Chapter 1
Introduction to Business Rules

Target audience

- *All*

In this chapter you learn

- *What are business rules*
- *What are the motivations behind the business rules approach*
- *In what ways do business applications with business rules differ from traditional applications*
- *Why do we need a different development methodology*

Key points

- *A business rule is a statement that defines or constrains some aspect of the business. Business rules have a business motivation and an enforcement regime.*
- *The business rules approach enables, (a) a better alignment between information systems and business, and (b) a greater business agility.*
- *Business rule applications externalize business logic and separate it from the underlying computational infrastructure where it can be managed by business.*
- *Business rule development differs from traditional application development in many ways: (1) it is business requirements-centric, (2) enterprise-level ownership – and management – of business logic, and (3) business-led implementation and maintenance of business logic.*

1.1 What Are Business Rules?

An on-line store might not accept a next-day delivery order if the order is received after 3:00 p.m.

J. Boyer and H. Mili, *Agile Business Rule Development*,
DOI 10.1007/978-3-642-19041-4_1, © Springer-Verlag Berlin Heidelberg 2011

My bank will not lend me money if my debt-over-income ratio[1] exceeds 37%

Section 152 of the US tax code defines a dependent as a person who is either a "qualifying child" or a "qualifying relative." A taxpayer's qualifying child for any taxable year is a person who:

- Is the taxpayer's child, sibling, step-sibling, or a descendant of any such relative
- Has the same principal residence as the taxpayer for at least half the taxable year
- Is younger than 19 at the end of the taxable year, or is a student who is younger than 24 at the close of the year, or is a student with disability – regardless of age
- Has provided for no more than half of her or his support for the taxable year

A *qualifying relative*, on the other hand.

My health insurance does not reimburse medical expenses incurred abroad if the claim is presented more than 1 year after the expenses had been incurred, or if the claimant has spent more than 182 days abroad within the past year.

Passengers with frequent flyer status Silver, Gold, Platinum, Super Platinum, and Super Elite Platinum may board at their leisure.

My car insurance does not cover drivers who have been convicted of driving while intoxicated (DWI) within the past 2 years; they are referred to a public no-fault insurance.

Fannie Mae will only underwrite mortgages on properties that have hazards insurance that protects against loss or damage from fire and other hazards covered by the standard extended coverage endorsement. The policy should provide for claims to be settled on a replacement cost basis. The amount of coverage should at least equal the minimum of:

- 100% of the insurable value of the improvements[2]
- The principal balance of the mortgage (as long as it exceeds the minimum amount – typically 80% – required to compensate for damage or loss on a replacement cost basis)

[1]The debt over income ratio is the ratio between total (monthly or yearly) debt obligations over gross income for the same period (monthly or yearly).

[2]For example, if a property is worth $200,000, $80,000 for land and $120,000 for the building, then the value of the improvements is $120,000.

Periodic interest payments made to the accounts of foreign entities who filed IRSform W-9 are subject to 28% backup withholding and need to be reported to the IRS in form 1099, with the box number 3 checked.

Citizens of NAFTA countries who travel into the USA by road need only show proof of citizenship.[3]

When mailing out monthly account statements, include marketing materials that match the customer profile.

Plane tickets purchased with Amex/Visa Gold/<insert your favorite card here> have built-in trip cancellation insurance.

If two alarms are issued by the same network node within 30 s of each other with the same alarm code, then group them under the same umbrella alarm.

If a wheel shows two consecutive temperature readings higher than 558°, then check for sticking brakes.

These are just a sampling of the types of rules that we have come across in our practice. Application areas include customer relationship management, marketing campaigns, the mortgage industry (retailers, mortgage insurance, secondary market), banking (credit cards, loans), car insurance, health insurance, loyalty programs, tax law, compliance, e-government, telecommunications, engineering, transportation, manufacturing, etc.

So, what *is* a *business rule*? If we break down the term "business rule" we get a *rule* of the *business*. Wordnet defines a *rule* as, among other things, "a principle or condition that customarily governs behavior," or "a prescribed guide for conduct or action." A rule of the *business* means that this principle or prescription is in the *business domain*, that is, it is part of the *requirements* (the *problem domain*), as opposed to a prescription dictated by a particular technological choice (the *solution domain*).

Business rule authors have proposed a number of definitions for business rules. Tony Morgan defines a business rule informally as "a compact statement about an aspect of the business . . . It is a constraint in the sense that a business rule lays down

[3]NAFTA: North American Free Trade Agreement, binding Canada, Mexico, and the USA.

what must or must not be the case" (Morgan 2002, p. 5). Ronald Ross defines a business rule as "a directive intended to influence or guide business behavior" (Ross 2003, p. 3). Barbara von Halle would like us to think of business rules as "the set of conditions that govern a business event so that it occurs in a way that is acceptable to the business" (von Halle 2001, p. 28).

The Object Management Group (OMG) defines a *rule* as a "proposition that is a claim of obligation or of necessity," and a *business rule* as a rule that is under business jurisdiction (OMG 2008). The *Business Rules Group*, which is an independent non-commercial peer group of business rule specialists, has produced a number of documents about the business rules approach, and has contributed to OMG's work on business process management and business rules. The Business Rules Group considers business rules from two perspectives, the business perspective, and the information systems perspective, defined as follows:

- From the business perspective: "... a business rule is guidance that there is an obligation concerning conduct, action, practice or procedure within a particular activity or sphere. Two important characteristics of a business rule: (1) there ought to be an explicit motivation for it, and (2) it should have an enforcement regime stating what the consequences would be if the rule were broken" (BRG 2008).[4]
- From the information system perspective: "... a business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure, or to control or influence the behavior of the business" (BRG 2008).

This distinction between the two perspectives is needed to account for the fact that a business process typically involves human actors and an information system, and business rules guide both. From the information system perspective, the rules talk about the data that is captured by the information system about the real world entities involved in the business process such as customers, products, or transactions. For example, in the insurance domain, a number of on-line quotation systems have three outcomes. In addition to "accept" and "decline" responses for clear-cut requests, borderline cases may receive a "manual referral" response so the request can be reviewed by a human underwriter. The human underwriter operates under a slightly different set of business rules from the ones automated in the information system. Such business rules would typically be captured in underwriting manuals.

While the bulk of this book is about the information system perspective, the early chapters address both perspectives.

Two characteristics of business rules stand out from the above definitions: (1) business rules are about business, and (2) business rules concern both the *structure* and the *behavior* of the business. We will elaborate these two characteristics further below.

[4]The Business Rule Group web site: http://www.businessrulesgroup.org/defnbrg.shtml.

1.1.1 Business Rules Are About the Business

Indeed, in the examples given, there is a *business motivation* behind the rule. To illustrate this point, consider our first rule about next-day delivery and the 3:00 p.m. deadline. Why would an on-line store put in place such a restrictive rule, and risk losing business as a consequence? A plausible justification could be that it *may* take more than 4 h to, (a) find a free warehouse clerk to fulfill the order, and for the assigned warehouse clerk to (b) locate the book in the warehouse, (c) prepare a package for delivery, and (d) deliver the package to the nearest Federal Express or UPS branch. Notice that the same *rule* would apply if the customer called by *phone* to place the order. Similarly, the rule about rejecting drivers with recent DUI convictions: the obvious business motivation is that such drivers present a high risk of causing accidents, and would cost the insurance too much money.

Von Halle says that "business rules are the ultimate levers with which business management is able to guide and control the business. In fact, the business's rules are the means by which an organization implements competitive strategy, promotes policy, and complies with legal obligations" (von Halle 2006). The Business Rules Group (BRG) has proposed a *Business Motivation Model* that attempts to formalize the link between business rules and business objectives (BRG 2007); the OMG's *Business Motivation Model Specification* is based on (BRG 2007). Roughly speaking, business rules are seen within the context of business plans: a business plan includes *ends* (business objectives) and *means* to achieve the *ends*. Business rules are part of the *means* that businesses deploy to achieve their goals (profitability, market share, customer loyalty, etc.); we will say more about the business motivation model in Chap. 4.

1.1.2 Business Rules Concern Both the Structure and the Behavior of the Business

This distinction is evident in the information systems perspective of the *business rules group* definition, and somewhat in the OMG definition, which distinguishes between *structural* or *definitional rules* and *operative* or *behavioral rules*. Roughly speaking, *structural rules* define the business information model. The statement "a sale record includes the buyer, the product, the quantity, the price, and any applicable discount" is a structural business rule. We can think of it as the definition of the **Sale** entity (or class). Similarly, the statement "an order can include one or several line items, one per product, indicating number of units and price" is also a structural business rule, which can be seen as defining the **Order** entity. A *behavioral rule*, on the other hand, is about how the business reacts to business events. Most of the example rules shown above are actually *behavioral rules*. The first rule (3:00 p.m. deadline) is relevant to *order entry*. The debt-over-income ratio is about loan application underwriting. The health insurance rule is relevant to the

processing of claims. And so forth. Generally speaking, *behavioral rules* kick in when something happens at the boundaries of the system. This distinction and others are described in more detail in Chap. 4.

1.2 Motivations for the Business Rules Approach

Before we talk about the business rules approach, let us talk about the "nonbusiness rules approach."

The sample of rules shown above has, *for the most part*, been successfully implemented in working information systems by people who have never heard of the business rules approach. So what is the hoopla about the business rules approach?

The next few real-life examples will illustrate three *major* issues that are adequately addressed by the business rules approach. We will present the examples first, and then identify the dominant issues:

- A company is in the natural gas business. It sells natural gas to public utilities. It draw 8–9 figure contracts with these public utilities, whose prices depend on the total volume (a certain volume of natural gas over the duration of the contract), throughput (a certain volume per hour), options to request a 10% (or 15% or 20%) increase of throughput within 6 h to accommodate consumption peaks, the possibility of storing the gas for low usage periods, etc. Beyond the raw volume (x cubic tons of gas), each one of these "options" has an infrastructure cost – and thus a price associated with it. The company's top management looks at the yearly numbers and figures two things: (1) given the volume that it sells, it should be making more money, and (2) overall, its customers are having a good deal, relative to the competition, and some customers have *very* good deals, but neither the company nor its lucky customers know it. We need to capture those pricing rules precisely so that (1) we can fine-tune the rules to make more money and yet remain competitive and (2) we can tell customers, precisely, how good a deal they are getting. As it turned out, those pricing rules walked out the door every day between 4:00 and 7:00 p.m., got stuck in traffic on most days, and called in sick some of the time – not to mention the occasional vacation. Not only that, but they took on separate lives in separate spreadsheets on the contract officers' laptops.
- A US state manages a number of social benefits (welfare) programs for people with disabilities, senior people, low-income people, single mothers, back-to-school single mothers, back-to-work programs for long-term unemployed people, food stamps, etc. Each one of these programs has eligibility guidelines, the contours of which have been defined by the laws that created those programs. Applications to the various programs are dispatched to "case workers" who assess the eligibility of the applicants and determine the benefits level. Case workers were overwhelmed, and their determinations were uncomfortably inconsistent.

Managers asked a couple of questions: (1) exactly *what* rules were being used, (2) how to ensure that those rules are used *consistently*, and (3) why processing times for straightforward cases were the same as for complex borderline cases.

These were but two of many examples of organizations that did not know precisely the rules under which they were operating, and consequently, operated under different – and often conflicting – sets of rules. Hence:

Issue 1: Organizations need to know which business rules they are using, and whether they are using them consistently.

- A phone company's core business is local phone service. The company was getting in the long-distance service. The local *public utility commission*[5] wants to ensure that phone companies with a monopoly on local phone service offer the same quality of service between customers who use them for long-distance service, and customers who use other carriers. Thus, "our" phone company has to file a report every month that shows quality of service statistics for *its* long-distance customers, and for the long-distance customers of other carriers. Because heavy penalties are levied when statistics show that the company gives preferential treatment to its long-distance customers,[6] an important part of the report filed with the PUC is the method of calculation. And, in the case of audit, our phone company has to be able to show that it *has, indeed, used those calculations to produce the report*.
- The *n*th user acceptance testing postmortem meeting. The customer complaints: "the system still does not do what it is supposed to." Technical lead: "Perhaps not, but it does what you told us to do." The customer: "I never told you to underwrite loans for customers with FICO score lower than 600." Technical lead: "You never told us the contrary either: you said underwriting decisions are based on our risk assessment score, not on FICO score alone." Customer: "yeah, but isn't the FICO score a big component of the risk assessment score." Technical lead, getting tired with all this fuzziness: "Define big." Customer: "Well, big as in 80%, perhaps more?" Technical lead turns to developer, whispers something, developer opens Eclipse on his laptop, and starts looking frantically through code, then his face illuminates: "well, we have it set at 90%." Customer, after doing calculations by hand, is adamant now: "Can't be! Show me." Developer looks at technical lead for a cue, and technical lead responds:

[5]In the USA, Public Utility Commissions (PUCs) are statewide regulatory commissions with a mandate to balance the needs of consumers and utilities (electricity, natural gas, water, telecommunications, etc.) to ensure safe and reliable utility service at reasonable, competitive rates.

[6]For example, both Jane and Joe have their local service with our company – they have no choice – but Jane chose our company for long-distance service, whereas Joe chose a competitor. If both Jane and Joe make a service call, say to report a problem with the line, the PUC wants to know if Joe's calls are handled as diligently as Jane's (how fast it takes customer service reps to get back to Jane vs. Joe, how many calls it takes to resolve the issue, what is the elapsed time between opening the case and closing it, etc.).

"Show them the code!" The developer starts looking for a cable to connect his laptop to the overhead projector. He does not find one, walks out of the room. The project manager, who called the meeting, asks "do we have to do this now? Because we have ..." The technical lead and customer answer emphatically: "Yes!" The developer comes back with a cable, and puts up the method `addFactor` from the prosaically named **`RAStrategyDataProxy`** class on the screen:

```
public void addFactor(float v, HashMap<Interval,Float>  penalties) {
        Iterator<Interval> intervals = penalties.keys();
        float pen = 0;
        while (intervals.hasNext()) {
                Interval next = intervals.next();
                if (next.contains(v)) {
                        pen = (penalties.get(next)).floatValue();
                        break;
                }
        }
        raScore = WEIGHT* raScore + (1-WEIGHT)*pen;
  }
```

The technical lead is happy with how intimidating this must look to the customer, and looks at her defiantly, as if taunting her "Ok, so what are you going to do with it?" The customer, unfazed, wastes no time throwing the curve ball back at him: "Don't look at me like that! Translate!"

Now it is *his* problem again: explain classes, methods, generics, hashmaps, and iterators to a business person! Luckily, this business person is a very smart lady who was once a programmer ... 30 years ago ... in COBOL. Lo and behold, after explanations about what the penalties hashmap represents, and through many detours through the code, for example, to find where the constant `WEIGHT` is defined, what `raScore` means, how it is initialized, and how it gets updated, they actually find the bug. True, `WEIGHT` is set to 90%, and the risk assessment score is initialized to the FICO score, but each time a new factor is taken into account, the underlying weight of the FICO score is actually decreased by 10%. This explains the discrepancy between the customer's hand calculations and the output of the program. It is 6:30 p.m., the tension has subsided, the meeting is finished, and as everybody walks out, the project manager sighs "There's gotta be a better way!"

This story ended well because the customer was smart, stubborn, no pushover, and was once a programmer. How many business customers are like that? Further, in this case, we were able to pull out a single Java method that enforces the business rule, and inspect it. We are seldom that lucky. Indeed, the business logic will often be scattered in many places: context-sensitive interaction screens based on customer profile or location, configuration data in external files, limited validation

functionality in input screens, control logic in functions, database integrity constraints, SQL code, and the nightmarish stored procedures. Hence:

Issue 2: Organizations need to describe the business rules that are embodied in their information systems in a way that all stakeholders can understand, and need a way of ensuring traceability between those rule descriptions and the actual implementations of the rules.

- An insurance company sells all kinds of policies to individuals and corporations. Its marketing department regularly evaluates its underwriting rules to assess the profitability of the various market segments. For example, assume that the insurance company covers drivers who are as young as 18 years old. Given that young drivers are more accident prone, one may ask whether the 18- to 19-year-old market segment makes money for the insurance. To this end, the marketing department compares the total claims paid out in the past 6 months, on policies held by drivers between the ages of 18 and 19, to the total value of premiums collected for that market segment. If the company collects more in premiums than it pays out in claims, then that market segment is cost effective. Else, it needs to make its rules more stringent to weed out the statistically losing market segment. All is good. The marketing department performs these simulations every month, on the data for the previous 6 months, and makes recommendations for new underwriting rules. IT takes a minimum of 4 months to implement such changes with the current technology. Hence, the company cannot react as rapidly to changing market conditions. Its reaction is always 4 months behind, and when IT is doing the final testing, everyone knows that the rules that are being tested are already 3 months obsolete.
- The mortgage division of a financial services and insurance company has reacted quickly to the sub-prime mortgage market crisis by tightening the eligibility requirements for mortgages as soon as the first signs of the crisis started showing on the radar, that is, in the late spring of 2007. By mid-July, new eligibility requirements were published internally and sent out to retail branches. By late fall, the online mortgage application system was still using the old eligibility criteria. Potential customers with shaky credit, who had been hearing about tightening credit from the 6 o'clock news, started believing in Santa Claus when the online system replied "Congratulations. Your application has been pre-qualified. A mortgage specialist will be in touch with you soon." Which specialist sometimes had the un-CRM task of calling the customer to say "we apologize: our on-line system still operates under the old eligibility rules." Not cool.
- An investment company buys and sells (trades) securities on behalf of its customers. For each trade, it chooses the best exchange market on which to execute the trade based on (1) the types of security (bonds, equities, etc.), (2) the actual security (e.g., Microsoft stock), (3) the volume (e.g., ten versus ten million), (4) the commission charged by the exchange market on such trades, (5) any contractual agreements between the investment company and the exchange, (6) any contractual agreements between the exchange and the

customer on behalf of which the trade is being made, and (7) the market conditions. Trade execution routing is automated through an application. The investment company would like the application to be responsive to changes in the various factors. However, the frequency of these changes goes from once in a lifetime (e.g., the emergence of a new exchange market or of a brokerage house) to the minute (market conditions), to anything in-between (weekly, monthly, etc.).

These are just three real-life examples of situations where the IT infrastructure of a company becomes an *impediment* to evolution, as opposed to an *enabler*. Hence:

Issue 3: Organizations need an agile development infrastructure/paradigm that enables them to react to the changing environment in a timely manner.

Having accepted that business rules should, and do, for the most part,drive our business information systems (Sect. 1.1), the several *real* examples showed a number of problems with the way business rules are typically implemented – or not, as for the case of the natural gas company – in information systems. The *business rules approach* addresses all of these problems. So what is it? Barbara von Halle defines the business rules approach as "a formal way of managing and automating an organization's business rules so that the business behaves and evolves as its leaders intended" (von Halle 2001). We like this definition because we feel that it captures the essence of the business rules approach in a single sentence:

- It is *a formal approach*: This means clearly defined processes, tasks, roles, and work products, that is, a methodology.
- *Managing* and *automating* business rules: Management and automation are related but separate concerns. *Management* includes collecting, recording, validating (for accuracy), assessing (for business worth), publishing, and evolving the business rules. This needs to be done – and can be done – whether those business rules are automated or not: as our natural gas supplier example showed, important rules of the business were not defined precisely and consistently across the enterprise. As for *rule automation*, it means making those rules *operational*, that is, come up with a <language, interpreter> pair so that enterprise applications can reference them.
- [The business] *behaves* and *evolves* as [. . .] *intended*: As our mortgage underwriting example duel between business and IT showed, language barriers between business and IT can make the first goal – *behave* as *intended* – difficult to achieve, and equally difficult to verify. As the last three examples showed, traditional development techniques *cannot* possibly meet the pace of change of the business environment.

We can think of this definition as a set of *requirements*. In the next section, we look at how typical *implementations* of the business rules approach look like.

1.3 How Do Business Rule Applications Differ from Traditional Business Applications?

What does a business application developed with the business rules approach look like? We know how a business rule application *should not* look like: it should *not* look like the rule-based systems that were developed in the 1980s: (1) custom (*from the ground up*) development methodologies with esoteric terminologies; (2) their own programming language – or at least one not used in business applications; (3) their own data storage (persistence) mechanisms; (4) poor scalability; and (5) little or no connectivity to any of the existing business systems. No wonder the technology failed to penetrate business information systems back then!

To understand what business applications developed under the business rules approach look like, we have to understand what the business rule approach entails. A *full implementation* of the business rules approach has three components:

1. A methodology for rule management, that is, collecting, recording, validating, assessing, publishing, and evolving the business rules
2. One or several more or less formal languages for expressing business rules at different stages of their life cycle and for different audiences (business, IT, and computer)
3. A tool set for managing and executing the rules, a *Business Rule Management System* (BRMS)

The three components are interrelated:

- The BRMS supports the methodology to various degrees through a shared repository for rule artifacts, workflow/process management functionalities, an enforcement of roles through access control, and so forth.
- The management functionalities of the BRMS support the creation and modification of rules expressed in the rule languages, and the translation of rules between the various languages.
- The rule automation (execution) functionalities of the BRMS support the execution of rules in one or several of the supported rule languages.

Some authors consider the provision of an executable rule language, as distinct from the application programming language, and the provision of rule execution functionalities by the BRMS as a highly desirable but not a necessary aspect of the business rules approach. We agree that it is highly desirable, and if we consider agility as an essential aspect of the business rules approach, then we will have to consider it *necessary*.

Figure 1.1 shows the three components of a *business rules approach implementation* and their dependencies. Part II of this book (Chaps. 3, 4, and 5) will deal with process. We introduce BRMS in general and JRules in particular, in Part III (Chaps. 6, 7, and 8). Rule authoring and rule languages are discussed in Part IV (Chaps. 9, 10, and 11). Rule execution is discussed in Part V (deployment, Chaps. 12 and 13)

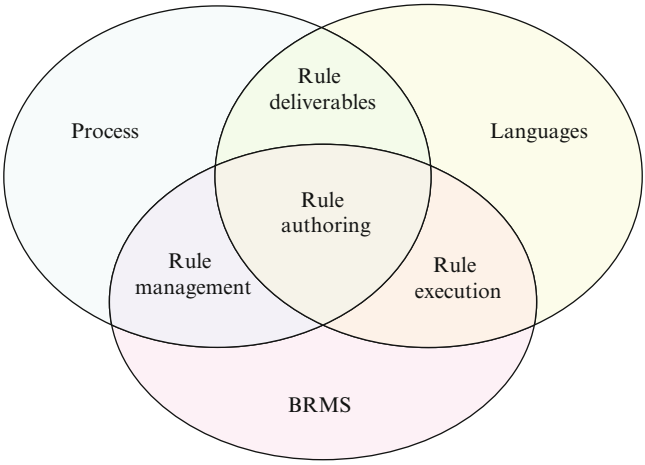

Fig. 1.1 The three components of a business rules approach and their interrelationships

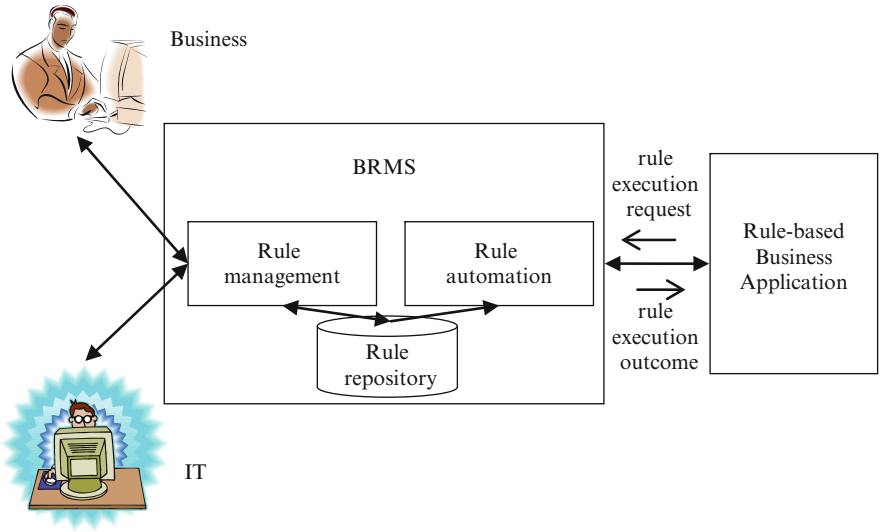

Fig. 1.2 The role of a BRMS in a business rule implementation

and Part VI (testing, Chaps. 14 and 15). Rule management is discussed in Part VII (rule governance, Chaps. 16 and 17).

Figure 1.2 shows the BRMS within the context of its operational environment. The BRMS has two components, a management component and an execution component, sharing a common repository of rules. The rule repository is read and modified by management functionalities, but read-only by automation (execution) functionalities. The rule repository may contain different representations of the

same business rules, depending on lifecycle stage and on audience. Figure 1.2 shows that both business and IT access the management functionalities. We will not try to be more precise at this point; Chap. 3 presents the different roles in more detail.

According to this scenario, the rules relevant to business applications are executed *outside* of the business applications: the rule automation component of the BRMS acts as a rule execution service on behalf of business applications. This is the most typical scenario for full-functionality BRMSs and shows one way that business applications developed with the business rules approach differ from traditional development methods. However, it is not the only way of executing rules; this and other issues will be discussed at length in Part III of this book.

In this context, business applications developed with the business rules approach – or business rule applications, in short – differ from traditional applications in four ways: (1) the code itself, (2) deployment, (3) run-time behavior, and (4) maintenance. We will discuss the four aspects in turn.

The code. A good application design with the business rules approach should exhibit very few code-level differences with *good* nonbusiness rule applications. The only difference is in the way control-intensive domain functionality is implemented. A good object-oriented design would typically assign each domain-specific function to a *facade* or *controller* method, which in turn would coordinate domain objects to produce the result. Take the property insurance coverage for mortgages rule (third example presented in Sect. 1.1). A good object-oriented application would have a method called "`checkPropertyHazardInsuranceCoverage()`" defined for the class **`MortgageApplication`**, or for some **`PropertyAssessmentService`** class, which returns `true` if the coverage is adequate, and `false` otherwise. In a nonbusiness rule-oriented application, the method would implement the business logic described by the rule in the implementation language (java or C# or Object Cobol!) with loops, **if**s, **then**s, and **else**s. A business rule application would, instead, code the business decision logic in a rule language and delegate its execution to the rule execution component of the BRMS, as illustrated in Fig. 1.2. Other than that, the code should look identical! In fact, we consider it *good practice* to circumscribe the parts of a business application that are "aware" of business rules, and that interact with a BRMS.

Deployment. With regard to deployment, a business rule application differs from a traditional application in that application logic is broken into two pieces: (1) business rules that are managed and executed by a BRMS and (2) a computational infrastructure that is responsible for everything else (materializing application objects and managing them, managing the application workflow, architectural services, etc.). These two pieces are packaged separately, and deployed separately, and often asynchronously; we will say more when we talk about maintenance.

Run-time. In terms of run-time behavior, we should see no difference between the *functional behavior* of a business rule application and that of a traditional one: they are supposed to be both implementing the same business rules, and thus we should get the same outcomes for the same inputs! In fact, this is one way that we

can validate a business rule application that is a reengineered version of a legacy application – as most rule projects are. In terms of run-time architecture, an implementation scenario such as Fig. 1.2 means that our business rule application needs to invoke an external service, although we could also embed a rule interpreter (called *rule engine*) in the business application in the same executable/run-time image.

Maintenance. Maintenance is probably the one aspect of a business rule applications that is most different from traditional applications. As we saw in Sect. 1.2, one of the key motivators for the business rules approach is the need for agility so that business rule applications can evolve as fast as the business needs it. Several factors make maintenance easier and faster:

1. *Understandability by business*. Business rules are expressed in languages that business users can understand, enabling them to either specify the rules themselves or to easily validate them.
2. *Separate deployment*. Because business rules are deployed separately from the code base of applications, we can have a rule maintenance and release cycle that is separate from – and hence much lighter-weight than – your average application maintenance and release cycle.
3. *Separate execution*. As a corollary of separate deployment, and based on the scenario shown in Fig. 1.2, business rules are executed by the BRMS, on demand from business applications. This means that we can have *hot deployment* of new business rules, without shutting down the business application. In fact, the Websphere ILOG JRules BRMS – JRules, in short – enables us to run *different versions* of business rules *simultaneously*. We will introduce JRules in Chap. 8 and talk about situations where we might need several versions of rules in Chap. 13.

Figure 1.3 illustrates the different release and maintenance cycles for the core of business applications and for the business rules.

The lower part of the figure shows the maintenance and release cycle for the application code, which should be fairly stable. After the first release of an application, we may have an update release or two within the first year, but after that, the pace of change slows down even further – often once a year or less, for back-office systems. With regard to the rules, we can have many smaller updates as frequently as needed, including daily, or even hourly, if quality assurance can follow!

1.4 Why Do We Need a New Methodology?

The business rules approach makes business rules explicit, separates them from other application requirements and development artifacts, and manages their development, their deployment, and their execution. The way that we develop the basic application infrastructure, however, need not change significantly. If you have been

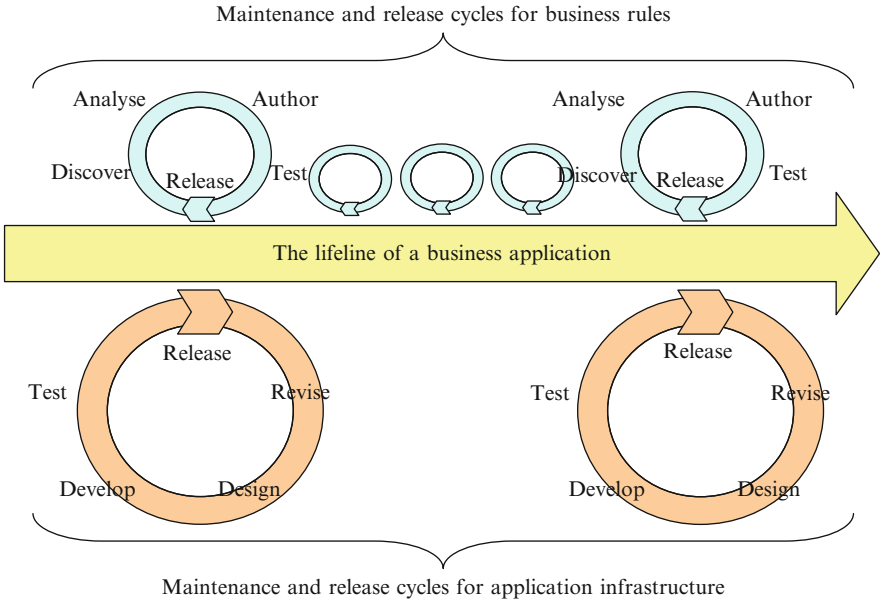

Fig. 1.3 Maintenance and release cycles for application core versus business rules

using some homegrown version of the Unified Process (UP), or some agile method, or flavorful combinations of the two such as OpenUp,[7] you need not change the way that you develop your application infrastructure: (1) you still use use cases or business process description (or whatever it is that you use) to capture functional requirements, (2) you still use object models to represent the business domain and the way it is captured in the software, and (3) you still design your architecture using the same criteria (distribution, scalability, performance, and security) and the same solutions. However, we need well-defined processes, roles, and deliverables to handle business rules, and their relationship to the application infrastructure. In the remainder of this section, we will discuss the ways in which the *process* of developing a business rule application differs from traditional application development. Part II of this book will go over our own methodology, *Agile Business Rule Development* (ABRD); in this section, we will content ourselves with highlighting the issues.

Synchronous versus asynchronous rule management. Before we start talking about various development activities, we need to make a distinction between two ways of developing and managing business rules, which have different methodological implications:

[7]OpenUP is an Eclipse project that uses the *Eclipse Process Framework* (http://www.eclipse.org/epf) to specify an agile version of the Unified Process.

- We can develop business rules as a separate activity, independent of specific business application projects, and project schedules. We can think of business rule management as part of a broader *knowledge management* practice within the organization. This means, among other things, the existence of a rule management organization within the enterprise, which can serve various business applications. The rule management organization is then responsible for collecting, codifying, validating, and publishing the business rules. The application project organizations will then reference a subset of those rules in their applications. In this case, we have a well-defined producer–consumer relationship between the rule management organization and the application project organizations. Figure 1.4 illustrates this scenario.
- We can also develop business rules as a by-product of specific business applications. In this case, the rules will be developed incrementally, and always within the context of a specific application project. However, the rules will be stored and managed in a shared repository. Figure 1.5 illustrates this scenario.

Which approach works best? Each of the two approaches has its advantages and disadvantages. The first approach may be more appropriate for a large and *mature* organization which will have a dedicated team of business analysts whose job is to create and manage business rules for the enterprise. This approach requires top-level management commitment since it requires significant up-front investment costs in human resources that are not easily linked to operational priorities. One of the methodological challenges that such teams would face is the scoping of their activities. Indeed, without any specific mandate at hand, they need to identify and prioritize the business areas that they need to address. Also, the chances are that in the first few months or years of operation, many project organizations will not find in the repository everything that they need. The advantages of this structure include

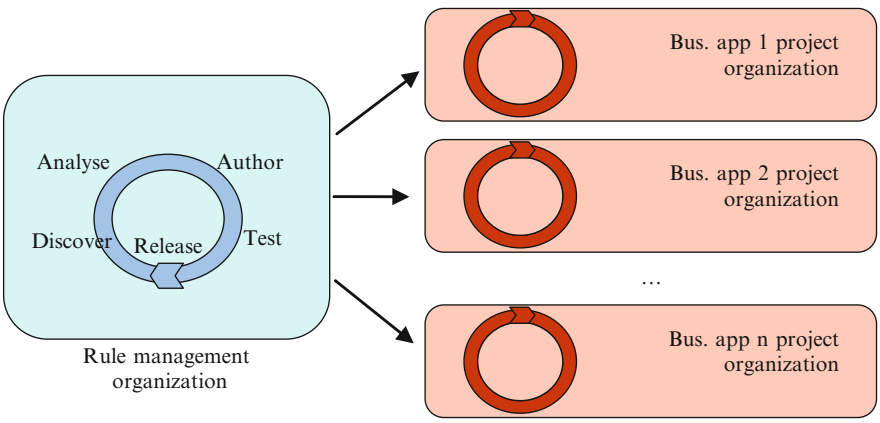

Fig. 1.4 Rule management is the responsibility of an independent organization that produces rules consumed by different project organizations

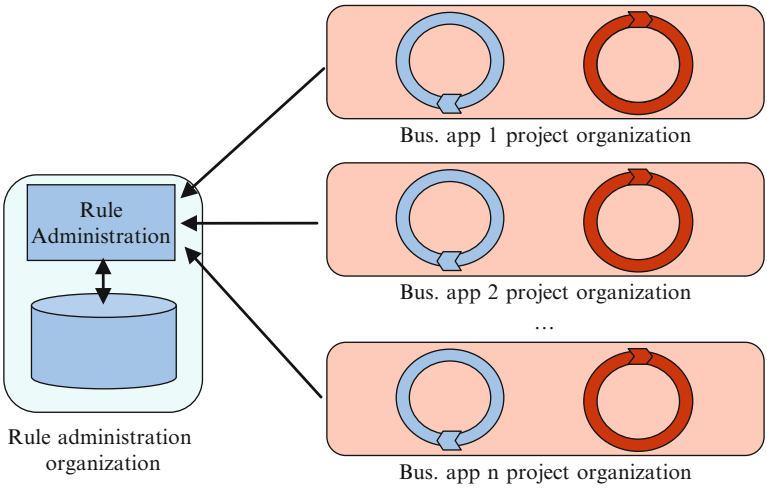

Fig. 1.5 Each project team develops and manages the business rules it needs for the application it is building

a de facto enterprise-wide visibility of rules, a more coherent rule repository, and a more consistent application of rules across business applications.

The second approach does not require substantial up-front investments that are hard to justify, will not suffer from "analysis paralysis" since rules will be collected within the context of specific applications, and each business application will have all the rules it needs by the time it is done. However, it has two major disadvantages: (1) a duplication of effort between various project teams, especially if several projects are running in parallel and (2) having to manage multiple sets of rules with a potential proliferation of variations on the same rules, or worse yet, conflicting versions of rules. Figure 1.5 shows this scenario. In this case, we have an enterprise-wide lightweight *rule administration* function, in terms of a shared repository and centralized access control, but each business application project team is responsible for managing its rules, from discovery to execution.

In practice, enterprises would use an organization that is between these two extremes, depending on its maturity level. An enterprise that is making its first foray into the business rules approach should use the organization shown in Fig. 1.5, for the first couple of pilots, typically in sequence. It is more likely in this case that the same people involved in the business rule component of the first application will also be involved in the second application, both to perfect their techniques and to act as *seeds* for other teams. As they get involved in more projects, these pioneers will also start developing a global view of the business rules, and start seeing opportunities for sharing and reusing rules between applications, and across business functions. They may eventually get integrated into an enterprise-wide *business rules expertise center* that includes expertise in business rules methodology, business rule implementation technology, and business knowledge. Some of these pioneers may be loaned to specific project teams, while others focus

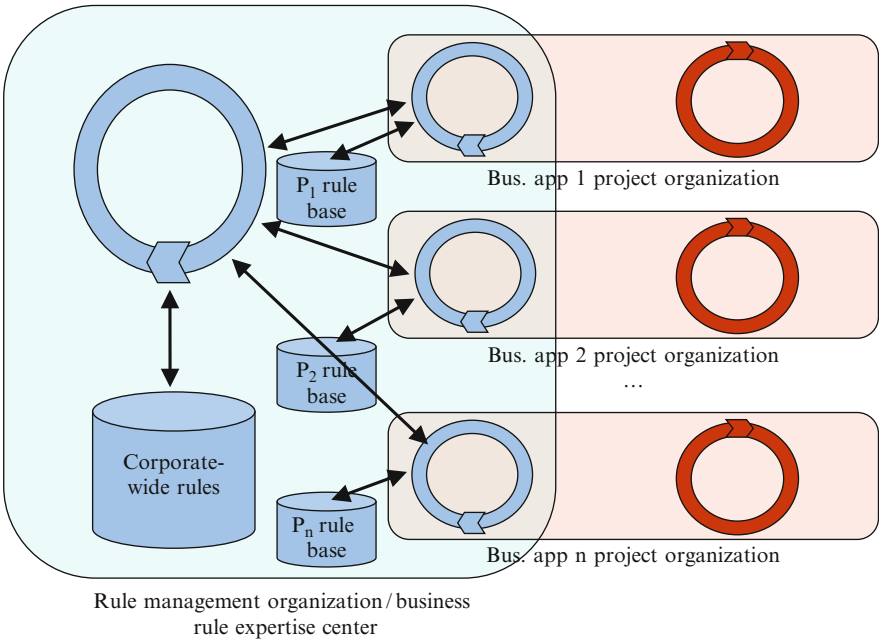

Fig. 1.6 An intermediary organization that combines the agility of synchronous development while leveraging common expertise and corporate-wide rules

on corporate-wide rules. Figure 1.6 illustrates such an organization, which we have seen operate successfully in some of the more mature organizations. Figure 1.6 shows that there is two-way communication between project-specific rule activities and corporate-wide rule activities. Indeed, project-specific rule teams will use the corporate-wide rule base as a potential source of rules relevant to the application at hand. Also, in the process of collecting rules for a specific application, they may find that some rules are generally applicable, and include them – or ask that they be included – in the corporate-wide rule base.

The methodology presented in this book, ABRD, is based on the synchronous model – Fig. 1.5.

New application development versus reengineering existing applications. Many of our engagements with customers dealt with new applications aiming at automating previously manual, decision-intensive business processes. Such projects have the necessary business focus from the beginning, and provide an opportunity to apply the principles of the business rules approach, almost by the book. However, *many more* engagements consisted of reengineering existing applications. The scope and depth of the reengineering effort determine the extent of freedom that the project team will have in implementing the new system, and the number of painful compromises that need to be made to accommodate the legacy system. Figure 1.7 shows different reengineering scopes in relation to a layered system

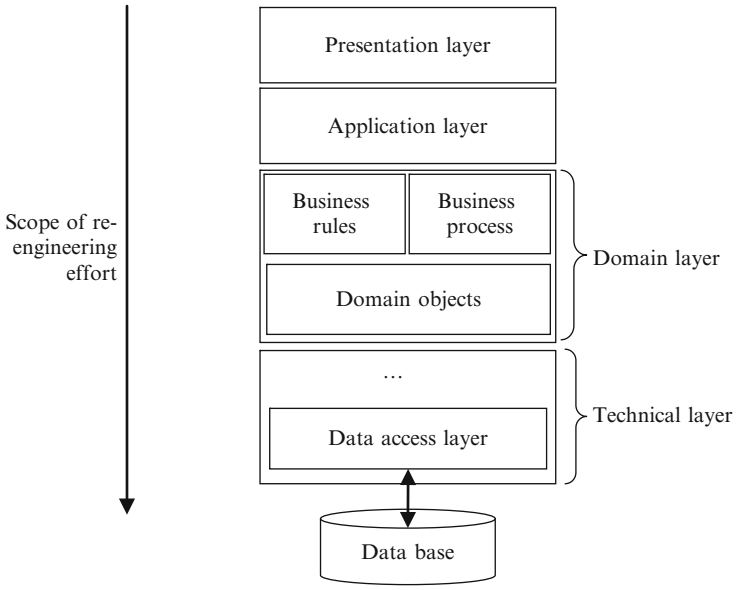

Fig. 1.7 The implications of the business rules approach depend on the scope of the reengineering project

architecture. We will comment on a few points in this space that correspond to the most typical situations.

A common scenario consists of introducing new technologies into a legacy system to make it more scalable, agile, modifiable, etc. In this case, business rules technology is introduced along with a mix of other technologies, including an object-oriented domain layer, a web-based presentation layer, a business process workflow engine, etc. In this case, the only thing that is salvaged from the legacy system is often limited to the legacy database (or EIS layer); anything from the data access layer up to the presentation layer is built from scratch. With the appropriate discipline (e.g., business focus), these projects may be managed – and feel like – new application development (*forward engineering*), with few constraints and compromises.

Another common scenario consists of reengineering the top layers of the application, going from the presentation layer down to, and excluding, the domain objects layer. This means that the domain objects are already built in Java or C#, and that we need "only" to reengineer the way the business rules are implemented and executed in the application. This scenario is not trivial as the existing domain object implementation may not readily lend itself to the expression and execution of business rules according to the business rules approach. The gap needs to be bridged through a combination of methodology and technology.

Figure 1.8 shows a methodology matrix that illustrates the methodological variants of the business rules approach. The STEP methodology (von Halle 2001)

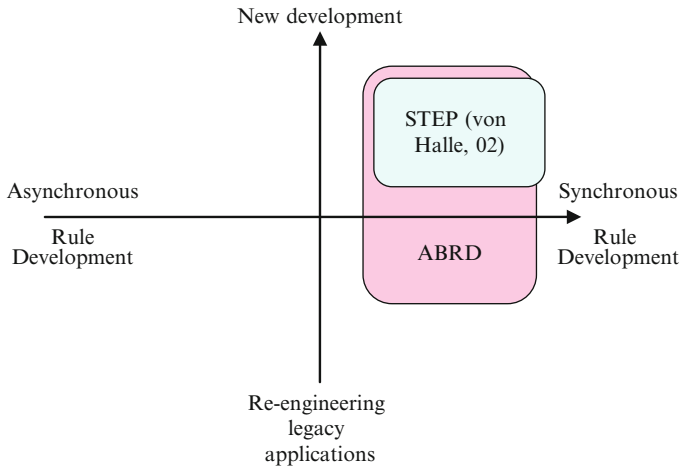

Fig. 1.8 A business rules methodology matrix

is an essentially synchronous, forward-engineering methodology for new applications built under the business rules approach, and addresses both the infrastructure of the application and the business rules component in the same framework. ABRD focuses on the business rules component and its interface with the application infrastructure, of which it is fairly independent.

We now turn our attention to the various development *activities* and see how they are affected by business rule methodologies, depending on where they fit in this matrix. For the sake of discussion, we will consider (a) requirements capture, (b) analysis and design, (c) coding/authoring, (d) testing, and (e) maintenance; the changes brought upon by the business rules approach are fairly independent from the actual *process* along which these activities are organized.

Requirements capture. In the synchronous mode, for new developments, we elicit the business rules as part of the requirements capture. However, the business rules are gathered in separate deliverables, which cross-reference other requirements deliverables such as domain models and business use cases. Further, there is an explicit emphasis on business rationale (business policies and motivations behind them), as opposed to focusing on the business actions that derive from such rationale. Accordingly, we need specific processes, roles, techniques, and deliverables to handle business rules. The processes and techniques for eliciting business rules, and the intermediary deliverables, depend on the requirements capture technique traditionally used by the organization. For example, if an organization relies on *use cases* for capturing functional requirements, the business rules will be captured in the context of *decision steps* within those use cases [see, e.g., the *use-case rule discovery roadmap* of the *STEP* methodology (von Halle 2001)]. If we have a reengineering project, the legacy system and its documentation are usually used as a potential source – seldom the only one – for business requirements

in general, and business rules in particular. In this case, the process and techniques for rule discovery are adapted accordingly.

In the asynchronous mode, we clearly need separate processes, roles, techniques, and deliverables for the discovery of enterprise business rules, independently of requirements capture for specific business applications.

Analysis and design. The analysis and design of the *infrastructure* of a business application are marginally affected by the adoption of the business rules approach, except for a more explicit business focus, and the reliance on a BRMS for performing business decisions (see, e.g., Fig. 1.2). However, there are lots of new things to analyze and design on the decision/business rule side of the application. There is such a thing as *rule analysis*, which deals with things such as breaking complex business rules into several simpler more atomic ones, detecting redundancies, overlaps or contradictions between rules, documenting the business motivations of rules, and so forth (see Chap. 4). Further, we need to package rules into coherent units of testing, deployment, and execution – called *rulesets* – depending on the underlying business process and on application design considerations (Chap. 9). We also need to specify and design the management component of the BRMS, including the structure of the rule repository (Chap. 9), the rule metadata, the enforcement of the rule change processes, etc. (Chaps. 16 and 17). Finally, we need to design the way in which the business application will interact with the BRMS for executing the business rules (Chaps. 7, 12 and 13).

Coding/authoring. The coding of the application infrastructure is not affected by the use of the business rules approach. However, decision logic is now coded separately as business rules through a BRMS system, and we need a new set of processes, techniques, skills, roles, and tools for *rule authoring*. One of the major consequences of this separation is that the two aspects of the application are decoupled and can progress independently. We have been involved in projects where the application infrastructure was completed before the first business rule was coded and tested. An incredulous CIO protested "how could you send half the development team home when you are still capturing requirements." We have also been to projects where all of the business rules have been coded, and many were tested, before a single domain Java class was coded. Rule authoring issues and solution patterns are fairly independent of where we stand in the methodology matrix (Fig. 1.8). Part IV of this book (Chaps. 9, 10, and 11) is dedicated to rule authoring.

Testing. In traditional system development, *functional testing* can only start after large chunks of an application have already been implemented. Further, *black box* functional testing provides little to no help in diagnosing an application's business logic, whereas *white box* functional testing requires us to identify and analyze logical paths within complex execution traces. With the business rules approach, we can test *individual* business rules, with *little infrastructure code*. This is like performing *functional unit testing* where we are able to identify, trace, and modify individual logical paths through the application code. The testability of individual rules is a powerful verification and validation tool. Part VI of this book (Chaps. 14 and 15) deals with rule testing.

Maintenance. In traditional system development, maintenance requests follow a similar implementation path, whether the request concerns business logic or infrastructure code: once a manager has signed off on a maintenance request, it falls into the hands of IT who implement it, test it, and deploy it. With the business rules approach, because business rules (decision logic) are developed and maintained separately, we have different processes in place that recognize the *business* nature of *business rule maintenance*, and that take advantage of the lighter deployment mechanisms for business rules. Business rule maintenance is part of a wider set of rule management activities that we refer to as *rule governance*. Rule governance processes depend heavily on the business rule approach variant along the synchronous versus asynchronous development dimension (see Figs. 1.4 to 1.6). Rule governance is discussed in Chaps. 16 and 17.

1.5 Summary and Conclusions

Organizations develop business information systems to support their business processes. These information systems should behave in a way that is consistent with the organization's business objectives and policies. They do so by enforcing *business rules*. Put another way, business rules embody the business soul of business applications. Both business and IT need to know what those rules are, and sometimes customers and regulators do too. The rules need to be expressed in a language that all the stakeholders can understand, and implemented in a way that enables us to change them at the speed of business, as opposed to the speed of IT. These are the motivations behind the so-called business rules approach.

The business rules approach consists of three interrelated components:

1. A methodology for creating and managing the business rules
2. One or more languages for expressing them at different stages of their life cycle and for different audiences
3. A tool set for managing and executing them on behalf of business applications

We saw in Sect. 1.4 that business rules methodologies come in different flavors, depending on the maturity of the organization with the business rules approach and on the nature of the project, that is, a new development versus a reengineering project. We also saw how the adoption of the business rules approach affects traditional development tasks such as requirements capture, analysis, design, coding, testing, and maintenance. The remainder of this book addresses all of these activities within the context of the Agile Business Rule Development methodology and the IBM Websphere ILOG JRules business rule management systems (BRMSs) – JRules in short.

So, is it an evolution or a revolution? We do not like revolutions. Revolutions start with destruction – destroying legacies – lead to initial chaos – even if temporary – and are often run by quasi-religious zealots. And the outcome is often unpredictable. The ingredients for the business rules approach have been

around for more than 20 years. It is their combination, in their current mature form, which gives the approach its revolutionary power.

In this chapter, we strove to focus on the basics, which does not necessarily do justice to the complex technological landscape of today's enterprise applications. More detail and nuances will be presented in the next 17 chapters of the book!

1.6 Further Reading

There are a number of resources about the business rules approach that the reader can consult to complement the information provided in the chapter.

- A book by Ronald Ross titled *Principles of the Business Rules Approach*, published by Addison Wesley, February 2003, Addison Wesley. As the title suggests, this is a foundational book. It talks about the essence of business rules, how they relate to business events, and proposes an extensive classification of rules. This book says very little about implementation, and does not present a step by step methodology for building business rule applications – nor was it its intent.
- The book *Business Rules and Information Systems: Aligning IT with Business Goals*, by Tony Morgan, Addison Wesley, March 2002. This is another foundational book – a *great one, nonetheless*. It presents the essence of the business rules approach by explaining what business rules are, what they are about, and attempts a rigorous approach to rule capture and analysis. There is little in terms of a step-by-step methodology and *very* little in terms of technology.
- Barbara von Halle's book, *Business Rules Applied: Building Better Systems Using the Business Rules Approach*, published by John Wiley & Sons, in 2001. This book presents the STEP methodology (*Separate*, *Trace, Externalize*, and *Position* rules for change). It does an excellent job of presenting methodology but is a bit short on design and very short on implementation.
- The business rules group web site (http://www.businessrulesgroup.org) contains links to the various papers published by its members. Topics addressed include the definition of business rules (see Sect. 1.1), the business rule motivation model, and the business rule maturity model.
- The Object Management Group (http://www.omg.org) has a number of active standards related to business rules, a number of which are based on (more readable) submissions of the business rules group.
- The *business rules forum* (http://www.businessrulesforum.com) is an annual conference for people interested in the business rules approach, and is a good opportunity for learning about new product features and cutting-edge thinking.

http://www.springer.com/978-3-642-19040-7

Agile Business Rule Development
Process, Architecture, and JRules Examples
Boyer, J.; Mili, H.
2011, XXVI, 567 p., Hardcover
ISBN: 978-3-642-19040-7