Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2143 — Ingeniería de Software 2020-1

NOMBRE: Matías Duhalde
SECCIÓN: 1

Interrogación 1 — Respuesta Pregunta 1

En primer lugar, para definir los requisitos funcionales, se usaron Relatos de Usuario, que corresponden con metodologías ágiles. Sin embargo, estos fueron hechos por los mismos desarrolladores, a partir de un documento de requisitos entregado por el cliente. Para que los relatos de usuario sean efectivos, estos deben ser escritos por el cliente, y además, es recomendable que estos surjan desde una conversación entre este y el equipo de desarrollo (no de un documento).

En segundo lugar, el flujo de trabajo se plasmó en una carta Gantt. Esto no es recomendable en procesos ágiles como Scrum, debido a que estas son poco mutables y no se adaptan correctamente a la dinámica de los Sprints (las tareas por hacer cambian en cada iteración). Es mucho más recomendable usar otro tipo de diagramas como Burnout Charts y Kanban Boards, en conjunto con un Product Backlog.

En tercer lugar, al dividir los grupos, se estableció un supervisor en cada uno, encargado de comunicarse con otros grupos. Esto en Scrum es contraproducente, debido a que entorpece el proceso de comunicación. Uno de los principios de las metodologías ágiles privilegia la comunicación entre todos los involucrados en el proyecto, y establecer un supervisor atenta contra este principio. Además, con respecto a los grupos, estos deberían ser autogestionados, no comandados por una sola persona.

En cuarto lugar, y relacionado al punto anterior, las reuniones se realizaron sólo entre supervisores y el gerente. Estas debiesen ser entre todos los integrantes del proyecto, con el objetivo de que todos estén al tanto del estado de este. Sumado a esto, las reuniones sólo eran acerca de lo que ya se realizó, pero también debería tomarse en cuenta lo que se realizará a futuro.

En quinto lugar, según el relato, los sprints tenían una duración variable de entre 1 a 3 semanas dependiendo de las tareas. En realidad, los sprints deberían tener una duración fija a lo largo de todo el proyecto (corta, pero no variable), para a organizar de mejor manera las tareas de cada sprint y mantener el ritmo.

En sexto lugar, las tareas de cada sprint se definieron a partir de la canta Gantt inicial. En un proceso Scrum, estas deberían surgir a partir de una reunión analizando el progreso y el feedback del cliente, no ser definidas desde un inicio.

En séptimo lugar, al final de cada sprint, para obtener feedback del cliente, a este se le entregó un documento que contenía los avances. En Scrum, se debería entregar un producto capaz de pasar a producción para mostrar los avances al cliente, con el objetivo de obtener mejor feedback de las features ya implementadas. Además, para obtener este feedback, debieron haberse organizado reuniones (Sprint Reviews) entre todo el equipo de desarrollo junto con los clientes y stakeholders. Junto con esto, la gerencia era la que estaba a cargo de idear requisitos adicionales a partir del feedback recibido. Estos deberían idearlos principalmente el cliente, en conjunto con todo el equipo de desarrollo.

Finalmente, la gerencia introducía directamente las nuevas tareas dentro del Sprint actual. Esto no debería hacerse en ningún caso, todos los objetivos del Sprint deben ser definidos antes de empezarlo, y no modificarlos a lo largo de este, para no entorpecer el proceso.

En una nota aparte, en el relato tampoco se aprecia el rol de Scrum Master, el cual es fundamental para todo el proceso. Al no existir este rol, no se puede aplicar correctamente Scrum ni corregir las malas prácticas que se esten dando actualmente.

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2143 — Ingeniería de Software 2020-1

NOMBRE: Matías Duhalde
SECCIÓN: 1

Interrogación 1 — Respuesta Pregunta 2

Para realizar la distribución de las tareas en Scrum, hay que tener en cuenta ciertos detalles que este proceso tiene:

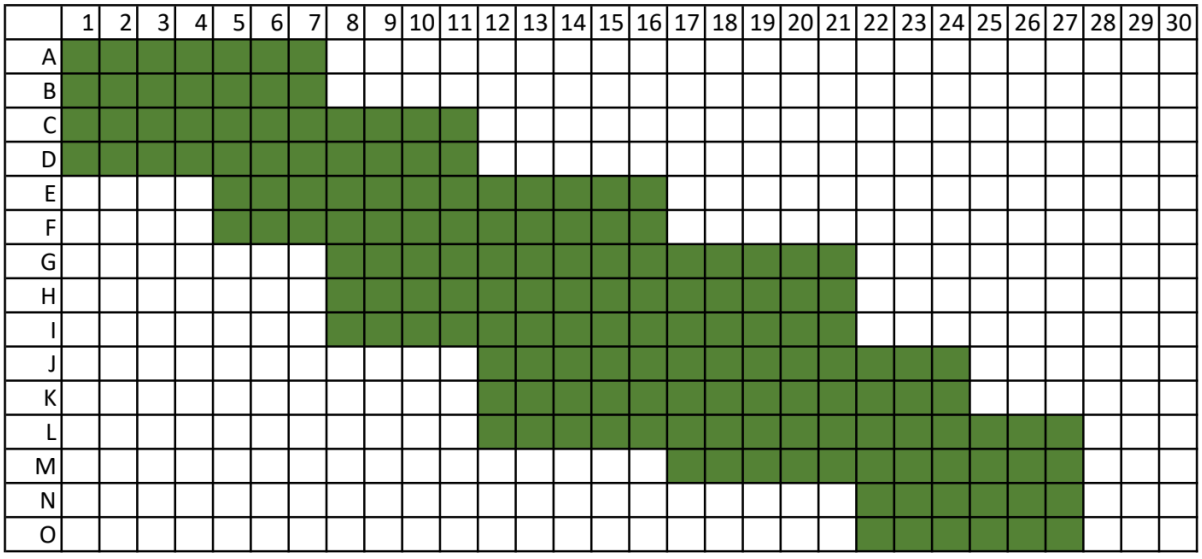
1. Los sprints son cortos, y tienen una duración constante a lo largo de todo el proyecto.

2. Las tareas no pueden quedar a medias o incompletas al finalizar un Sprint.

Tenemos que en el Kanban board, al día 28 ya están todas las tareas listas. No se explicita un tiempo máximo en el enunciado para el proyecto en Scrum, pero sí se explicita que deben estar todas las features implementadas, por lo que estamos mirando a una planificación con fixed-scope.

Se supone que las tareas que están en el Kanban board no se pueden dividir en relatos de usuario más pequeños, por lo que se mantendrán como están.

No tenemos información día por día en el Kanban board, por lo que se asume que las tareas se empiezan el primer día que se muestran en **In-Progress**, y ya están terminadas el primer día que se muestran en **Done** (terminan un día antes del día que se muestra en Done). A partir de esto, se pueden resumir las tareas en el siguiente gráfico:



En este gráfico las tareas están a la izquierda, el número de día arriba, y los recuadros verdes representan los días en el cual la tarea se trabajó. Si suponemos que un recuadro equivale a 1 SP, tenemos que en el diagrama de arriba hay un total de 167 SP. Estos, se trabajaron en un total de 27 días, por lo que obtenemos una velocidad promedio de 6.2 SP por día.

En nuestro proceso Scrum, para adaptarse mejor a lo que pide el enunciado, se define un sprint de una semana de duración (5 días hábiles). Entonces, tenemos que en el proceso Kanban la velocidad promedio por semana fue de 31 SP por semana.

Finalmente, dado que las tareas tienen distintas prioridades, es necesario atender a las más importantes primero, por lo que se debe seguir el orden "alfabético" en este caso.

Con todo esto en cuenta, se tiene que las tareas que se atendieron por semana fueron las siguientes: **Nota:** En cada Sprint (semana en este caso) en un proceso Scrum no sólo se trabajan las features, sino que también se realizan reuniones como la Sprint Review, las Daily Meetings, Sprint Planning, y Sprint Retrospective, en donde el equipo de desarrollo, el Scrum Master, los clientes y los stakeholders discuten el flujo del proyecto, que tareas se harán, revisan el backlog, se entrega feedback del progreso, discuten como mejorar el proceso, entre otras actividades. Por simplicidad, estas no se incluyen en la descripción de cada semana, pero se asume que también se realizan.

Semana 1: Se trabajan las tareas A (7 SP), B (7 SP) y C (11 SP). Suman un total de 25 SP, por debajo de la velocidad del equipo.

Semana 2: Se trabajan las tareas D (11 SP), E (12 SP) y F (12 SP). Suman un total de 35 SP, ligeramente por sobre la velocidad del equipo.

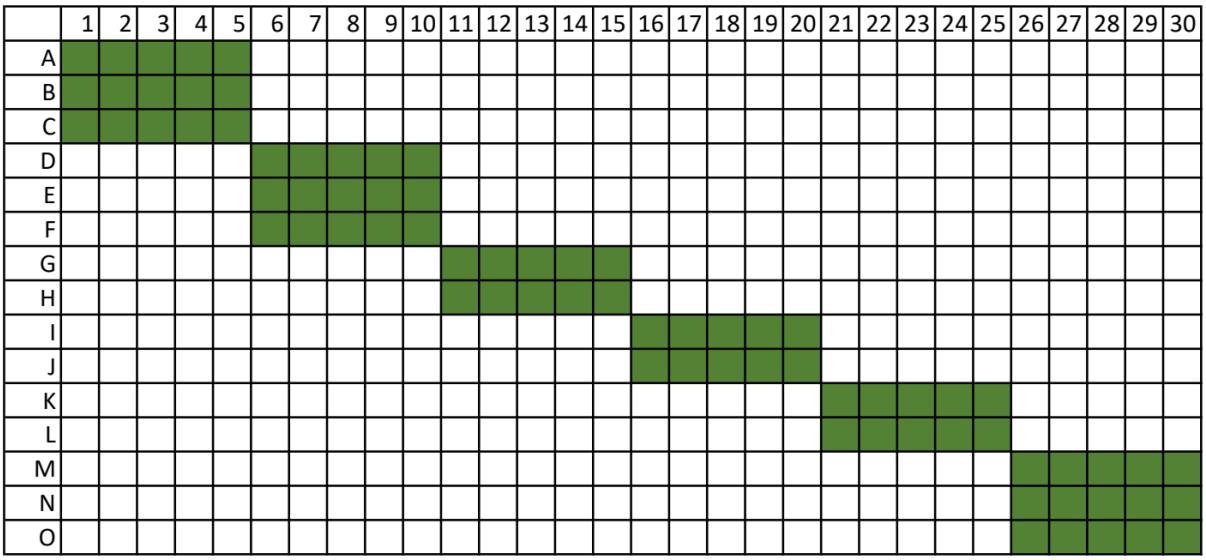
Semana 3: Se trabajan las tareas G (14 SP) y H (14 SP). Suman un total de 28 SP, por debajo de la velocidad del equipo.

Semana 4: Se trabajan las tareas I (14 SP) y J (13 SP). Suman un total de 27 SP, por debajo de la velocidad del equipo.

Semana 5: Se trabajan las tareas K (13 SP) y L (16 SP). Suman un total de 29 SP, por debajo de la velocidad del equipo.

Semana 6: Se trabajan las tareas M (11 SP), N (6 SP) y O (6 SP). Suman un total de 23 SP, por debajo de la velocidad del equipo.

La distribución de las tareas en los días del proceso Scrum se resumen en el siguiente diagrama:



Notar como a diferencia del diagrama anterior, todos los sprints son de duración constante, y no se dejan tareas a medio hacer al terminar cada sprint (también notar que ya no se mantiene la equivalencia de 1 recuadro verde = 1 SP).

Con esta distribución, seríamos capaces de desarrollar el software en un proceso Scrum, cumpliendo los requisitos, las deadlines, y satisfaciendo al usuario.

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2143 — Ingeniería de Software 2020-1

NOMBRE: Matías Duhalde
SECCIÓN: 1

Interrogación 1 — Respuesta Pregunta 3

Para empezar, en el enunciado se nos exige tener listo el primer release para el día 2 de Junio, partiendo hoy 5 de Mayo (para un total exacto de 4 semanas). Notar que dado que se nos exige esta fecha, debemos planificar en base a una fecha fija (fixed-date planning).

Para calcular los Story Points, se asignan las equivalencias basadas en una escala de potencias de 2, de acuerdo a la siguiente tabla:

Relato de Usuario	Cantidad	Story Points
Simple	10	1
Simple a Intermedio	6	2
Intermedio	8	4
Complejo	2	8

En base a esta tabla, se tiene que el primer release consiste en un total de 70 SP.

Del enunciado se pueden obtener que la velocidad promedio del equipo de desarrolladores es $v_p = 14$, con $v_{min} = 12$ y $v_{max} = 18$. Si definimos sprints de dos semanas, estos valores serán 28, 24 y 36, respectivamente.

Si multiplicamos velocidad por la cantidad de sprints que realizaremos, obtenemos:
$p_{avg} = 56$, $p_{worst} = 48$, $p_{best} = 72$.

Dado que nuestro "best case scenario" excede a los puntos requeridos por muy poco, y el "worst case scenario" junto con el average están bien por debajo de los requeridos, lo más sensato es renegociar las features de nuestro primer release.

En el enunciado no se especifica cuales de los requerimientos son los más críticos, pero se puede suponer que el primero (Información sobre distribución de las personas que han sido confirmados como positivas) es el más crítico, debido a que el objetivo de el proyecto es ayudar a reducir la tasa de contagio. Tampoco se especifica cuales requerimientos implican los relatos de usuario más complejos, pero es seguro asumir que se pueden dejar de lado relatos de usuario de los otros features menos importantes, siempre que el cliente lo acepte, para poder llegar a una meta alcanzable con seguridad.

Basta con eliminar un relato complejo, dos relatos intermedios, dos simples a intermedios, y dos simples (o cualquier otra combinación de estos, para completar un total de 18 SP), para poder alcanzar la meta en el "worst case scenario", y poder asegurar al cliente que obtendra un release con las features acordadas, dado nuestro equipo y los límites de tiempo.