BAB 4 Konsep Bisnis Open Source

Sejarah Free Software

Free Software atau non-proprietary software telah ada sejak penemuan komputer pertama kali sekitar pertengahan tahun 1940, dan dalam beberapa tahun hal ini berlangsung tanpa masalah. Namun pembuatan, penyebaran, dan penggunaan free software ini hanya terbatas pada kalangan tertentu saja seperti engineer, ilmuwan, dan orang-orang tertentu yang memiliki akses komputer yang dianggap sebagai teknologi yang mahal dan langka. Di dalam universitas dan sektor publik (terutama lembaga militer di negara maju) dimana fasilitas komputer berada, pertukaran berbagai perangkat lunak berupa kode-kode program berlangsung bebas dan mudah antar sesama programmer yang dibayar untuk usahanya membuat program bukan untuk kepemilikan programnya.

Beragamnya jenis program yang ada saat itu telah memicu pembuatan sistem operasi (Operating System) untuk menyatukan program yang masih terpisah-pisah tersebut sehingga dapat berjalan dengan mudah dalam satu komputer. Sampai awal tahun 1970, sistem operasi dijalankan dalam komputer mini dengan mainframe tertentu sehingga membuat program menjadi tidak kompatibel, akibatnya program harus ditulis ulang untuk setiap jenis mesin. Inilah awal munculnya proprietary software yang digunakan oleh IBM, Burroughs, Honeywell dan pembuat komputer besar lainnya untuk membantu membedakan jenis program dan menyesuaikannya dengan merk mesin tertentu.

Pada tahun 1970 juga, beberapa programmer dari laboratorium AT&T Bell berhasil membuat sistem operasi UNIX yang ditulis dengan bahasa C dan dapat berjalan pada berbagai merk mesin komputer. Pada tahun 1980, revolusi komputer mencapai puncaknya dengan dibuatnya jenis komputer PC. Penggunaan PC ini meningkat secara eksponensial hingga menjamah sektor bisnis. Seiring dengan ekspansi komputer pada berbagai sektor bisnis ini, programmer tidak hanya dibayar untuk pembuatan program tetapi juga untuk program yang dibuatnya. Dengan demikian, perkembangan perangkat lunak proprietary relatif lebih cepat dibandingkan non-proprietary software.

The Second Era (1965-1975) was also characterized by the use of product software and the advent of “software houses”. Software was developed for widespread distribution in amultidisclipinary market. Programs for mainframes and minicomputers were distributed to

hundreds and sometimes thousands of users. Enterpreneurs from industry, government and academia broke away to develop the ultimate software package and earn a bundle of money. (Pressman, 1997)

Pengembangan propietary software juga dilatarbelakangi munculnya renaissance, revolusi industri, dan kapitalisme yang muncul pada tahun-tahun tersebut. Ideologi modern untuk membuat berbagai produk dan memasarkannya pun juga mulai masuk ke lini software. Sementara itu, free software yang lebih bersifat post-modernisme, muncul secara resmi pada tahun 1984 untuk membendung dan mengimbangi gerakan propietary software. Sekaligus untuk mengembalikan pengembangan software yang bersifat free dan open source pada awal kemunculannya.

Pada tahun 1984, Richard Stallman membuat proyek yang dinamakan GNU (GNU's Not Unix) di laboratorium Artificial Intelligence MIT. GNU ini merupakan sistem operasi yang dibuat untuk "melawan" komersialisasi perangkat lunak yang dilakukan perusahaan pembuat UNIX. Dengan usahanya ini, Stallman menjadi pionir free software melalui projek GNU dan pembentukan Free Software Foundation (FSF). Pada tahun 1990, adanya jaringan internet menstimulasi perkembangan free software dengan terbentuknya komunitas free software di seluruh dunia yang tidak hanya tertarik dengan sistem operasi tetapi juga dalam pengembangan aplikasinya. Pada tahun 1994, GNU menjadi sistem operasi sempurna dengan kontribusi kernel Linux yang dirilis oleh Linus Torvalds. GNU/Linux menjadi sistem operasi alternatif selain UNIX.

Saat ini, FOSS (Free or Open Source Software) tumbuh pesat dengan berbagai sistem dan aplikasinya menjadi solusi alternatif dari pemakaian proprietary software. Penggunaan perangkat lunak open source menjadi pilihan utama dibeberapa negara dengan berbagai kelebihan yang dimiliki OSS seperti sekuritas, reliabilitas sistem, dan kelebihan lainnya. Dengan sifatnya yang "open" memberikan keuntungan lebih bagi pengguna OSS yang juga seorang pengembang (developer) perangkat lunak, karena pengembang dapat memodifikasi program yang tersedia sesuai dengan keinginannya. Pengguna atau konsumer yang juga seorang pengembang atau produsen disebut sebagai "prosumer". Selain itu, secara ekonomis, pengguna tidak perlu mengeluarkan biaya untuk OSS ini dibandingkan dengan penggunaan proprietary software.

Pelajaran dari Fenomena Open Source Bagi Dunia Bisnis

Saat ini, berbagai perusahaan telah memberikan perhatian lebih banyak kepada sistem open source. Sepuluh tahun yang lalu seolah terlihat bahaya Microsoft akan memonopoli pasar server. Akan tetapi, patut disyukuri bahwa open source telah mencegah terjadinya hal tersebut. Survey terbaru menunjukkan bahwa 52% perusahaan sedang mengganti server Windows mereka dengan Linux server.

Lebih penting lagi untuk diperhatikan 52% yang manakah mereka? Saat ini, seseorang yang berencana untuk menggunakan server Windows harus memberikan penjelasan apa yang mereka lebih ketahui daripada Google, Yahoo dan Amazon tentang server. Sebab, Google, Yahoo, dan Amazon menggunakan Linux sebagai mesin server mereka.

Namun hal terbesar yang harus dipelajari dunia bisnis dari dunia open source bukanlah mengenai Linux atau Firefox, namun tentang kekuatan yang ada dibalik lahir dan berkembangnya Linux dan Firefox sebagai produk open source. Kekuatan tersebut lebih berpengaruh dibandingkan aplikasi apa yang Anda gunakan.

Kita dapat menemukan kejelasan mengenai kekuatan tersebut dengan membandingkan open source dan blogging. Blog berasal dari kata web log yang berarti catatan di web. Blog biasanya berupa catatan pribadi seseorang mengenai hal apapun yang diletakkan di web, baik melalui layanan khusus blog maupun dalam bentuk situs pribadi biasa. Sebagaimana yang telah Anda duga, keduanya meiliki banyak kemiripan.

Sebagaimana open source, blogging adalah sesuatu yang dilakukan orang untuk dirinya sendiri, tanpa biaya, dan dengan senang hati. Sebagaimana hacker (dalam pengertian sebenarnya, bukan dalam pengertian cracker) open source, para blogger kerap bersaing dengan mereka yang bekerja untuk uang, dan mereka seringkali menang. Metode yang digunakan untuk mengukur kualitas pun sama: teori Darwin tentang seleksi (*survival of the fittest*). Penggunaan teori ini tidak membenarkan bahwa manusia berasal dari monyet. Berbagai perusahaan memastikan kualitas melalui seperangkat aturan yang mencegah para pegawai untuk berbuat seenaknya. Akan tetapi, hal tersebut tidak diperlukan jika mereka dan pembaca/audiens dapat berkomunikasi satu sama lain. Mereka hanya perlu membuat apa yang mereka mau, yang memiliki kualitas baik akan tersebar, dan yang buruk akan diabaikan. Dan dalam kedua kasus, masukan dari audiens akan meningkatkan kualitas terbaik.

Kesamaan lain yang ada adalah keduanya memanfaatkan web. Banyak orang ingin melalukan hal besar tanpa memerlukan biaya, namun, sebelum adanya web, sulit sekali untuk menyapa audiens atau berkolaborasi dalam suatu proyek open source.

Amatir versus Profesional

Hal terpenting yang harus dipelajari oleh dunia bisnis adalah fakta bahwa seseorang bekerja lebih keras ketika mengerjakan sesuatu yang mereka senangi. Fakta ini memang bukan berita baru. Namun, bagaimana dunia bisnis dapat disimpulkan harus mempelajarinya? Meeka bukan tidak mengetahui hal ini, namun sistem bisnis mereka yang menunjukkannya.

Dunia bisnis masih mengacu pada model yang lebih kuno, yang dapat disederhanakan dengan kata Perancis untuk bekerja: *travailler*. Kata ini memiliki padanan dalam bahasa Inggris, *travail*, yang berarti *torture* (menyiksa, menyakitkan).

Meski demikian, kata untuk bekerja yang sekarang dipakai adalah *work*. Ketika masyarakat semakin kaya, mereka mempelajari sesuatu berkaitan dengan bekerja, yang sangat mirip dengan apa yang mereka pelajari tentang diet. Saat ini, kita tahu bahwa makanan yang paling sehat adalah apa yang biasa dimakan oleh leluhur petani kita karena mereka miskin. Mereka hanya memakan apa yang mereka tanam. Bahkan, mereka yang tinggal di desa masih ada yang seperti itu sampai sekarang dan lebih sehat daripada kita yang tinggal di kota. Sebagaimana makanan yang kaya gizi, waktu senggang hanya terasa dibutuhkan ketika kita tidak mendapatkannya seecara cukup. Kita memang didesain untuk bekerja, sebagaimana kita telah didesain untuk memakan serat dengan jumlah tertentu. Tubuh kita akan menjadi kurang sehat bila kita kurang mengkonsumsi serat.

Ada sebuah kata yang digunakan untuk menggambarkan seseorang yang mengerjakan sesuatu karena mencintai apa yang dikerjakannya, yaitu: *amateur*. Kata ini sekarang memiliki konotasi yang buruk karena kita melupakan akar katanya. Kata amatir pada awalnya lebih merupakan kata pujian. Sayangnya dunia di abad ini lebih mementingkan profesionalisme, di mana amatir ini tidak profesional, secara definisi.

Oleh karena itu dunia bisnis sangat terkejut dengan sebuah pelajaran dari dunia open source: bahwa orang yang bekerja karena mencintai pekerjaannya seringkali lebih bagus hasil kerjanya dibandingkan dengan orang yang bekerja dengan niat mencari uang. Para pengguna tidak beralih dari Internet Explorer ke Firefox karena mereka ingin mengutak-atik kode programnya. Namun, lebih karena Firefox memang browser yang lebih baik.

Microsoft bukannya tidak berusaha. Mereka tahu bahwa menguasai pasar browser akan mengukuhkan monopoli mereka. Hanya saja, mereka tidak cukup mampu membayar orang yang akan bekerja lebih baik dibandingkan sekelompok *hacker* terinspirasi, yang akan mengembangkan browser tanpa minta dibayar.

Profesionalisme pun tampaknya hampir selalu dipandang secara berlebihan (*overesimate*), tidak hanya karena pengertian definitifnya bekerja untuk uang, namun juga beberapa konotasi seperti formalitas dan dikesampingkannya emosi. Sebagaimana terlihat pada tahun 1970, profesionalime lebih merupakan dorongan *fashion* yang muncul pada era industrialisasi saat itu.

Salah satu hal yang paling berpengaruh saat itu adalah adanya jalur distribusi/komunikasi yang diistilahkan sebagai *channel*. Istilah ini digunakan baik dalam konteks penyebaran produk/barang maupun penyebaran informasi. Ada jalur distribusi dan ada saluran televisi dan radio.

Sempit dan terbatasnya suatu jalur menyebabkan seorang profesional tampak begitu unggul dibandingkan para amatir. Misalnya, hanya ada sedikit lowongan pekerjaan sebagai jurnalis/wartawan profesional. Sehingga persaingan yang ada menyebabkan rata-rata wartawan dianggap cukup baik kualitasnya. Sementara pada sisi amatir, setiap orang bebas mengekspresikan pendapatnya mengenai suatu kejadian di mana saja. Akan tetapi, ungkapan pendapat seseorang yang sedang ngobrol di warung kopi dianggap seperti kata-kata orang bodoh; bila dibandingkan dengan seorang wartawan yang menulis tentang masalah yang sama.

Di internet/web, penghalang untuk mengemukakan ide jauh lebih rendah. Anda bebas berbicara mengenai apa saja tanpa harus membeli kopi, bahkan anak kecil pun bisa menulis di web. Jutaan orang menulis di web. Dan seperti yang kita duga, sebagian besar tidak begitu bagus. Hal ini menyebabkan beberapa media mengganggap blog tidak menimbulkan ancaman apapun melainkan hanya pekerjaan orang iseng demi kesenangan sendiri.

Kesalahan yang sebenarnya adalah pemaknaan blog oleh sebagian besar media. Blogger tidak mereka artikan sebagai seseorang yang menulis di web dalam format weblog, melainkan siapa saja yang mempublikasikan tulisan secara online. Hal ini akan menjadi masalah karena web akan menjadi media standar untuk publikasi. Bagaimana bila kata *blogger* kita ganti dengan *writer*?

Ada satu poin penting yang dikesampingkan oleh pemilik media cetak, yaitu kenyataan bahwa blog pada umumnya tidak ada pembacanya. Pada sistem lama (sistem *channel*), ini merupakan sesuatu dengan kualitas standar. Karena anda suka atau tidak, hanya itulah yang akan anda dapatkan. Namun saat ini, Anda dapat membaca tulisan dari penulis mana saja yang anda suka. Jadi, media cetak tidak bersaing dengan blog rata-rata, melainkan bersaing dengan blog terbaik. Dan sebagaimana Microsoft, mereka kalah.

Saya mengetahuinya dari pengalaman saya sendiri sebagai pembaca. Saya lebih suka membaca review dari seorang penulis yang saya kenal kredibilitasnya dibandingkan membacanya dari situs berita. Atau saya akan membaca beberapa review untuk mengimbangi satu berita yang saya baca. Misalnya, ketika ada berita peluncuran OpenOfice 3.0, meskipun disinggung juga mengenai fitur-fitur barunya, membaca review seorang “online writer” mengenai hal ini akan lebih menarik.

Meskipun sebagian besar media cetak saat ini memiliki situs web, namun seringkali saya tidak menemukan artikel yang ingin saya baca dari halaman depan situs webnya. Melainkan menggunakan situs pencari semacam Google, atau memanfaatkan *aggregator* seperti Google News, Slashdot atau Delicious. Situs Kompas, misalnya, adalah kumpulan artikel dari orang-orang yang bekerja untuk Kompas. Sedangkan Delicious berisi kumpulan artikel yang menarik.

Selain itu sebagian besar artikel di media cetak membosankan. Tidak ada hal baru yang dapat kita pelajari. Yang ada hanyalah data-data yang disusun agar enak dibaca. Tidak ada hal yang baru dalam berita buruk; selain nama dan tempat terjadinya peristiwa. Misalnya, seorang anak diculik, seorang gadis diperkosa, kecelakaan kereta, dan sebagainya.

Sebagaimana di dunia software (perangkat lunak), ketika para profesional menghasilkan sesuatu, tidak mengejutkan bila para amatir dapat melakukannya dengan lebih baik. Karena mereka hidup bergantung pada chanel, mereka juga dapat mati karena channel. Bila bergantung pada sistem oligopoli, anda akan tenggelam dalam kebiasaan buruk yang melenakan, sehingga akan sulit menghadapi persaingan yang muncul tiba-tiba. Penulis novel yang bagus pun, biasanya tidak menulis untuk mencari uang; tapi menulis dari hati.

Kantor sebagai Tempat Bekerja

Kemiripan lain dari blog dan software open source adalah sebagian besar dari blog dan software open source dibuat oleh mereka yang bekerja di rumah. Hal ini barangkali tidak terlihat mengejutkan. Namun, seharusnya ini merupakan sesuatu yang tidak lazim. Analogi yang sama adalah bahwa pesawat yang dihasilkan di rumah dapat menembak sebuah F-18. Berbagai perusahaan dan organisasi membangun gedung dengan biaya yang tidak sedikit dengan satu tujuan: untuk menyediakan tempat untuk bekerja. Da orang yang bekerja di rumah, yang tidak didesain untuk bekerja, ternyata malah lebih produktif.

Hal ini membuktikan sesuatu yang sudah kita duga. Kantor pada umumnya merupakan tempat yang menyedihkan untuk menyelesaikan pekerjaan. Dan sebagian besar penyebabnya adalah berbagai kriteria yang berhubungan dengan profesionalisme. Kantor yang bersih diharapkan mendorong efisiensi. Namun, mendorong efisiensi tidaklah sama dengan benar-benar bekerja secara efisien.

Atmosfir produktivitas yang ingin ditanamankan di kantor mirip dengan nyala api yang digambar di dinding luar mobil untuk menunjukkan kecepatan. Kenyataan yang dilakukan banyak pegawai justru lebi buruk.

Banyak hal berbeda diterapkan pada sistem *startup*. Biasanya sistem ini dimulai di sebuah apartemen/rumah. Calon *startup* dapat memilih dan membeli kursi dan meja yang mereka suka. Mereka dapat bekerja pada jam yang tidak lazim dan mengenakan pakaian yang nyaman dan kasual sekali bagi mereka. Tanpa harus mengenakan kemeja atau dasi. Mereka dapat melihat apapun di internet tanpa diganggu oleh peringatan ”anda mengakses sesuatu yang tidak berhubungan dengan pekerjaan”. Pembicaraan dengan gaya formal dapat diganti dengan humor yang lebih santai. Sebuah perusahaan atau organisasi pada tahap ini barangkali menjadi kondisi yang paling produktif.

Barangkali ini bukan merupakan kebetulan. Namun ini juga dapat berarti bahwa beberapa aspek profesionalisme memang seuatu yang tidak menguntungkan.

Hal yang menyumbangkan pada kerusakan moral paling buruk pada sistem kantor tradisional adalah bahwa karyawan harus berada di kantor pada jam yang sudah ditentukan. Biasanya ada beberapa orang yang memang harus berada di kantor, namun alasan terbesar bahwa sebagian besar karyawan harus bekerja pada jam tertentu adalah karena perusahaan tidak mampu mengukur produktivitas karyawannya.

Ide pokok di balik adanya jam kantor adalah jika kita tidak bisa membuat orang bekerja, setidaknya kita mencegah mereka untuk bersenang-senang. Jika mereka harus berada di dalam kantor dari jam 8 pagi hingga jam 4 sore mialnya, dan mereka tidak boleh melakukan kegiatan selain bekerja, maka dapat dipastikan meerka akan bekerja. Demikian teorinya. Kenyataannya banyak dari mereka justru tidak bekerja dan tidak juga bersenang-senang.

Jika Anda dapat mengukur kinerja seseorang, banyak perusahaan tidak perlu memiliki jam kerja yang tetap. Anda hanya perlu mengatakan: ini yang harus anda kerjakan. Kerjakanlah kapan pun anda mau, dan di mana saja anda suka. Jika pekerjaan anda memerlukan pertemuan dengan seseorang, maka anda perlu berada di sini sesuai waktu yang diperlukan. Sisanya terserah anda.

Hal ini terlihat seperti tidak mungkin alias mustahil. Namun inilah yang disampaikan Paul Graham kepada orang yang akan bekerja di perusahaannya. Tidak ada patokan mengenai jam kerja. Paul Graham sendiri tidak pernah muncul di kantor sebelum jam 11. Namun ini bukan berarti bahwa kita memberi kelonggaran atau berbaik hati kepada karyawan. Inti sebenarnya dari perkataan tersebut adalah: jika anda bekerja di sini, kami berharap anda menyeleaikan banyak hal. Jangan mengelabui kami hanya dengan seringnya anda hadir di sini.

Permasalahan lain yang muncul adalah seseorang yang berpura-pura sedang bekerja akan mengganggu mereka yang sedang benar-benar bekerja. Diyakini, hal inilah yang menyebabkan sebuah organisasi besar memerlukan banyak sekali rapat. Sebuah organisasi besar biasanya memiliki tingkat pencapaian per kapita yang rendah. Dan mereka harus tetap berada di tempat selama delapan jam setiap hari. Ketika banyak waktu dihabiskan untuk pencapaian yang rendah, diperlukan sesuatu untuk mengatasinya. Dan rapat biasanya menjadi mekanisme yang dilakukan untuk menutupi kekurangannya.

Paul Graham misalnya, sempat bekerja selama setahun dengan model kantor tradisional dari jam 9 pagi hingga jam 5 sore. Dia masih ingat dengan baik perasaan nyaman yang aneh setiap kali diadakan rapat. Ia merasa sangat nyaman dan waspada, bahwa ia dibayar untuk membuat program (programming). Ia membayangkannya seolah ada mesin di mejanya yang mengeluarkan uang setiap menit entah apapun yang sedang dia kerjakan. Termasuk ketika dia sedang berada di kamar mandi. Namun karena mesin bayangan ini selalu ada, ia merasa harus selalu bekerja setiap saat. Dan rapat merupakan hal yang sangat menyenangkan dan melegakan. Sebab, setiap menit dalam rapat dihitung sebagai bekerja, seperti saat dia sedang menulis program. Tetapi rapat begitu mudah dilakukan. Cukup duduk manis dan terlihat memperhatikan apa yang sedang dibahas.

Rapat dan pertemuan-pertemuan mirip dengan candu ditambah dengan efek relasional. Dan selain menambah biaya langsung dari waktu yang digunakan untuk pertemuan, ada biaya untuk pembagian waktu seseorang menjadi potongan kecil yang terlalu kecil untuk dapat berguna.

Anda dapat melihat ketergantungan pada hal ini dengan menghilangkannya seketika. Misalnya, bagi perusahaan besar, dapat dilakukan percobaan sebagai berikut. Tentukanlah sebuah hari yang disebut sebagai hari bekerja. Pada hari itu, setiap orang diharuskan bekerja selama delapan jam, tanpa melakukan komunikasi dengan orang lain. Artinya mereka harus mencari

pekerjaan yang menghabiskan waktu delapan jam dan dapat mereka kerjakan sendiri tanpa istirahat.

Masalah lainnya adalah seringkali mereka yang pura-pura bekerja terlihat lebih baik dibanding mereka yang benar-benar bekerja. Ketika menulis program, Paul Graham menghabiskan waktu untuk berpiir saja hampir sama dengan waktu yang dihabiskannya untuk menulis program di komputer. Sisanya ia habiskan dengan duduk minum the atau berjalan-jalan di sekitar ruang kerjanya. Dan inilah waktu yang kritis, ketika ide-ide bermunculan. Dan ia masih merasa bersalah ketika melakukannya di kebanyakan kantor, di mana sebagian besar pegawai yang lain terlihat sibuk.

Adalah hal yang sulit untuk menunjukkan kesalahan yang sudah biasa dilakukan sampai ada contoh lain yang dapat diperbandingkan. Dan itulah satu alasan mengapa open source dan blogging dalam beberapa hal menjadi sangat penting. Mereka menunjukkan kepada kita bagaimana bekerja yang sebenarnya.

Saat ini, Paul Graham sedang mendanai delapan *startup*. Seorang temannya terkejut ketika mengetahui bahwa mereka disuruh tinggal di apartemen mana pun sebagai kantor. Dan ia tidak melakukannya untuk menghemat biaya. Akan tetapi, ia melakukannya agar mereka membuat software yang berkualitas. Bekerja dalam lingkungan yang informal akan membuat mereka mengerjakannya dengan benar tanpa menyadarinya. Ketika anda memasuki sebuah kantor, maka pekerjaan dan kehidupan pun terpisah.

Salah satu kunci profesionalisme adalah terpisahnya pekerjaan dari kehidupan pribadi. Dan ini saya yakini sebagai sesuatu yang salah.

Bottom-Up

Pelajaran besar yang ketiga dari open source dan blogging adalah bahwa ide dapat muncul dari bawah ke atas; tidak melulu dari atas ke bawah. Orang-orang membuat sesuatu yang mereka inginkan, dan hasil yang berkualitas akan tetap eksis.

Anda tidak salah bial ini terdengar familiar. Ini adalah prinsip ekonomi pasar. Ironisnya, meskipun open source dan blog dikerjakan secara *free*, justru terlihat lebih mirip dengan ekonomi pasar. Sementara sebagian besar perusahaan, dengan berbagai pernyataan mengenai nilai dari pasar bebas, malah berjalan seperti negara komunis secara internal.

Ada dua wewenang yang bersama-sama berperan dalam perancangan: apa yang harus dilakukan selanjutnya, dan menjaga kualitas. Pada sistem channel tradisional, keduanya mengalir dari atas ke bawah. Misalnya, seorang editor sebuah surat kabar akan meugaskan reporter untuk meliput dan menulis suatu hal lalu mengedit hasil tulisan reporternya.

Oepn souce dan blogging menunjukkan bahwa berbagai hal tidak harus berjalan dengan cara itu. Ide-ide dan bahkan kontrol kualitas pun dapat mengalir dari bawah ke atas. Dari bawahan ke pimpinan. Dan dalam kedua kasus tersebut, hasilnya tidak hanya dapat diterima, namun malah lebih baik. Contohnya adalah software open source yang lebih reliabel karena sifatnya yang open source; setiap orang dapat menemukan kesalahan dalam kode pogram.

Hal yang sama berlaku pula dalam dunia tulisan. Ketika sebuah esai yang dipajang di web telah dibaca oleh ribuan orang, penulisnya bisa merasa percaya diri. Jika belum ada yang membaca,

rasanya seperti merilis program tanpa melakukan pengujian. Selain itu, menulis secara online dapat dilakukan kapan saja dan tentang apa saja. Sedangkan jika kita mengirim artikel ke suatu media cetak, tulisan itu tidak akan dibaca orang selama beberapa bulan.

Banyak karyawan ingin melakukan sesuatu yang hebat bagi perusahaannya, namun lebih sering pihak manajemen tidak mengizinkannya. Berapa kali Anda mendengar mengenai kasus seperti ini? Kasus yang paling populer barangkali adalah Steve Wozniak. Saat itu, ia masih menjadi karyawan Hewlet-Packard (HP) dan mengajukan diri untuk membangun mikrokomputer. Namun, idenya ditolak oleh perusahaannya. Akhirnya ia pun keluar dari HP dan menjadi arsitek komputer Apple Macintosh bersama Steve Jobs. Saat ini, komputer dan sistem operasi Mac adalah sesuatu yang eksklusif dan "wah". Bahkan, ada yang menyatakan bahwa bila seseorang pernah mencoba Mac, dia pasti ingin memilikinya. Atau setidaknya, bila tidak terjangkau, ia akan berangan-angan untuk memilikinya suatu saat nanti.

Hal seperti ini sering terjadi. Akan tetapi, biasanya kita tidak mendengar beritanya. Sebab, untuk membuktikan kebenaran pernyataannya, kadang-kadang seseorang harus keluar dan mendirikan perusahaannya sendiri, seperti yang dilakukan Wozniak.

Startups

Seperti apakah wujud bisnis di masa depan ketika bisnis telah menyerap pelajaran dari open source dan blogging? Hambatan terbesar yang mencegah kita melihat masa depan bisnis adalah asumsi bahwa orang yang bekerja untuk anda harus menjadi pegawai atau karyawan anda. Tapi pikirkan apa yang terjadi sebenarnya: perusahaan mempunyai sejumlah uang, lalu membayarkannya kepada para karyawan dengan harapan para karyawan akan membuat sesuatu yang bernilai lebih besar daripada bayaran yang mereka terima. Padahal ada jalan lain untuk mengatur hubungan tersebut. Sebagai ganti membayar karyawan dengan gaji, mengapa tidak anda bayar dengan investasi? Kemudian, sebagai ganti atas kehadiran mereka di kantor anda untuk mengerjakan proyek milik anda, mereka dapat bekerja pada proyek-proyek mereka sendiri di mana saja mereka mau.

Ada beberapa permasalahan dalam hubungan karyawan – atasan. Misalnya, sebagai karyawan, saya akan bekerja sebaik-baiknya sesuai permintaan pelanggan. Namun saya tidak suka diberitahu mengenai apa yang harus saya kerjakan. Menjadi atasan juga terkadang bisa membuat anda frustrasi. Seringkali lebih mudah mengerjakan sendiri segalanya daripada meminta orang lain mengerjakannya.

Contoh paling jelas bahwa sistem karyawan atau pegawai bukanlah hubungan ekonomis murni yaitu kasus ketika sebuah perusahaan dituntut di pengadilan karena memecat karyawannya. Dalam suatu hubungan ekonomis murni, anda bebas melakukan apa saja. Jika anda ingin berhenti membeli kayu dari *supplier* A dan beralih membelinya ke *supplier* B, anda tidak perlu menjelaskan alasannya. Tidak ada yang bisa menyalahkan anda karena tidak adil kepada *supplier*. Keadilan (secara hukum) menunjukkan akibat dari suatu ikatan paternal bahwa tidak ada transaksi atau hubungan yang setara di sana.

Sebagian besar aturan hukum bagi pengusaha berisi atau bertujuan untuk melindungi hak-hak karyawannya. Tapi jangan mengharapkan sesuatu tanpa imbal balik yang serupa. Jadi jangan berharap pengusaha atau pemilik perusahaan tempat anda bekerja mempunyai semacam tanggungjawab paternal (layaknya seorang ayah), kecuali anda sebagai karyawan akan dianggap sebagai anak-anaknya.

Ada hubungan yang lebih baik, yaitu *investor-founder*. Paul Graham bisa melihat kerugian pada hubungan pegawai-pengusaha karena sudah berpengalaman sebagai *investor* maupun *founder*. Keuntungan terbesar dari sistem *investor-founder* dibandingkan dengan sistem pengusaha-pegawai adalah bahwa orang yang bekerja untuk proyeknya sendiri akan bekerja jauh lebih produktif. Seperti halnya pada sistem open source dan blogging. Startup adalah sebuah proyek pribadi dalam dua aspek yang sama pentingnya, yaitu secara kreatif dan secara ekonomi.

Google adalah sebuah contoh langka mengenai perusahaan besar yang menerapkan konsep yang sudah dibahas sebelumnya. Mereka berusaha keras membuat kantor-kantor mereka kurang steril. Mereka juga menghargai karyawan yang bekerja dengan sangat baik. Yaitu dengan menghadiahkan sejumlah besar saham perusahaan. Mereka bahkan membebaskan hacker menghabiskan 20% waktunya untuk proyek pribadi masing-masing.

Mengapa orang-orang tidak sekalian saja diminta menghabiskan 100% waktunya untuk proyek mereka sendiri. Kemudian, daripada memperkirakan nilai dari apa yang mereka buat, berikan nilai saham yang ada di pasar saat ini. Hal ini tidaklah mustahil. Sebab, hal seperti inilah yang dilakukan oleh *venture capitalist*. Mereka menanamkan saham pada proyek-proyek yang bernilai.

Saat ini, orang yang paling pintar di bangku kuliah pun seringkali hanya berpikir untuk mencari pekerjaan setelah lulus. Padahal tidak harus demikian. Sebenarnya yang perlu dia lakukan hanyalah menciptakan sesuatu yang bernilai. Maka uang akan datang dengan sendirinya. Pekerjaan (menjadi pegawai) adalah salah satu cara untuk melakukannya. Namun bila dibandingkan, anda akan mendapat uang lebih banyak bila bekerja untuk investor daripada bekerja untuk pengusaha.

Para hacker cenderung berpikir bahwa bisnis hanyalah untuk para lulusan MBA (Master of Business Administration). Tetapi dalam startup, bukan anda tidak akan melakukan administrasi bisnis. Yang perlu dilakukan adalah membuat atau menciptakan bisnis. Tahap pertama dalam menciptakan bisnis adalah membuat produk, yaitu aktivitas hacking. Inilah bagian yang berat. Jauh lebih berat membuat sesuatu yang disukai (atau bahkan dicintai) orang daripada mengambil sesuatu yang sudah disukai orang dan memikirkan bagaimana menghasilkan uang darinya.

Sebab lain yang membuat orang ragu untuk memulai startup adalah resikonya. Memang tidak semua bisnis akan langsung sukses. Ada juga yang mengalami kegagalan pada awalnya. Namun, sebagaimana open source dan blogging, anda akan lebih senang mengerjakannya, meski anda gagal. Karena anda akan mengerjakan sesuatu milik anda sendiri, bukan mengerjakan apa yang diperintahkan oleh orang lain.

Semoga dalam jangka panjang, semua orang berpikir untuk menuju ke sana. Yaitu menerapkan efek terbesar dari kekuatan yang ada pada open source dan blogging. Lalu menggantikan sistem pengusaha-pegawai dengan sistem yang lebih murni secara ekonomi, investor-founder. Sistem investor-founder adalah sistem ekonomi murni. Mengapa? Karena kedua belah pihak berada dalam posisi yang setara. Jika semua pemuda di Indonesia berpikir untuk menciptakan pekerjaan dan bukan mencari pekerjaan, maka tidak akan ada pengangguran di negeri yang langganan juara olimpiade sains ini.

Apa itu Bisnis

Entitas bisnis dapat diartikan sebagai suatu perusahaan atau organisasi yang melakukan kegiatan-kegiatan ekonomi untuk mengumpulkan keuntungan berupa laba. Keuntungan tersebut dibukukan dari besarnya seluruh pendapatan dikurangi dengan besarnya seluruh pengeluaran. Kegiatan ekonomi adalah kegiatan yang dilakukan untuk memenuhi kebutuhan. Dengan demikian banyak sekali bidang bisnis yang ada, sejalan dengan makin banyaknya jenis kebutuhan manusia yang harus dipenuhi.

Prinsip ekonomi yang biasanya berusaha dicapai adalah “dengan modal atau pengeluaran sekecil mungkin harus diperoleh keuntungan sebesar-besarnya”. Maka, untuk mencapai hal tersebut, suatu perusahaan harus melakukan efisiensi dan peningkatan kualitas kinerja.

Setiap bisnis selalu berkaitan dengan pemasaran atau *marketing*. Pemasaran adalah proses perencanaan suatu produk atau jasa, lalu menentukan harga, mempromosikan, dan mendistribusikannya kepada para pelanggan. Pada perusahaan besar, fungsi marketing mendahului pembuatan suatu produk. Termasuk di dalamnya adalah penelitian pasar (*market research*) dan perancangan, pengembangan, dan uji coba produk.

Pemasaran berkonsentrasi pada konsumen. Mereka mencari tahu kebutuhan dan keinginan para pelanggannya terlebih dahulu. Setelah itu mereka mengembangkan strategi untuk mendidik para pelanggan mengenai fitur-fitur utama suatu produk, mempengaruhi mereka untuk membelinya, lalu meningkatkan kepuasan pelanggan dengan layanan purna jual atau dukungan teknis pasca pembelian.

Bisnis Open Source: Business Enabler versus Business Differentiator

Ketika berbicara mengenai bisnis open source, ada dua jenis penggunaan open source dalam dunia bisnis. Pertama, adalah sebagai teknologi yang memungkinkan bisnis dapat berjalan. Kedua, menggunakan open source sebagai alat untuk membuat bisnis kita lebih menarik bagi pelanggan dibandingkan pesaing kita.

Akibat Ekonomis dari Software

Sejak Microsoft menjadi contoh yang dominan ketika orang membicarakan industri software, mari kita bertanya: apakah akibat ekonomis terbesar dari kesuksesan Microsoft?

Apakah akibat ekonomis terbesar Microsoft adalah kekayaan Bill Gates? Tentu saja tidak. Banyak orang iri dengan kekayaan Bill Gates, namun kekayaan Bill hanya sebagian dari keseluruhan kue yang dibagi-bagikan di industri softwarenya.

Apakah jawabannya adalah fakta bahwa Microsoft telah mengumpulkan 40 milyar dolar di rekeningnya yang terus bertambah, sebagai perusahaan software terbesar di dunia, dengan 55 ribu karyawan di 85 negara (termasuk Indonesia)? Tidak juga.

Akibat ekonomis terbesarnya adalah kenyataan bahwa Microsoft memungkinkan berjalannya banyak bisnis yang luar biasa. Pelanggan atau pengguna produk Microsoft menjalankan

bisnisnya dengan lebih efisien. Bahkan ada yang ada yang tidak dapat menjalankan bisnisnya sama sekali tanpa adanya software dari Microsoft. Inilah akibat ekonomis terbesarnya.

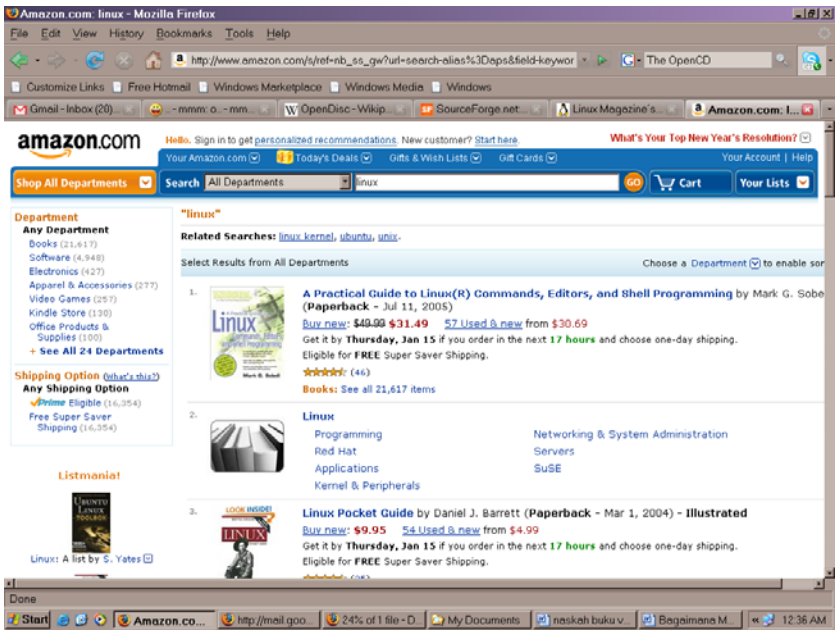
Microsoft hanyalah pembuat alat (*tools*). Akibat ekonomis bagi pembuat alat jauh lebih kecil dibandingkan dengan akibat ekonomis dari berbagai bisnis yang menggunakan alat yang dibuat oleh Microsoft. Akibat ekonomi sekunder yang muncul dari individu dan dunia bisnis yang menggunakan alat tersebut lebih besar daripada akibat primer dari uang yang dibayarkan untuk menggunakan teknologi Microsoft.

Hal yang sama berlaku juga pada sistem open source. Software open source hanyalah alat. Nah, jika ada alat yang lebih baik dalam menjalankan bisnis, mengapa belum banyak orang yang meliriknya?

Penting untuk diingat bahwa sebagian besar perusahaan tidak berbisnis dalam pembuatan software. Namun sebagain besar hampir pasti menggunakan komputer dalam menjalankan bisnisnya. Banyak perusahaan menjual berbagai macam barang di atas bumi ini: buku, baju, sepatu, makanan, maianan, dan sebagainya. Walau demikian, perusahaan kecil pun menggunakan software untuk menjalankan bisnisnya dalam batas tertentu. Bayangkan perencanaan keuangan sebelum adanya software *spreadsheet* semacam Excel, dan atau korespondensi sebelum adanya email. Saat ini kita membutuhkan software untuk menjalankan bisnis. Kebutuhan tersebut sudah sangat krusial sehingga sebuah bisnis dengan 50 karyawan atau lebih sudah perlu mempekerjakan seorang pogrammer, desainer web, atau seorang administrator sistem dengan kemampuan script programming. Bagi bisnis-bisnis tersebut, software digubakan sebagai teknologi untuk menjalankan bisnis, bukan sebagai produk yang mereka jual.

Business Enabler versus Business Differentiation

Teknologi yang memungkinkan bisnis anda berjalan adalah hal yang sangat penting, namun teknologi tersebut bukanlah sesuatu yang anda jual. Jika anda menjual buku, maka buku adalah sumber keuntungan anda. Software menjadi biaya yang tidak dapat dihindarkan demi menjalankan bisnis anda. Teknologi informasi menjadi bukanlah sumber keuntungan anda. Anda membutuhkannya sebatas sebatas untuk menjalankan bisnis anda.



Situs Web Amazon

Ada dua macam teknologi yang menjadi sumber biaya dalam memungkinkan bisnis anda berjalan. Yaitu *differentiator* dan *non-differentiator*. Jenis pertama, differentiator, adalah teknologi yang membuat bisnis anda lebih menarik bagi pelanggan anda ketimbang bisnis milik pesaing anda. Misalnya, jika anda berkunjung ke Amazon.com dan mencari sebuah buku, situs Amazon akan memberitahu anda tentang beberapa buku lain yang juga dibeli oleh orang-orang yang tertarik dengan buku yang akan anda beli. Seringkali buku-buku yang disarankan tersebut cukup menarik untuk anda beli juga. Namun jika anda berkunjung ke situs Barnes and Noble, anda tidak akan menemui fitur ini. Wajar saja bila Amazon berhasil menjual lebih banyak buku secara online.

Jadi, bagi Amazon, aplikasi ”rekomendasi” adalah *business differentiator*. Jelas saja, meng-open-source-kan *business differentiator* anda merupakan suatu kesalahan. Sebab, bisnis milik pesaing anda akan menjadi sama menariknya dengan bisnis milik anda. Bisnis anda tidak khas atau eksklusif lagi.

Sebaliknya, bisnis anda tidak akan rugi jika pesaing anda mengetahui setiap *bit* dari cara kerja *software non-differentiator* anda. Bahkan pesaing anda bisa menjadi kawan terbaik anda dalam hal ini. Tentu saja jika kerja sama dilakukan sebatas pada bagian yang tidak membedakan bisnis anda dengan yang lain. Sebab, kebutuhan anda dan kebutuhan pesaing anda adalah sama dalam hal ini. Fakta ini terlihat setiap hari di dunia open source. Misalnya HP dan IBM saling membantu mengembangkan software yang akan membantu penjualan sistem dari kedua vendor. Namun, mereka menjadi pesaing sejati pada level yang lebih tinggi dalam software stack. Yaitu, pada bagian yang memungkinkan perbedaan bisnis mereka secara efektif.

Barangkali 90% dari software yang digunakan dalam bisnis adalah non-differentiator. Sebagian besar merupakan infrastruktur, yaitu basis untuk mengembangkan teknologi differentiator. Termasuk dalam kategori infrastruktur antara lain sistem operasi, web server, database, server

aplikasi Java, middle-ware lainnya, tampilan grafis, dan alat bantu yang umum digunakan seperti web browser, klien email, pengolah lembar kerja, pengolah kata, dan aplikasi presentasi. Software apa saja yang memberikan nilai tambah pada perusahaan non-software dibangun di atas salah satu atau lebih komponen infrastruktur di atas.

Indikator penting untuk mengetahui apakah suatu software merupakan differentiator adalah seberapa besar kemungkinan pesaing anda dapat memperoleh software yang sama. Baik software Microsoft maupun Linux dan Open Source tidak dapat menjadi differentiator bisnis anda dalam jangka waktu yang lama,. Sebab, mereka tersedia bagi siapa saja. Mereka menjadi differentiator satu sama lain. Namun mereka tidak menjadi differentiator bagi bisnis anda. Salah satunya dapat membuat anda menghemat uang atau membuat pekerjaan anda menjadi lebih efisien. Akan tetapi, pada dasarnya mereka tidak akan membuat bisnis anda menjadi lebih atraktif bagi pelanggan anda.

Indikator penting lainnya adalah apakah pelanggan anda bisa melihat efek dari software tersebut. Pelanggan anda tidak peduli mengenai sistem operasi yang anda gunakan, kecuali bila terlalu sering crash. Mereka juga tidak peduli apakah anda menggunakan Microsoft Office atau Open Office. Mungkin anda punya alasan yang bagus dalam hal ini. Namun pelanggan anda tidak dapat melihatnya.

Dengan demikian, untuk membuat bisnis anda lebih menarik bagi pelanggan anda, sebaiknya anda menghabiskan lebih banyak perhatian dan dana anda pada software differentiator. Selain itu, anda juga perlu mengurangi perhatian dan dana anda pada software yang tidak akan membuat bisnis anda terlihat berbeda atau lebih menarik. Open source adalah kunci untuk mengurangi pengeluaran pada software non-differentiator.

Tentu saja anda harus melihat dengan jelas mengenai software apa yang menjadi differentiator dalam bisnis anda dan mana yang tidak. Banyak perusahaan memiliki sindrom ”Not Invented Here”. Mereka tidak percaya pada hasil kerja pihak lain. Sebab mereka tidak percaya bahwa hasil kerja pihak lain bisa sebaik yang mereka buat. ”NIH” adalah penyakit yang mahal. Jika terjadi, karyawan anda bisa jadi melakukan dua kali kerja, karena mereka mengerjakan sesuatu yang sudah tersedia di luar sana. Alih-alih mereka seharusnya dapat menghabiskan lebih banyak waktu untuk mengembangkan software yang dapat membedakan bisnis anda dengan yang lain.

Partisipas di dunia open source menjadikan sumber pengeluaran software anda lebih efisien dan efektif. Sisa dananya bisa menjadi tambahan keuntungan atau digunakan pada bidang lain yang lebih membutuhkannya.

Paradigma Ekonomi dalam Pengembangan Software

Semua bidang bisnis saat ini membutuhkan software sebagai pendukung aktivitasnya; maka software yang dibutuhkan tentu harus dikembangkan. Ada beberapa paradigma pengembangan software yang biasa digunakan untuk memenuhi kebutuhan tersebut, yaitu:

- Retail
- In-House dan Contract
- Kolaborasi Tanpa Lisensi Open Source
- Open Source.

Setiap paradigma tersebut memiliki perbedaan dalam hal:

- Distribusi biaya pengembangan
- Distribusi resiko kegagalan pengembangan
- Efisiensi pembiayan pengembangan software dibandingkan dengan biaya *overhead*
- Tingkat eksklusivitas penggunaan software yang dihasilkan; seberapa besar kemungkinan software yang dihasilkan tidak digunakan oleh pihak lain

Faktor-faktor tersebut menentukan paradigma mana yang paling cocok untuk mengembangkan software yang dibutuhkan. Perlu diingat bahwa sember dana utama dalam pengembangan software adalah para pengguna (*customer*). Mereka dapat mengarahkan dananya untuk menerapkan paradigma ekonomi manapun dalam rangka memenuhi kebutuhan softwarenya.

Paradigma Retail

Paradigma retail merupakan yang paling familiar bagi sebagian besar masyarakat. Dari seluruh software yang dikembangkan di Amerika, kurang dari tiga puluh persennya yang menggunakan paradigma ini. Pengguna akan membeli lisensi pakai dari sebuah software siap pakai yang dikembangkan oleh perusahaan software tertentu. Biasanya biaya pengembangan software ditanggung oleh satu perusahaan saja. Perusahaan pembuat softwarenya akan berusaha mendapatkan kembali modalnya sekaligus mencari laba dengan menjual berbagai produk softwarenya. Maka biaya pengembangan harus tersedia di hap awal pengembangan. Biaya tersebut mencakup seluruh biaya yang diperlukan untuk membangun software sebelum perusahaan dapat mengembalikan modalnya. Resiko kegagalan dalam mengembangkan produk yang menguntungkan juga ditanggung sepenuhnya oleh perusahaan tersebut.

Setelah sekian lama, biaya pengembangannya akan didistribusikan kepada para pengguna, jika produknya sukses di pasaran. Karena paradigma retail ini tidak dapat langsung mendistribusikan biaya dan resiko hingga produk softwarenya matang, biasanya perusahaan menganggap penting untuk memasuki pasar investasi sebagai mekanisme eksternal untuk mendistribusikan biaya dan resiko. Perusahaan memerlukan modal dari luar dalam jangka waktu yang cukup lama, sebab pengembangan software bisa memakan waktu hingga beberapa tahun. Setelah itu masih diperluan waktu untuk mebangun pasar hingga biaya pengembangan dapat diperoleh kembali dan memungkinkan untuk mendapatkan laba. Bursa saham dapat meningkatkan likuiditas investasi dengan mengizinkan investor untuk memperkirakan masa depan potensial perusahaan, ditunjukkan oleh harga saham, dan bukan pendapatan perusahaan saat ini. Lalu perusahaan yang sukses dapat menginvestasikan kembali keuntungan yang diperolehnya untuk pengembangan software berikutnya.

Biaya *overhead* dari paradigma retail tradisional ini sangat tinggi, kurang dari 10% dari uang yang dibayarkan oleh pengguna untuk membeli software yang akhirnya digunakan untuk pemasaran produk, pengembangan software, dan dokumentasi. Microsoft hanya menggunakan 16,8% dari seluruh pendapatannya untuk penelitian dan pengembangan software. Sisanya digunakan untuk hal-hal yang tidak menguntungkan pengguna secara langsung. Misalnya biaya yang sangat tinggi untuk menemukan pengguna produk Microsoft, mulai dari iklan, desain dan pembuatan kemasan yang sangat menarik (yang dilupakan setelah penjualan), pembayaran kepada penjual pengecer yang telah menyediakan ruang untuk memajang poduknya, staff penjualan, dan keuntungan. Angka 16,8% tersebut belum memperhitungkan peningkatan harga di tingkat pedagang besar

dan pedagang pengecer. Sehingga bagian dari harga produk yang mencerminkan pengembangan software akan menjadi kurang dari 10%.

Cara ini dapat pula menjadi tidak efisien dari sisi pembeli, karena dapat ditemukan perbedaaan antara kebutuhan pengguna dan software yang dibeli. Pengguna seringkali membeli software yang setelah dievaluasi lebih dekat tenyata tidak dapat digunakan untuk memenuhi kebutuhan aplikasinya. Seringkali pengguna tidak dapat mengembalikan biaya yang dia keluarkan untuk "shelfware" ini. Ada pula kerugian bila perusaahan pembuat software memutuskan menghentikan pengembangan dan tidak lagi menyediakan dukungan untuk software yang belum menghasilkan uang bagi pengguna atau pembeli software tersebut. Karena tidak tersedianya kode program (*source*) bagi produk yang tidak dilanjutkan lagi, maka software tersebut akan berada dalam keadaan tanpa dukungan. Dan pengguna hanya memiliki sedikit pilihan selain membukukan kegagalan investasinya.

Dengan mengombinasikan faktor-faktor resiko tersebut kita bisa mendapat kesimpulan estimasi konservatif bahwa 50% dari software retail yang dibeli pada akhirnya tidak dipakai, penjualan dikatakan gagal. Jika kita kalikan efisiensi kurang-dari-10% dari biaya yang digunakan untuk pengembangan software dengan tingkat kegagalan 50%; kita dapat menemukan bahwa efisiensi pembiayaan pengembangan software dengan paradigma retail ini kurang dari 5%.

Akibat terpenting dari efisiensi yang sangat rendah ini adalah bahwa paradigma retail ini hanya cocok untuk memproduksi software dengan pasar yang sangat luas. Pasar yang besar dapat menutupi ke-tidak- efisien-an dari paradigma ini, sebab setiap pengguna dapat membayar biaya yang relatif kecil dibandingkan biaya pengembangan software. Meskipun setelah dijumlahkan, mereka masih membayar lebih dari dua puluh kali lipat dari biaya pengembangan software.

Ada banyak produk software penting yang tidak dapat dengan mudah dikerjakan dengan paradigma retail. Sebab produk-produk software tersebut tidak memiliki pasar yang cukup besar untuk mengembalikan biaya pengembangan dan biaya overhead yang besar dari paradigma ini. Selain itu, banyak produk baru yang dapat menghasilkan pasar yang besar namun tidak dilirik oleh pabrik pembuat software. Hanya karena perusahaan dan investor tidak dapat diyakinkan bahwa pasar untuk produk baru tersebut dapat dibangun, atau risiko kegagalan yang terlalu tinggi. Hal ini cenderung menurunkan tingkat inovasi pada pabrik pembuat software yang menggunakan paradigma retail.

Salah satu contoh kegagalan inovasi pada paradigma retail adalah fakta bahwa inovasi terpenting dalam sepuluh tahun terakhir dalam bisnis komputer dunia, yaitu we server dan browser, harus dikembangkan sebagai produk open source yang dibiayai oleh pemerintah federal dalam bentuk riset di laboratorium universitas. Tidak ada perusahaan yang sebenarnya dapat menyelesaikan pekerjaan ini percaya bahwa ini dapat mendatangkan uang. Bahkan, satu-satunya perusahaan yang menginvestasikan dana yang cukup signifikan dalam pengembangan web (Autodesk, berinvestasi pada proyek Xanadunya Ted Nelson) memilih untuk tidak menyelesaikan proyek sebab model pendapatan yang diperkirakan oleh Xanadu untuk web (bayaran dari pembuat konten untuk setiap kata, dengan penelusuran yang kompleks untuk setiap pekerjaan turunan dan referensi) terlalu sulit untuk dibangun. Sementara pengembang web yang sukses, Tim Berners-Lee, tidak melihat kebutuhan untuk mengembangkan model pendapatan kanonik, namun menyerahkannya kepada para pengguna untuk menemukan cara memperoleh pendapatan melalui web.

Dan sebagaimana telah kita diskusikan, bisnis retail harus memperoleh sebanyak mungkin pelanggan untuk mendapatkan keuntungan setinggi mungkin, sehingga software retail umumnya

tersedia bagi siapapun. Sehingga sangat sulit untuk menjadi differentiator bagi bisnis milik pelanggan.

In-House dan Kontrak

Pengembangan secara In-House dilakukan oleh programmer internal perusahaan yang akan menggunakan softwarenya. Sedangkan pengembangan software pada sistem kontrak dilakukan oleh perusahaan lain yang bekerja sementar (disewa) untuk penggunaan eksklusif perusahaan yang menyewanya. Dalam kedua kasus, programmer umumnya dibayar karena menulis program, bukan karena produk softwarenya seperti pada kasus paradigma retail.

Bentuk lain dari pengembangan secara kontrak adalah kustomisasi ekstensif suatu produk dari vendor tertentu. Misalnya, sebuah perusahaan skala menengah sampai besar dapat membeli sebuah lingkungan web server standar lalu membayar vendor pembuat web server tersebut untuk melakukan kustomisasi yang sesuai dengan kebutuhan bisnis perusahaan tersebut.

Pelanggan memiliki kendali yang sangat baik dalam pengembangan secara in-house dan kontrak. Sebab programmer tidak akan dibayar bila tidak mengerjakan apa yang diminta oleh pelanggan. Klien dapat menentukan apakah software yang dihasilkan akan digunakan secara eksklusif atau boleh digunakan oleh pihak lain. Karena klien dapat mengontrol distribusi, paradigma ini sangat baik untuk mengembangkan sisi pembeda dari software. Bahkan, mungkin hanya inilah paradigma yang dapat benar-benar diterapkan untuk mengembangkan software yang menjadi pembeda bagi bisnis non-software milik klien.

Secara umum, seorang kontraktor bisa saja mencoba menggunakan sebagian dari hasil kerja untuk seorang klien untuk mempermudah bisnisnya dengan klien lain dengan menjual kembali hasil kerja yang sudah dibayar. Seberapa besar kesuksesan yang dilakukan kontraktor dengan cara ini bergantung pada kesepakatan kontrak dan kejujuran kontraktor. Jika kontraktor anda yakin bahwa ia dapat menjual kembali hasil kerja yang dia lakukan untuk anda, ia dapat meminta bayaran yang lebih rendah dan anda dapat memperoleh keuntungan dari distribusi biaya dan resiko kepada kontraktor dan klien berikutnya. Meskipun demikian, paradigma ini merupakan yang skenario kurang tepat untuk pengembangan software yang akan menjadi pembeda bagi bisnis anda, sebab bila kontraktor menjual kembali sesuatu yang menjadi pembeda dalam bisnis anda, maka bisnis anda dapat berada dalam bahaya dalam waktu singkat.

Sebagai ganti atas kendali menyeluruh pada proses pengembangan software secara in-house atau kontrak, pada umumya pelanggan menanggung seluruh biaya dan resiko pengembangan. Karena biaya dan resiko tersebut, in-house dan kontrak tidak dapat menjadi metode yang paling efektif dari segi biaya untuk pengembangan software yang tidak menjadi pembeda bagi bisnis klien. Jika pelanggan membayar keseluruhan biaya untuk mengembangkan sebuah software baru yang menggandakan fungsi-fungsi yang tersedia pada software yang telah ada dan tersedia bagi pelanggan tersebut dengan biaya yang lebih rendah, hal ini disebut “re-inventing the wheel’ dan merupakan pemborosan biaya. Jika pelanggan tidak bermasalah dengan tersedianya software yang sama bagi pihak lain, dan tidak membuatuhkan kendali penuh pada proses pengembangan, maka paradigma open source atau retail mungkin dapat menjadi paradigma yang lebih efektif dari segi biaya.

Pengembangan secara in-house dan kontrak sangat efisien dalam mengarahakn setiap rupiah yang dihabiskan untuk pengembangan software. Efisiensinya dapat mencapai 50% hingga 80%. Sumber inefisiensi dari paradigma ini antara lain biaya untuk mendapatkan klien baru bagi bisnis

kontraktor, biaya untuk menjaga keahlian khusus yang tidak selalu dipakai setiap saat, dan mark-up yang dilakukan oleh kontraktor.

Banyak proyek in-house dan kontrak yang gagal menghasilkan software yang digunakan dlam bisnis klien dan memenuhi tujuan yang telah ditentukan. Maka kita dapat menyimpulkan tingkat kesuksesan sebesar 50%, seperti yang kita terapkan pada paradigma retail.

Usaha Kolaborasi Tanpa Lisensi Open Source

Konsorsium biasa digunakan sebagai metode standar kolaborasi beberapa perusahaan dalam pengembangan software. Konsorsium pengembangan software secara tertutup memiliki catatan kegagalan yang tinggi. Namun, karena dilakuakn secara tertutup terkadang kegagalan tersebut tidak terlihat, seperti gunung es yang tidak terlihat oleh kapal Titanic ketika menabraknya dan tenggelam. Beberapa waktu terakhir, ada proyek konsorsium milyaran dolar yang akhirnya digantikan oleh proyek open source. Kita bisa melihat Taligent dan Monterey, dua konsorsium yang berniat untuk membuat pengganti Unix. Linux telah mendahului mereka. Kemudian ada proyek *Common Desktop Environment*, telah digantikan oleh proyek desktop open source GNOME, padahal banyak perusahaan yang mendukung CDE saat itu.

Beberapa tahun yang lalu, Cargill, sebuah perusahaan agrikultur besar, mendirikan sebuah konsorsium yang bertujuan untuk menyediakan keuntungan open source sekaligus menyediakan kerahasiaan dan berbagi manfaat dari software hanya untuk anggota konsorsium. Ini disebut komunitas terkontrol. Dua tahun kemudian, Cargill meninggalkan konsorsium yang didirikannya. Konsorsium tertutup merupakan struktur yang salah untuk pengembangan software yang tidak memiliki unsur pembeda bagi bisnis msing-masing. Alangkah lebih baik bila pintu dibuka lebar-lebar ketika tidak ada unsur pembeda yang harus dilindungi atau disembunyikan. Dan akuilah anggota yang dapat memberikan kontribusi yang berguna meskipun tidak dapat memberikan kontribusi dalam bentuk dana. Biaya semakin besar dalam sistem konsorsium karena hanya ada sedikit anggota yang akan berbagi biaya dan resiko dibandingkan dengan proyek open source. Selain itu, ada banyak struktur dan biaya overhead diabndingkan proyek open source yang serupa. Konsorsium tertutup biasanya dipimpin dengan metode *pay-for-say*, yaitu mereka yang memberikan kontribusi dana yang lebih besar merasa berhak untuk lebih menentukan arah pengembangan software. Sementara proyek open source lebih mengedepankan kemampuan teknis masing-masing anggota. Dengan *pay-for-say*, seorang anggota dapat mengarahkan konsorsium kepada kegagalan proyek secara keseluruhan jika hal itu menguntungkan bagi anggota tersebut. Perencanaan produk dalam sebuah konsorsium dapat mengarah kepada argumen yang tidak terselesaikan di antara berbagai perusahaan, sebab masing-masing memiliki ide pemasaran yang berbeda dan argumentasi pemasaran di atara berbagai perusahaan adalah hal yang subyektif dan sulit untuk dicarikan solusinya.

Sejarah yang buruk dari pengembangan software melalui konsorsium dan sebaliknya tingkat kesuksesan yang tinggi dari proyek-proyek besar open source yang dilakukan oleh kelompok perusahaan yang sama, menunjukkan bahwa ketiadaan diskriminasi yang diakibatkan oleh adanya lisensi open source merupakan komponen penting untuk menghasilkan kolaborasi yang efektif di antara sejumlah besar pihak dengan kenginan yang berbeda-beda.

Paradigma Open Source

Dalam paradigma open source, berbagai pihak (individu, perusahaan, institusi pendidikan, dan lain-lain bergabung untuk bekerja sama mengembangkan sebuah software. Umumnya,

pengembangan awal dilakukan oleh satu pihak seperti dalam paradigma in-house dan kontrak. Kemudian, software dirilis ke publik ketika software tersebut sudah memiliki fungsionalitas yang memadai untuk dimanfaatkan. Namun, software tersebut belum dianggap sebagai barang jadi. Software tersebut dirilis ke publik jauh lebih awal dibandingkan dengan rilis software pada paradigma retail.

Saat software tersebut sudah dapat digunakan, pihak lain akan mengambil manfaat dari software tersebut. Paradigma open source akan bermanfaat sepenuhnya bila software tersebut sudah berguna bagi pihak lain. Dengan demikian berbagai pihak memiliki motivasi positif untuk menggunakannya. Setelah menggunakan software tersebut, pihak-pihak lain ini akan memiliki motivasi positif untuk memperluas kemampuan software tersebut dengan menambahkan fitur-fitur yang menarik bagi mereka. Perluasan ini dapat dilakukan oleh programmer internal perusahaan yang menjadi pengguna atau kontraktor yang mengerjakannya sesuai keinginan perusahaan.

Peningkatan biaya untuk penambahan fitur jauh lebih rendah dibandingkan biaya untuk pengembangan keseluruhan. Berbagai pihak yang membuat modifikasi mempunyai motivasi positif untuk menulis perubahan tersebut dalam bentuk yang dapat diterima oleh pengembang lain dalam proyek tersebut. Selain itu, modifikasi tersebut dapat digabungkan dengan kode program inti software tersebut yang dapat diakses oleh seluruh pengembang software tersebut. Jika penggabungan ini tidak terjadi, biaya berkelanjutan untuk merawat fitur tambahan tersebut akan menjadi lebih tinggi. Sebab, pengembangnya terpaksa menelusuri perubahan di kode inti dan menjaga kompatibilitas dengan perubahan pada kode program inti.

Sebagai hasilnya, open source cenderung menumbuhkan komunitas pengembang software yang memberikan kontribusi pada software yang bermanfaat. Biaya dan resiko pengembangannya didistribusikan di antara pengembang dalam komunitas dan berbagai kombinasi di antara mereka dapat melanjutkan proyek jika yang lain meninggalkan proyek. Distribusi biaya dan resiko dimulai ketika proyek sudah cukup matang untuk membangun komunitas di luar pengembang awal.

Open source dikembangkan secara langsung oleh penggunanya. Misalnya, fitur-fitur dalam server web Apache ditambahkan oleh berbagai perusahaan yang membutuhkan fitur-fitur tersebut untuk menjalankan situs web mereka, atau terkadang dilakukan oleh kontraktor yang bekerja untuk berbagai perusahaan tersebut.

Pengguna produk open source tertentu biasanya mengidentifikasikan dirinya sendiri: mereka mencari produk yang mereka butuhkan dalam direktori software open source, lalu mereka mengunduhnya, dan menguji softwarenya. Jika software tersebut lolos pengujian, mereka lalu mendistribusikan penggunaan software tersebut secara resmi. Selanjutnya, mereka akan mendapat ketertarikan yang berkelanjutan pada produk tersebut. Pada titik ini, mereka menginginkan fitur tambahan pada software tersebut, dan mereka pun memiliki motivasi positif untuk menjadi *co-developer* software yang mereka gunakan.

Berbagai perusahaan yang bergabung dalam kolaborasi open source menggunakan software pada sisi yang tidak membedakan bisnisnya dengan perusahaan lain. Bagi perusahaan-perusahaan tersebut tidaklah penting apakah open source itu sendiri menghasilkan keuntungan. Software bukanlah sumber pendapatan mereka. Mereka menggunakan software sebagai teknologi yang memungkinkan bisnis mereka dapat berjalan. Agar sumber pendapatan mereka dapat berjalan, investasi harus mereka lakukan pada sumber biaya yang ada. Pada kasus software yang menjadi pembeda bagi bisnis mereka, pilihannya adalah pengembangan in-house atau kontrak. Sebab, mereka harus mencegah resep rahasia bisnis mereka agar tidak jatuh

kepada para pesaingnya. Sedangkan software *non-differentiator*, pilihan mereka adalah paradigma retail atau open source. Manakah yang lebih efisien?

Karena pengguna mengidentifikasikan dirinya sendiri, maka open source tidak memiliki inefisiensi pada paradigma retail yang harus menggunakan mekanisme iklan atau mekanisme mahal lainnya untuk mencari pelanggan. Setidaknya, open source lebih efisien dalam mengarahkan pundi-pundi perusahaan untuk pengembangan software dibanding harus membiayai overhead dalam paradigma in-house atau kontrak.

Karena efisiensi yang lebih besar dalam mengalokasikan sumber daya untuk pengembangan daripada overhead, paradigma open source dapat digunakan untuk mengembangkan produk yang tidak memiliki pasar yang luas yang tidak dapat dikembangkan secara ekonomis menggunakan paradigma retail.

Craig Small, seorang *builder* infrastruktur jaringan, mengikhtisarkan keuntungan-keuntungan open source bagi pelanggannya:

Penggunaan open source memangkas biaya-biaya awal pengadaan software. Hal ini penting karena merupakan pembuka jalan bagi pelanggan tersebut. Saya sedang berencana untuk membangun sistem pengelolaan jaringan dari nol, sampai saya menemukan sebuah proyek yang cukup mendekati apa yang saya butuhkan. Saya bisa hanya menghabiskan sebagian kecil dari waktu yang dibutuhkan jika dibandingkan dengan mengembangkannya dari nol.

Mengembangkan dan mengkustomisasi produk open source jauh lebih mudah dilakukan. Hal ini dikarenakan para pengembang sejak awal berharap bahwa orang-orang akan mengembangkan hasil kerja mereka lebih lanjut dengan cara-cara yang tidak terpikirkan oleh mereka sebelumnya. Sehingga mereka menulis software dalam bentuk yang lebih mudah untuk dikembangkan lebih lanjut. Sementara pembuat software propietary menulis program seolah tidak akan pernah ada seorang pun di luar perusahaannya yang akan membaca kode program mereka. Kami pun mengambil sebuah proyek open source yang menggunakan perangkat keras yang sma sekali berbeda dari yang kami miliki, namun dengan mudah kami membuatnya memonitor peralatan kami.

Saya telah mengembalikan extension yang saya kembangkan ke proyek open source tersebut. Sehingga saya percaya jika saya meninggalkan perusahaan, extension yang saya buat akan tetap dipelihara, setidaknya oleh anggota lain proyek tersebut.

Saat ada komunitas yang terdiri dari kesamaan kebutuhan teknis, baik pengembang software maupun penggunan non programmer yang berkembang di sekitar proyek software yang kami berkolaborasi di dalamnya. Kami berdiskusi tentang bagaimana memenuhi keinginan kami dan menggabungkan ide-ide yang tidak pernah terpikirkan jika kami mengerjakannya sendiri.

Software propietary memiliki biaya yang baru saja diperhatikan oleh dunia bisnis. Organisasi seperti *Business Software Alliance* terus-terusan memeriksa kalangan bisnis bersama polisi nasional, meminta dokumentasi bahwa sebuah perusahaan membeli lisensi untuk setiap komputer yang ada. Banyak sekali waktu dan usaha yang dihabiskan untuk memastikan berbagai perusahaan memenuhi ketentuan lisensi yang semakin ketat saja. Mengapa misalnya, kita tidak dapat memindahkan Windows XP dari laptop kita ke komputer desktop kita secara legal? Open source menghindari hal-hal yang tidak masuk akal dan tidak produktif seperti ini.

Tidak semua proyek open source menuai kesuksesan. Banyak proyek open source yang belum matang, berguguran ketika masih berupa bibit, atau tetap bertahan sebagai proyek solo dan

menggunakan sumber daya yang kecil. Akan tetapi biaya yang dikeluarkan untuk proyek-proyek tersebut tidaklah signifikan. Sebab, proyek tersebut mati tanpa harus menarik perhatian pelanggan yang signifikan atau komunitas pengembang terlebih dahulu. Proyek open source yang lebih matang bisa juga mati ketika kondisi sudah lebih baik. Namun, seringkali kode program, data, dan keterampilan dari satu proyek dimungkinkan untuk digunakan pada proyek lain. Sehingga investasi dalam pengembangan proyek tidaklah hilang.

Biaya partisipasi dalam proyek open source yang sudah matang sangatlah berbeda dengan biaya pengembangan dalam paradigma retail, in-house atau kontrak. Pengeluaran terbesar adalah biaya untuk meluangkan waktu berpartisipasi. Ilustrasinya adalah kombinasi biaya personalia dalam pengujian software, sumber daya perorangan atau kontraktor yang dihabiskan untuk mengadaptasi produk open source yang sudah ada agar memenuhi kebutuhan dan memberikan dukungan open source bagi pengguna internal, serta kemungkinan investasi waktu pada software yang digantikan karena gagal menemukan pelanggan. Biaya terbesar dari open source akan terjadi ketika tidak ada komunitas selain pengguna: ini akan menjadi sama saja dengan biaya pengembangan in-house atau kontrak. Yakni, seorang pelanggan menanggung seluruh biayanya. Biaya tersebut bisa menjadi lebih rendah bergantung pada jumlah anggota yang berpartisipasi secara aktif dan seberapa besar pekerjaan yang diperlukan. Investasi yang bisa hilang umumnya adalah waktu personil. Dengan memperhitungkan hal ini, paradigma open source setidaknya menghasilkan efisiensi eknomis sebesar paradigma in-house dan kontrak, serta jauh lebih efisien dibanding paradigma retail.

Ringkasan Metode Bisnis dalam Produksi Software

Paradigma open source memiliki beberapa keuntungan ekonomis yang signifikan daripada produksi software secara retail atau in-house dan kontrak. Open source mengombinasikan efisiensi alokasi sumber daya untuk pengembangan software dengan distribusi biaya dan resiko yang lebih baik daripada paradigma yang lain. Open source dapat memungkinkan pengembangan produk yang tidak dapat dilakukan dengan paradigma retail karena tidak ada pasar yang luas. Selain itu, biaya dan resiko juga terdistribusi, sementara pada in-house dan kontrak, seluruh biaya dan resiko ditanggung oleh satu pihak saja. Open source mendistribusikan biaya dan resiko secara langsung, tidak memerlukan investasi di bidang pemasaran. Distribusi resiko dan biaya dilakukan jauh lebih awal daripada paradigma retail. Open source memberikan kendali penuh kepada penggunanya untuk mengkustomisasi produknya. Meski demikian, pengguna tidak diberi hak untuk mengatur siapa saja yang dapat mengakses produk tersebut. Sehingga open source kurang bagus bila digunakan untuk mengembangkan software yang menjadi pembeda bisnis (differentiating software).

Bisnis (segala jenis bisnis) yang membutuhkan software yang tidak menjadi pembeda, sangat disarankan untuk mengalihkan beberapa pengembangannya ke open source agar dapat mengambil efisiensi ekonomis yang lebih besar. Partisipasi dalam komunitas open source seharusnya menjadi bagian penting dari strategi keseluruhan pada bisnis apapun untuk mengalihkan biaya pengembangan dari software yang tidak menjadi pembeda bisnis (*non-differentiating software*) ke software yang menjadi pembeda bisnis (*differentiating software*).

Bisnis Open Source versus Konsep Free?

Apakah Open Source Bisa Bertahan?

Banyak orang yang mengalami kesulitan dalam memahami bagaimana open source dapat terus bertahan hidup dalam jangka panjang. Padahal open source tidak berjalan sesuai dengan paradigma pengembangan software secara retail. Dari mana open source mendapatkan dana pengembangan? Sebab, proyek open source tentu memerlukan dana dalam proses pengembangannya. Banyak orang yang juga memiliki kekhawatiran bahwa open source akan mati karena bersifat gratis dan tidak ada yang membiayai proses pengembangannya.

Sebenarnya, open source juga dibiayai oleh para pengguna atau pelanggannya. Jadi, sumber dana pada paradigma open source sama saja dengan paradigma retail. Ingat, sumber dana suatu pabrik software bukan dari dalam pabrik software itu sendiri, melainkan dari kantong para pelanggan/pengguna yang membeli software mereka. Metode open source dibiayai oleh anggaran pengeluaran pada berbagai perusahaan yang membutuhkan produk open source tersebut. Baik secara langsung maupun tidak langsung.

Mereka bersedia melakukan investasi dalam pembuatan software tersebut melalui paradigma open source. Mengapa? Karena mereka dapat berbagi biaya dan resiko pengembangannya dengan banyak pihak yang berkolaborasi di dalamnya. Dengan demikian, biaya yang mereka keluarkan lebih sedikit dan anggaran pengeluaran mereka untuk software menjadi lebih efisien, daripada menggunakan paradigma retail.

Apa Akibat Ekonomis Open Source?

Kita telah mendefinisikan akibat ekonomis dari Microsoft sebagai:

Fakta bahwa mereka memungkinkan berjalannya berbagai bisnis besar (milik para pelanggan), menjadikan bisnis dapat berjalan lebih efisien, dan mempunyai bisnis yang tidak berjalan sama sekali tanpa adanya software Microsoft.

Definisi yang sama berlaku untuk Open Source.

Faktanya, open source telah menjadi tulang punggung internet dan web server hingga hari ini, mayoritas aplikasi dasar untuk pengiriman email (mail server), memungkinkan terkirimnya email dan berjalannya berbagai bisnis lain, organisasi, maupun pencapaian perorangan. Bisnis apa yang tidak membutuhkan internet? Jadi, jika dihitung dengan uang, efek ekonomis open source mungkin bisa mencapai puluhan milyar dolar. Atau bahkan lebih.

Setiap pengembangan teknologi yang mengizinkan bisnis berfungsi lebih efisien berarti sistem ekonomi juga berjalan lebih efisien. Pada kasus ini, open source memungkinkan dunia bisnis mengeluarkan lebih sedikit anggaran untuk software. Sekaligus, mendapatkan software dengan kualitas yang lebih baik dan mempunyai kendali lebih besar pada software yang dipakainya. Bukan malah dikendalikan oleh perkembangan software. Uang yang berhasil dihemat tidaklah hilang, melainkan dapat digunakan untuk hal yang penting bagi masing-masing bisnis.

Lalu, Apa Keuntungan yang Didapat Oleh Produsen Software?

Kita telah banyak membahas dari sudut pandang dunia bisnis yang menggunakan software dalam bisnisnya. Yaitu, mereka yang mengeluarkan biaya pengembangan software. Namun, kita belum membahas dari sudut pandang perusahaan pembuat software. Apa keuntungan open source bagi pembuat software?

Jika anda menjalankan bisnis pembuatan software, apakah anda selalu menjual software yang benar-benar sempurna dan final? Beberapa bisnis menjual kegiatan pembuatan software alih-alih menjual software itu sendiri. Secara khusus, hal ini dilakukan oleh perusahaan penyedia jasa konsultasi dan solusi software. Pelanggan anda akan membayar atas jasa pembuatan suatu software open source.

Pada kasus bisnis yang lebih memilih untuk membuat software untuk dijual daripada menjual jasa pemrograman atau pelatihan, metode open source agak sulit dijalankan. Namun, bukan berarti hal ini mustahil. Ada beberapa contoh sukses yang dapat anda simak pada Bab 6 yang membahas berbagai model bisnis open source.

Barangkali open source memang menyebabkan penurunan permintaan propietary software. Namun, hal ini tidak akan menurunkan permintaan pemrogram, karena permintaan software secara umum tidak akan turun. Data juga menunjukkan bahwa sebagian besar software tidak dibuat untuk dijual. Melainkan untuk memenuhi kebutuhan internal. Pemrogram propietary mungkin akan beralih ke organisasi yang membuat software open source dengan metode yang sukses secara ekonomis.

Mungkin saja pemrogram yang berpindah ke tempat yang kurang berorientasi pada bisnis mendapatkan penghasilan yang lebih sedikit. Tetapi pendapatan surplus pada perusahaan pembuat software biasanya jauh lebih banyak dibagikan kepada pihak manajemen daripada staf. Sangat jarang programmer yang bisa mendapatkan penghasilan dari nilai saham perusahaan yang memiliki produk dominan di pasaran.

Permasalahan *Free-Rider*

Free-Rider adalah sesuatu yang umum dijumpai dalam sistem ekonomi. Apa yang akan anda lakukan pada orang-orang yang menggunakan suatu produk atau jasa tanpa memberikan imbalan kepada penyedia produk dan jasa tersebut? Apakah anda memiliki mekanisme yang efektif untuk mencegah adanya free-rider?

Semua pengguna open source berawal dari status free-rider. Mereka mengunduh suatu aplikasi open source, lalu menggunakannya, dan barangkali juga menyebarkannya kepada teman-temannya. Pada umumnya mereka melakukannya tanpa terpikir untuk memberikan kontribusi dalam pengembangannya. Sampai suatu ketika mereka menginginkan tambahan fitur.

Jika mereka menginginkan tambahan fitur, mereka bisa saja menambahkannya sendiri daripada membayar developer aslinya. Pada titik ini, ia tidak lagi menjadi free-rider. Berbagai bisnis yang bergabung sebagai developer dalam proyek open source memberikan kontribusi beberapa software, dan seluruh bisnis tersebut mengambil keuntungan ekonomis dari penggunaan software tersebut. Minat setiap developer umumnya sama, karena mereka telah memilih sendiri produk software tertentu yang berguna bagi mereka. Sehingga kontribusi developer tertentu umumnya juga berguna bagi developer yang lain.

Ada juga developer yang tidak didorong oleh keinginan menyediakan software bagi dunia bisnis. Individu-individu seperti ini biasanya meempunyai motivasi seperti seniman, atau peneliti ilmiah.

Para voluntir mendapatkan pemenuhan kebutuhan emosional untuk memiliki pengguna bagi software mereka. Sebagaimana seorang seniman memenuhi kebutuhannya atas apresiasi orang lain pada lukisannya. Bagi para voluntir, pengguna memberikan keuntungan tidak terlihat. Keuntungan yang memang diinginkan oleh para voluntir. Dengan demikian pengguna juga tidak bisa dianggap sebagai free-rider.

Perusahaan-perusahaan yang memiliki kepentingan pada produk open source tertentu cenderung menyewa developer yang sudah mendapatkan status sebagai developer produk tersebut. Maka individu yang sebelumnya tidak memiliki minat pada uang dalam proyek open source cenderung menemukan pekerjaan (yang dibayar) pada organisasi yang tertarik. Partisipasi mereka pada proyek open source kerap mendatangkan keuntungan ekonomis. Ini adalah alasan lain bahwa pengguna tidak dapat begitu saja dianggap sebagai free rider.

Peneliti sains mempunyai paradigma sendiri mengenai pertukaran pengetahuan berkesinambungan. Serupa dengan semangat saling berbagi dan saling belajar di komunitas open source. Sebab, sains berkembang jauh lebih cepat ketika penemuan-penemuan diumumkan kepada saintis lainnya yang dapat menambahkan ide-ide dan intuisi mereka pada penemuan-penemuan tersebut. Saintis memperoleh kepuasan dengan memublikasikan hasil kerja mereka. Karena publikasi tersebut akan meningkatkan statusnya di mata saintis lainnya. Selain itu, hasil penelitian tersebut umumnya juga menentukan kesuksesan karir mereka sebagai peneliti. Saintis sudah berulang kali menggunakan open source sebagai media publikasi komponen software dari pekerjaan mereka. Sebagai tambahan, para saintis biasanya didorong oleh keinginan memberi manfaat kepada masyarakat. Mereka juga ingin menyaksikan orang lain mendapatkan manfaat dari menggunakan software yang dikembangkan melalui penelitian ilmiah.

Para pengguna adalah orang-orang yang kita rekrut untuk nantinya menjadi partisipan yang lebih aktif dalam proyek open source, sebagaimana pengecer berusaha merekrut orang-orang yang mau membeli software yang mereka buat.

Lalu, apakah permasalahan free-rider ini berpengaruh signifikan pada software dan open source? Free-rider pada penumpang bus atau KRL Jabotabek, misalnya, akan menggunakan sisa ruang dalam bus atau KRL. Hal ini bisa saja membuat penumpang yang membayar jadi tidak bisa naik bus atau KRL tersebut karena sudah penuh. Sedangkan free-rider yang menggunakan salinan Microsoft Windows tanpa membeli lisensi bisa saja mengurangi jumlah Windows yang dibeli secara resmi. Meski bisa juga tidak. Namun pengguna Windows bajakan tidak lantas menghalangi kemungkinan orang lain untuk menggunakan Windows seperti pada kasus bus dan KRL. Free-rider yang menggunakan produk open source sama sekali tidak mengurangi pasar produk open source yang bersangkutan ataupun mengecualikan orang lain untuk dapat menggunakannya.

Dengan demikian, dapat disimpulkan bahwa free-rider pada dunia open source tidak menimbulkan kerugian dalam pengembangan open source.

Mengapa Tidak Ada Mobil Open Source?

Paradigma open source bekerja dengan baik pada banyak produk yang nilai utamanya terletak pada rancangannya. Hingga saat ini, metodologi open source telah sukses digunakan untuk membuat software, ensiklopedia, dan rancangan IC (integrated circuit).

Rancangan IC bisa ditanamkan ke sebuah alat khusus yang disebut *programmable gate-array.* Alat ini lalu bertindak sebagai sirkuit sesuai dengan rancangan tersebut. Ini adalah contoh kasus bahwa perangkat keras sangat mirip dengan perangkat lunak.

Namun demikian, sebagian besar benda bukanlah software. Hampir tidak dibutuhkan biaya untuk menyalin software dari disket yang satu ke disket yang lain. Atau, yang sedang populer saat ini adalah melalui FTP (*Flashdisk Transfer Protocol*). Menyalin buku yang sedang anda baca ini tentu membutuhkan biaya fotokopi yang tidak sedikit. Menggandakan kursi tetap membutuhkan palu, paku, kayu, gergaji dan alat-alat pertukangan yang lain.

Mobil jauh lebih kompleks dari buku dan kursi. Selain itu juga dibutuhkan banyak proses fisik dengan alat-alat berat dan mahal. Untuk membuat motor elektrik, seseorang harus menambang logam, memperhalusnya, lalu menarik kawat, menggulung lembaran logam, membuat cetakannya dan merakitnya dalam bentuk akhir yang harus memiliki presisi sangat tinggi. Tidak mengherankan bila dibutuhkan seluruh entitas ekonomi untuk membuat mobil. Sementara software yang berkualitas bisa saja dibuat oleh satu orang saja.

Bila suatu hari nanti kita bisa menghasilkan produk fisik yang kompleks hanya dengan membuat rancangannya saja dan memerintahkan sebuah mesin untuk menerapkan rancangan tersebut pada material logam yang dan listrik tersedia, ekonomi kita akan berubah secara radikal. Hingga hari ini, kita masih terbatas pada pembuatan komponen dengan mesin yang dikendalikan komputer, ditambah dengan proses yang masih melibatkan intervensi manual. Komunitas open source sudah berkembang pada mesin-mesin dan kita sudah mulai melihat mereka berbagi rancangan. Namun, ilmu dan teknologi manufaktur yang dikendalikan oleh komputer masih harus berkembang sangat pesat sebelum kita bisa memiliki mobil open source.

Mengapa Open Source Bisa Berbisnis?

CEO (Chief Executive Officer) Netscape, Jim Barksdale sering berkata, “there is more to being in business than simply making money”. Secara khusus, ia sering mengutip ucapan Peter Drucker, “the purpose of any business is to create customers and keep them”.

Pelanggan adalah organisasi atau individu yang melihat nilai dari produk dan jasa yang anda sediakan. Mereka membayar sejumlah uang untuk mendapatkannya. Anda menciptakan pelanggan dengan meyakinkan orang bahwa anda dapat menyediakan sesuatu yang bernilai atau berguna bagi orang tersebut. Kemudian anda merayu dan mempengaruhi mereka untuk memberikan uang kepada anda sebagai ganti atas nilai atau kegunaan tersebut.

Anda menjaga keberadaan pelanggan dengan menyediakan nilai yang berkesinambungan bukan nilai sesaat saja. Nilai ini harus memadai untuk membuat mereka terus memberikan uang kepada anda sebagai ganti atas nilai-nilai yang terus mereka terima dari anda.

Bagi para pelanggan, nilai tidak harus berasosiasi dengan fitur-fitur tertentu dari suatu produk. Faktanya, produk dengan fitur yang sama namun ditawarkan oleh dua perusahaan yang berbeda bisa mempunyai nilai yang berbeda bagi pelanggan. Secara lebih umum, nilai suatu produk berkaitan dengan sejauh mana produk tersebut menyelesaikan permasalahan yang dihadapi pelanggan. Semakin banyak atau semakin besar dan pentingnya masalah yang dapat diselesaikan oleh produk tersebut, semakin besar pula nilainya bagi pelanggan.

Para pelanggan bisa saja menyelesaikan permasalahan tersebut dengan bantuan fitur sebenarnya dari suatu produk. Misalnya, web browser seperti NCSA Mosaic dan Netscape

Navigator awalnya ditujukan untuk mengatasi masalah dalam mengakses internet tanpa peralatan (utilitas) yang kompleks. Mereka juga bisa mengatasi masalah dengan bantuan atribut lain dari suatu produk. Misalnya, merk dan reputasi kualitas yang baik bisa membantu menyelesaikan masalah dalam membuat keputusan pembelian yang “aman”.

Agar tetap sukses seiring waktu berjalan, sebuah perusahaan dapat mencoba memberikan nilai tambah pada jajaran produknya dengan berbagai cara. Cara yang dapat ditempuh antara lain: menambahkan fitur baru pada produk yang sudah ada, membuat produk baru, menambahkan metode baru dalam menggunakan produk, dan lain sebagainya. Agar perusahaan anda terus berkembang, anda harus menemukan cara-cara baru untuk memberikan nilai kepada para pelanggan. Sekaligus cara-cara baru untuk mengubah nilai tersebut menjadi uang yang mengisi pundi-pundi perusahaan anda.

Tapi jangan lupa, pesaing anda juga bisa melakukan hal yang sama. Pesaing anda juga bisa menambahkan fitur baru, produk baru, cara baru dalam menggunakannya, dan dalam banyak kasus mereka memiliki sumber daya yang lebih besar untuk melakukannya. Jika anda ingin meraih kesuksesan pada pasar yang kompetitif, anda bisa mencoba dengan menawarkan suatu produk lebih awal daripada pesaing anda. Atau menciptakan nilai yang melebihi tawaran produk pesaing anda. Atau membuat nilai bagi produk dan jasa ada dengan cara tertentu sehingga pesaing anda tidak dapat menirunya dengan mudah.

Pada era inovasi yang sangat cepat, memenangkan persaingan berarti keharusan mengejar keuntungan dari nilai relatif yang diturunkan dari fitur-fitur produk. Karena nilai dari fitur produk umumnya hanya bersifat sementara, maka harus dikombinasikan dengan usaha yang lebih strategis untuk mengubah keseimbangan kompetisi dengan suatu cara yang kurang dikuasai atau diantisipasi oleh pesaing anda.

Dalam hal ini, perpindahan menuju model bisnis open source tidak lantas menjadi langkah taktis untuk menyelesaikan masalah bisnis tertentu. Namun model bisnis open source lebih bersifat sebagai bagian dari keseluruhan strategi untuk mengubah aturan-aturan kompetisi dalam *market space* anda atau dalam industri software secara keseluruhan.

Giving Away the 'Crown Jewels'?

Mari kita asumsikan bahwa anda sependapat dengan pembahasan di atas, dan anda setidaknya mau mempertimbangkan bahwa open source menawarkan cara yang mungkin bagi perusahaan anda untuk bersaing dengan lebih baik di pasar. Namun demikian, dapat diduga bahwa dalam batas tertentu ada pertanyaan yang biasa muncul berkaitan dengan pengalaman anda mengenai konsep open source bagi anda. Pertanyaan tersebut adalah, “*Why would a software company want to give away its crown jewels to the world?*"

Gagasan mengenai *source code* sebagai “crown jewels” adalah metafor yang umum dipakai. Metafor yang bisa menjadi kekuatan besar bagi hal yang baik atau buruk, tergantung konteksnya. Crown berarti mahkota, dan jewel berarti berlian. Metafor ini bermakna bahwa kode program milik perusahaan software adalah sumber daya yang sangat berharga dan harus dilindungi dari siapapun di luar perusahaan dengan berbagai cara. Sebagian besar perusahaan software komersial merilis kode program dari produk mereka hanya pada kondisi yang sangat khusus. Mereka juga hanya melakukannya disertai dengan penetapan aturan khusus. Termasuk di dalamnya persetujuan untuk tidak memperlihatkannya kepada pihak lain dengan istilah dan syarat yang sangat ketat dalam redistribusi. Selain itu masih harus ditambah dengan biaya pindah tangan yang sangat mahal.

Akan tetapi, *source code* hanya memperoleh nilai komersial hanya jika anda bisa menggunakannya untuk membuat produk dan atau jasa yang anda jual dengan imbalan uang. Secara khusus, anda bisa melakukannya dengan sukses hanya dalam konteks sebuah perusahaan dengan merek yang sudah dikenal baik dan memiliki posisi menguntungkan di pasar. Misalnya, meski anda memiliki *source code* Windows yang pernah bocor di internet beberapa waktu yang lalu, anda tidak akan lantas menjadi sekaya Bill Gates.

Merilis kode program salah satu produk anda sebagai open source tidak lantas berarti perusahaan lain dapat menduplikasi kesuksesan anda atau menimbulkan kerugian bagi bisnis anda. Kesuksesan tersebut lebih ditentukan oleh banyak faktor, bukan hanya fitur-fitur produk yang ada dalam *source code*. Faktor-faktor lain yang berpengaruh yaitu: kepercayaan dan dikenalnya nama dari *brand* anda, kualitas dan reputasi layanan atau jasa anda, efektivitas jalur distribusi dan penjualan anda, dan sebagainya. Faktor-faktor tersebut masih sangat dipengaruhi oleh anda, bahkan dalam kondisi ketika anda tidak memiliki kendali pada source code itu sendiri.

Nilai yang Terkandung dalam Source Code

Bila anda sudah menerima kemungkinan bahwa merilis source code salah satu atau lebih dari produk-produk anda bisa menjadi bagian dari keseluruhan strategi bisnis anda, akan muncul dua pertanyaan. Pertama, bagaimana distribusi souce code bisa menghasilkan nilai bagi para pelanggan. Kedua, bagaimana perusahaan anda mengubah nilai dari source code tersebut menjadi penghasilan dan keuntungan (misalnya, dalam bentuk uang).

Mari kita definisikan terlebih dahulu pengertian “source code” dalam pembahasan ini. Source code suatu produk adalah pernyataan-pernyataan dalam bahasa pemrograman yang menjadi dasar atau pokok dari produk tersebut (termasuk informasi yang berkaitan, seperti *make files* dan *error message files*, dan lain-lain). Source code harus dapat dibaca oleh manusia, dapat dimodifikasi untuk membuat perubahan dan penambahan, lalu digunakan untuk membangun versi fungsional (bentuk binari) produk tersebut atau produk turunan lain yang menggunakan sebagian atau keseluruhan source code yang sama.

Pada pembahasan ini, diasumsikan bahwa anda sebagai pemilik produk bersedia merilis keseluruhan source code yang berkaitan. Pada praktiknya, hal seperti ini tidak terjadi pada semua kasus. Misal, awalnya anda mungkin mendistribusikan komponen tertentu hanya dalam bentuk binari, atau bahkan tidak merilis sama sekali komponen tertentu.

Banyak software telah didistribusikan dalam bentuk source code sesuai tradisi. Ada beberapa cara yang dapat diterima umum mengenai peningkatan nilai suatu produk secara langsung karena menyediakan source code untuk produk tersebut. Peningkatan nilai produk tersebut bagi pelanggan terjadi karena tersedianya source code dapat membantu pelanggan memecahkan salah satu atau lebih dari permasalahan-permasalahan berikut ini:

1) Pelanggan dapat melindungi investasinya pada suatu produk software secara lebih baik. Permasalahan mungkin baru muncul jika vendor software tersebut gulung tikar atau memutuskan penghentian pengembangan dan dukungan bagi suatu produk yang sangat penting dalam operasional bisnis milik pelanggan.

   Source code escrow atau mekanisme penyediaan source code yang disimpan oleh pihak ketiga sebagai wasiat yang tertulis dalam kontrak juga memberikan perlindungan yang sama. Namun, secara khusus hal ini biasanya hanya berlaku untuk kasus ketika vendor

keluar dari dunia bisnis alias bangkrut. *Source code escrow* juga tidak memberikan keuntungan-keuntungan tambahan yang tidak langsung.

2) Pelanggan dapat memahami dengan lebih baik mengenai bagaimana software tersebut bekerja atau digunakan. Hal ini bisa dirasakan bila dokumentasi yang dibuat oleh vendor tidak lengkap atau membingungkan.

3) Pelanggan dapat mencari dan memperbaiki kerusakan atau potensi lubang keamanan yang dapat menimbulkan kerugian dalam operasional bisnis pelanggan.

4) Pelanggan dapat memperbaiki *bugs* sendiri, khususnya bila vendor tidak bisa atau tidak punya niat untuk melakukannya.

5) Pelanggan dapat mengajukan atau melakukan pemeriksaan independen pada dukungan Millenium Bug Year 2000 (atau Year 2030 yang diramalkan akan menimpa sistem Unix) dan syarat-syarat lainnya serta memperbaiki permasalahan yang ditemukan. Tidak tersedianya source code waktu itu menambah kerumitan penyelesaian masalah Millenium Bug Y2K. Sedangkan pada kasus Unix yang baru diramalkan akan terjadi, sudah tersedia beberapa alternatif penyelesaian masalah karena tersedianya source code.

6) Pelanggan dapat memodifikasi software tersebut agar berjalan pada sistem operasi atau perangkat keras lain yang tidak didukung oleh vendor. (melakukan *porting* aplikasi, sekaligus memperluas pasar pengguna produk yang bersangkutan)

7) Pelanggan bisa menggunakan source code untuk membuat versi kustom dari produk software yang asli, diperluas fungsinya atau ditingkatkan kemampuannya, atau membuat satu aplikasi baru dan menghindari kebutuhan untuk membangun atau menulis aplikasi tersebut dari nol alias "from scratch."

Distribusi source code suatu produk juga dapat meningkatkan nilai produk tersebut bagi pelanggan, secara tidak langsung, yaitu:

1) Penulis dan pelatih atau tutor yang tidak memiliki asosiasi dengan vendor bisa membuat dokumentasi dan materi pelatihan atau tutorial yang lebih lengkap dan tepat. Mereka tidak harus terlalu sering bergantung pada dokumentasi asli dari vendor (yang terkadang tidak lengkap). Sehingga proses mempelajari produk tersebut menjadi lebih mudah. Pelanggan pun dapat melatih pengguna akhir dan developer mereka secara lebih mudah dan lebih murah.

2) Konsultan dan integrator sistem menjadi lebih familiar dengan teknologi dan teknik yang tertanam dalam produk tersebut. Mereka pun menjadi lebih terampil dalam menerapkan sistem yang berbasiskan atau dikembangkan dengan memanfaatkan produk tersebut. Hal ini menciptakan lebih banyak orang yang mampu menerapkan aplikasi yang kompleks memanfaatkan produk tersebut. Alhasil, pelanggan dapat menerapkan sistem dengan lebih murah karena ada persaingan di antara orang-orang yang memiliki keahlian minimum yang diperlukan. Atau, pelanggan bisa mendapatkan implementasi sistem dengan fungsionalitas yang lebih dengan harga yang sama, karena adanya source code menjadikan implementator yang paling ahli mempunyai keahlian yang lebih dibandingkan bila tidak tersedia source code.

3) Pihak ketiga bisa menemukan *bugs* dan *security flaws* serta membuat perbaikannya. Perbaikan tersebut bisa dipaketkan ke dalam produk aslinya. Dengan demikian produk

tersebut menjadi lebih bebas-bug dan lebih aman daripada bila tidak tersedia source code. Hal ini dapat meningkatkan nilai produk karena menolong pelanggan menurunkan biaya operasional dan perawatan software.

4) Penyedia software lainnya bisa menggunakan kembali source code tersebut (dengan asumsi hal ini diizinkan dan dianjurkan) untuk membuat modifikasi dan *add-on* yang menyedikan fitur-fitur tambahan yang tidak ada pada produk aslinya. Pelanggan dapat menggunakan produk-produk tersebut untuk menerapkan sistem dengan fungsionalitas lebih. Akhirnya hal ini akan meningkatkan nilai keseluruhan dari produk asli yang dianggap sebagai platform untuk membuat modifikasi dan add-on.

Berbagai metode untuk meningkatkan nilai sudah cukup familiar bagi yang sudah mengenal beberapa produk open source seperti kernel Linux, GNU C++ *compiler*, Apache *web server*. Produk-produk tersebut biasanya didistribusikan bersama dengan source code, atau tersedia source code secara terpisah yang dapat diakses oleh publik (misalnya melalui web). Mereka saling membantu menjadikan produk-produk open source sebagai produk keseluruhan (*whole product*).

Whole product dalam pengertian ini adalah: sebuah produk yang disertai dengan berbagai fasilitas tambahan yang diperlukan untuk memastikan bahwa nilai-nilai yang dirasakan pelanggan dari produk tersebut sesuai dengan yang dijanjikan oleh vendor. Fasilitas tambahan tersebut meliputi: dukungan, dokumentasi, pelatihan, konsultasi dan integrasi sistem oleh pihak ketiga, serta komunitas developer yang berkembang pesat agar produk dapat digunakan dengan cara-cara yang lebih inovatif dan menyediakan berbagai add-on. Selain itu produk dilengkapi juga dengan sekumpulan standarisasi *de jure* dan atau *de facto* sebagai basis pengembangan bagi vendor dan pihak lain.

Jika anda bersaing dengan perusahaan-perusahaan yang lebih besar, berbagai fasilitas tambahan tersebut adalah kekuatan mereka. Agar mampu dan sukses bersaing seiring berjalannya waktu, anda tidak boleh hanya menciptakan produk yang memiliki fungsionalitas yang lebih daripada pesaing anda. Melainkan anda harus menemukan cara-cara untuk mendekati atau menyamai nilai *whole product* milik pesaing anda. Cara-cara yang tidak mensyaratkan perusahaan anda menanggung seluruh beban sendirian (pada kondisi sumber daya anda tidak sama dengan yang dimiliki oleh pesaing anda).

Meski demikian, sebagaimana pada perusahaan komersial, tidak cukup hanya dengan meningkatkan nilai dari produk anda. Anda juga harus menemukan cara yang secara berkelanjutan mampu membuat para pelanggan mengembalikan nilai-nilai tersebut kepada anda dalam bentuk penghasilan dan keuntungan yang meningkat.

Sebelum mempertimbangkan cara-cara yang dapat ditempuh sebagai bagian dari strategi open source, ada baiknya me-review bagaimana perusahaan software komersial menghadapi permasalahan ini. Setelah itu, baru pikirkan apa saja yang perlu diubah pada kasus bisnis open source.