Análisis de métricas de DevOps

Las métricas van a ser analizadas solo para los bugs y features de este proyecto, y no para las tareas extras que hayamos añadido a los tableros, ya que estos últimos no tienen mucho interés.

Observaciones de las métricas

Lead Time & Cycle Time

Para medir estas métricas, se decidió utilizar el equivalente de días en esfuerzo realizado. En un proyecto de la vida real tiene sentido utilizar los días, ya que se entiende que los desarrolladores trabajan regularmente a lo largo de la semana. En este proyecto sin embargo, muchas veces se dio que gran parte del esfuerzo fue realizado en pocos días. Es decir, en vez de haber invertido 3 hs-p en 3 días, quizás se puso un esfuerzo de 9hs-p en un solo día (esto es un ejemplo).

Realizar este cambio es importante para que las métricas posteriores reflejen mejor el trabajo realizado. Para mostrarlo, ponemos un ejemplo. Supongamos que el *Lead Time* es 2 días, y el *Cycle Time* es 1 día, pero el esfuerzo realizado en ese día es 7 hs-p. Si hacemos $\frac{1}{2}$ para el *Flow Efficiency* (explicado más abajo), nos va a dar un 50%. Sin embargo el trabajo realizado claramente implica más que el 50% del tiempo: si asumimos 4 hs-p por día, esto equivaldría a casi el 100% del tiempo si se hubiera repartido en ambos días.

Nosotros vamos a asumir un tiempo de trabajo de 3 hs-p por día para los días en los que no se trabajó en el *Lead Time*. El *Cycle Time* va a ser igual al esfuerzo.

Flow Efficiency

Dado que en el contexto de nuestro proyecto manejar el *Touch Time* fue difícil (empezamos a trabajar, cortamos para comer, salir hacer algo, etc. Es difícil estar constantemente “parando y arrancando el reloj”), el *Flow Efficiency* lo vamos a calcular de la siguiente manera:

$$Flow\ Efficiency\ = \frac{Cycle\ Time}{Lead\ Time}$$

Es decir, sustituimos el *Touch Time* por el *Cycle Time*. Claramente esto es un workaround muy *ad hoc*, que se ajusta únicamente a nuestro contexto.

Ley de Little (Throughput)

La Ley de Little es una métrica que no vamos a utilizar, ya que requiere trabajar con el WIP. El WIP solo tiene sentido trabajarlo si nosotros lo definimos previamente, pero esto es algo que no hicimos. Sin embargo, el throughput no deja de ser importante, y esto sí lo vamos a analizar.

En cuanto al período seleccionado, nos interesó elegir cada entrega para poder comparar los throughput de ambas y ver si se parece mantener cierta tendencia.

Entregas que no necesitan métricas

En la primera entrega nos enfocamos únicamente en analizar la deuda técnica del proyecto a utilizar.

En la cuarta entrega (actual), el enfoque fue en la automatización de tests utilizando Selenium y en el análisis de métricas de DevOps.

Dado que solo vamos a calcular métricas para features y bugs, estas dos entregas no las necesitan (además, métricas básicas como el esfuerzo ya se encuentra en cada entrega).

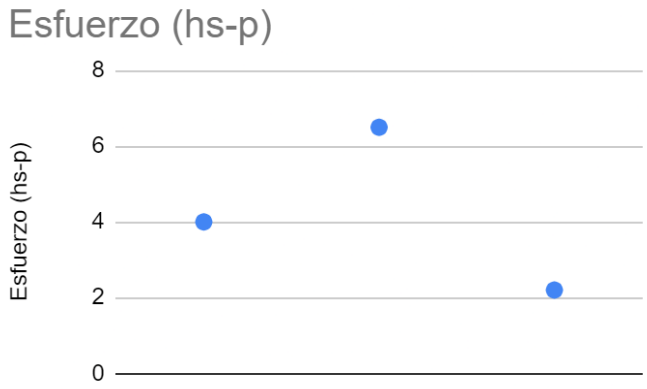
Entrega 2

Lead time, Cycle Time & Esfuerzo

$$Lead\ Time\ =\ Done\ -\ Todo\ Start\ Date$$
$$Cycle\ Time\ =\ Done\ -\ Desarrollo\ Start\ Date$$

Lead time, Cycle Time & Esfuerzo						
US	**Todo Start Date**	**Desarrollo Start Date**	**Done**	**Lead Time**	**Cycle Time**	**Esfuerzo (hs-p)**
Introducir un número de trackeo inexistente muestra un mensaje de error del backend #27	26-Sep-2023	27-Sep-2023	28-Sep-2023	7	4	4
Se pueden agregar cantidad negativa de medicamentos al carrito #18	26-Sep-2023	27-Sep-2023	28-Sep-2023	9.5	6.5	6.5
El sistema permite agregar cantidades negativas a una stock request #12	26-Sep-2023	27-Sep-2023	28-Sep-2023	5.2	2.2	2.2
					Promedio	
				7.233333333	4.233333333	4.233333333

Como podemos ver, las 3 tarjetas tuvieron el mismo Lead Time y Cycle Time, por lo que no hubo desviaciones



En cuanto al esfuerzo, no hubo desviaciones grandes con respecto al promedio.

Flow Efficiency

Flow Efficiency	
Cycle Time	4.233333333
Lead Time	7.233333333
Flow Efficiency	58.53%

Throughput

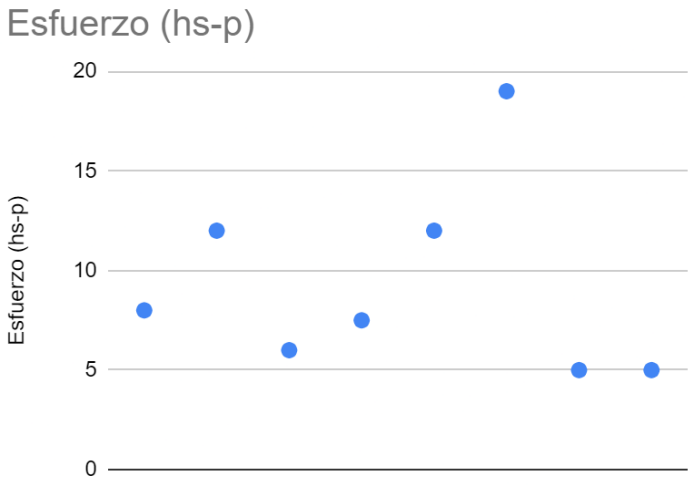
Throughput
3

Entrega 3

Lead time, Cycle Time & Esfuerzo

$$Lead\ Time\ =\ Done\ -\ Todo\ Start\ Date$$
$$Cycle\ Time\ =\ Done\ -\ Requirement\ Definition\ Start\ Date$$

Lead time, Cycle Time & Esfuerzo						
US	**Todo Start Date**	**Requirement Definition Start Date**	**Done**	**Lead Time**	**Cycle Time**	**Esfuerzo (hs-p)**
FE - Modificacion Producto#56	17-Oct-2023	21-Oct-2023	23-Oct-2023	20	8	8
BE - Modificacion Producto#55	17-Oct-2023	23-Oct-2023	23-Oct-2023	30	12	12
FE - Baja Producto#51	17-Oct-2023	21-Oct-2023	23-Oct-2023	18	6	6
BE - Baja producto#52	17-Oct-2023	19-Oct-2023	23-Oct-2023	13.5	7.5	7.5
FE - Alta de Producto#50	17-Oct-2023	17-Oct-2023	23-Oct-2023	12	12	12
BE - Alta de Producto#49	17-Oct-2023	17-Oct-2023	23-Oct-2023	19	19	19
FE - Purchase products#58	17-Oct-2023	23-Oct-2023	23-Oct-2023	23	5	5
BE - Purchase products#59	17-Oct-2023	23-Oct-2023	23-Oct-2023	23	5	5
					Promedio	
				19.8125	9.3125	9.3125



Como se puede ver, en este caso sí hubo una desviación significante. La tarjeta “BE - Baja producto#52” tuvo un esfuerzo de 19 hs-p que está muy por encima del promedio 9.3.

Mirando los registros de esfuerzo, esto se dio porque esta tarjeta estuvo atascada y se necesito a más de un desarrollador para poder avanzarla.

Flow Efficiency

Flow Efficiency	
Cycle Time	9.3125
Lead Time	21.8125
Flow Efficiency	42.69%

Throughput

Throughput
8

Conclusiones

Una de las primeras cosas que pudimos ver, es que en cada entrega la fecha de llegada a la columna Done siempre se mantuvo igual (aunque esto no sea una métrica propiamente dicho, pero sí las afecta). Esto se dio por dos razones. Primero, el equipo siempre definió en una Planning Meeting todas las tareas a realizar, por lo que todas entran al backlog al mismo tiempo. En segundo lugar, y más importante, las US solo pasan a Done cuando previamente se ha tenido una reunión con el PO quien valida (o no) la tarjeta. Sin embargo, nosotros solo realizamos una reunión con el PO por cada una de las entregas al final de la etapa. Esto significó que muchas US quedaron atascadas en la etapa de Review por varios días, ya que independientemente de si una tarjeta sea terminada muchos o pocos días antes del final de la etapa, de todos modos se valida al final de la misma.

Esto es importante porque no refleja muy bien el trabajo que hicimos en las métricas. Dado que las tarjetas están atascadas en Review, el *Lead Time* se agranda “artificialmente”, mientras que el *Touch Time* (esfuerzo en nuestro caso) permanece igual. Esto significa que el *Flow Efficiency* va a parecer más chicos de lo que realmente es. Si estuvieramos midiendo en días, el problema es similar, ya que se agranda el *Lead Time* y *Cycle Time*.

La solución obvia e inmediata es asegurar que el PO valide el trabajo realizado tan pronto como este llegue a la etapa de Review. De esta manera las tarjetas que van siendo finalizadas se van pasando a la columna de Done.

En cuanto al *Flow Efficiency*, podemos notar que es bastante alto. Buscando online, pudimos encontrar que flows de más de 40% son considerados excepcionales. Según las heurísticas que vimos en la clase, algo mayor a 40% es un buen valor. En definitiva, resolvimos que tuvimos un buen valor. La realidad es que no tuvimos muchos bloqueos, y lo que encontramos en internet habla generalmente en un contexto laboral. En ese contexto, muchas veces los

desarrolladores tienen que atender a muchas cosas por fuera del desarrollo de una feature, tal como: reuniones con cliente, Dailys, Retrospectivas, responder mails, etc. Todo esto hace que el *Flow Efficiency* baje. En nuestro caso, todo el tiempo que destinamos al proyecto fue enfocado únicamente a las tareas, por lo que hubieron pocas distracciones.

Por último tenemos al *Throughput*. Lo que nos pareció interesante es intentar observar si parece haber una tendencia en el correr de las entregas. Si consideramos que en la primera entrega realizamos en promedio 7 hs-p y tuvimos un throughput de 3, si realizamos una regla de 3 vemos que en las 21 hs-p de la tercera entrega se esperaría que sea 9. 8 se encuentra dentro de un rango razonable, y más aún considerando que nuestro *Flow Efficiency* es un poco peor en la segunda entrega.