ISSN 2085-4579

Analisa Integrasi Aplikasi Akuntansi Berbasis SaaS

Studi Kasus XYZ.com

Danny Y. Djahidin[1]
[1] Program Studi Sistem Informasi, Universitas Mercu Buana, DKI Jakarta, Indonesia
danny.yudin@mercubuana.ac.id

Diterima 3 November 2019
Disetujui 15 Juni 2020

***Abstract*—SaaS application users get strategic benefits, among them the speed of implementation, reliability, and flexibility also optimization of infrastructure capacity. Meanwhile integration benefits users with increase of productivity, data accuracy, speed and flexibility. Those benefits drive XYZ.com, an online store in Jakarta, to integrate its web commerce application with SaaS accounting application. System integration design is carried out by analysis and observation of current business processes using the business objectives model. It is concluded that with system integration, XYZ.com wants to achieve 3 business objectives, i.e. (1) increasing the number of orders recorded in the accounting application, (2) improving stock accuracy, and (3) reducing the number of order cancellations due to unavailability of goods. System integration is designed to enable information retrieval such as products, contacts, transactions, and journals from accounting application through available API. Further research needs to be carried to realize the list of requirements that have been described in the feature tree to the stage of software development and implementation so that the objectives set by the business can be measured and improved.**

***Index Terms*—API, Business Objectives Model, Feature Tree, Integration**

I. PENDAHULUAN

Dalam kehidupan masyarakat dewasa saat ini, sistem informasi memainkan peranan penting dalam mediasi antara sumber daya informasi dan pengguna informasi di berbagai bidang. Pada masa awal bisnis komputasi, sebagian besar perangkat lunak ditulis dari nol menyebabkan pengembangan perangkat lunak menjadi mahal dan lama sehingga tidak bisa mengejar dengan cepat perkembangan teknologi internet dan PC yang membuka pasar baru perangkat lunak serta arsitektur perangkat lunak baru [1], [2]. Namun biaya *bandwidth* yang menurun memberikan kesempatan bagi perusahaan untuk membeli tingkat konektivitas tertentu dengan harga yang terjangkau memungkinkan aplikasi online atau SaaS untuk beroperasi dengan baik [3].

Beberapa studi [4]–[6] mengatakan bahwa pengguna diuntungkan dengan kecepatan dari implementasi SaaS, keandalan, serta fleksibilitas dan optimasi kapasitas infrastruktur. Hal ini memberikan pengguna keuntungan strategis untuk dapat memberikan nilai lebih baik konsumen dibandingkan para pesaing lain. Namun pada sisi lain pengguna selalu menginginkan agar SaaS dapat disesuaikan dengan kebutuhan serta terintegrasi dengan sistem berjalan mereka, dan apabila pengguna memiliki arsitektur TI yang telah matang maka pengguna cenderung menginginkan penyesuaian terhadap aplikasi mereka [7]. Sebagian besar pebisnis solusi *cloud* telah mengembangkan *Application Programming Interfaces* (API) untuk mendukung interaksi antar-mesin, data masuk dan keluar dari *cloud*, dan dari perusahaan ke perusahaan [8]. Dengan kata lain, tersedianya API memungkinkan organisasi untuk melakukan penyesuaian, pengembangan dan integrasi antara sistem yang sudah berjalan sebelumnya dengan SaaS tanpa harus memahami apa yang terjadi dibalik SaaS.

Gleghorn menjelaskan bahwa perusahaan melakukan integrasi dengan tujuan untuk meningkatkan produktivitas, akurasi data, kecepatan dan fleksibiltas [9]. Hal ini juga yang mendasari XYZ.com, salah satu *online store* di Jakarta, untuk melakukan integrasi aplikasi web *commerce* dengan aplikasi akuntansi berbasis SaaS. Integrasi antara kedua aplikasi menjadi sangat penting bagi XYZ.com untuk mengurangi kesalahan data pada aplikasi akuntansi yang digunakan karena saat ini pengguna harus melakukan pengisian data kembali ke dalam aplikasi akuntansi berbasis SaaS atas dasar laporan penjualan aplikasi web *commerce* secara manual. Untuk itu peneliti mengusulkan untuk memanfaatkan teknologi API yang disediakan oleh penyedia aplikasi akuntansi berbasis SaaS agar dapat terintegrasi dengan web *commerce* XYZ.com.

II. LANDASAN TEORI

A. *Software as a Service*

Pada masa awal bisnis komputasi, sebagian besar perangkat lunak ditulis dari nol sehingga

pengembangan perangkat lunak menjadi mahal dan lama [1]. Hal ini menyebabkan pengembangan perangkat lunak konvensional tidak bisa mengejar dengan cepat perkembangan teknologi internet dan PC yang membuka pasar baru perangkat lunak serta arsitektur perangkat lunak baru [2]. Produk *commercial off the shelf* (COTS) hadir sebagai sebuah sistem perangkat lunak yang dapat disesuaikan dengan kebutuhan pelanggan yang berbeda tanpa mengubah kode sumber sistem [10]. Keinginan untuk menurunkan biaya perangkat lunak, mengurangi waktu, mempermudah implementasi, dan menghindari "*reinventing the whell*" telah mendorong peralihan menuju akusisi perangkat lunak berbasis COTS, walaupun di sisi lain pengguna tidak dapat menambahkan fitur maupun proses sesuai dengan kebutuhan bisnis mereka [1], [11].

Secara tradisional, penyedia menjual perangkat lunak berbasis COTS ke pengguna dan membantu memasangnya di situs pengguna sehingga pengguna harus menyediakan infrastruktur, perangkat keras, dan layanan dukungan TI untuk memungkinkan penggunaan perangkat lunak secara berkelanjutan [12]. Namun biaya *bandwidth* yang menurun memberikan kesempatan bagi perusahaan untuk membeli tingkat konektivitas tertentu dengan harga yang terjangkau memungkinkan aplikasi online untuk beroperasi dengan baik [3]. Hal ini yang kemudian melahirkan model bisnis perangkat lunak sebagai layanan, atau *software as a service* (SaaS). Dengan model ini pengguna hanya membayar bila mereka menggunakan perangkat lunak dan dalam banyak kasus, SaaS mungkin terbukti lebih murah daripada memiliki dan memelihara sistem dan teknologi secara *in-house*, sehingga memungkinkan pengguna menghemat uang untuk dukungan dan biaya upgrade, infrastruktur, personil, dan implementasi [3], [13]. Selain penghematan biaya kepemilikan saat membeli sebuah perangkat lunak, pengguna juga diuntungkan dengan kecepatan dari implementasi SaaS, keandalan, serta fleksibilitas dan optimasi kapasitas infrastruktur [4]–[6].

B. Integrasi SaaS

Saat ini transaksi elektronik diselesaikan melalui internet, namun bila tidak ada sistem ERP sebagai sarana pendukung proses internal untuk perusahaan, sulit untuk membayangkan bagaimana sistem dapat menerima pesanan melalui internet, dicetak, ditransfer ke sistem manajemen produksi, dan kemudian akan menjadi input untuk proses bisnis pada internet [14]. Sehingga integrasi menjadi penting karena memungkinkan pertukaran informasi dua arah atau sinkronisasi dengan melibatkan dua (atau lebih) sistem termasuk sistem ERP dan juga *e-commerce* [15].

Proses bisnis sendiri bisa menjadi sangat rumit dan harus dijalankan di banyak aplikasi dan layanan. Dengan meningkatnya kompleksitas proses bisnis, maka aplikasi atau layanan SaaS tunggal dalam perangkat lunak *cloud* tidak dapat memenuhi semua persyaratan bisnis dengan sendirinya. Untuk memenuhi kebutuhan bisnis sambil memanfaatkan layanan berbasis SaaS, aplikasi bisnis harus berintegrasi dengan aplikasi *on-premise* dan atau SaaS lain [4]. Integrasi memungkinkan manipulasi dan pertukaran data antar aplikasi yang dapat dilakukan di atas sebuah *interface* dengan membatasi dua entitas independen, dalam hal ini aplikasi, sehingga dapat berinteraksi atau berkomunikasi satu sama lain. *Interface* sendiri dinyatakan dalam bentuk operasi parameter sehingga kode sumber internalnya tidak pernah terungkap [10].

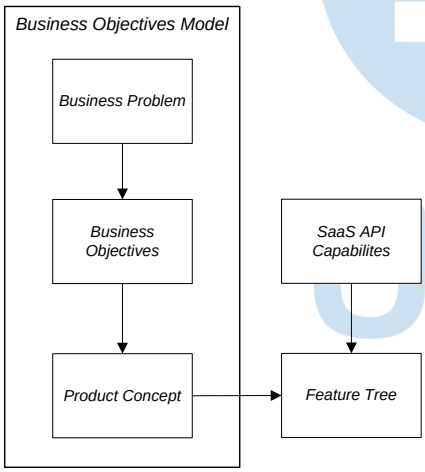
SaaS sebagai bagian dari solusi *cloud*, telah memiliki *Application Programming Interfaces* (API) untuk mendukung interaksi antar-mesin, data masuk dan keluar dari *cloud*, dan dari perusahaan ke perusahaan lain [8]. API tersebut dibangun diatas protokol HTTP dengan menggunakan standar XML atau JSON dan hanya dapat dimanipulasi dengan menggunakan empat metode utama protokol HTTP yaitu GET, PUT, DELETE dan POST [8], [16]. Dengan keterbukaan data pada SaaS, maka dimungkinkan untuk melakukan integrasi dengan sistem lama maupun baru melalui API. Keterbukaan SaaS terhadap integrasi didorong oleh keinginan perusahaan agar SaaS dapat disesuaikan dengan kebutuhan dan diintegrasi dengan sistem berjalan, dan apabila pengguna memiliki arsitektur TI yang telah matang dan telah mengotomatisasi proses bisnis strategis, maka pengguna akan cenderung menginginkan penyesuaian yang ekstensif terhadap aplikasi [7].

III. METODE PENELITIAN

Pengembangan sistem dibuat dengan mengikuti apa yang dibutuhkan oleh bisnis (*business requirements*); apa yang perlu dilakukan oleh pengguna (*user requirements*); apa yang harus dilakukan perangkat lunak (*functional requirements*); karakteristik yang harus dimiliki sistem (*non-functional requirements*); dan bagaimana sistem harus dibangun (*system requirements*) sedangkan fungsi perangkat lunak yang diinginkan biasanya kompleks, sehingga itu dibutuhkan analisa kebutuhan bisnis dari pengguna sebelum melakukan integrasi [17]. Kebutuhan perangkat lunak harus menunjukkan kualitas perangkat lunak yang dikembangkan atau disesuaikan untuk menyelesaikan masalah tertentu seperti otomatisasi, mendukung proses bisnis, memperbaiki kekurangan perangkat lunak yang ada, mengontrol perangkat keras, dan lain-lain [18]. Kebutuhan perangkat lunak sendiri terbagi menjadi dua jenis, yaitu kebutuhan fungsional yang menyatakan apa layanan yang harus disediakan sistem dan bagaimana sistem harus bereaksi terhadap input

dan situasi tertentu dan kebutuhan non-fungsional yang menyatakan performa sistem yang diinginkan termasuk interoperabilitas dengan sistem lain [10]. Dengan begitu perangkat lunak biasanya merupakan kombinasi kompleks dari kebutuhan pengguna yang berbeda di berbagai tingkat organisasi dan lingkungan di mana perangkat lunak akan beroperasi.

Kebutuhan integrasi dengan SaaS sendiri dibagi menjadi tiga kategori, pertama dimana aplikasi utama bisnis mengambil dan memperbaharui data dalam SaaS, kedua dimana SaaS menjadi aplikasi pemrosesan logika bisnis, atau terakhir, gabungan keduanya yang berarti SaaS dapat mengambil dan mengolah serta memproses data tersebut menggunakan logika bisnis dimana data dan aturan yang digunakan berasal dari aplikasi lain yang terintegrasi [4]. Dapat disimpulkan bahwa untuk melakukan integrasi diperlukan pengetahuan atas data dan logika bisnis yang disediakan oleh SaaS melalui API. Pemahaman yang jelas tentang fungsi yang diinginkan pengguna akan menghasilkan elemen minimum yang diperlukan dalam sinkronisasi maupun replikasi data antara SaaS dengan aplikasi lain [19], sehingga dibutuhkan analisa lagi terhadap kapabilitas API untuk menyelaraskan kebutuhan bisnis dari pengguna dengan SaaS itu sendiri.



Gambar 1. Alur penelitian

Dalam menentukan kebutuhan bisnis, pertama harus diketahui dahulu masalah apa yang dihadapi dan hal apa saja yang menghalangi bisnis dalam mencapai tujuannya sedangkan kebutuhan bisnis harus ditentukan sebelum fitur yang ada pada sistem dapat sepenuhnya ditetapkan [20], [21]. Fitur sendiri merupakan satu atau lebih kapabilitas sistem yang saling terkait secara logis dan dijelaskan oleh serangkaian persyaratan fungsional untuk memenuhi kebutuhan bisnis serta memberikan nilai tambah kepada pengguna [20], [21]. Statistik mengungkapkan bahwa faktor kebutuhan bisnis berpengaruh signifikan pada kesuksesan integrasi aplikasi [22].

Dengan demikian peneliti perlu menggali terlebih dahulu permasalahan yang dimiliki oleh bisnis untuk mendapatkan kebutuhan yang diinginkan dan kemudian membangun konsep fitur yang akan ditangani oleh sistem namun dengan penyesuaian atas kapabilitas API dari aplikasi akuntansi berbasis *SaaS*. Dari analisa atas kedua data tersebut, kemudian disusun fitur yang harus dimiliki oleh integrasi sistem dengan menggunakan *feature tree* dan juga persyaratan fungsional yang dibutuhkan. *Feature tree* sendiri digunakan kareana dapat memberikan pandangan yang lebih baik atas fitur – fitur yang akan dibangun dalam sebuah aplikasi [20].

IV. HASIL DAN PEMBAHASAN

A. Analisa Kebutuhan Sistem

Dalam membangun sistem, ada persyaratan bisnis yang menjelaskan mengapa proyek pengembangan sistem dilakukan sehingga dapat membantu menentukan tujuan keseluruhan sistem serta mengklarifikasi kontribusi yang terukur bagi organisasi [17]. Untuk dapat memahami kebutuhan bisnis dari XYZ.com, dilakukan pengumpulan kebutuhan dari sisi bisnis dan juga pengguna dengan menggunakan wawancara, observasi lapangan, dan analisa data. Kebutuhan bisnis dapat berasal dari sponsor pendanaan, eksekutif perusahaan, manajer pemasaran, atau visioner produk [21] dan dalam penelitian ini kebutuhan bisnis didapat dengan wawancara kepada pemilik XYZ.com yang juga merupakan eksekutif perusahaan dan divalidasi dengan analisa terhadap data atas proses bisnis berjalan.

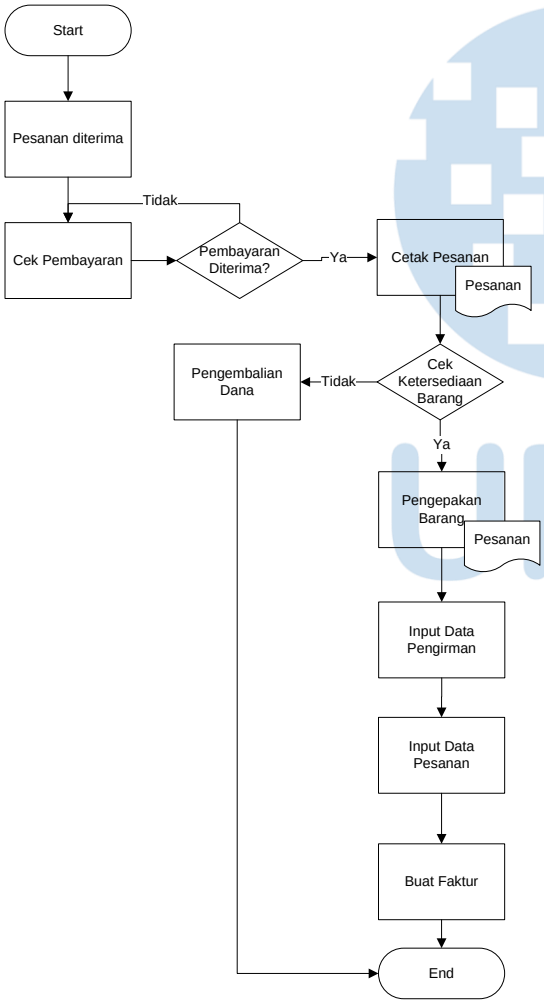
Dari hasil wawancara ditemukan sebuah masalah, yaitu informasi penjualan yang dihasilkan oleh aplikasi akuntansi tidak akurat dan tidak dapat diandalkan oleh pengguna. Masalah tersebut juga didukung oleh data jumlah pesanan yang diperoleh dari aplikasi akuntansi dan juga data penjualan *website* seperti yang ditunjukkan oleh pada tabel 1. Data menunjukkan bahwa pesanan yang tercatat pada aplikasi akuntansi hanya sebanyak 67% dari total 516 pesanan yang tercatat pada *website* di periode 13 September – 19 September 2018.

Tabel 1. Jumlah pesanan tercatat pada aplikasi akuntansi

Tanggal	Total Pesanan *Website*	Total Pesanan Aplikasi Akuntansi	%

13/09/2018	141	44	31%
14/09/2018	90	78	87%
15/09/2018	111	80	72%
17/09/2018	83	71	86%
18/09/2018	82	67	82%
19/09/2018	9	8	89%
Total	516	348	67%

Berdasarkan hasil wawancara dan analisa data penjualan, penelitian dilanjutkan dengan memahami proses bisnis berjalan untuk mengetahui akar dari permasalahan yang terjadi. Analisa proses bisnis diperlukan untuk menentukan ruang lingkup proyek dan membantu mengidentifikasi proses yang memerlukan dukungan serta mengidentifikasi kebutuhan integrasi aplikasi [23].



Gambar 2. Proses bisnis berjalan

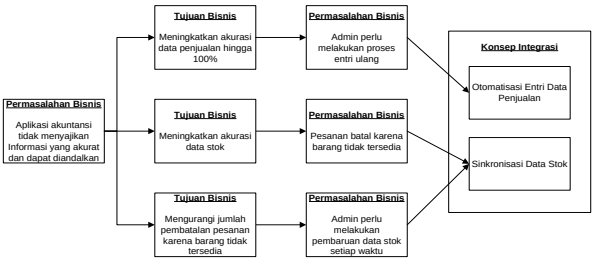
Gambar 2 menunjukkan proses bisnis berjalan yang dimulai dari pengelolaan data penjualan *website* sampai dientri pada aplikasi akuntansi. Admin menerima pesanan dari *website* dan kemudian mengecek mutasi rekening bank melalui layanan *internet banking* untuk mengetahui apakah pembayaran sudah dilakukan oleh pembeli atau belum. Apabila pembayaran sudah diterima, maka admin akan mencetak pesanan dan melakukan pengemasan barang apabila stok tersedia. Barang yang telah dikemas dalam sebuah paket kemudian dikirim melalui logistik rekanan, dimana admin akan menerima tanda terima pengiriman dan menginput nomor tanda terima ke dalam *website* untuk memberitahukan kepada pembeli bahwa pesanan sudah dipenuhi dan dikirimkan. Setelah pesanan terkirim pada akhir proses, baru kemudian admin menginput data pesanan ke dalam aplikasi akuntansi dan membuat faktur untuk mencatat penjualan.

Dari hasil pengamatan atas alur proses bisnis berjalan, peneliti menemukan bahwa pengguna tidak disiplin dalam mengelola data penjualan pada aplikasi akuntansi dilakukan setelah selesai mengirimkan pesanan yang diterima dari *website* sehingga data pesanan tidak sepenuhnya dientri pada aplikasi akuntansi. Hal ini mengakibatkan data penjualan dan stok pada aplikasi akuntansi menjadi tidak dapat diandalkan oleh para penggunanya. Pada sisi lain proses pembaruan data ketersediaan barang tidak dijalankan oleh admin sehingga dapat terjadi kesalahan informasi pada *website* XYZ.com. Kesalahan informasi tersebut menyebabkan pembeli membuat pesanan atas barang yang sudah tidak tersedia lagi.

Business objectives model digunakan dalam merancang solusi karena pemodelan tersebut memungkinkan peneliti bersama dengan para pemangku kepentingan untuk mengidentifikasi nilai dan esensi dari sebuah proyek dan kemudian menggunakan nilai tersebut pada setiap kesempatan dalam membangun persyaratan sistem [20]. *Business objectives model* memiliki 3 komponen yang saling terhubung, yaitu permasalahan bisnis, tujuan bisnis, dan konsep produk beserta fitur, dimana pemodelan selalu dimulai dari permasalah bisnis yang memiliki sebuah tujuan bisnis.

Berdasarkan pemodelan pada gambar 3, diketahui bahwa ada 3 tujuan bisnis yang akan dipenuhi dari integrasi sistem yang dirancang yaitu, (1) meningkatkan jumlah pesanan yang tercatat pada aplikasi akuntansi, (2) memperbaiki akurasi stok, dan (3) mengurangi jumlah pembatalan pesanan akibat tidak tersedianya barang. Beatty & Chen [20] mengatakan bahwa tujuan bisnis harus dapat terukur secara kuantitatif, sehingga peneliti dan para pemangku kepentingan menentukan bahwa (1) pesanan dari *website* harus tercatat seluruhnya (100%) pada aplikasi akuntansi, (2) meningkatkan akurasi stok menjadi 100%, yang secara langsung akan (3) meniadakan pembatalan pesanan akibat stok kosong. Sehingga untuk mencapai seluruh tujuan bisnis

tersebut, integrasi yang dikembangkan harus mampu melakukan otomatisasi entri data pesanan dan sinkronisasi ketersediaan barang antara *website* XYZ.com dengan aplikasi akuntansi.



Gambar 3. *Business objectives model*

B. *Rekomendasi dan Usulan*

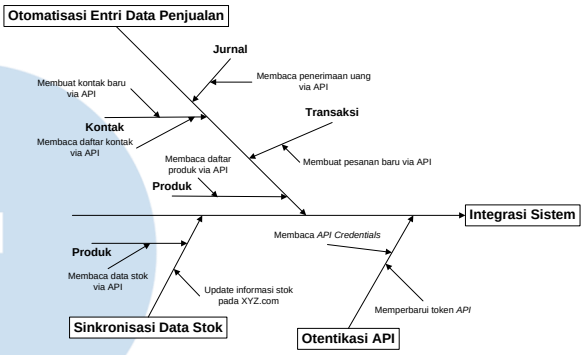
Agar informasi penjualan dan stok dapat tersinkronisasi secara otomatis dan lebih tepat waktu maka integrasi antara XYZ.com dan aplikasi akuntansi harus memungkinkan terjadinya proses pertukaran data melalui protokol internet karena baik XYZ.com dan aplikasi akuntansi merupakan dua layanan terpisah yang diakses melalui internet. Aplikasi akuntansi yang digunakan merupakan sebuah layanan *SaaS* dengan *API* sebagai salah satu fitur sehingga memungkinkan penggunanya membangun integrasi dengan aplikasi lainnya. Integrasi dapat dilakukan dengan menggunakan arsitektur *Representational State Transfer* atau REST yang mengandalkan komunikasi *stateless*, *client-server*, dan *cacheable* menggunakan protokol HTTP [8].

Tabel 2. Akses *API*

Objek	Keterangan	Akses
Kontak	Menampung daftar kontak yang melakukan suatu transaksi	*Read, Create, Update*
Produk	Menampung daftar produk yang dapat dijual maupun dibeli beserta informasi ketersediaan barang	
Pembelian	Daftar transaksi pembelian	
Penjualan	Daftar transaksi penjualan	
Jurnal	Jurnal akuntansi yang dibentuk setelah transaksi pembelian, pembayaran, penjualan, dan penerimaan terjadi	

Kapabiltas *API* aplikasi akuntansi pada Tabel 2 diperoleh dengan menganalisa dokumentasi yang disediakan oleh penyedia layanan dimana pengguna dapat mengambil informasi dari lima objek yang tersedia yaitu: kontak, produk, pembelian, penjualan dan jurnal serta melakukan entri atau pembaruan data. Akses terhadap seluruh objek didapat setelah melakukan otentikasi aplikasi dengan menggunakan protokol *OAuth* versi 2.0. Hasil analisa ini kemudian digunakan dalam membuat daftar fitur yang diperlukan untuk memenuhi konsep integrasi yang telah disepakati.

Gambar 4 memuat pohon fitur beserta objek yang perlu diakses melalui *API* agar konsep integrasi dapat diwujudkan. Tidak semua objek digunakan untuk mengembangkan fitur, hanya objek yang berkaitan dengan penjualan dan ketersediaan barang saja yang perlu diakses. Untuk dapat melakukan entri pesanan secara otomatis, sistem harus dapat membaca dan menulis ke dalam 4 objek yaitu (1) kontak, dimana informasi pembeli disimpan dan dibaca, (2) produk, dimana informasi produk beserta nilai jual perlu dibaca, (3) penjualan, dimana data pesanan dan informasi penjualan disimpan dan (4) jurnal, dimana informasi transaksi penerimaan uang dibaca.

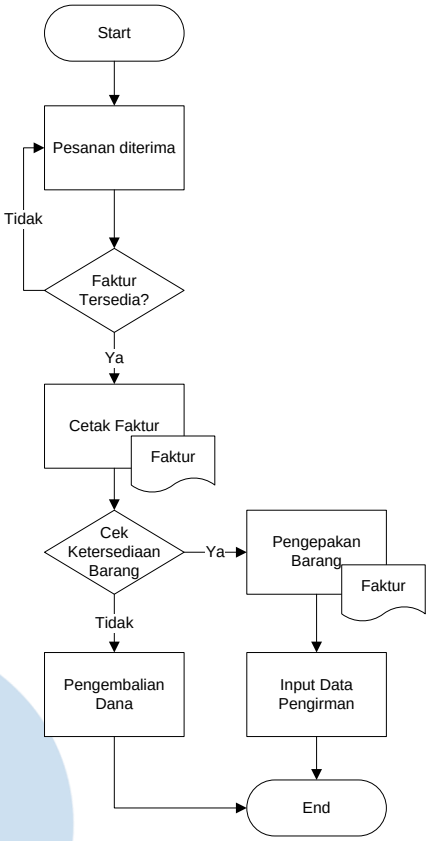


Gambar 4. Pohon fitur

Integrasi dimulai dengan melakukan otentikasi terlebih dahulu agar mendapatkan akses terhadap objek-objek yang ada. Otentikasi dilakukan dengan metode *OAuth* 2.0 dimana aplikasi perlu mendaftarkan diri pada layanan akuntansi *SaaS* dengan menggunakan id dan kunci rahasia untuk meminta token akses kepada server. Jika berhasil, maka aplikasi akan mendapatkan token beserta *refresh* token dan dengan menggunakan token akses yang digunakan sah, server akan melayani permintaan dan memberikan respon data kepada aplikasi. Token akses memililki umur selama 24 jam sehingga setelah melewati umur tersebut aplikasi perlu melakukan permintaan *refresh* token dan jika permintaan valid, maka server akan memberikan token akses baru kepada aplikasi.

Setelah mendapatkan akses, integrasi dimulai dengan mencocokkan data produk yang ada di *website* XYZ.com dan aplikasi akuntansi. Hal ini

bertujuan agar kedua aplikasi mendaftarkan kode produk yang sama sehingga dapat mengidentifikasi produk yang sejenis. Dengan begitu *website* XYZ.com dapat membaca informasi stok yang ada pada aplikasi akuntansi melalui *API* produk dan memperbarui informasi ketersediaan barang secara cepat dan otomatis. Otomatisasi entri data pesanan lebih kompleks dibandingkan proses pembaruan stok karena harus membaca objek kontak, produk, transaksi, dan jurnal melalui *API*. Entri data ke aplikasi akuntansi dimulai dengan ketika pembeli menyelesaikan proses belanja pada XYZ.com, dimana sistem akan mengirimkan data pembeli terlebih dahulu untuk menentukan apakah pembeli merupakan pelanggan baru atau lama dengan mencocokkan nomor telepon atau alamat email yang terdaftar. Apabila data pembeli sudah terdaftar, maka sistem akan menggunakan data tersebut sebagai pemilik pesanan yang akan dientri, bila tidak maka sistem akan secara otomatis membuat kontak baru dengan tipe pembeli pada aplikasi akuntansi melalui *API* yang tersedia. Selanjutnya sistem akan membuat pesanan melalui *API* dengan menggunakan data pesanan yang ada pada *website* XYZ.com dan juga secara otomatis akan menandai pesanan sebagai sudah terbayar. Proses menandai pesanan sebagai sudah terbayar memanfaatkan fitur rekonsiliasi transaksi yang dimiliki oleh aplikasi akuntansi dimana pesanan dapat secara otomatis terkonfirmasi apabila transaksi uang masuk telah dicocokkan dengan pesanan yang dientri. Dengan terkonfirmasinya pesanan, maka pengguna dapat mengetahui pesanan mana yang perlu diproses lebih lanjut secara otomatis. Proses baru ini juga memberikan nilai tambah lain bagi pengguna dimana proses pengembalian dana dapat dicatat dengan baik karena dilakukan setelah data penjualan dibentuk dalam sistem akuntansi.



Gambar 5. Proses bisnis usulan

Dengan otomatisasi proses entri data pesanan dan stok maka diperlukan perubahan proses bisnis untuk mendukung berjalannya integrasi antar sistem, seperti yang ditunjukkan pada Gambar 5. Pada proses bisnis yang diusulkan, admin tidak perlu lagi menerima pesanan dan mengecek mutasi rekening bank secara manual, karena proses tersebut akan dijalankan secara otomatis melalui integrasi dengan aplikasi akuntansi. Apabila pembayaran sudah diterima, maka admin dapat langsung membuat dan mencetak faktur penjualan lalu dilanjutkan dengan pengemasan barang apabila stok tersedia. Proses setelah barang dikemas dalam dan dikirim melalui logistik rekanan tetap berjalan sama seperti proses sebelumnya, dimana admin akan menerima tanda terima pengiriman dan menginput nomor tanda terima ke dalam *website* XYZ.com untuk memberitahukan kepada pembeli bahwa pesanan telah terkirim. Proses entri faktur penjualan yang dilakukan akan menyebabkan stok berkurang pada aplikasi akuntansi, hal ini dapat dimanfaatkan oleh sistem integrasi untuk melakukan pembaruan informasi ketersediaan barang secara otomatis ke *website* XYZ.com.

V. SIMPULAN

Pebisnis solusi *cloud* telah mengembangkan *Application Programming Interfaces* (API) untuk mendukung interaksi antar-mesin, data masuk dan keluar dari *cloud*, sehingga memungkinkan pengguna untuk melakukan penyesuaian, pengembangan dan integrasi antara sistem yang sudah berjalan dengan *SaaS* yang pada akhirnya dapat meningkatkan produktivitas, akurasi data, kecepatan dan fleksibiltas. Hal ini juga yang mendasari XYZ.com, salah satu *online store* di Jakarta, untuk melakukan integrasi aplikasi web *commerce* dengan aplikasi akuntansi berbasis *SaaS*. Perancangan integrasi sistem dimulai dengan melakukan analisa serta observasi terhadap proses bisnis berjalan dengan menggunakan *business objectives model*. Disimpulkan bahwa dengan integrasi sistem, XYZ.com ingin mencapai 3 tujuan bisnis yaitu, (1) meningkatkan jumlah pesanan yang tercatat pada aplikasi akuntansi, (2) memperbaiki akurasi stok, dan (3) mengurangi jumlah pembatalan pesanan akibat tidak tersedianya barang. Dengan demikian integrasi sistem dirancang untuk dapat mengambil informasi seperti produk, kontak, transaksi, dan jurnal dari aplikasi akuntansi melalui API yang tersedia.

Diharapkan daftar kebutuhan yang telah dijabarkan pada *feature tree* dapat direalisasikan ke tahap pengembangan perangkat lunak dan implementasi sehingga tujuan yang telah ditetapkan oleh bisnis, yaitu: (1) pesanan dari *website* harus tercatat seluruhnya (100%) pada aplikasi akuntansi, (2) meningkatkan akurasi stok menjadi 100%, yang secara langsung akan (3) meniadakan pembatalan pesanan akibat stok kosong, dapat diukur pencapaiannya dan ditingkatkan menjadi lebih baik lagi.

DAFTAR PUSTAKA

[1] M. Keil and A. Tiwana, "Beyond Cost : The Drivers of COTS Application Value," *IEEE Softw.*, vol. 22, pp. 64–69, 2005.

[2] M. Aoyama, "New Age of Software Development: How Component-Based Software Engineering Changes the Way of Software Development?," *1998 Int. Work. CBSE*, no. July 1998, pp. 1–5, 1998.

[3] A. Dubey and D. Wagle, "Delivering software as a service," *McKinsey Q.*, vol. 6, no. May, pp. 1–12, 2007, doi: 10.1021/cr068365a.

[4] F. Liu, W. Guo, Z. Q. Zhao, and W. Chou, "SaaS integration for software cloud," *Proc. - 2010 IEEE 3rd Int. Conf. Cloud Comput. CLOUD 2010*, pp. 402–409, 2010, doi: 10.1109/CLOUD.2010.67.

[5] A. Susarla, A. Barua, and A. B. Whinston, "Myths about Outsourcing to Application Service Providers," *IT Prof.*, vol. 3, no. June, pp. 32–35, 2001.

[6] B. Waters, "Software as a service: A look at the customer benefits," *J. Digit. Asset Manag.*, vol. 1, no. 1, pp. 32–39, 2005, doi: 10.1057/palgrave.dam.3640007.

[7] M. Xin and N. Levina, "Software-as-a Service Model : Elaborating Client- Side Adoption Factors," *29th Int. Conf. Inf. Syst.*, 2008.

[8] G. R. Vijay and A. R. M. Reddy, "Cloud Application Programming Interface Based On REST Framework," *Int. J. Eng. Res. Technol.*, vol. 2, no. 6, pp. 2202–2206, 2013.

[9] R. Gleghorn, "Enterprise application integration: A manager's perspective," *IT Prof.*, vol. 7, no. 6, pp. 17–23, 2005, doi: 10.1109/MITP.2005.143.

[10] I. Sommerville, *Software Enginnering*, 9th ed. Addison-Wesley, 2011.

[11] D. McKinney, "Impact of commercial off-the-shelf (COTS) software on the interface between systems and software engineering," *Proc. 21st Int. Conf. Softw. Eng. - ICSE '99*, no. May, pp. 627–628, 1999, doi: 10.1145/302405.302721.

[12] D. Ma and A. Seidmann, "The pricing strategy analysis for the 'software-as-a-service' business model," *Grid Econ. Bus. Model.*, pp. 103–112, 2008.

[13] D. Ma, "The business model of 'Software-as-a-Service,'" *IEEE Int. Conf. Serv. Comput. SCC*, no. July, pp. 701–702, 2007, doi: http://doi.ieeecomputersociety.org/10.1109/SCC.2007.118.

[14] T. Cai and L. Liu, "Integration of B2B E-commerce and ERP in Manufacturing Enterprise and its Application," *Int. Conf. Manag. Educ. Inf. Control*, no. June, pp. 801–806, 2015, doi: 10.2991/meici-15.2015.141.

[15] R. Kazman, C. Nielsen, and C. Nielsen, "Understanding Patterns for System-of- Systems Integration Understanding Patterns for System-of- Systems Integration," *Res. Showcase@CMU*, no. December, 2013.

[16] H. Han *et al.*, "A RESTful Approach to the Management of Cloud Infrastructure," *2009 IEEE Int. Conf. Cloud Comput.*, pp. 139–142, 2009, doi: 10.1109/CLOUD.2009.68.

[17] R. M. Dennis, A., Wixom, B. H., & Roth, *System Analysis and Design*. John wiley & sons., 2008.

[18] J. W. Moore, A. Abran, P. Bourque, R. Dupuis, and L. L. Tripp, *Guide to the Software Engineering Body of Knowledge*, no. 2005921729. The Institute of Electrical and Electronics Engineers Inc, 2004.

[19] H. Hai and S. Sakoda, "SaaS and integration best practices," *Fujitsu Sci. Tech. J.*, vol. 45, no. 3, pp. 257–264, 2009, doi: 10.1080/07421222.2001.11045665.

[20] A. Chen and J. Beatty, *Visual Models for Software Requirements*. Redmond, Washington: Microsoft Press, 2012.

[21] K. Wiegers and J. Beatty, *Software Requirements*, 3rd ed. Redmond, Washington: Microsoft Press, 2013.

[22] A. Gericke, M. Klesse, R. Winter, and F. Wortmann, "Success factors of application integration: an exploratory analysis," *Commun. Assoc. Inf. Syst. Vol.*, vol. 27, no. November, 2010.

[23] N. Erasala, D. C. Yen, and T. M. Rajkumar, "Enterprise application integration in the electronic commerce world," *Comput. Stand. Interfaces*, vol. 25, no. 2, pp. 69–82, 2003, doi: 10.1016/S0920-5489(02)00106-X.