Survei: Aplikasi dan Pengembangan Model Software-as-a-Service pada Cloud Computing

I Gusti Ngurah Ady Kusuma
STMIK STIKOM Bali
Jalan Raya Puputan No. 86 Renon, Denpasar, Bali. (0361) 244445
e-mail: ady_kusuma@stikom-bali.ac.id

Abstrak

Aplikasi berbasis layanan atau *Software-as-a-Service* merupakan aplikasi yang dibangun terdiri dari beberapa layanan atau *service*. Pada umumnya aplikasi ini merupakan aplikasi yang ditempatkan pada sebuah *server* dan terhubung dengan jaringan internet untuk mengakses *service* dari penyedia layanan lainnya. Konsep ini berkembang dari pemrograman berorientasi objek dimana tujuan dari *Software-as-a-Service* atau SaaS dapat digunakan oleh pengguna tanpa harus memiliki pengetahuan yang tinggi mengenai teknik-teknik yang digunakan. SaaS telah banyak berkembang pada pembangunan sebuah infrastruktur dalam *mobile* dan *ubiquitous computing*, dan *cloud computing*. Dalam paper ini dirangkum kembali penggunaan SaaS dalam bidang tersebut dan menyajikan bagaimana penerapan penggunaan SaaS yang dapat dilakukan berdasarkan pelaku penerapan tersebut yang dilihat dari sisi *server, broker,* pengelola, dan pengguna.

Kata kunci: *Software-as-a-Service, mobile computing, ubiquitous computing, cloud computing*

Abstract

An application based on service or Software-as-a-Service is application that consist of some service. Generally, this application placed on server and connected to internet for accessing other service from another provider. This concept advanced from object oriented programming with purpose that Software-as-a-service or SaaS could be used by another user who is not specialized on SaaS. SaaS already advancing of infrastructure in mobile and ubiquitous computing and significantly in cloud computing. In this paper, author reviewing the uses of SaaS on cloud computing and shows how the application of SaaS could be done based on side of view from sever, broker, administratorm and user..

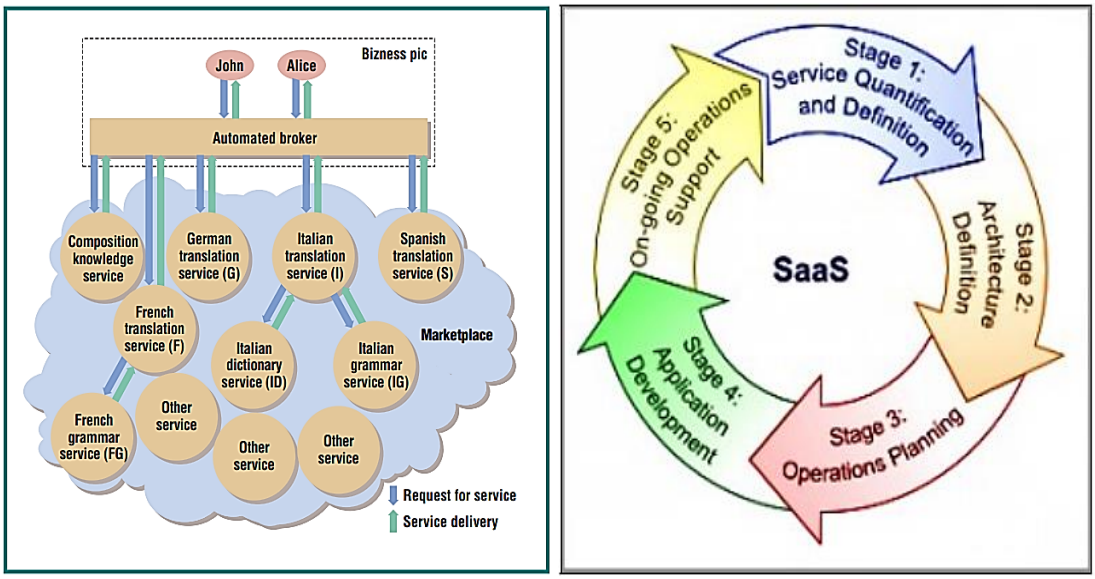
Keywords*: Software-as-a-Service, mobile computing, ubiquitous computing, cloud computing*

1. Pendahuluan

Pada perkembangan aplikasi beberapa tahun belakangan ini, penyedia layanan internet memberikan penawaran penggunaan *service* tanpa membeli *software*, memasang dan merawat *software* tersebut. Sifat dinamika dan *online* tersebut mulai mengubah asumsi tradisional *software* menjadi sebuah layanan yang bersifat *pay-as-you-go* yaitu pengguna hanya perlu membayar biaya sesuai dengan fungsi yang digunakan. Paradigma orientasi *software* ini dikenal dengan nama *Software-as-a-Service* atau SaaS [1].

SaaS menggunakan jaringan komputer untuk penggunaaan *service*. *Software* berada pada data *center* dan dipelihara oleh penyedia SaaS yaitu SaaS *provider*. Pengguna hanya memerlukan jaringan yang terkoneksi dengan SaaS *provider* untuk menikmati layanan *service* yang bisa melalui Internet atau jaringan nirkabel atau *wireless*. SaaS bisaanya dibangun menggunakan protokol yang berbasis pada *web* dengan *browser* pada client untuk mengakses *service*. SaaS memiliki karakteristik yaitu *service* bisa diakses oleh *multi-user* secara sekaligus, penghitungan biaya hanya pada *service* yang digunakan, mendukung semua fungsi dasar pembangungan aplikasi dan meminimalisir biaya implementasi dan integrasi.

Seiring dengan berkembangnya teknologi yang tercipta pada beberapa tahun terakhir, berbanding lurus dengan meningkatnya kemampuan sebuah *server* sehingga banyak fungsi-fungsi atau layanan yang



(a) (b)

Gambar 1. (a) Ilustrasi dari arsitektur *Software-as-a-Service* [1] dan (b) *Delivery Model* dari *Software-as-a-Service* [3]

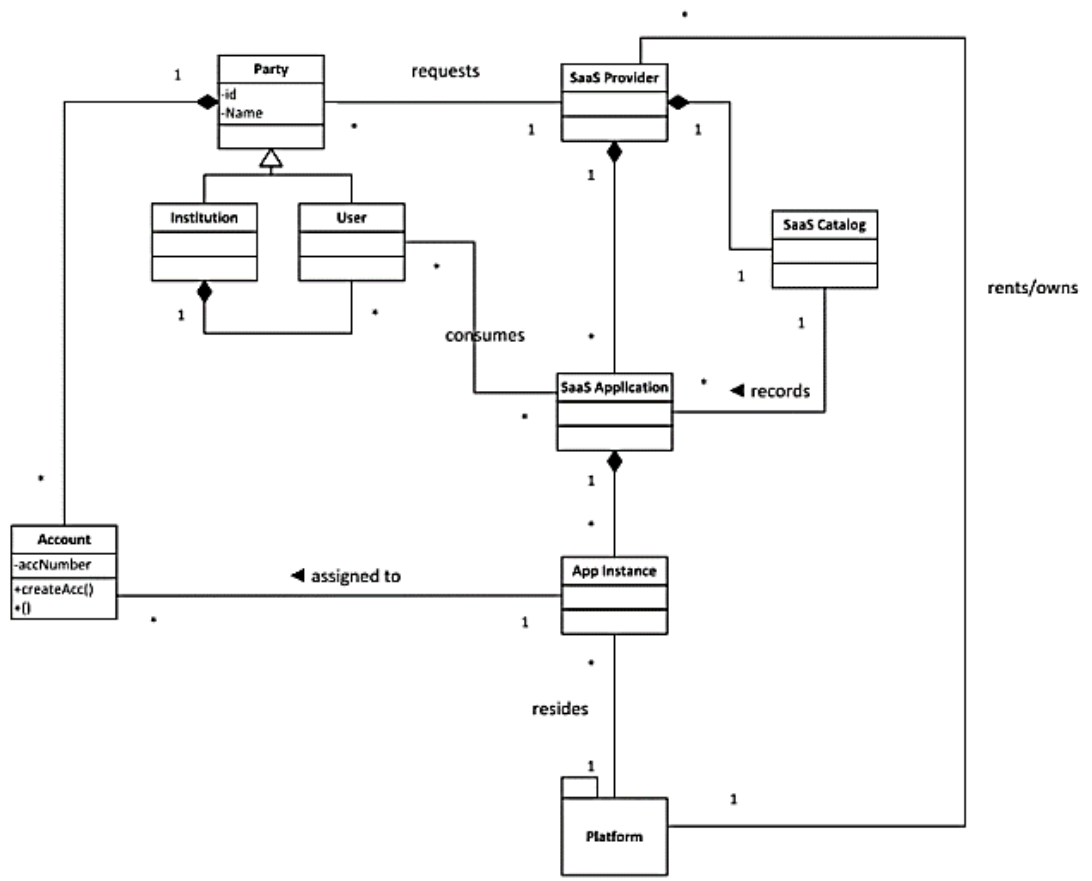
dulunya diproses pada sisi *client-side* kini diproses penuh oleh *server* dan *client* hanya menerima hasil akhirnya. Layanan (atau *service*) yang disediakan inipun menggunakan *service* dari *server* yang lain. Perkembangnya pun kian pesat, bahkan Google pun mulai banyak memberikan *service* melalui GoogleAPI seperti konversi mata uang, translasi bahasa dan lainnya. Perkembang ini sangat menjanjikan bagi kemudahan dan kecanggihan dari teknologi informasi ini, sehingga sangat layak untuk dilakukan sebuah survei untuk melihat sejauh apa teknologi SaaS ini mampu dikembangkan dan diperdayakan

Pada SaaS terdapat peran kunci pelaksaanaanya yaitu *end-user* atau *customer*, SaaS *provider* sebagai penyedia layanan, SaaS Developers sebagai pembuat *service*, SaaS Host sebagai penyedia Infrastrukstur dan Hub *provider* yang me-integrasi-kan semua *service*. Keuntungan yang dimiliki oleh SaaS model yaitu Time-to-Market yang berarti *customer* lebih cepat dalam penangan solusi, Reduced-I/T-Skill yaitu *customer* tidak perlu memiliki kemampuan yang tinggi, Direct-Acoountability, pengurangan biaya di depan dan perawatan dan Manajemen Sekuritas. Semua keuntungan tersebut bertujuan untuk mengurangi biaya operasi dan meminimalisir biaya-tak-terduga untuk memaksimalkan keuntungan.

SaaS dibangun pada sebuah platform yang bisaanya platform tersebut dibangun diatas Open-Source program. Pengguna platform tersebut dibagi menjadi 3 yaitu Platform Administrator, SaaS Administrator dan Subscriber's User and Administrators. Platform SaaS dapat dikatagorikan dalam beberapa grup yaitu Access Service, Platform Support Service dengan sub-group Business Support Service (BSS) yang menyediakan layanan interaksi dan Operation Support Service (OSS) yang menangani *middleware* SaaS, Security Service, SaaS Management Service, SaaS Instance yang merupakan fungsi aktual *service* dari SaaS yang akan digunakan, Data Service dan Integration Service. [2]

Masing-masing *service* bisa bersifat *independent* atau *dependent* terhadap *service* lainnya. Pengguna atau *end-user* hanya mengakses hingga penyedia layanan *service* dan penyida layanan *service* atau *service broker* menampilkan mengenai layanan-layanan yang ada. Gambar 1 merupakan ilustrasi dari layanan SaaS.

Cara penggunaan SaaS ini seperti yang terdapat pada Gambar 1.a adalah pengguna (John dan Alice) mengakses *broker* untuk menggunakan *service* yang diperlukan. *Service* yang tersedia akan terdaftar pada *broker* dan kemudian *service* yang terhubung dengan *broker* dapat dipilih sesuai dengan fungsi yang diperlukan oleh pengguna. Pada ilustrasi tersebut terdapat juga *service* yang menggunakan *service* lainnya seperti *service* French Translation Service yang menggunakan *service* dari French Grammar Service. Semua *service* tidak langsung terhubung dengan pengguna melainkan melalui *broker* atau pihak ketiga. Sehingga terdapat 3 pihak dalam ilustrasi tersebut yaitu pengguna, penyedia *service*



Gambar 2. Pola *Class Diagram* dari *Software-as-a-Service* pada *Cloud Computing*. [5]

dan *broker*. Tujuan dari adanya *broker* adalah membantu pengguna dalam pencarian *service* yang diperlukan sesuai dengan kebutuhan serta memastikan *service* tersebut dapat digunakan.

Peningkatan penjualan informasi dari sebuah produk *service* diadopsi oleh SaaS *delivery model*. Normalnya sebuah perusahaan yang menggunakan SaaS memiliki dua ketentuan harga penjualan yaitu berdasarkan penggunaan dan berdasarkan harga tetap yang bisaanya berasal dari inisialisasi biaya awal [3]. Gambar 1.b merupakan *delivery model* yang digunakan pada SaaS.

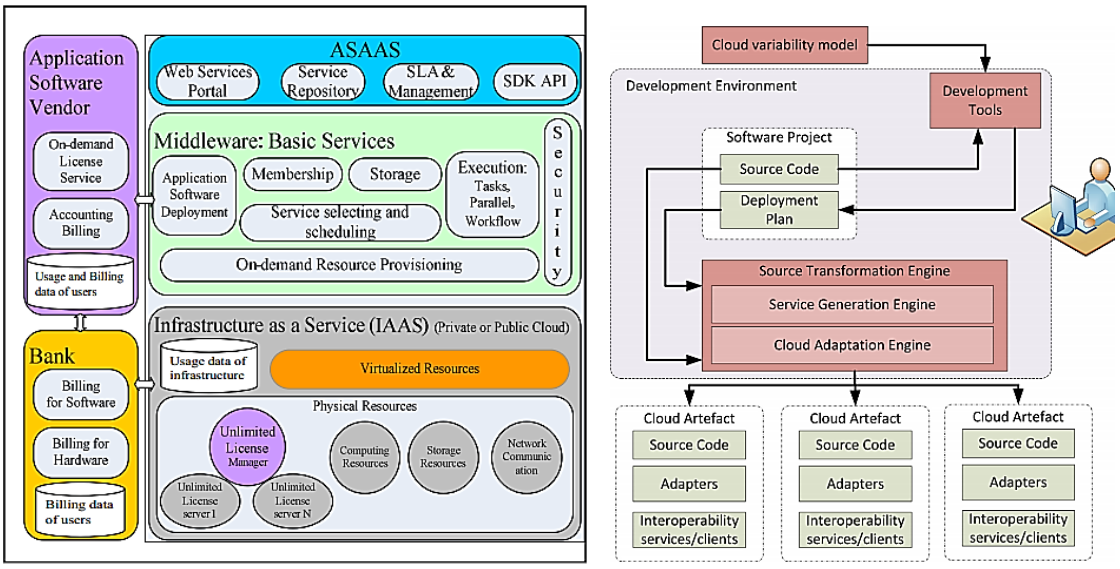
Pada *development cycle* terdapat 3 tambahan fase yaitu Evaluation, Subscribing dan Operations yang merupakan perbedaan dengan *software* yang bersifat *on-premise* (*software* yang terpasang pada komputer pengguna) [3]. Pada fase Platform Evaluation menentukan kecocokan platform kepada *service*, fase Subscribing bertujuan untuk mendapatkan status dari kualitas kecocokan dari kualitas platform untuk menentukan strategi Backup dan Recovery.

2. Tinjauan Pustaka

A. Mobile dan Ubiquitous Computing

Pada *mobile* dan *ubiquitous computing*, *service provider* yang bisaanya menyediakan jaringan internet bertindak sebagai *service* integrator dan aplikasi dibangun oleh pihak ketiga dan dipasang pada SaaS Platform. Penggunaan SaaS pada *mobile* dan *ubiquitous computing* ini diasumsikan bahwa pengguna memiliki jaringan internet yang berkesinambungan.

Terdapat pebedaan dasar dari karakteristik *mobile* dan *ubiquitous computing* dengan komputer desktop yaitu ukuran yang lebih kecil, memiliki *bandwidth* yang rendah dan sewaktu-waktu dapat terputus dari jaringan internet. Perbedaan ini menyebabkan penyedia SaaS harus mampu mencari solusi agar layanan tetap berjalan pada *mobile device*. Solusinya adalah membangun SaaS yang bersifat *adaptive* yakni mampu beradaptasi sesuai dengan jenis *device* dari *end-user* yang menggunakan *service*. Namun pada sisi *customer* memiliki juga peran penting dalam koneksi sehingga diperlukan beberapa



(a) (b)

Gambar. 3. (a) Arsitektur *web-service* dari *Application Software-as-a-Service* (ASAAS) pada *cloud computing* [7] dan (b) Desain *framework development* SaaS [6]

kode dasar untuk menangani *offline caching* ketika tidak terdapat koneksi ke SaaS *provider* dan sinkronisasi ketika kapanpun mendapatkan koneksi ke SaaS *provider*. [2]

B. Cloud Computing

Cloud yang dimaksud disini adalah sebuah jaringan dari pusat data yang merupakan banyak komputer yang terhubung dan mampu menjalankan fungsi yang sama sehingga memberikan pengguna berupa *software* yang *powerfull* dan dapan diakses melalui internet. *Cloud computing* didasari pada dua konsep dasar yaitu Service Oriented Architecture (SOA) dan virtualisasi. *Cloud computing* bisaanya dikenal dengan ”*Everything as a Service*” atau EaaS/XaaS. Secara garis besar *service* diklasifikasikan menjadi 3 jenis yaitu *Infrastructure-as-a-Service* (IaaS), *Platform-as-a-Service* (PaaS) dan *Software-as-a-Service* (SaaS) [4].

Ketiga klasifikasi tersebut memiliki perananan penting dalam pembangunan sebuah *cloud computing*. IaaS berfungsi dalam pembangunan infrastruktur *cloud computing* seperti komputer *server*, media penyimpanan dan jaringan komputer, PaaS berfungsi sebagai penyedia platform yang digunakan dan SaaS adalah dimana aplikasi disimpan dan dipelihara oleh penyedia layanan *cloud* [5].

SaaS aplikasi bisaanya didistribusikan kepada konsumen melalui sebuah internet berupa *web-service* atau API’s yang memungkinan pengguna mengakses aplikasi secara *on-demand*. Hal ini memerlukan strutkur *class diagram* dari SaaS dan *cloud computing* yang ditunjukan oleh Gambar 2 [5].

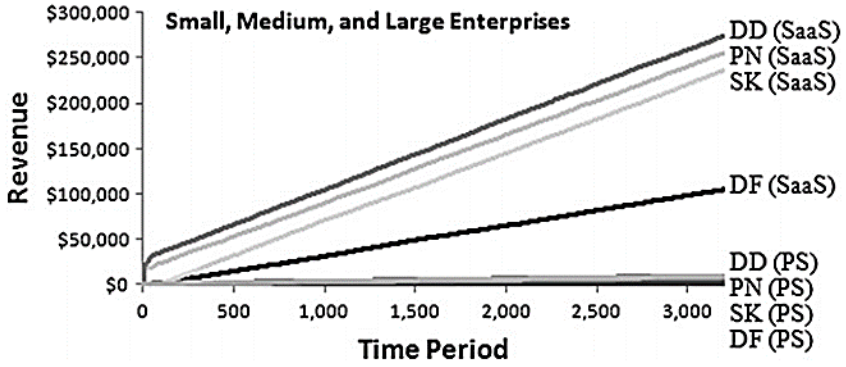
Pada Gambar 2 SaaS *provider* menawarkan SaaS Application yang terdaftar pada SaaS Catalog. Pada *class diagram* bisa terdapat *single* Application Instance yang digunakan oleh banyak pengguna atau satu *instance* untuk satu pengguna. Aplikasi yang ditawarkan oleh SaaS diletakan pada PaaS dan IaaS yang dimana PaaS dan IaaS dapat dimiliki oleh pihak ketiga.

Pada sisi *development*, *service* yang dibuat haruslah bisa bekerja pada banyak platform, karena pengguna menggunakan dari berbagai platform. Terdapat juga kemungkinan ketika sebuah *service* mengalami gagal platform sehingga harus dipindah ke platform lain. Hal ini memerlukan sebuah *framework* sebagai *middleware* dari berbagai macam platform. Hal ini memungkinan *development* hanya memerlukan *framework* untuk membangun *service*. Gambar 3.b merupakan desain *development framework* dari SaaS [6].

Secara umum SaaS memiliki *interface webservice* kepada para pengguna SaaS yang dapat diakses melalui jaringan internet sehingga pengguna dapat menggunakannya kapan saja dan *service* selalu tersedia (*on-demand*). Pengguna tidak perlu membayar sebagai pemilik dari aplikasi melainkan membayar biaya *service* karena menggunakan. Hal ini membuat pengguna dapat memilih fungsi-fungsi yang hanya diperlukan. Gambar 3.a merupakan arsitektur dari desain *webservice* [7].

Perpaduan SaaS dan *cloud computing* telah mengurangi beban biaya yang diperlukan dan kini telah banyak digunakan oleh beberapa penyedia layanan yang besar seperti Google dan iPhone Apps.

Perusahaan berskala kecil pun mampu membangun sebuah *cloud computing* dan SaaS model dengan organisasi kompleksitas seperti universitas [8].



Gambar. 4. Komparasi dari keempat skema penjualan pada penyedia skala kecil, menengah dan besar pada SaaS dan PS. [10]

Dengan *delivery model* menggunakan SaaS pada *cloud computing* telah memberikan sebuah paradigma pembangunan *software* dengan memanfaatkan penggunaan kembali dan memberikan akses *on-demand*. Kualitas dari penggunaan *delivery model* ini telah diuji menggunakan Fuzzy Logic dan hasilnya *medium quality* dengan nilai 0,486 [9].

C. Skema Biaya dan Keuntungan [10]

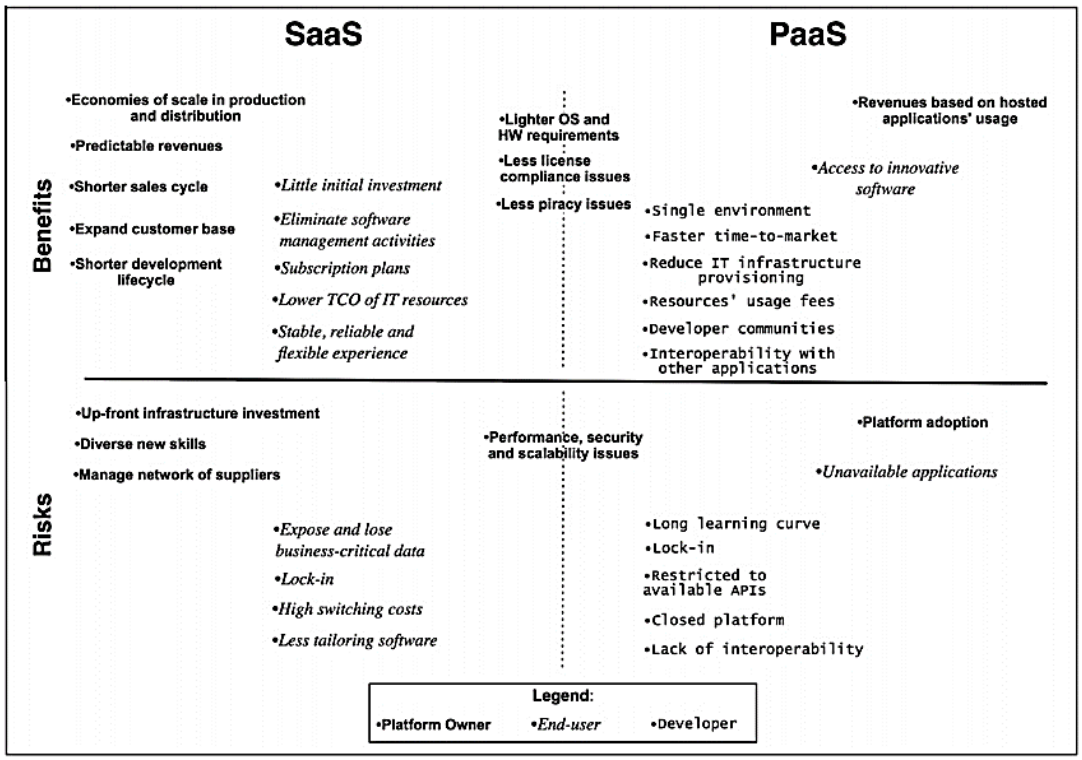
Pada skema biaya dan keuntungan, *software* SaaS dibandingkan dengan *perpetual software* (PS) dengan sistem lisensi dengan skema harga yang berbeda. Penyedia *software* yang berusaha menaikan keuntungan akan menggunakan informasi mengenai pengguna, persaingan dan harga jual pasar untuk menentukan harga.

Pada analisis penjualan *software* digunakan sebuah basis agen simulasi untuk penentuan harga *software* secara dinamis dengan fokus pada SaaS dan *perpetual software licensing* model. Skema yang digunakan ada sebanyak 4 dengan 1 tambahan skema sebagai refrensi yaitu penghargaan Demand-Driven (DD), penghargaan Derivative-Follower (DF), penghargaan Penetration (PN), dan penghargaan Skimming (SK). Untuk refrensi menggunakan *flat-rate* (*fixed-price*) atau FP. Menggunakan metode Analytic Hierarchy Process (AHP) sebagai penentuan mekanisme penghargaan dengan 4 kriteria utama yang disorot yaitu finansial, kemampuan *software*, organisasi dan penyedia. Metode AHP dikembangkan oleh Saaty yang merupakan metode penghitungan bobot yang diturunkan dari Direct Weighting, SMART, SWING dan SMARTER.

Derivative-Follower (DF) merupakan sistem penghargaan yang digunakan penyedia *software* yang memiliki pengetahuan mengenai kondisi pasar dan permintaan aktual dari pengguna. Demand-Driven (DD) merupakan sistem penghargaan dengan menawarkan harga berdasarkan nilai yang diberikan oleh pengguna yang memerlukan informasi mengenai keinginan serta kebisaaan dari pengguna. Penetration-Pricing (PN) merupakan sistem penjualan dengan menawarkan harga yang rendah untuk mendapatkan keuntungan dengan menaikan harga selanjutnya yang dimana target penjualan adalah pengguna yang memiliki sensitifitas yang tinggi namun daya beli yang rendah. Skimming-Pricing (SK) merupakan sistem penjualan dengan memasang harga tertinggi dan kemudian menurunkan harga secara pelan-pelan dengan target penjualan pada pengguna yang benar-benar ingin menggunakan dan membayar *software* tersebut.

Simulasi dari skenario mewakilkan sebuah penjualan *software* dimana pengguna membeli lisensi *software* dan penyedia menawarkan lisensi *software* yang bergantung pada skema penghargaan yang digunakan. Pengguna/*customer*s menentukan pilihan apakah membeli aplikasi SaaS atau *perpetual license* (PS) dari penyedia *software*. Namun pengguna tidak dapat memilih untuk tidak membeli dari kedua pilihan yang ada. Gambar 4 merupakan hasil dari keseluruhan ujicoba pada keempat skema penjualan.

Pada simulasi yang dihasilkan dapat disimpulkan bahwa skema penjualan DD adalah skema penjualan yang paling efektif untuk mendapatkan keuntungan dari pada skema yang lainnya. Namun pada kenyataan mendapatkan informasi yang sempurna mengenai pengguna dan persaingan sangatlah kecil



Gambar 5. Rangkuman dari keuntungan (*benefit*) serta ancaman yang mungkin muncul (*risk*) pada SaaS dan PaaS. [11]

kemungkinannya sehingga skema DD sangat susah untuk diterapkan. Penyedia *software* bisa memilih skema penjualan PN atau SK karena kedua metode tersebut hampir efektif terhadap skema penjualan DD.

D. Perbandingan dengan Platform-as-a-Service [11]

Software-as-a-Service telah diperkenalkan oleh ICT Market Analysts dan Software Industry Assosication sebagai bentuk dari penyempurnaan versi dari model Application Service Provider (ASP) yang menawarkan host dan memberikan akses terhadap *software* melalui sebuah jaringan. Manajemen dari aplikasi ini bersifat terpusat dan menawarkan model penggunaan *one-to-many*. SaaS telah berevolusi menjadi *web-based application* dan memiliki atribut lain seperti *multi-tenant eficiency*, *configurability* dan *scalability*. Penyedia *service* akhirnya mampu menawarkan *service* dari yang besifat minor hingga aplikasi major melalui sebuah *web*site sehingga memberikan fleksibilitas pada sisi penggunaan waktu dan lokasi dari akses.

Platform-as-a-Service (PaaS) merupakan pendekatan *software* yang memfokuskan pada *development cycle* dan penawaran aset untuk aplikasi baru dengan memperbolehkan investasi dan pemeliharaan pada infrastruktur. PaaS memfasilitasi keseluruhan dari *software life cycle* dengan menawarkan layanan untuk desain aplikasi, *development*, *testing* dan *hosting*. Layanan ini memungkinkan kalangan professional maupun dari kalangan nonprofessional dari pembuat *software* membuat aplikasi dari SaaS yang telah disediakan seperti force.com atau membangun sebuah aplikasi baru seperti Google App Engine.

Pembangunan *software* dengan hanya berbasis pada PaaS atau berbasis pada SaaS memiliki keuntungan dan kerugian yang berbeda-beda. Namun kedua metode tersebut menawarkan keuntungan dari sisi kebutuhan kemampuan *hardware* yang digunakan tidak terlalu tinggi dan memiliki isu yang kecil pada lisensi penggunaan. Gambar 5 merupakan rangkuman perbandingan dari model SaaS dan PaaS pada keuntungan dan ancaman yang mungkin muncul.

E. Penerimaan/Kepuasan Pengguna [12]

Untuk mengetahui tingkat penerimaan SaaS di sisi pengguna dilakukan survei empiris yang dilakukan sebanyak 4 kali di sebuah perusahaan e-commerce di China yaitu Alibaba. Aspek dasar yang dinilai ada empat yaitu kemudahan penggunaan (*easy to use*), keamanan *software* (*security*), *reability* dan

responsiveness.

Berdasarkan hasil survei ternyata terdapat faktor lain selain empat yang telah disebutkan yaitu pengaruh sosial. Seorang pengguna *software* bisaanya akan meminta pendapat kepada kerabatnya sebelum memutuskan untuk menggunakan *software*. Meskipun terdapat faktor pengaruh sosial, para pengguna memiliki tingkat penerimaan SaaS masih tinggi. Dari 19 hipotesa awal mengenai dukungan pengguna terhadap SaaS, hanya 3 hipotesa yang tidak terdukung sisanya 16 hipotesa didukung oleh hasil penelitian.

3. Metode Penelitian

A. Optimasi Performa [13]

Pada optimasi performa pada pemasangan aplikasi SaaS dapat dilakukan dengan beberapa metode salah satunya dengan pendekatan Performance Improvement Opportunities (PIO). PIO memberikan titik optimasi penggunaan *hardware* salah satunya adalah permasalahan pada *bootlneck*. Teknik ini menggunakan kombinasi dari *assocication rules* dan perhitungan *performance* untuk membentuk sebuah *matrix* sebagai pendeteksi dari keadaan *bootleneck.*

Matrix yang digunakan adalah SARatio *metric* yang memberikan ketepatan *starting point* untuk PIO melakukan deteksi *bootleneck* daripada *traditional metrics*. Dengan metode ini pendeteksian *bootleneck* mendapatkan ketepatan 83% dari 12 PIO yang dianalisis dan waktu respon sistem yang berhasil diklassifikasi dengan tingkat 100%.

B. Manajemen Sumber Daya (Scheduler)

Pada manajemen sumber daya terdapat beberapa metode yang dapat digunakan dengan konsentrasi manajemen yang berbeda pula. Beberapa metode atau pendekatan yang digunakan yaitu Knapsack Problem untuk manajemen infrastruktur [14], Service Level Agreement (SLA) untuk manajemen lisensi serta penanganan pelanggaran [15] dan manajemen sumber daya untuk meminimalisir biaya [16], dan basis pengguna layanan (*tenant-based*) untuk manajemen penggunaan *hardware* [17].

Pada manajemen infrastruktur menggunakan Knapsack Problem, terdapat virtualisasi pada infratruktur untuk menambah daya pemrosesan data dan membagi pekerjaan pemrosesan data. Penggunaan Knapsack Problem sebagai manajemen *scheduler* rating keberhasilan pemrosesan data berada pada level 83,3% hingga 86,7% [14].

Pada manjemen lisensi dan penanganan pelanggaran dari Service Level Agreement (SLA), SLA digunakan untuk menentukan *Quality of Service* (QoS) dengan tujuan meinimalisir penggunaan biaya dan meningkatkan *Customer Satisfication Level* (CSL). Algoritma yang digunakan adalah SLA-*based resource provisioning* untuk menangani permintaan konsumen yang bersifat dinamis dengan metode BFResvResource, BestFit dan BFReschedReq [15].

Pada penangan biaya (*cost*) menggunakan SLA, terdapat 3 algoritma yang diusulkan yaitu maksimalkan keuntungan dengan meminimalisir jumlah VM (ProfminVM), memaksimalkan keuntungan dengan *rescheduling* pekerjaan (ProfRS), dan memaksimalkan keuntungan dengan memanfaatkan *penalty delay* (ProfPD). Dari ketiga algoritma yang diusulkan ProfPD memiliki performa yang terbaik untuk meminimalisir biaya dan memaksimalkan keuntungan [16].

Pada manajemen sumber daya menggunakan *tenant-based* terdapat 2 permasalahan yaitu penggunaan sumber daya yang tidak mencukupi (*overutilization*) dan penggunaan sumber daya yang berlebihan (*underutilization*). Terdapat 4 pendekatan yang digunakan yaitu berbasis pada autentikasi, *persistence*, alokasi VM, dan penyamarataan penggunaan sumber daya (*load-balancing*). Dengan pendekatan tersebut *server-hours* berhasil dikurangi sebanyak 32% dan rata-rata *over* dan *underutilization* berhasil dikurangi [17].

C. Semantic Interoperability dan Cloud [18]

Pada sebuah *cloud computing* bisa terdiri dari beberapa sub-*cloud computing* lainnya yang saling berhubungan. Masing-masing sub-*cloud* bisa bertukar informasi dan mendukung satu sama lainnya. Hal ini disebut dengan *interoperability* dimana hal ini berbeda dengan *portability* yaitu merupakan menyalin sebuah *service* dan dijalankan di *cloud* masing-masing.

Interoperability memerlukan sebuah manajemen dalam penggunaannya yang bersifat *semantic* sehingga bisa saling berkomunikasi walupun *cloud* bersifat heterogen. Kerangka kerjanya memanfaatkan sebuah *cloud broker* sebagai penyatu dari masing-masing *cloud* yang ada. *Cloud broker* bertugas sebagai

penggabung dari masing-masing *cloud* dan mediator dari komunikasi antar *cloud*. Dengan metode ini ketika pengguna ingin mengakses lebih dari berbagai *service* pada berbagai sumber *cloud* cukup mengakses satu *cloud broker*.

D. Copyright and Agreement Paradigm [19]

Setiap *software* memiliki lisensi dan hak cipta untuk melindungi kekayaan intelektual dan yang lainnya yang terkandung dalam *software* termasuk SaaS. SaaS memiliki perbedaan dalam lisensi dan hak cipta dengan *perpetual license* karena perbedaan ruang lingkup cakupan lisensi. Perbedaan tersebut terletak dalam beberapa hal.

Salah satu perbedaannya terletak pada perjanjian pembayaran pengguna. *Perpetual license* bisaanya disertai dengan biaya diawal dan terdapat pilihan-pilihan tambahan yang tidak termasuk pada biaya awal atau mungkin pilihan tersebut tidak dibutuhkan namun masuk dalam biaya awal. Sedangkan SaaS *agreement* bersifat lebih fleksible dalam struktur pembayaran oleh pengguna yang bergantung pada fungsi apa saja yang digunakan.

Perbedaan selanjutnya terletak pada kualitas *software*. Pada *software* dengan *perpetual license* ketika *software* memiliki masalah, masalah tersebut tidak termasuk di dalam *copyrightability* dan menjadi objek pada perjanjian lisensi sehingga pengguna bisa saja tidak bisa komplain mengenai masalah yang mungkin muncul ketika menggunakan *software* karena tidak termasuk dalam perjanjian. Namun pada SaaS yang memiliki Service Level Agreements (SLA) yang merupakan standar perjanjian *business-to-business* mencakup berbagai macam parameter untuk menyamakan persepsi dari expetasi pengguna dengan performa *service* sesungguhnya.

E. Sekuritas dan Integritas [20]

Karena sifat terbuka dan saling berbagi SaaS pada *cloud computing* maka memberikan celah keamanan yang dapat diserang sehingga mengganggu integritas dari *cloud*. Dalam hal ini diusulkan sebuah pemeriksaan integritas dari sebuah *service* sebelum dipasarkan yaitu IntTest. IntTest merupkan sebuah program yang mampu menganalisa dan menampilkan grafik dari lokasi serangan yang mungkin akan terjadi secara akurat. IntTest tidak memerlukan spesial *hardware* atau kernel untuk menjalakannya sehingga bisa diaplikasikan pada *cloud* dengan skala kecil hingga besar.

Tujuan dari IntTest ini adalah melacak lokasi dan keberadaan dari penyedia *service* yang mencurigakan yang menawarkan *service* yang belum bisa dipercaya. IntTest mengasumsikan bahwa semua komponen *service* adalah *black boxes* yang tidak memerlukan akses khusus ke *hardware* maupun kernel dari platform. Hasilnya IntTest mampu menghemat trafik dari data hingga 40% ketika terdapat serangan pada *cloud* sehingga integritas dan sekuritas tetap terjaga.

4. Hasil dan Pembahasan

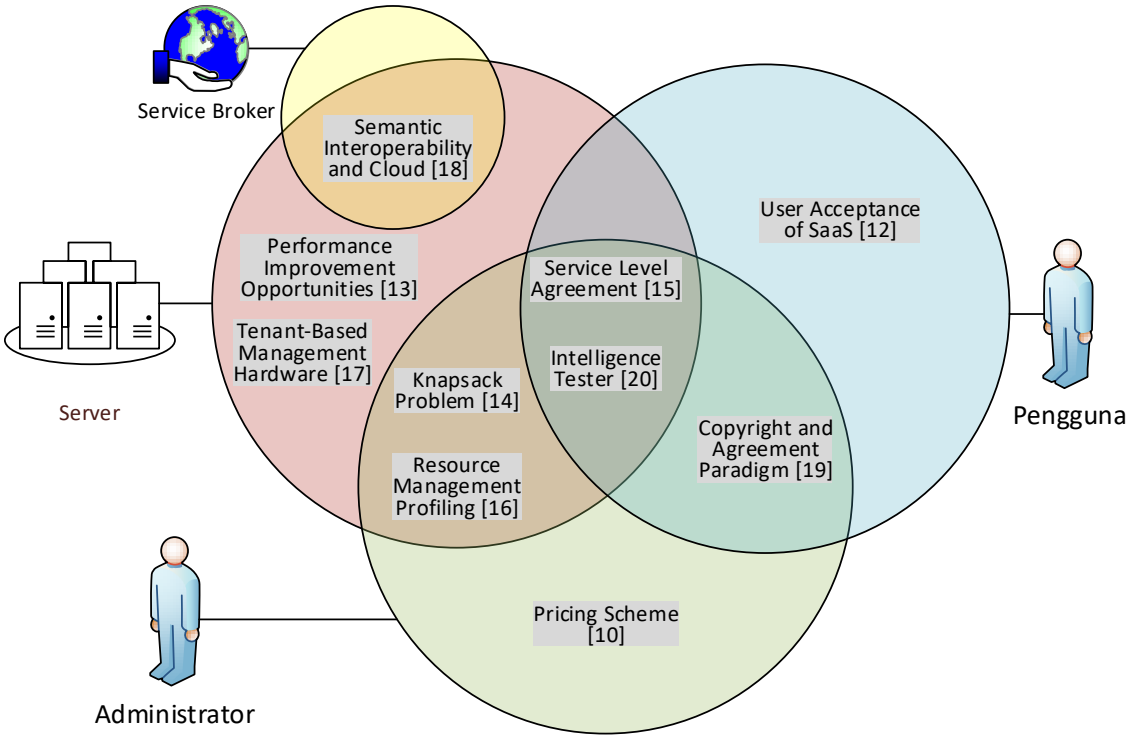
Berdasarkan dari survei pengembangan penggunaan SaaS, dapat disajikan pada bagian apa diterapkan perkembangan dari beberapa hasil penelitian tersebut. Terdapat empat sisi penerapan yang dapat dilihat dari hasil survei yaitu pada sisi *server*, *service broker*, pengelola (*administrator)*, dan pengguna (*user*). Gambar 6 merupakan hasil analisis dari penerapan teknologi yang ada pada SaaS.

Teknologi PIO dan Tenant-Based Management dapat diterapkan pada sisi server untuk meningkatkan performa dari *server*. Pengelola dapat melakukan manajemen infrastruktur dan sumber daya dengan memanfaatkan Knapsack Problem dan Profiling Resource. Pengelola juga dapat memanfaatkan *Pricing Scheme* untuk menentukan harga dari masing-masing *servcice* yang ditawarkan. *Server* juga dapat memanfaatkan layanan lain yang sudah tersedia dan memilih layanan tersebut untuk membantu menopang kebutuhan *server* dengan teknologi semantik.

Antara pengelola, *server*, dan pengguna dapat melakukan ujicoba pada realibilitas *server* dengan memanfaatkan teknologi Intelligence Tester beserta mengatur layanan apa saja yang akan digunakan dengan memanfaatkan SLA. Sebagai bahan pertimbangan penggunaan SaaS, pengguna juga dapat memanfaatkan data dari laporan *acceptance* beserta perjanjian hak milik yang telah disepakati antara pengguna dan pengelola server.

5. Simpulan

Software-as-a-Service atau SaaS membawa paradigma baru pada pembangunan sebuah *software* karena biaya yang dikeluarkan lebih kecil dari pada paradigma tradisional. Hal ini didukung dengan semakin berkembangnya teknologi jaringan yaitu *cloud computing*. SaaS dan *cloud computing* memiliki persamaan dalam objek yang ditawarkan yaitu sebuah *service*.



Gambar 6. Hasil Analisis dan Penerapan Teknologi SaaS

Pada *cloud computing* dengan hanya menggunakan SaaS tidak dapat membangun sebuah *cloud* melainkan diperlukannya model lain yaitu *Infrastructure-as-a-Service* (IaaS) dalam pembangunan infrastruktur dan *Platform-as-a-Service* (PaaS) dalam pembangunan platform dasar.

Service yang ditawarkan oleh *cloud computing* yang dibuat dengan model SaaS dan memanfaatkan *webservice* atau API memiliki portabilitas yang tinggi karena mampu dijalankan dari mana saja dan kapan saja sehingga mewujudkan sifat *on-demand application*. Hal ini diperkuat dengan biaya yang dikeluarkan lebih murah karena pengguna hanya perlu membayar atas *service* yang digunakan atau diperlukan dan pemeliharaan *service* menjadi tanggung jawab penyedia layanan.

Dalam pemasangan SaaS pada *cloud computing* diperlukan optimasi dan manajemen sumber daya dalam menjalankannya. Kemampuan *interoperability* juga diperlukan untuk menghubungkan masing-masing *cloud* yang berbeda dengan tetap menjaga integritas dan keamanan dari *service* yang ditawarkan.

SaaS yang diperkuat dengan Service Level Agreement akan lebih memberikan rasa nyaman kepada pengguna sehingga dapat diterima oleh pasar dengan tetap melindungi hak cipta dari penyedia *service*. Oleh karena itu model pembangunan *software* dengan SaaS dapat dipadukan dengan model dari *Cloud* Computing sehingga dapat memberikan keuntungan lebih banyak.

Daftar Pustaka

[1] D. Wang, Y. Zhang, B. Zhang and Y. Liu, "Research and Implementation of a New SaaS Service Execution Mechanism with Multi-Tenancy Support," in *First International Conference on Information Science and Engineering*, Nanjing, China , 2009.

[2] S. Poslad, Ubiquitous Computing Smart Devices, Environments and Interactions, London: A John Wiley and Sons Ltd, 2009.

[3] G. Kulkarni, P. Chavan, H. Bankar, K. Koli and V. Waykule, "A New Approach to Software as Service Cloud," *International Conference on Telecomunication Systems, Services, and Applications*, vol. 7, pp. 196-199, 2012.

[4] K. K. Fung Yuen, "Software-as-a-Service Evaluation in Cloud Paradigm: Primitive Cognitive Network Process Approach," *Signal Processing, Communication and Computing*, pp. 119-124, 2012.

[5] K. Hashizume, E. B. Fernandez and M. M. Larrondo-Petrie, "A Pattern for Software-as-a-Service in Clouds," *IEEE International Conference on BioMedical Computing,* pp. 140-144, 2012.

[6] J. Guillen, J. Miranda, J. M. Murillo and C. Canal, "A Service-Oriented Framework for Developing Cross Cloud Migratable Software," *The Journal of Systems and Software,* vol. 86, pp. 2294-2308, 2013.

[7] Z. Hou, X. Zhou, J. Gu, Y. Wang and T. Zhao, "ASAAS: Application Software as a Service for High Performance Cloud Computing," *IEEE International Conference on High Performance Computing and Communications,* vol. 12, pp. 156-163, 2010.

[8] J. Leonard, "Dynamics of Cloud-based Software as a Service in Small Communities of Complex Organization," *Hawaii International Conference on System Science,* vol. 47, pp. 3118-3787, 2014.

[9] N. Baliyan and S. Kumar, "Quality Assessment of Software as a Service on Cloud using Fuzzy Logic," *Cloud Computing in Emerging Markets,* pp. 1-6, 2013.

[10] J. Rohitratana and J. Altmann, "Impact of Pricing Schemes on a Market for Software-as-a-Service and Perpetual Software," *Future Generation Computer System,* vol. 28, pp. 1328-1339, 2012.

[11] V. Goncalves and P. Ballon, "Adding Value to The Network: Mobile operators' experiments with Software-as-a-Service and Platforma-as-a-Service Models," *Telematic and Informatics,* vol. 28, pp. 12-21, 2011.

[12] J. Du, J. Lu, D. Wu, H. Li and J. Li, "User acceptance of Software as a Service: Evidence from Costumers of China's leading E-Commerce Company, Alibaba," *The Journal of Systems and Software,* vol. 86, pp. 2034-2044, 2013.

[13] C.-P. Bezemer and A. Zaidman, "Performance Optimization of Deployed Software-as-a-Service Applications," *The Journal of Systems and Software,* vol. 87, pp. 87-103, 2014.

[14] F. Aisopos, K. Tserpes and V. Theodora, "Resource Management in Software as a Service using The Knapsack Problem Model," *Int. J. Production Economics,* vol. 141, pp. 465-477, 2013.

[15] L. Wu, S. K. Garg, S. Versteeg and R. Buyya, "SLA-Based Resource Provisioning for Hosted Software-as-a-Service Application in Cloud Computing Environments," *IEEE Transaction on Services Computing,* vol. 7, pp. 465-485, 2014.

[16] L. Wu, S. K. Garg and R. Buyya, "SLA-based Admission Control for a Software-as-a-Service Provider in Cloud computing Environments," *Journal of Computer and System Science,* vol. 78, pp. 1280-1299, 2012.

[17] J. Espadas, A. Molina, G. Jimenez, M. Molina, R. Ramirez and D. Concha, "A Tenant-Based Resource Allocation Model for Scaling Software-as-a-Service Application over Cloud Computing Infrastructure," *Future Generation Computer System,* vol. 29, pp. 273-286, 2013.

[18] R. Rezaei, T. K. Chiew, S. P. Lee and Z. S. Aliee, "A Semantic Interoperability Framework for Software as a Service Systems in Cloud Computing Environments," *Expert System with Application,* vol. 41, pp. 5751-5770, 2014.

[19] A. Savelyev, "Software-as-a-Service - Legal Nature: Shifting the Existing Paradigm of Copyright Law," *Computer Law & Security Review,* vol. 30, pp. 560-568, 2014.

[20] J. Du, D. J. Dean, Y. Tan, X. Gu and T. Yu, "Scalable Distributed Service Integrity Attestation for Software-as-a-Service Clouds," *IEEE Transaction on Parallel and Distributed Systems,* vol. 25, pp. 730-739, 2014.