de desarrolladores
Salida(i)	-	Salida(i)	
Actualizac	Actualizac	-	

Tablas 2. Tabla de eventos independientes

En las Tablas 2 y 3, "i" toma valores entre 1 (inclusive) y la cantidad de desarrolladores (inclusive)

Tabla de eventos futuros
Tiempo de próxima llegada: Tiempo de próxima llegada de un ticket al backlog (minutos).
Tiempo de próxima salida(i): Tiempo de próxima salida de un ticket resuelto por un desarrollador (minutos).
Tiempo de próxima actualización: Tiempo en el que se actualizaran las prioridades de los tickets en los casos que correspondan (munitos).

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).

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 "johnsonsb"

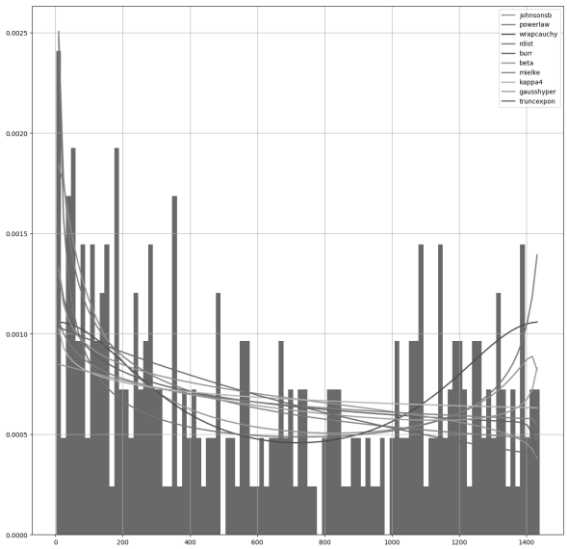

Figura 1. FDP del intervalo entre arribos de tickets

De la misma forma se pueden observar las FDPs del tiempo de resolución de un ticket y del puntaje de estimacion de un ticket. Las distribuciones que mas se adaptaban a nuestros datos eran "weibull_max" y "alpha" respectivamente, tal y como se puede ver en las figuras 2 y 3

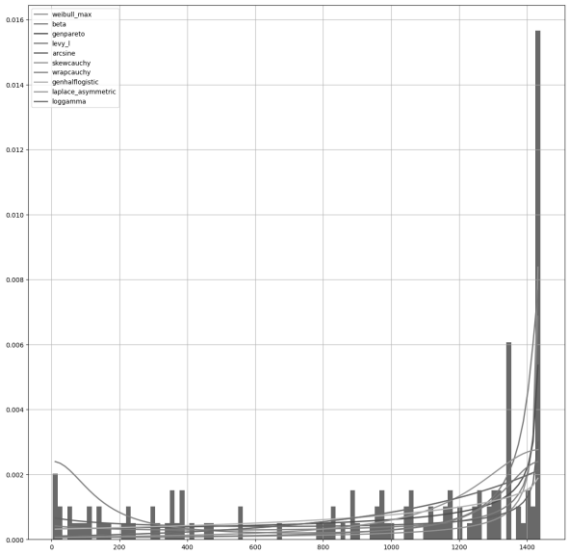

Figura 2. FDP del tiempo de resolución de tickets

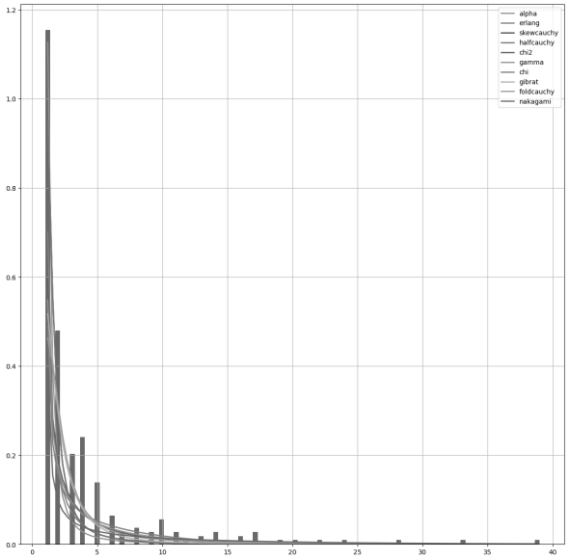

Figura 3. FDP del puntaje de estimación de un ticket

Luego de obtener las FDPs pasamos a construir el programa para realizar la simulación. Utilizamos principalmente Java, pero además generamos un pequeño servidor en Python que llamamos desde el sistema principal para que nos devuelva valores basados en las FDPs. En la Figura 4 se puede ver un esquema básico

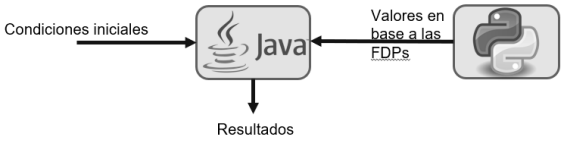

Figura 4. Esquema básico del programa realizado

Resultados

Durante la simulación se analizaron 6 escenarios. El primer escenario es el más simple, contando solo con un desarrollador, 2 días de actualización de prioridades y cada voto son 12 horas de desarrollo. En este caso nos dio que sería una opción inviable ya que se atrasaría mucho el tiempo de desarrollo de tickets, estos van subiendo de prioridad constantemente ya que no son agarrados para su desarrollo a tiempo, haciendo así que el tiempo de permanencia de ticket en el sistema sea 40.8 días.

El segundo escenario que probamos fue con 3 desarrolladores manteniendo el resto de las variables de control estables. Aquí encontramos que, si bien es más eficiente, aun así, hay un desfasaje grande al momento del desarrollo de cada ticket, haciendo que el promedio de desfasaje vaya a 24.6 días de atraso
Luego probamos con 5 desarrolladores, manteniendo constante el tiempo de actualización de prioridad de los tickets y el valor de las estimaciones.
En este caso podemos ver que se volvió más eficiente el agarre de tickets, llevando el tiempo promedio de permanencia en el sistema a tan solo 2.6 días y no tenemos tanto desfase de resolución de ticket, este solo siendo 10.8 horas.
Como próximo caso de prueba, cambiamos la cantidad de desarrolladores a 10, manteniendo el tiempo de actualización y la estimación constantes.
En este último, podemos ver que no necesariamente más desarrolladores significa que se va a volver más eficiente. En este caso el tiempo de permanencia de cada ticket en el sistema subió a 4.9 días, aunque el desfasaje haya aumentado positivamente a 59.1 horas, indicando que se resuelven los tickets más rápido.
En el siguiente escenario, volvemos a seleccionar 5 desarrolladores, pero esta vez decidimos cambiar el tiempo de actualización a 7 días manteniendo la estimación constante. Con esto, pudimos ver que, si bien los tickets se mantienen en el sistema el mismo tiempo que el caso de actualización de 2 días y 5 desarrolladores, podemos ver que el desfasaje es menor, haciendo que los tickets no se resuelvan tan rápido como antes con la misma cantidad de desarrolladores, pasando de 10.8 horas a tan solo 4.8 horas.
Por último, tenemos el escenario donde solo hay 5 desarrolladores, mantenemos el tiempo de actualización a 2 días, pero cambiamos el valor de la estimación de tickets. En este caso un punto equivale a 8 horas. Podemos ver que, en este caso, si bien el tiempo de permanencia en el sistema

es similar, aun así, el tiempo de desfasaje bajo a tan solo 3.8 horas.

	N°1	N°2	N°3	N°4
Cantidad de programadores	1	3	5	10
Actualización de prioridades de tickets	2dias	2dias	2dias	2 días
Valor de cada punto de estimación de tickets	12	12	12	12
Resultados				
Tiempo promedio de permanencia en el sistema de los tickets	40.8 días	1.7 días	2.6 días	4.9 días
Promedio de tickets resueltos por semana	15.8	15.6	15.6	15.7
Promedio de desfase de ticket	-32.6 horas	-24.6 horas	10.8 horas	59.1 horas

Tablas 4. Escenarios planteados y resultados (parte 1)

	N°5	N°6
Cantidad de programadores	5	5
Actualización de prioridades de tickets	7dias	2dias
Valor de cada punto de estimación de tickets	12	8
Resultados		
Tiempo promedio de permanencia en el sistema de los tickets	2.6 días	2.6 días
Promedio de tickets resueltos por semana	15.7	15.7
Promedio de desfase de ticket	4.8 horas	3.8 horas

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 resolverían tickets más rápidamente, el escenario con diez desarrolladores resultó en un aumento del tiempo de permanencia de los tickets en el sistema. Esto sugiere que, más allá de un cierto punto, la adición de recursos humanos puede generar complejidades adicionales que contrarrestan los beneficios esperados.

Asimismo, la frecuencia de actualización de prioridades juega un papel crucial en la eficiencia del sistema. Una actualización cada siete días, en lugar de cada dos, disminuyó el desfase de los tickets, indicando que una frecuencia de actualización demasiado baja puede permitir que los tickets menos urgentes se acumulen y no se resuelvan a tiempo. Sin embargo, una frecuencia de actualización más alta (cada dos días) ayudó a mantener una carga de trabajo más equilibrada y priorizada, reduciendo el tiempo total de permanencia de los tickets en el sistema.

Uno de los resultados esperados fue que con un desarrollador, el tiempo de permanencia de los tickets en el sistema fue extremadamente alto (40.8 días). Esto se debe a que un solo desarrollador no puede manejar la carga de trabajo de manera eficiente, lo que causa un rápido aumento en la acumulación de tareas pendientes. Este escenario extremo sirve para enfatizar la importancia de un equipo adecuadamente dimensionado.

Conclusión

Llegamos como conclusión a que la cantidad optima de desarrolladores es de 5, el tiempo de actualización es de 2 días y la estimación adecuada es de 8 horas cada punto. Sin embargo, de todas formas, decidimos elegir el escenario nro. 3, con 5 desarrolladores, tiempo de actualización de 2 días y el valor de cada punto siendo 12 horas debido a que queremos que nuestros desarrolladores tengan algo de tiempo ocioso para poder resolver tickets mas prioritarios en el caso de surgir, como por ejemplo: bugs de producción.

Agradecimientos

Agradecemos a la profesora encargada del

curso Silvia Monica Quiroga y al ayudante Hernan Dario 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:

Ornella Fasciolo. UTN FRBA.
ofasciolo@frba.utn.edu.ar.
Gabriel Spisso. UTN FRBA.
gspisso@frba.utn.edu.ar.
Alejandro Deheza. UTN FRBA.
dehezagarvizu@frba.utn.edu.ar.