# PEMBANGUNAN *FRAMEWORK* UNTUK DOMAIN KASUS KULINER

Daeng Rosanda[1]

[1]Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
E-mail : daengrosanda@gmail.com[1]

## ABSTRAK

*Framework* dibangun dengan tujuan untuk meningkatkan produktifitas perangkat lunak, karena *framework* telah menyediakan fungsional umum yang ada pada suatu domain kasus aplikasi. Domain kasus aplikasi kuliner dipilih 24 dari 30 programmer dari berbagai latar belakang. Hal ini didasari oleh belum adanya *framework* dalam bidang tersebut. Selanjutnya, fungsional yang umum pada *framework* didapatkan melalui hasil analisis proses bisnis yang ada pada setiap aplikasi yang telah berjalan. Implementasi fungsional yang pada *framework* juga dianalisis melalui analisis perancangan pola. Hasil implementasi tersebut diuji dengan menggunakan metode pengujian berorientasi objek dan pengujian implementasi perangkat lunak dari *framework*.

**Kata kunci :** *framework*, kuliner, *design pattern*.

## 1. PENDAHULUAN

Pengembangan perangkat lunak kuliner dipicu oleh keanekaragaman kuliner yang tersebar diseluruh Indonesia. Perangkat lunak kuliner merupakan aplikasi yang dapat menyajikan data tempat kuliner berdasarkan urutan tertentu. Berdasarkan hasil observasi di PT GITS Indonesia, perangkat lunak kuliner menjadi media promosi bagi pelaku bisnis kuliner. Namun demikian, pembangunan perangkat lunak tidak terlepas dari produktifitas para pengembangnya. Menurut prediksi IDC, di Indonesia ada sekitar 56,5 ribu pengembang perangkat lunak pada tahun 2006. Tetapi, Malaysia dan Singapura ternyata mampu memproduksi perangkat lunak lebih banyak jika dibandingkan dengan Indonesia meskipun jumlah pengembang mereka lebih sedikit. Hal tersebut menunjukan bahwa kuantitas bukan merupakan faktor penentu produktifitas [ HYPERLINK \l "1" 1 ].

### 1.1 *Framework*

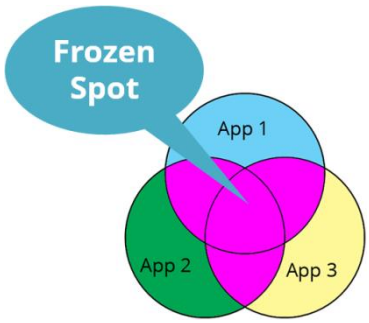
*Framework* adalah perangkat lunak setengah jadi yang didalamnya terdapat komponen yang bisa digunakan secara berulang baik sebagian maupun seluruh dari sistem yang direpresentasikan oleh sekumpulan kelas abstrak beserta relasinya untuk membangun perangkat lunak yang baru 2]**}**. *Framework* terdiri dari sekumpulan fungsionalitas umum sehingga pembangunan aplikasi bisa dimulai terhadap hal yang spesifiknya. *Framework* digunakan sebagai bahan dasar untuk pembangunan suatu aplikasi. *Framework* dapat digunakan secara berulang, karena *framework* memiliki semua fungsional yang ada pada domain kasus tertentu.

#### 1.1.1 Analisis Domain

Analisis domain dapat ditempuh dengan cara mengelompokan berbagai aplikasi dari segi proses bisnis. Proses bisnis ini yang nantinya akan membedakan atau menyamakan fungsional antar aplikasi. Setiap proses bisnis yang ada masing-masing aplikasi maka disebut *hotspot*. Sedangkan setiap proses bisnis yang ada diantara salah satu aplikasi maka disebut dengan *frozen spot* [ HYPERLINK \l "2" 2 ].

#### 1.1.2 *Frozen spot*

*Frozen spot* merupakan suatu fungsional yang identik ada pada suatu domain kasus tertentu 2]**}**. *Frozen spot* biasanya berupa fungsional keseluruhan maupun bagian dari suatu fungsional yang ada. Pada dasarnya, irisan dari setiap fungsional yang ada antar aplikasi disebut *frozen spot*. *Frozen spot* juga bisa dibilang bahan mentah suatu fungsional yang nantinya ada kemungkinan diurai menjadi *hotspot*. Namun, *frozen spot* tidak selalu ada pada setiap aplikasi pada domain kasus yang sama. Sehingga jika ada fungsional yang tidak bisa diurai maka akan tetap menjadi *frozen spot*.




**Gambar 1 Ilustrasi *Frozen spot***

### 1.1.3 *Hotspot*

*Hotspot* merupakan inti dari perancangan *framework* yang akan menentukan jenis arsitektur perangkat lunak pada domain kasus tertentu [ HYPERLINK \l "2" 2 ]. *Hotspot* terdiri atas sekumpulan fungsional yang pasti ada dalam setiap aplikasi dalam suatu domain kasus tertentu. *Hotspot* pada dasarnya merupakan fungsional yang ada pada masing-masing aplikasi dalam suatu domain kasus tertentu. *Hotspot* ini bisa didapatkan melalui penguraian dari *frozen spot*.

Selain dari penguraian *frozen spot*, *hotspot* juga bisa muncul karena kebutuhan *hotspot* yang telah terdefinisi. Penguraian *hotspot* dibagi kedalam dua bagian yaitu *whitebox* dan metode *blackbox* 2]**}**. Metode pendefinisian *hotspot* secara *whitebox* merupakan pendefinisian *hotspot* berdasarkan kebutuhan dari *hotspot* lainnya. Sedangkan secara *blackbox*, *hotspot* didefinisikan melalui penguraian dari *frozen spot*. Syarat dari metode ini yaitu fungsional harus ada pada setiap aplikasi dalam suatu domain kasus. Sehingga cara ini dapat ditempuh dengan cara mendefinisikan *frozen spot* terlebih dahulu.

### 1.2.1 *Creational Pattern*

Penciptaan objek dari suatu kelas disebut instansiasi [ HYPERLINK \l "Met06" 5 ]. Pada saat instansiasi objek diciptakan dari suatu kelas. Instansiasi ini memiliki pola yang disebut dengn pola penciptaan. Pola penciptaan merupakan tahap dimana penciptaan kelas menjadi objek 4]**}**. Sehingga pada penciptaan objek beserta penyusunnya seperti properti dan metode telah terdefinisi dengan pola ini.

Adapun pola *structural* yang digunakan dalam penelitian ini adalah pola *Adapter* dan pola *Decorator*. Pola *adapter* memungkinkan untuk menghubungkan dua kelas yang berbeda tanpa mengubah struktur dan kode keduanya dengan menggunakan *adapter* 7]**}**. Sedangkan Pola dekorator merupakan salah satu pola dimana kelas membutuhkan perangkat tambahan terhadap objek secara dinamik [ HYPERLINK \l "4" 4 ]. Dekorator lebih memberikan fleksibilitas terhadap *method*nya jika dibandingkan dengan penurunan (*inheritance*).

Sedangkan pola *Iterator* adalah pola yang digunakan untuk penelusuran sekumpulan objek tanpa membongkar implementasinya [ HYPERLINK \l "Fre04" 7 ]. Objek yang ditelusuri oleh *iterator* biasanya objek terunut. *Iterator* dapat menelusuri sampai dengan tahap properti dari objek. Dengan demikian pola yang ditelusuri oleh *iterator* bersifat agregat karena *iterator* tidak memiliki komposisi dari pola lainya.

## 1.3 Perangkat Lunak Uji

Pengujian perangkat lunak merupakan tahap pengujian dimana semua pola yang telah dirancang akan diuji secara fungsionalitas 3]**}**. Tujuan dari pengujian perangkat lunak untuk memeriksa kesesuaian antara kebutuhan dengan fungsionalitas perangkat lunak yang telah dibangun. Dalam penelitian ini, pengujian yang akan dilakukan meliputi pengujian fungsionalitas dari setiap kelas yang dibangun beserta pengujian implementasi perangkat lunak yang dibangun berdasarkan pada *framework*. Kedua bagian tersebut masuk kedalam pengujian orientasi objek (*OO Testing*) dan uji implementasi *framework*.

### 1.3.2 Uji Implementasi

Pengujian implementasi *framework* merupakan pengembangan perangkat lunak yang didasari oleh *framework*. Pengujian ini berupa pembangunan perangkat lunak berdasarkan *framework* [ HYPERLINK \l "2" 2 ]. Dengan demikian hasil dari *framework* ini dapat dibuktikan fungsionalitasnya berdasarkan hasil implementasi.

## 2. ISI PENELITIAN

Dalam penelitian ini terdiri dari 3 bagian yaitu analisis proses bisnis, *analisis frozen spot*, analisis *hotspot*, perancangan kelas, dan perangkat lunak uji.

## 2.1 Analisis Proses Bisnis

Analisis proses bisnis akan dilakukan sebagai tahap awal pembangunan *framework*. Tahap ini akan menganalisis setiap aplikasi yang telah berjalan pada domain kasus kuliner. Tahap ini dimulai dari perbandingan proses bisnis dari 3 aplikasi kuliner yang telah berjalan. Masing-masing aplikasi tersebut didapatkan dari urutan *rating* tertinggi dari Google Play. Adapun rincian tentang aplikasi tersebut dapat dilihat pada Tabel 1