1.1. Software-based business

As of today, there is a great paradigm shift. In past decades, software was the unavoidable way to “automate business” in the logic of cost and time savings, productivity, and better quality in product and service delivery. More recently, “software became recognized not just as an automation tool but more broadly as a strategy for providing products and services not yet offered” [FAV 11]. In other words, nowadays, software is a non-removable part of products and services. Software may be embedded in a car, for instance, leading to attractive functionalities (assisted parking). Another example could be a jewel reseller who is able to provide online authenticity certificates for its products through its accession to a trustable international organization in charge of regulating such certificates (respect of laws sale tracking, etc.). In both cases, software delivers some business added value.

Companies whose primary activity is selling software are reputed to provide intangible goods [POP 11]. The distinction between these and other companies is tending to disappear. Car manufacturers of the future will thus, instead of selling “a car”, sell “a computer” and hardware/software interoperating with an engine, a chassis, an interior, a steering wheel, etc. Jewelers will probably be in a similar situation due to the irreversible interpenetration between the Internet and business activities.
The shift is the fact that the business model of “modern companies” is changing, critically relying on software. In this context, transforming car engineers into software engineers would be a huge challenge, or, in the opposite way, a very bad idea; this is the same for jewelers. So, new business models have to be invented to tame software.

From a software engineering viewpoint, we mean it is important to build software differently and beyond this to have software evolution under control because of proliferation. In this line of reasoning, most of the classical software providers still suffer from handmade practices. Introducing these practices in non-software companies might be a nightmare. Software divisions of future companies will include software builders/maintainers or not. In the negative case, at least, business analysts and innovators will constitute these divisions to offer differentiating, and thus competitive, goods and/or services. Finally, stand-alone software will no longer exist to the benefit of cooperative pervasive (more or less big) software components irrigated by the Internet.

1.2. Information-driven business

The value coming from software is the computed information. Forthcoming software-based business models must then focus on information-as-a-revenue and try to diminish the costs generated by software creation, maintenance and utilization.

Today’s entrepreneurship success is thus strongly ruled by information. As an immediate result, organizations (companies, administrations, etc.) continuously grow their dependency upon information and thus information technology (IT).

In this context, business processes increasingly rely on high-end information: undisruptive availability, liveliness, sharpness, easy digestion (even “digestibility”), rich semantics and creation of meaningful knowledge from computed information.
Business processes are powered by information systems whose criticality, optimality and dynamicity, i.e. efficiency in short, are key concerns of business analysts, software project managers, software architects and software developers. These people think about and maintain applications on a daily basis, which are edges of an ill-delimited graph, even imbroglio, of information channels (hardware and software). Over the years, nobody has the global overview of this graph. Worse still, everybody wonders why this graph does not collapse as a paper castle built from a card game. The rule of the game is now clear: the crash of the information graph is the straightforward bankruptcy of the organization.

Figure 1.1 shows a common vision of information and information systems in organizations. On the right-hand side, the computer layer not only goes on providing operating means for business automation, but it must also be a booster.
1.2.1. Adaptation to business

Information systems are an abstract view (Figure 1.1, left-hand side) of software applications and databases, middleware platforms, operating systems and related hardware (servers, mainframes, personal devices and computers) and infrastructures (power feeding, server cooling, networks, both local area networks and wide area networks, etc.). In essence, information systems constitute a logical view, which focuses on the immaterial assets of computer environments: information, its structuring, organization, production and delivery means.

As a metaphor, information systems are similar to a set of services offered by a town: municipal libraries, book loans, magazine consultation, buses, green car renting, kindergarten children’s entertainment events, etc., with related synchronicity, e.g. bus schedules fit to libraries’ working hours, children’s entertainment events, etc. In such a context, town citizens do not care about librarian and bus driver salaries, fuel in buses, libraries’ heating, etc.

In this line of reasoning, it has always been tempting, even healthy, to isolate information from its physical implementation. This approach aims at better considering information-as-a-service. Instinctively, information consumers do not pay attention to computing environments being hardware or software.

Designers of information systems thus have the permanent difficulty of guaranteeing and maintaining high-quality services wrapping information processing. The difficulty mainly lies in hiding intrinsic problems from piled (hardware and software) layers and recurrent failures. As an analogy, a bus drivers’ strike would probably diminish the quality of the town’s services to citizens.

For a long time, the ideas of architecture and urbanization have taken a prominent place in IT. It is important to notice that we consider architecture or urbanization of information systems in a logical way. As discussed before, information systems are mind views while in practice bits move about within circuits. Thus,
information pieces, building blocks, etc., have virtual connections, links, etc., whose awareness is a key aspect of information management at large. Architecture is related to software that powers information systems, while urbanization is a macroscopic wrapper including information channels, forms, circulation, restitution, etc. Both urbanization and architecture act as a basis for, respectively, information systems and applications. Cartographic representations of these (sample in Figure 1.2) can be made more or less explicit, depending on their rational nature. Rationality aims in essence at controlling useless complexity.

As an analogy, Figure 1.2 shows the case of tourist flow management for the Eiffel Tower. Urbanization (right-hand side, top of Figure 1.2) copes with transportation infrastructure in connection with tourist visit routes and coarse-grained throughputs. Architecture (right-hand side, bottom of Figure 1.2) is concerned with “solutions” (e.g. signage). Information boards about visit routes are components of a chosen architecture to perform tourist flow management “at runtime”. Services rely on components, for instance, displaying on
boards the next times of bus, boat, subway, etc., arrivals at closer transportation stations.

Gradually, information system designers face newer challenges. While architectures take time to become optimal, nowadays, they are expected to be/become variation-prone. Today’s economical contexts (globalization, trend reversals boosted by the Internet and consumers’ zapping) call for changing business: practices and processes at the organizational level. At the underlying level, information and logic engraved in information systems are surely subject to modifications as well. While organizations may require business practices and processes to rapidly adapt, information systems do not have the same latency. Re-architecturing is above all an offline activity. Fortunately, not all business adaptations involve re-architecturing, but information systems must be thought by designers to cushion business shocks: that is the new deal.

Returning to the case of tourist flow management for the Eiffel Tower, re-architecturing could be the review of the existing information systems for dealing with sporadic phenomena (e.g. cold/heat waves) or frequent events (e.g. sport shows), which may increase or decrease the presence of tourists. The case of a heat wave may, for instance, call for fit-like-a-glove services: boat traffic and arrivals to the Seine river embankments have increase to allow people to refresh themselves on the water when departing or arriving. In other words, customers will prefer boats to the detriment of subways, buses, etc.

So, nowadays, since architecture variability cannot be ignored, information systems should gain more flexibility. Typically, architecture components must, on demand, collaborate in a different way and/or extend collaborations with third-party components often unknown at design time. As mentioned in the introduction, service-oriented architecture (SOA) is a solution principle, but a lot of progress is expected in this research field.

In [BAT 14], it is especially revealed that acting on architectures is often infeasible due to excessive complexity. The difficulty to sort out
business logic from this complexity is high. Instability of architectures (the contrary of variation-prone) is thus the phenomenon when interventions in architectures’ inner workings generate long periods before recovering stability.

As an overview, business pressure is such that information systems must demonstrate a kind of real-time evolvability. In this scenario, attenuating the adherence between information systems (as the immaterial value of organizations) and computer facilities (both hardware and software) seems to be a perpetually renewed challenge. The well-known weakness of information systems is their poor reactivity in terms of requirement adaptation while, in contrast, today’s business is subject to very frequent variations, even shocks. In other words, long-term strategies related to information management poorly comply with volatile short-term business activities.

1.3. The case of tourism industry

The sector of tourism is indicative of the increasing and inescapable intertwining between IT and business. Gallo and Krupka in [GAL 08] argue “(...) travel companies will face the need to introduce in-depth changes to their business strategies in order to adapt to the changes affecting their customers. (...) The development of new products and services and the adaptation of the offer to global customer trends require a great deal of innovation” (emphasis ours). In reality, as in many other sectors, tourism to a great extent relies on IT to support this innovation. Nonetheless, IT can also be a source of possible setback when companies are slowed down by rigid information systems.

As an illustration, Figure 1.3 shows what might be an economical process whose aim is the customization of travel offers for new customer profiles, namely singles. Invariably, the creation of new business services leads to new software services (and their tricky connection with what is existing). At the bottom of Figure 1.3, software evolution is caught in a cost vise. Two contradicting requirement streams drive changes: innovation in scope and daily
business. Experience especially shows that change implementation is a source of regression. Namely, one may observe what follows: what works perfectly at a given time after months of effort can spontaneously become out of order. As an illustration, the addition of new services for singles is both an extension and a modification (coupling with the existing architecture’s components). To get the job done well, modification may call for “adaptation” in existing components. Afterward, these do not serve the daily business (unexpected failures) while they did before. The expected innovation and its associated revenues may then be significantly penalized by the impossibility of driving software evolution in a timely manner under controlled costs.

Figure 1.3. IT and software evolution positioning in the fluctuating tourism industry
In all business sectors, in people’s minds, IT is often rightly considered as an aspirator of financial resources. That is true when IT is no longer observed as a business developer. Moreover, people outside the IT world do not understand why IT is costly (a euphemism) while, *par excellence*, it is the technical field where competition is fiercer, innovations are bigger and, accordingly, costs linked to hardware/software parts (e.g. open-source software libraries) are increasingly lower.

Regarding the tourism industry, it should benefit from both the Internet (as an ever unbound marketplace) and IT advances, which together reshape the Internet-based possibilities of doing business. Nonetheless, over the years, the tourism industry has been unsettled by the Internet, which created an excessive, even confusing, offer with an exacerbated competition. In fact, the globalization of tourism business diminishes sales margin, relying on adaptive information systems not to miss pioneering revenue opportunities.

New players, new deals, new rules of the game, etc., appear in quasi-real-time. The paradox is that IT makes possible this liveliness, while software applications must accordingly behave differently to cushion new business events. Ultimately, this leads us to ask developers to change code and in the worst case to reformat software architectures. The latter is both a source of stress and risk and, unfortunately, software crash before reaching a new stable situation, which, in turn, does not meet the very last business expectations. This infernal circle can only be broken with flexible software frameworks.

Tourism players, such as hotel chains, tour operators, tourism agencies/organisms/consortia, transporters and car rental companies, are involved in both business to customer (B2C) and business to business (B2B) commerce. For instance, hotel chains may buy excursions from tour operators while the latter buy bedrooms from these chains.

New players are, for instance, health centers because a confirming trend is the fact that customers associate travels with the possibility of care: dental care, plastic surgery, fitness, etc. Another trend is the possibility of collaborating with real estate agents, which can supply
different kinds of accommodation, and thus multiply the types of lodging on offer.

New deals can be joint and/or bulk purchasing, subcontracting, product/service sharing, partnership with price comparison Websites, etc.

The new rules of the game are, for instance, the fact that end customers include implicit concerns when ordering travels. These are security, sustainability, privacy, responsible tourism, etc. In the best case, such values might be transformed into paying services, which probably require collaboration with specialists. In the worst case, these values may be in contradiction to cheap offers.

Intuitively, from a software viewpoint, it turns out that, a minima, tourism applications must be able to exchange data. Beyond this, we may simply imagine, for example, the connection between a health care center software and a travel management platform to book and arrange care stays within touristic stays. This link is similar to service interoperation between travel and dating Websites in Figure 1.3. Each business adaptation case would probably lead to a specific software technical problem. Reasoning case-by-case results in numerous induced problems whose piling is inevitable and resolution is very long.

For software experts, SOA, later discussed in this book, is an appropriate approach for organizing software so that interoperability succeeds beyond data: applications may evolve incrementally through new services (i.e. functionalities) and/or new service composition. In the case of the mentioned travel management platforming, we should have the possibility of easily, straightforwardly and transparently calling for services accessible from the healthcare center software, provided that the latter has been thought up, designed and equipped with interoperability abilities, say, secure Web services since medical data require more privacy.

Beyond the excitement provided by the Internet, there is an actual potentiality for IT to favor reactivity in business. More generally, IT
and information systems must be the springboard for business adaptation in shorter and shorter cycles.

In the common business-oriented language (COBOL) world, this vision is a myth. In the Internet and cloud computing worlds, using Java platforms/technologies in particular, SOA is a technical reality. Nonetheless, from a business perspective, SOA often remains a (later reachable) goal: no company, in the tourism sector in particular, has developed such appealing adaptation capabilities to absorb very high business fluctuations. In very rare cases, only software aims at changing. In effect, business processes around applications also have to mutate in involving users differently, modifying usages (roles, tasks, documents, frequencies, etc.). Mutations generate natural inertia, which is most of the time incompatible with the time slots required to have software applications that instantly suit requirement fluctuations.

So, software modernization, with a focus on COBOL, is not only a technical issue to be addressed. There is a crucial need to have enough reactivity in business processes, information systems and software applications/components that simply help rapid development/maintenance. There is a challenge in moving legacy systems to renewed ones. This challenge especially amounts to, as much as possible, separating business concerns from technical constraints.

More generally, the top of Figure 1.4 shows that IT may sometimes be a hindrance when rigidity in information systems prevents any kind of adaptation. Beyond technical issues and the particular case of COBOL, the idea of software modernization is thus the progressive erasing of such rigidity.

1.4. IT progress acceleration

Theoretically, IT progresses are the source of inexorable improvements in the functioning of information systems. Empirically, this position statement is false. It turns out that the migration of any information system, or information systems part (applications, components, services, etc.) from one “legacy” technology to a
“modern” technology, may be, without experience feedback and expertise, a nightmare, never mind the costs.

Broadly speaking, high tech may be viewed as a lure promoted by “evangelists” who never, in the past, present and future, used/use/will experience the high tech they have built and promote aloud. Lies about high tech are in essence its masked inefficiency due to its (natural) intrinsic lack of maturity, weak testing, poor adoption and small-scale utilization. High tech with maturity, representative experimentations, rich feedbacks and lessons learned, etc., is actually no longer high tech.

From a business viewpoint, high tech is of little interest if it does not address business issues. Results are not necessarily immediate and tangible. If they are deferred, the high tech implementation method must, however, provide guarantees in time; tangible progresses through returns on investment especially must occur after a certain period of time. Observable cases are rare; they are often accompanied by debatable numerical data and statistics. However, there is a poor communication on absolute failures whose counting and deep analysis is thus illusory. This context does not favor experience exchanges in an impartial manner and contributes to freeze the opinion of legacy people: they are still negative about software (unjustified) “sophistication” at large.

Figure 1.4. IT may be both a business stimulator and a brake
For COBOL professionals, model-driven development (MDD), SOA, agile software development, cloud computing, etc., may thus be considered as high tech whose maturity needs to be proved first. The specificity of business sectors, companies and well-isolated activities may also be an argument against going out of legacy contexts. The adhesion of people is the rule. Feasibility surveys and studies are helpful; proofs of concept are essential to convince COBOL people. High tech inventors are respectable in their recognized role for IT progress acceleration, but they generally have a tight vision on business.

Roughly speaking, there are at least two very different coarse-grained categories of computer professionals: business application builders and generic software providers. The latter develop open-source software, commercial off-the-shelf (COTS) products, etc., and act as suppliers for the former. The former meet end-users that are difficult to synchronize with domain requirements. The main task of application builders is to push and acquire information to and from the reality of organizations. This task does leave time for integrating new software technologies in application development frameworks: for instance, switching to an object-oriented programming language, adopting and setting up an agile software development method, etc. So, pragmatism is as follows: high tech has to be thought like any science contribution: effectively shared and beneficial for humanity; if not it will be forgotten or postponed to the next century.

1.5. Legacy world

The legacy (software) world is the sum of legacy technologies, legacy information systems, legacy applications and “legacy people”, anything apparently aged but still delivering the expected business services in time and quality – is this a paradox? Not really… Behind the “legacy” term is probably a lot of expertise, long experience, background and wisdom. Briefly, summarizing “legacy” as something pejorative is often misplaced.

In [BAT 14], there is a very recent interesting summary on interviews of IT practitioners about their own perception of legacy
software systems and their possible modernization. Recognized qualities are: “(76.7%) business-critical, (52.8%) proven technology, (52.3%) reliable system and (24.4%) performance”. This directly confirms the idea that “legacy” conveys positive values. In contrast, it is concomitantly agreed that strong factors impose modernization: “(1) high maintenance costs, (2) lack of knowledge, (3) to remain agile to change and (4) prone to failures”. Discerning readers may detect questioning contradictions in this survey, for instance, how a legacy system may at the same time be a “reliable system” and “prone to failures”? In fact, legacy systems are diverse in nature: people share common characteristics like “something aged”, but they may disagree about criteria like “reliability” above.

In this line of reasoning, the qualification of a software system as “legacy” is not systematically linked to a programming language such as COBOL: “more than half of the informants do not agree that the programming language is a determining factor for a system to be legacy, while the rest were in agreement” [BAT 14]. However, half of the respondents’ legacy systems are known to be built on top of COBOL.

In [NAS 08, p. 6], another survey on legacy systems is given: “inability to be adequately supported, maintained, or enhanced” (82.8% of interviewees) is the premium discriminating criterion for “legacy”.

Evolvability, or more precisely, the proven absence of this potentiality is, on the spot, what better qualifies “legacy”. In relation to our prior analysis on the new business deal, the question is: “why such a significant concern about evolvability?”. The answer that comes is the same: “inability to meet business needs or system not agile enough to continually meet the challenging needs of the organization” [NAS 08, p. 6] (79.3% of interviewees). This criterion is not independent of the first criterion, since evolvability mainly results from business need fluctuations. The interesting word in the previous text extract is “continually”. IT practitioners no longer view maintenance as discrete, but as inevitably continuous. Driving changes without break periods has definitely become “the job”.
1.5.1. Exiting the legacy world

In fact, economical considerations prevail over technical ones; this is mostly true because the latter are induced from the former. More precisely, the ratio between the immaterial value of information systems and the cost of ownership and the technical debt associated with these systems is a balance indicator. The worry factor, i.e. decrease in this ratio, is the door to software modernization.

As discussed previously, information systems are the core source of information-as-a-revenue, but they may behave like an old car whose oil consumption is no longer consistent with the essential services to be delivered: transport from point to point. In this metaphor, using a public bus is similar to replacing a legacy software system by a COTS software package.

So, exiting the legacy world is just a breaking point with respect to the inadmissible deviation of financial indicators. Software modernization then becomes an actual concern before being an obsession. Nonetheless, in all surveys, people never claimed that filling the gap with the newest technologies is the motivation behind software modernization.

Strategically, organizations want to avoid technological silos like developing solutions based on an evident isolation with the Internet, cloud computing, etc. Beyond this, using these newer technologies may be unacceptable, particularly because of entry costs. In [NAS 08, p. 12], “funding” has rank 5 (in a scale of 1–5 with 1 being “not challenging” and 5 being “extremely challenging”). This criterion is recognized as the first major obstacle for modernization. In these times of crisis, budget constraints drastically limit the spectrum of candidate methods for modernization.

Returning to the metaphor of the energy-consuming car, modernization is not just the replacement without awareness of the old car by a cost-saving car. In other words, a smooth ride, for example, can be an existing practice to be kept. More generally, old car usages are probably associated with the best cost-saving practices: best-
known circuits, shortcuts, car sharing, etc. We mean the immaterial value of information systems is nothing but the business value buried in computer memories and storages (information), as well as programs (functions and rules, logic in short). Only modernization methods based on a solid extraction and an intelligible reconsolidation of this business value make sense. This approach may attenuate the “funding” disease through the fact that modernization is first and foremost porting business intelligence from one target to another.

In this spirit, high tech and legacy technologies are not opponents. They are just different means, whose appropriateness is strictly linked to different time slots. Accordingly, we may then write that any high tech is the legacy technology of tomorrow. This strongly confirms that modernization methods cannot be proposed in terms of point-to-point technology mapping and transfer. The consistent and complete expression of legacy systems, once ported, independent of new technologies, is thus so vital.

1.5.2. Legacy world professionals

IT is strongly characterized by mutations, which apparently and permanently call for “people brain updates” in terms of acquired knowledge, technology comprehension and so on. There is, understandably, a natural reluctance to follow up these mutation cycles, which are numerous, frequent, but sometimes volatile and unjustified. Worse, they are sometimes just hype. Being open-minded must a priori be the rule in IT, but experience shows that most of the worldwide software development stakeholders have no professional time to devote to IT news, in terms of knowledge enhancing especially. Beyond this, the volume of technology releases (products, versions, application programming interface (API), standards, even paradigms, etc.) is simply too huge.

We cannot, without any nuance, talk about the inability of people to apprehend technology jumps; it is certainly only a matter of time. In this context, software modernization is either an opportunity to invest in new knowledge or it may be viewed as the end of
“tranquility”. As mentioned above, people share the positive opinion on legacy software systems as being “reliable”. Behind the “reliable” word is the fact that any long professional life with only one line-of-product concern (e.g. COBOL) is a sure way to converge to “reliable” systems, i.e. we must read here: “systems with full controllability”. More generally, technology capitalization contradicts high tech. The source of stability and full control of information systems relies on keeping old technologies under long-term utilization despite the fashions.

Software modernization is in essence the moment that has been pushed away for a long time. Technology jumps in non-chosen moments are then problematic because they are human-centric. Several psychological and cultural barriers may strongly slow down the process to move forward. By translating software to ill-known technological targets, feelings such as creativity vanishing, loss of control, being software robots, etc., may increase for individuals or groups.

There is another source of trouble in IT. The persisting craftsmanship in IT is the consequence (or perhaps the cause) of the not-invented-here (NIH) syndrome. In effect, software reuse might be considered as a semi-failure at the beginning of the 2010s, while the origin of the software crisis was put forward in the 1960s. People persist in considering that they build so-specific software. Any the software from outside is, in this scenario, suspicious. This is both true and false. This is false because, as a counterexample, any new employee after learning periods must be able to play a significant role in existing software evolution. This is true because the proximity with end-users is irreplaceable. To that extent, software outsourcing is nowadays identified for certain types of software only.

Another key human factor of software modernization is the “graying” of IT staff. Employee retirement is an everyday event in organizations. A driver for software modernization is then often this human factor. However, beyond the loss of human (technical) resources is the loss of business intelligence. Indeed, computing wrongly remains a technical discipline omitting the raison d’être of an information system as being the nervous system of organizations.
Organization management is preponderant. As already discussed, the careers of IT people close to retirement are almost always based on an economical background. Software modernization in this scenario is the true opportunity to mine this business intelligence before retirement.

1.6. Conclusions

A justified criticism against IT is the fact that it was created to assist organizations in management and business, but in increasing the number of applications, information systems tend to become incoercible. We mean, in an organization, IT components (hardware + software) not only become more complex but also tend to exist to only feed each other. Keeping IT and business converging is an everyday battle, which calls for more and more effort, means and money. IT people are skeptical about newer technologies because they do not actually deliver what they promise. Non-IT people do not understand why previous important investments in computing infrastructures do not solve problems in a timely manner. These people only want to relate to information systems from the surface. On the opposite side, IT people cannot easily argue that information systems’ inner workings are very difficult to monitor and manage.

Honestly speaking, IT people are overwhelmed. They cannot step back. They suffer IT. Concretely, in COBOL for instance, programs come from nowhere. Technically, each appears as yet-another-retaining-wall. From the business viewpoint, over the years, IT components have begun to look like patches whose direct positive impact on the business is often imperceptible. In this context, legacy information systems are naturally guilty. Why then modernize with the risk of standing still?

This depressing vision contradicts news in IT magazines, Web blogs, great-fanfare announcements, success stories, etc. Indeed, this chapter shows that a new deal may exist: software is no longer the means for information processing; it is the source of extended business through the idea of information-as-a-revenue and that of
service in SOA. With the Internet’s unfinished culmination, there is indisputably a paradigm switch. Software and information are no longer only helpers or boosters; they are “the value”. Precisely, services as consumer goods and services as software artifacts become increasingly less distinctive. Intentionally, SOA and the Cloud are the up-to-date software supports to favor such a convergence. The case of the travel industry is representative through the endless opportunity to develop and sell new services. In such a revolution, software is componentized; components are business-related and pervasive including high availability and strong dependability.

Because business without people is meaningless, this chapter also mentions that revolutions, even though technological, cannot ignore people’s aspirations, cultures, experience, know-how, etc. COBOL software modernization arises in line with this healthy observation.