COMPACT
INFO

LOGIC Management via Decision Tables

LOHRFINK
software engineering GmbH & Co. KG

www.lohrfink.de

1 Introduction

Such issues as: Business Rules Management, unclear specifications and often missing language between technical teams and IT, Business-IT alignment, have been relevant for years.

In this context, an extremely powerful method is unjustly neglected: the ***decision table***.

This document highlights the benefits of decision tables and explains why this method alone, if used correctly, offers an unrivalled performance in ***consistent and secure mastering of logic at all project stages***.

2 Business rules

Business rules define precisely which sequence of conditions in a business flow leads to which actions:

WHEN	funds are due
AND	period allowed for payment is over
AND	the given customer doesn't have VIP status
AND	the outstanding amount is classified as a minor contribution
AND	the issue is about the initial first payment
THEN	send a warning reminder based on Article 38 of the contract
AND	mark in the database that the customer is subject to contract breach or in danger of contract termination

What is often ignored is that a business rule is never standing alone but is always necessarily logically related to numerous other business rules whereby the relationship results directly from the conditions used.

Here is an example of such a related rule:

WHEN	funds are due
AND	period allowed for payment is over
AND	the given customer doesn't have VIP status
AND	the outstanding amount is classified as a minor contribution
AND	the issue is about subsequent payment
THEN	send a warning reminder based on Article 39 of the contract
AND	initiate contract termination

In order to make sure that decision-making processes are described correctly, business rules must be put together in groups, whereby each group must ensure that all possible combinations of the conditions are taken into account and that they are complete as well as free of redundancies and contradictions.

3 Decision tables

Decision tables are a very compact, precise and long-term-proven method of grouping related business rules in a standardized way, see e.g. international standards such as CSA Z243.1-1970 or DIN 66241.

The two verbal rule examples from the previous chapter correspond to the decision table rules R04 and R05 here:

		R01	R02	R03	R04	R05	R06	R07	R08
B01	Account balance?	<0	=0	>0	>0	>0	>0	>0	>0
B02	Payment deadline reached?	-	-	Y	Y	Y	Y	Y	N
B03	Customer has VIP status?	-	-	Y	N	N	N	N	-
B04	Minor payment?	-	-	-	Y	Y	N	N	-
B05	Initial first payment?	-	-	-	IFP	SP	IFP	SP	-
A01	No action	-	X	-	-	-	-	-	X
A02	Write log entry	X	-	X	-	-	-	-	-
A03	Send reminder	-	-	-	A.38	A.39	A.38	A.39	-
A04	Schedule contract termination	-	-	-	X	-	-	-	-
A05	Terminate contract	-	-	-	-	X	-	-	-

When modeling via a software tool, it is vital that this standardized structure is maintained:

- All conditions (C01 to C05) are listed in the upper left section, all actions (A01 to A05) are listed in the lower left section and every column on the right exactly represents one of the business rules (R01 to R08).

This form of presentation is a prerequisite for numerous important functions that enable efficient and secure handling of rules in the first place:

- We can condense rules, expand rules or resort them, we can remodel or move conditions and actions, missing rules can be found and added automatically, and existing redundancies and conflicts can be detected, etc.

4 Logic brought to perfection

The usage of decision tables comes along with various advantages:

- The quality of a group of rules can already be checked when describing decision-making processes: we can easily find out if they are complete, free of redundancies and contradictions
- The complexity of descriptions can be measured (via the number of conditions, actions and rules); as a result, further expenditure, e.g. on implementation, testing and acceptance, can be planned much more precisely

5 Create source code straight from decision tables

If you want to guarantee that your system behavior corresponds exactly to description, there is no better way than to generate your program code directly from the decision tables.

Any subsequent rule changes must always be made in the decision table first; the program code is synchronized with functional changes by being generated.

Specifications, i.e. decision tables, remain the leading documents and never become outdated if you follow this procedure.

6 Platform independence

In principle, the program code for any platform can be generated from decision tables, if necessary, even for several platforms at the same time.

For example, the program code for Java, C# and ABAP can be generated simultaneously from one decision table.

7 Supporting communication

Communication is facilitated through standardized and compact form of presentation, especially for complex issues.

A high degree of detail can be achieved through clear and unambiguous references (as in a coordinate system: decision table, rule, condition, action).

8 Easier work with descriptions

Descriptions of decision-making processes in the form of decision tables are generally perceived by users as "less strenuous" and "less tiring" than descriptions provided e.g. in text form.

This effect is intensified if the tools that are used provide functions such as automatic addition of missing rules or simulation of rule behavior for specific case examples.

9 Easier work during technical implementation

The programming effort can be reduced to the necessary minimum thanks to the use of generators; in extreme cases, it is even possible to simply regenerate after rule changes.

10 Existing solutions, legacy applications, migration

Unfortunately even nowadays the implementation of business logic is still in many cases performed by writing the logic straight into executable program code.

When using decision tables instead:

- an abstract and highly understandable level of logic description is achieved (post-documentation using a standardized "unbundling" method)
- and technical conversion can then be carried out by means of generation into the same or another target platform without any loss of quality.

Decision tables are thus an excellent framework for reengineering projects.

11 Future-proof qualities

Decision tables are mature, robust and globally standardized, for example via DIN 66241 in Germany.

They aren't bound to a specific platform and can even later be used for generating program codes for completely different or brand new platforms.

12 Requirements for a tool for decision tables

The most critical success factor is user acceptance, which is why the tool must seamlessly integrate into the existing well known user's work environment.

- Powerful help functions must inspire the user ("We haven't had anything like this before", "We feel that our descriptions are now better and more complete")
- For the software developer, the tool must be able to communicate well with the respective development environment
- The software developer must find the code generation helpful and be able to consider the generated code "high quality"
- Functions and mechanisms of the tool must be easy and quick to understand

Important functionality sets for a powerful and convenient usability when dealing with decision tables are:

- Automated checks for completeness and well as freedom of redundancies and contradictions
- Automated generation of missing rules
- Automated condensing of rules e.g. by removing irrelevant conditions
- Resorting of rules based on parameterized aspects
- Showing and hiding rules to focus on specific aspect refinements
- Filtering of rules according to arbitrarily complex criteria
- Interactive simulation of decision table behavior for specific cases
- Linking and nesting decision tables, e.g. in order to use a decision table in another decision table as a condition or action
- System-wide analysis (retrieval) of conditions or actions already used elsewhere
- Dashboarding and presenting numbers and KPIs for projects and subprojects: number of decision tables, conditions, actions and rules

Further aspects:

- The toolset should ideally be platform-independent i.e. it should be able to be used in and with any platform
- It should be possible to generate the program code for various programming languages from a decision table
- Maximum transparency: the generated program code should offer the possibility to run a detailed analysis and evaluation of system behavior (i.e. rule execution) at runtime
- It should be possible to generate test cases for various test management and test automation tools from one decision table
- There should be a possibility to integrate the toolset also in CI/CD pipelines
- The tool should be a pure development toolset, no third-party runtime components should be needed in specific production environments (e.g. no rule engines, no rule server, no "black-box")

13 Further topics

This document only provides an overview of the benefits and synergies when using decision tables in combination with a powerful toolset and framework when describing technical regulations.

Further subjects for a closer look would include e.g.:

- Loops and nested loops and their associated logic
- State machines
- Nesting of decision tables, i.e. decision tables as conditions or actions for other decision tables
- Complete secured generation of all relevant test cases directly from decision tables
- Code generation for Java, JavaScript, TypeScript, Python, ABAP, PL/SQL, Natural, Cobol, C#, C++, C, VB, VBA, Delphi, PHP and others
- Automated logging of rule executions
- Statistical analysis of rule executions also for long running processes
- Automated research functionality to find out which test cases have and which have not been covered yet

14 LF-ET - The decision table tool

LF-ET (www.lohrfink.de/en/solutions/lf-et/) fulfills all requirements referred to in this document and enables an easy yet powerful and consistent use of decision tables in any IT projects.

Using decision tables with LF-ET is intuitive and LF-ET, with its Excel-like interface and numerous powerful functions, enables even very complex sets of rules to be securely mastered.

Optional use of generators can guarantee automatically that the delivered software module exactly meets the specifications that it is based upon. Actual programming effort can thus be significantly reduced.

Experience has shown that significant cost savings can be expected, particularly in the case of later maintenance and care, with the risk of errors being significantly reduced at the same time.

If necessary, LF-ET enables you to create a program code in various programming languages from one decision table.

LF-ET is a truly development toolset, ***no additional external software*** is needed to execute the generated programs.

15 Contact us

We have been supporting numerous authorities and companies from various industries as an IT-consultancy and service provider for many years.

No matter what technical context is given in your environment – the decision table is an essential method that enables process flows to be described, implemented and tested more easily and precisely.

Feel free to contact us with any further questions or queries.

LOHRFINK
software engineering GmbH & Co. KG

Marie-Curie-Str. 6
D-70736 Fellbach

Telefon 0711/3424 897-0
Telefax 0711/3424 897-15
info@lohrfink.de
www.lohrfink.de
www.lohrfink.de/lf-et