**BAB 2**

**TINJAUAN PUSTAKA DAN DASAR TEORI**

**2.1 Tinjauan Pustaka**

Tinjauan pustaka merupakan acuan utama dalam beberapa studi yang pernah dilakukan yang berkaitan dengan penelitian ini. Terdapat beberapa penelitian mengenai implementasi *Scrum* dalam berbagai studi kasus.

Møller, dkk (2009) telah merancang aplikasi yang kompatibel mendukung *Scrum* sebagai metode manajemen proyek yang berfokus pada pengembangan antarmuka pengguna yang intuitif dan aplikasi yang mudah untuk ditangani.

Dalam penelitian yang dilakukan oleh Segers pada tahun 2012 berhasil menganalisa manfaat *Scrum Board* yang menggunakan kertas maupun perangkat lunak serta mengevaluasi *Scrum Board* berbasis perangkat lunak yang telah dikembangkan oleh beberapa perusahaan perangkat lunak.

Pada tahun 2013, Ingason, dkk dalam penelitiannya mengatakan bahwa *Kanban Wall* tidak mendefinisikan aturan tentang cara kerja harus dilakukan ataupun tidak dilakukan, sehingga perlu menggunakan *Scrum* untuk mengembangkan pekerjaan.

Perbedaan antara penelitian ini dengan penelitian lain yang telah ada adalah dalam penelitian ini difokuskan dalam mengimplementasikan *Scrum Board* ke dalam aplikasi *mobile* berbasis android.

Tabel 2.1 Perbandingan Penelitian

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter  Penulis | Platform | Metode | Obyek |
| Louise Smed Møller, dkk (2009) | Desktop | *Scrum* | *Scrum Master*,  *Product Owner* dan  Tim Pengembang |
| Jan Segers (2012) | Kertas dan Desktop | *Scrum* | *Scrum Master*,  *Product Owner* dan  Tim Pengembang |
| Helgi Thor Ingason, dkk (2013) | Kertas | *Kanban* dan  *Scrum* | *Scrum Master*,  *Product Owner* dan  Tim Pengembang |
| Usulan (2015) | Android | *Scrum* | *Scrum Master*,  *Product Owner* dan  Tim Pengembang |

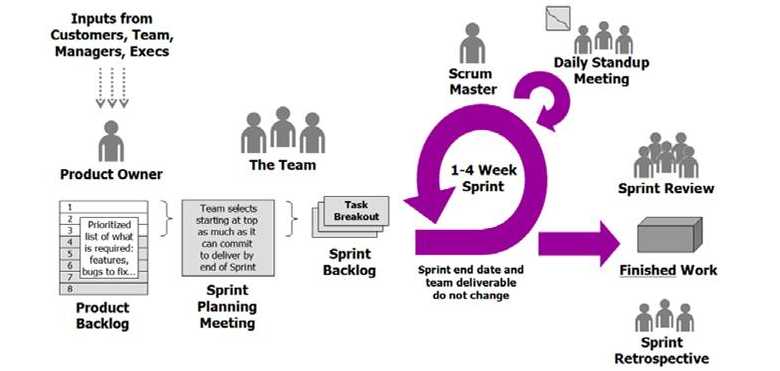
**2.2 Dasar Teori**

**2.2.1 *Agile Software Development***

*Agile Software Development* adalah salah satu metodologi dalam pengembangan perangkat lunak. Kata *Agile* artinya menjadi cepat, ringan, bebas bergerak, waspada. *Agile* adalah kata yang digunakan untuk menggambarkan konsep model proses yang berbeda dari konsep model proses yang ada (Martin, 2003). Konsep *Agile Software Development* diciptakan oleh Kent Beck dan 16 rekannya dengan menyatakan bahwa *Agile Software Development* adalah cara untuk membangun perangkat lunak dengan melakukannya dan membantu orang lain untuk membangunnya sekaligus (Dingsøyr, 2010). Dalam *Agile Software Development* interaksi dan personil lebih penting daripada proses dan alat-alat, perangkat lunak yang bekerja lebih penting daripada dokumentasi yang lengkap, bekerja sama dengan klien yang lebih penting dari negosiasi kontrak, dan responsif terhadap perubahan lebih penting daripada mengikuti rencana tersebut. Namun, seperti model proses lain, *Agile Software Development* memiliki sendiri keuntungan dan tidak cocok untuk semua jenis proyek, produk, orang dan situasi. *Agile Software Development* memungkinkan model proses yang toleran akan perubahan kebutuhan sehingga respon terhadap perubahan dapat dilakukan lebih cepat. Terdapat beberapa metode di dalam A*gile Software Development*, yaitu *Scrum*, *Crystal*, *Behavior-Driven Development* (BDD), *Test-Driven Development* (TDD), *Feature-Driven Development* (FDD), *Adaptive Software Development* (ADP), *Extreme Programming* (XP) dan masih banyak lagi yang lain.

**2.2.2 *Scrum***

*Scrum* termasuk salah satu yang paling popular di kalangan pengembang perangkat lunak. *Scrum* adalah sebuah kerangka berpikir dalam mengembangkan *software* yang memberi penekanan pada *self-organisation* dan semangat kolektivitas (Partogi, 2015). Gambar 2.1 menunjukan alur kerja *Scrum*.

Gambar 2.1 *Scrum Flow*

**2.2.3 Manajer-manajer di dalam *Scrum***

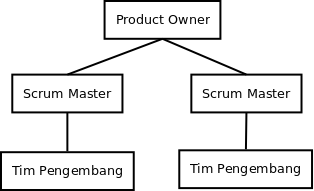
Dalam *Scrum*, peran manajer didesentralisasikan ketiga peran yang masing-masing memiliki akuntanbilitas terhadap areanya masing-masing. Ketiga peran tersebut adalah *Scrum Master*, *Product Owner* dan Tim Pengembang.

*Scrum Master* adalah peran manajerial yang memiliki banyak dampak yang cukup luas. *Scrum Master* bertanggung jawab dalam meningkatkan tingkat *Agility* di dalam organisasi. *Scrum Master* bertanggung jawab untuk melayani *Product Owner*, Tim Pengembang dan *stakeholder* lainnya di dalam organisasi.

*Product Owner* adalah manajer lainnya di dalam kerangka kerja *Scrum*. *Product* *Owner* bertanggung jawab dalam mengelola produk untuk meningkatkan nilai valuasi perusahaan di pasar. *Product Owner* memastikan produk yang dikembangkan oleh Tim Pengembang menghasilkan nilai tambah untuk pelanggan dan perusahaan di mana dia berada.

Tim Pengembang bertanggung jawab untuk mengelola proses pengembangan *software*.

Pada manajemen proyek tradisional, *Scrum Master* dan *Product Owner* dikenal sebagai *Project Manager* sedangkan Tim Pengembang terdiri dari *Programmer*, *Tester*, *Designer* dan *Analyst*.

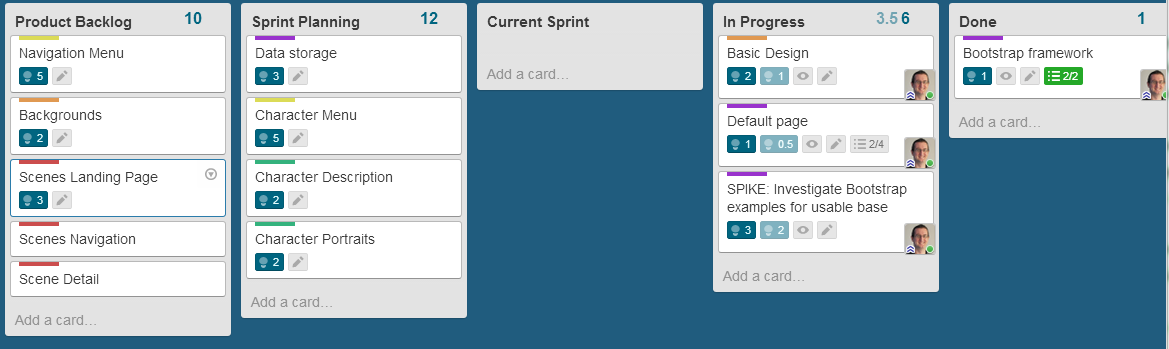
Gambar 2.2 Manajer-manajer di dalam *Scrum*

Gambar 2.2 menunjukan hirarki dari manajer-manajer di dalam *Scrum* dimana posisi *Scrum Master* berada di tengah-tengah antara *Product Owner* dan Tim Pengembang serta *Product Owner* dapat memimpin lebih dari satu tim *Scrum*.

**2.2.4 *Scrum Board***

*Scrum Board* adalah alat untuk mengimplementasikan *Scrum*, yang menggambarkan *Sprint* dan digunakan untuk mempermudah jalannya *Stand-up Meeting*. *Scrum Board* berisi kolom proses pengerjaan yang di dalam setiap kolom terdapat daftar tugas. Tugas yang harus dikerjakan dalam *Scrum Board* biasa disebut dengan istilah tiket. Kolom tersebut biasanya terdiri dari *Backlog* (daftar rencana tiket yang akan ada di dalam *software*), *Sprint backlog* (daftar rencana mengenai bagaimana Tim Pengembang berkolaborasi selama satu *Sprint* ke depan), *In Dev* (daftar tiket yang sedang dikerjakan), *In Test* (daftar tiket yang sedang dalam pengecekan) dan *Done* (daftar tiket yang sudah lolos pengecekan dan siap untuk *deployment* di akhir *Sprint*).

Gambar 2.3 merupakan contoh dari *Scrum Board* yang dibuat menggunakan aplikasi Trello. Terdapat 5 kolom pada contoh *Scrum Board* tersebut, yaitu *Product Backlog*, *Sprint Planning*, *Current Sprint*, *In Progress* dan *Done*.

**Gambar 2.3 Contoh *Scrum Board*

**2.2.5 *Sprint***

*Sprint* adalah kumpulan tugas-tugas yang akan dikerjakan dalam periode tertentu. Periode *Sprint* ditentukan maksimal 30 hari, lebih dari itu kemungkinan besar keadaan pasar sudah berubah.

**2.2.6 Tiket**

Tiket adalah istilah pengganti kata tugas pada *Scrum Board*. Pada sebuah tiket terdapat nomer tiket, judul tiket dan poin tiket. Nomer tiket biasanya terdiri dari inisial project diikuti dengan nomer urut pembuatan tiket. Poin tiket sendiri digunakan untuk penanda besar kecilnya pengerjaan sebuah tiket. Perhitungan poin tiket menggunakan bilangan Fibonacci dimana semakin kecil angkanya semakin kecil pula pengerjaan sebuah tiket.

**2.2.7 *Stand-up Meeting***

*Stand-up Meeting* adalah rutinitas tiap awal hari dan tidak lebih dari 30 menit. Pada S*tand-up Meeting*, masing-masing individu yang ada di dalam tim menjelaskan tentang tugas apa saja yang sudah dikerjakan kemarin, tugas yang akan dikerjakan hari ini dan hambatan apa saja dalam menjalankan tugas.

**2.2.8 Java**

Java adalah sebuah teknologi dimana pada teknologi tersebut mencakup Java sebagai bahasa pemrograman yang memiliki sintaks dan aturan pemrograman tersendiri, juga mencakup Java sebagai platform dimana teknologi ini memiliki *virtual machine* dan *library* yang diperlukan untuk menulis dan menjalankan program yang ditulis dengan bahasa pemrograman Java. Alasan terbesar dalam pembuatan bahasa pemrograman Java adalah keinginan akan terbentuknya suatu bahasa pemrograman yang bisa berjalan di berbagai perangkat tanpa harus terikat oleh *platform*, sehingga Java ini bersifat *portable* dan *platform independent* (tidak tergantung mesin atau sistem operasi).

**2.2.9 Android**

Android adalah sebuah sistem operasi untuk ponsel yang berbasis linux. Android SDK (*Software Development Kit*) menyediakan *tools* dan API (*Application Programming Interface*) yang diperlukan bagi para pengembang untuk membuat dan mengembangkan aplikasi yang digunakan pada ponsel bersistem operasi Android dengan menggunakan bahasa pemrograman Java.

**2.2.10 UML**

Menurut Pender (2002), UML adalah standar untuk menciptakan model yang mewakili perangkat lunak berorientasi objek dan sistem bisnis. UML memiliki standarisasi notasi tetapi tidak mendikte bagaimana menerapkan notasi. UML mencakup spesifikasi untuk sembilan diagram berbeda yang digunakan untuk berbagai dokumen perspektif dari solusi perangkat lunak dari awal proyek sampai instalasi dan pemeliharaan mikrofinansial.

***Use Case* Diagram**

Menggambarkan fitur dimana pengguna mengharapkan sistem untuk menyediakan. Lima elemen pemodelan yang membentuk *Use Case* Diagram adalah *actor*, *Use Case*, *association* dan *dependency*.

***Sequence* Diagram**

Semua *Sequence* Diagram lebih dimodelkan pada tingkat objek daripada tingkat kelas untuk memungkinkan skenario yang menggunakan lebih dari satu instance dari kelas yang sama dan bekerja pada tingkat fakta, data uji dan contoh. *Sequence* Diagram menggunakan tiga elemen notasi mendasar yaitu *object*, *message/stimuli* dan *object lifetime*.

***Class* Diagram**

*Class* Diagram terdiri dari tiga kompartemen (ruang persegi panjang) yang mengandung informasi yang berbeda diperlukan untuk menjelaskan sifat-sifat satu jenis objek.

Notasi dalam *class* diagram adalah sebagai berikut :

1. *Class Name* digunakan untuk mendefinisikan *class* (tipe objek) dalam sebuah paket.
2. *Attribute* berisi semua definisi data.
3. *Operations* berisi definisi untuk setiap perilaku yang didukung oleh jenis objek.