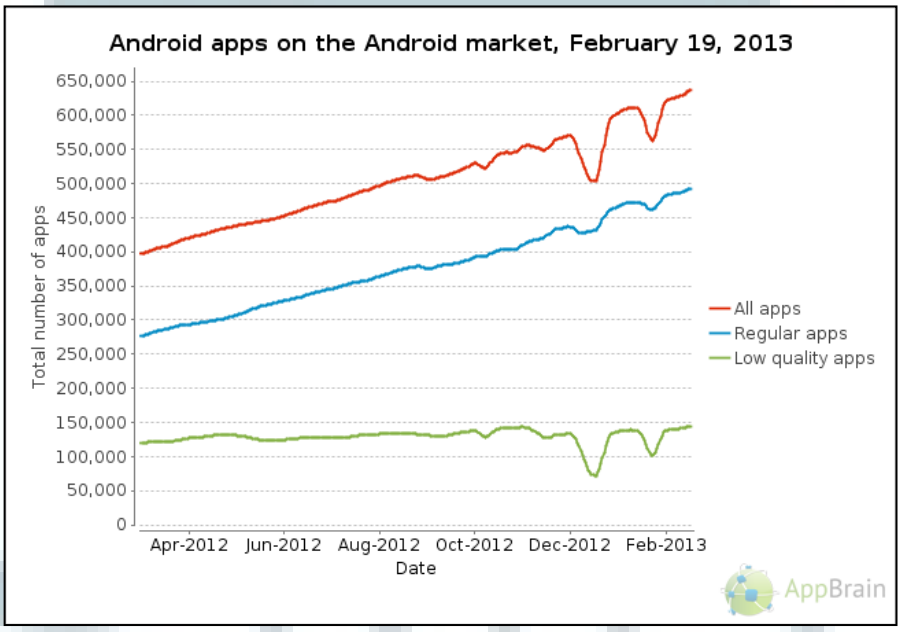
BAB I

PENDAHULUAN

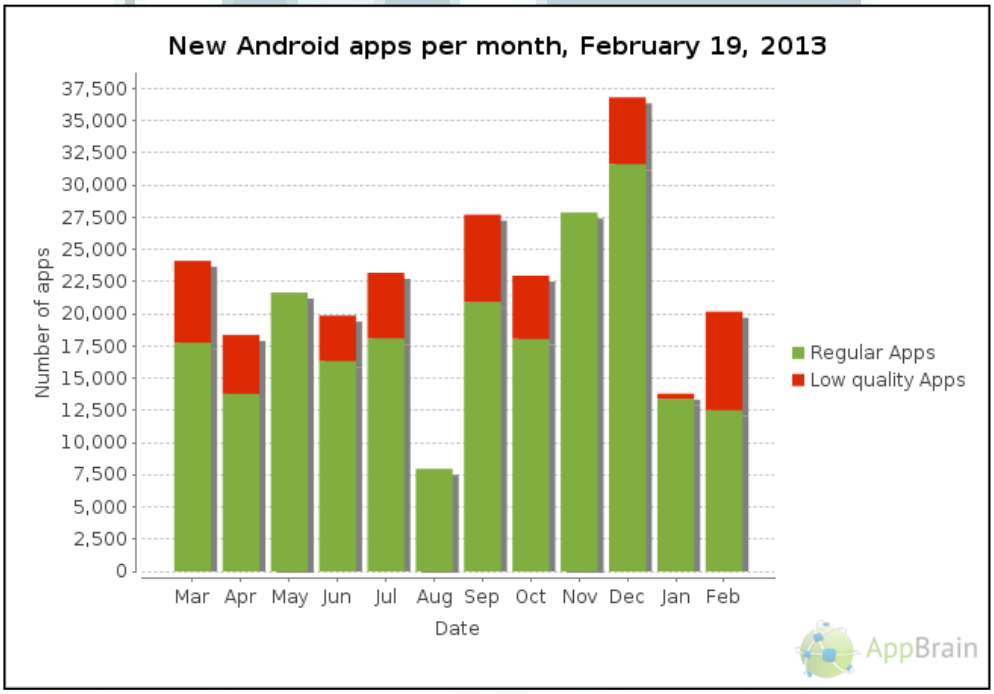
1.1 Latar Belakang

Perangkat (*device*) berbasis Android sudah merupakan suatu hal yang umum untuk ditemui beberapa tahun belakangan ini. Pada akhir tahun 2010 , sistem operasi Android sudah mulai menggantikan sistem operasi lainnya khususnya untuk perangkat *smartphone* dan *tablet*. Pada September 2012, tercatat sudah ada lebih dari 500 juta perangkat Android yang diaktifkan di seluruh dunia dan setiap harinya terus bertambah sebesar 1,3 juta perangkat setiap harinya (F, Alan).



Gambar 1.1 Statistik Jumlah Aplikasi Android (April 2012 – Februari 2013)
Sumber : (AppBrain, 2013)

Selain penggunanya yang sudah sangat banyak, aplikasi Android juga terus bertambah setiap harinya. Berdasarkan data yang diambil dari AppBrain (www.appbrain.com), pada tanggal 19 Februari 2013 sudah terdapat lebih dari lima ratus ribu aplikasi Android yang saat ini telah dikembangkan dan telah dipasarkan.



Gambar 1.2 Statistik Pertambahan Aplikasi Android Baru (Maret 2012 – Februari 2013)
Sumber : (AppBrain, 2013)

Berdasarkan data tersebut, pertambahan aplikasi Android setiap bulannya sangat banyak. Bertambahnya jumlah aplikasi Android ini tentunya juga akan menambah jumlah *Android Project*, dimana setiap *project* biasanya ditujukan untuk menghasilkan suatu aplikasi tertentu.

Dalam mengerjakan suatu *Android Project*, tentunya para pengembang akan berurusan dengan *source code* yang biasanya terdiri dari beberapa file dan dikerjakan oleh suatu tim. Sebuah tim umumnya akan terdiri dari *programmer* dan *project manager*. Jumlah *programmer* sangat bergantung kepada skala dari aplikasi yang sedang dikerjakan. Sebuah *project* mungkin saja dikerjakan oleh seorang *programmer* untuk *project* dengan skala kecil atau bahkan sampai empat atau lima *programmer* untuk *project* dengan skala yang cukup besar. Pada tahap pengembangan aplikasi ini, para *programmer* akan saling membagi tugas dan mengerjakan *project* secara terpisah. Ketika masing-masing *programmer* telah selesai dengan bagiannya masing-masing, dilakukanlah integrasi.

Integrasi ditujukan untuk menggabungkan bagian-bagian yang terpisah yang telah dikerjakan oleh *programmer* menjadi suatu kesatuan sehingga aplikasi dapat berjalan sesuai yang diinginkan. Integrasi dapat dilakukan dengan melakukan *patching* terhadap *source code* atau *resource file* seperti gambar atau suara. *Patching* adalah suatu proses untuk memperbaharui aplikasi atau kode tertentu. *Patching* dapat bertujuan untuk menambal celah keamanan yang ada pada aplikasi, memperbaiki kinerja suatu fungsi tertentu, atau menambahkan suatu fungsi baru pada aplikasi. Untuk proses *patching* terhadap *source code*, biasanya dilakukan secara manual yaitu dengan membaca *log message* tentang perubahan-perubahan apa saja yang telah dilakukan. Tim akan mengikuti *log message* ini dan melakukan pengubahan terhadap *source code* sehingga dihasilkan suatu *code* baru. *Patching* biasanya dilakukan dalam tahap integrasi, yaitu tahap penggabungan proyek atau kode yang telah dimodifikasi secara terpisah menjadi satu.

Proses *patching* ini menjadi tahap yang penting, karena kesalahan pada tahap ini dapat menyebabkan proses integrasi berakhir dengan kegagalan. Kesalahan dapat terjadi ketika kode akan diintegrasikan antara *programmer* dan *lead programmer*. *Programmer* adalah orang yang bertanggung jawab dalam pembuatan program untuk proyek dengan menuliskan kode-kode dalam bahasa pemrograman tertentu, sementara *lead programmer* adalah seseorang yang bertanggungjawab untuk memimpin tim *programmer* dalam seluruh proses yang berhubungan dengan pemrograman seperti proses integrasi dan *unit testing*. Posisi *lead programmer* dapat ditempati oleh *project manager* atau *programmer* senior.

Umumnya *programmer* yang mengubah kode akan memberikan informasi tentang kode yang diubahnya kepada *lead programmer*. *Programmer* akan menuliskan suatu pesan tentang pengubahan yang dilakukannya, sementara *lead programmer* akan membaca pesan tersebut kemudian melakukan penggabungan kode. Bila dilaksanakan secara manual, mungkin saja *programmer* yang mengubah kode memberikan pesan yang kurang lengkap, atau *lead programmer* yang membaca pesan tersebut salah mengartikan pesan yang diberikan sehingga kode yang digabungkan menjadi kode yang salah dan proses integrasi menyebabkan program tidak berjalan dengan seharusnya.

Oleh karena itu dibutuhkan suatu *tools* yang dapat membantu proses integrasi dalam suatu *Android Project*. Integrasi yang dilakukan meliputi *patching source code* dan penggabungan *resources* untuk aplikasi. *Tools* yang dihasilkan harus mudah digunakan dan dapat dipakai bersamaan dengan *Integrated Development Environment* (IDE) yang populer digunakan dalam pengembangan

Android Project. *Eclipse IDE* merupakan suatu IDE yang cukup populer digunakan dalam pembuatan *Android Project*. Oleh karena itu *tools* yang dibuat berupa *plug-in* untuk *Eclipse IDE* sehingga integrasi dan pengembangan suatu *Android Project* dapat dilakukan satu aplikasi saja, yaitu Eclipse IDE.

1.2 Rumusan Masalah

Masalah yang dapat dirumuskan untuk penelitian ini adalah:

1. Bagaimana membuat suatu *plug-in Eclipse* yang dapat memodifikasi suatu *Android Project* berdasarkan suatu informasi yang didapat dengan membandingkan dua *Android Project* yang sama, tetapi dengan versi yang berbeda.
2. Bagaimana cara untuk membandingkan dua buah *source code* kemudian menemukan *Longest Common Subsequence* (LCS).
3. Bagaimana cara untuk menghasilkan suatu *Shortest Edit Script* (SES) yang dapat mentransformasi suatu *source code* dari versi yang lama ke versi yang lebih baru.

1.3 Batasan Masalah

Beberapa batasan yang harus ditentukan dalam penelitian ini adalah sebagai berikut

1. Versi *Eclipse IDE* minimal yang dibutuhkan untuk *plug-in* ini adalah *Eclipse Classic* versi 3.6.

2. *Android Project* yang akan di-*patch* harus dikerjakan menggunakan *Eclipse IDE* yang telah dilengkapi dengan *plug-in* yang dihasilkan dalam penelitian ini.
3. Pembuatan *patch* baik terhadap *source code* maupun *resources* dilakukan oleh masing-masing anggota tim berdasarkan suatu *Original Android Project* yang sama.
4. *Patch* yang dihasilkan oleh masing-masing tim akan diterapkan terhadap *Original Android Project* secara bersamaan sekaligus.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. Membuat *plug-in Eclipse* yang mampu menemukan *Longest Common Subsequence* dari dua buah *source code* yang sama, tetapi dalam versi yang berbeda dalam suatu *Android Project*, kemudian menghasilkan *Shortest Edit Script* yang digunakan sebagai panduan untuk memodifikasi kode.
2. Membuat *plug-in Eclipse* yang mampu memberikan panduan pada proses integrasi sehingga mempercepat proses pengembangan proyek dan kesalahan saat proses integrasi dapat diminimalisasi.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Adanya sebuah *plug-in eclipse* yang mampu melakukan modifikasi terhadap suatu *Android Project* yang didasarkan pada informasi yang didapat dari proyek yang sama, tetapi dengan versi berbeda sehingga proses integrasi menjadi lebih mudah dan kesalahan dapat diminimalisasi.
2. Memudahkan kerja dari para pengembang aplikasi Android dan menambah efektifitas serta kualitas dari aplikasi yang dihasilkan.

1.7 Sistematika Penulisan

Sistematika penulisan laporan skripsi ini dijelaskan sebagai berikut.

Bab I Pendahuluan

Bab ini berisikan pendahuluan tentang penulisan penelitian ini. Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penulisan, manfaat penulisan, dan sistematika penulisan.

Bab II Tinjauan Pustaka

Bab ini berisikan landasan teori yang digunakan untuk memulai dan membantu penelitian. Landasan teori yang akan dijelaskan mengenai Longest Common Subsequence (LCS), Shortest Edit Script (SES), Algoritma Miller-Myers, Android, Android Project, Eclipse, dan Eclipse Plug-In Development Environment.

Bab III Analisis dan Perancangan *Plug In*

Bab ini berisikan spesifikasi umum kebutuhan dan perancangan *plug-in*. Bab ini akan mencakup tentang spesifikasi umum dari *plug-in*, diagram kerja dan fugsi *plug-in*, masukan serta keluaran yang dari *plug-in*, desain kerja *plug-in*, dan desain antarmuka *plug-in*.

Bab IV Implementasi dan Uji Coba

Bab ini berisi penjelasan mengenai implementasi dan hasil uji coba *plug-in*. Pada bab ini akan dibahas bagaimana perancangan yang telah dilakukan diterapkan secara nyata menjadi suatu *plug-in*. Selain itu, *plug-in* yang dihasilkan juga diuji coba untuk mengamati hasil dan performanya apakah sesuai dengan tujuan awal pembuatan dan apakah *plug-in* dapat menyelesaikan permasalahan yang telah dijelaskan sebelumnya.

Bab V Simpulan dan Saran

Bab ini berisi simpulan dan saran dari penelitian ini. Simpulan berisikan informasi mengenai berhasil tidaknya penelitian ini menyelesaikan permasalahan yang dihadapi. Saran akan berisikan mengenai kekurangan dari penelitian ini yang tentunya dapat dijadikan suatu landasan untuk melakukan penelitian selanjutnya agar dihasilkan suatu penelitian yang lebih baik lagi.