Visual Models for Software Requirements

Use visual models to improve the way you capture and analyze software requirements—and deliver better results for your business. The authors—two experts in software requirements methodologies—guide you through a simple but comprehensive language of visual models that has been used on hundreds of real-world, large-scale projects.

Discover how to apply visual-modeling best practices to transcend the limitations of text-based approaches. And enable all your stakeholders—from line-of-business reps and executives to your development and test teams—to better ‘see’ and substantiate the solution.

- Use visual models to capture requirements, identify gaps, and validate solutions more rigorously than traditional approaches
- Get practical, scenario-based guidance on how and when to combine models for best outcomes
- Apply visual models successfully across development scenarios, including agile, iterative, and waterfall
- Practice the book’s concepts as you work through the chapters
- Change your focus from writing a good requirement to analyzing an entire solution

Includes a full set of visual modeling templates
Ready to download at http://go.microsoft.com/fwlink/?Linkid=253518

About the Authors

Joy Beatty drives the creation and implementation of requirements modeling methodologies, including new and best practices. A leader in the requirements community, Joy has trained thousands of business analysts and helped major organizations build business-analysis centers of excellence.

Anthony Chen runs a professional services company that creates software requirements for Fortune 500 companies. He helped pioneer key requirements-modeling concepts, including Requirements Modeling Language (RML), and shares his extensive expertise through writing, consulting, and mentoring.

Joy Beatty and Anthony Chen
Sample Chapters
Copyright © 2010 by Seilevel
All rights reserved.

To learn more about this book visit:
http://go.microsoft.com/fwlink/?LinkId=253519
# Contents

*Foreword* ................................................................. xxvii  
*Introduction* ............................................................ xxix  

## PART I AN INTRODUCTION TO MODELS

### Chapter 1 Introduction to RML 3

- RML Defined .............................................................. 4  
- Challenges with Traditional Software Requirements Practices .......... 4  
  - Limitations of the Human Brain. ................................. 4  
  - Pictures Are Easy, Words Are Hard .............................. 6  
- Requirements Models .................................................. 7  
  - Why Not UML? ....................................................... 8  
  - Requirements vs. Design. ......................................... 8  
  - Requirements Models Are Not the End Game .................... 10  
- Using RML on Projects ................................................. 11  
- Additional Resources .................................................. 11  
- References .............................................................. 11

### Chapter 2 Model Categorization 13

- Objectives, People, Systems, and Data Models ....................... 14  
  - Bounding Models .................................................... 15  
  - All Four Categories Are Needed ................................. 15  
- Objectives Models .................................................... 17  
- People Models .......................................................... 18  
- Systems Models ........................................................ 19  
- Data Models ............................................................ 20  
- References .............................................................. 20  


# Contents

## PART II OBJECTIVES MODELS

### Chapter 3 Business Objectives Model

- Business Objectives Model Template ........................................ 25
- Example ............................................................................... 26
- Creating Business Objectives Models .................................... 29
  - Identify Business Problems .................................................. 30
  - Identify Business Objectives ............................................... 31
  - Define Additional Problems and Objectives ............................ 32
  - Define the Product Concept ............................................... 33
  - Describe Success Metrics ................................................... 34
  - Questions to Ask to Complete the Business Objectives Model .... 34
- Using Business Objectives Models .......................................... 35
  - Providing a Common Understanding of a Project’s Value ............ 35
  - Bounding the Solution Space ............................................... 36
  - Understanding Projects That Are Underway ............................ 36
  - Deriving Requirements ....................................................... 39
  - When to Use ...................................................................... 39
  - When Not to Use ................................................................ 39
- Common Mistakes .................................................................... 39
  - Not Understanding the Business Problem ............................... 39
  - Defining Business Objectives That Are Not Measurable .......... 39
  - Articulating the Wrong Type of Information in Business Objectives ......................................................... 40
- Related Models ....................................................................... 40
- Exercise .............................................................................. 40
- Instructions .......................................................................... 41
- Scenario ............................................................................... 41
- Additional Resources ............................................................ 41
- References ........................................................................... 41
## Chapter 4  Objective Chain  43

<table>
<thead>
<tr>
<th>Objective Chain Template</th>
<th>44</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>45</td>
</tr>
<tr>
<td>Creating Objective Chains</td>
<td>49</td>
</tr>
<tr>
<td>Identify Business Objectives and Features</td>
<td>49</td>
</tr>
<tr>
<td>Select the Features to Analyze in Objective Chains</td>
<td>49</td>
</tr>
<tr>
<td>Identify the Objective Factors</td>
<td>50</td>
</tr>
<tr>
<td>Create the Objective Chain Hierarchy</td>
<td>53</td>
</tr>
<tr>
<td>Define the Objective Equations</td>
<td>55</td>
</tr>
<tr>
<td>Using Objective Chains</td>
<td>57</td>
</tr>
<tr>
<td>Comparing the Relative Values of Features to Cut Scope</td>
<td>57</td>
</tr>
<tr>
<td>Determining Project Success</td>
<td>59</td>
</tr>
<tr>
<td>Deriving Requirements</td>
<td>59</td>
</tr>
<tr>
<td>When to Use</td>
<td>60</td>
</tr>
<tr>
<td>When Not to Use</td>
<td>60</td>
</tr>
<tr>
<td>Common Mistakes</td>
<td>60</td>
</tr>
<tr>
<td>Not Creating Objective Chains Because Data Doesn’t Exist</td>
<td>60</td>
</tr>
<tr>
<td>Skipping Levels in the Hierarchy</td>
<td>60</td>
</tr>
<tr>
<td>Related Models</td>
<td>60</td>
</tr>
<tr>
<td>Exercise</td>
<td>61</td>
</tr>
<tr>
<td>Instructions</td>
<td>61</td>
</tr>
<tr>
<td>Scenario</td>
<td>61</td>
</tr>
<tr>
<td>Additional Resources</td>
<td>62</td>
</tr>
<tr>
<td>References</td>
<td>62</td>
</tr>
</tbody>
</table>

## Chapter 5  Key Performance Indicator Model  63

<table>
<thead>
<tr>
<th>KPIM Template</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>65</td>
</tr>
<tr>
<td>Creating KPIMs</td>
<td>67</td>
</tr>
<tr>
<td>Identify Business Processes</td>
<td>67</td>
</tr>
<tr>
<td>Identify KPIs</td>
<td>68</td>
</tr>
<tr>
<td>Create the KPIM</td>
<td>69</td>
</tr>
</tbody>
</table>
Chapter 7 Requirements Mapping Matrix

RMM Template ........................................... 88
Example .................................................. 90
Creating RMMs ........................................... 92
  List Process Flow Steps .............................. 92
  Map Requirements to Process Flow Steps ........ 93
  Identify Missing Mappings ......................... 96
Using RMMs ............................................. 96
  Reviewing in an Easy-to-Read Structure ........ 97
  Identifying Missing Requirements ................ 97
  Identifying Extraneous Requirements or Missing Steps .... 97
  Prioritizing Scope ................................... 98
  Advantages of Using a Requirements Management Tool ...... 98
Deriving Requirements ................................. 98
When to Use ............................................ 99
When Not to Use ...................................... 99
Common Mistakes ..................................... 99
  Not Mapping to Process Flows ...................... 99
  Not Using or Updating RMMs ....................... 99
Related Models ......................................... 99
Exercise ................................................... 100
  Instructions .......................................... 100
  Scenario ............................................. 100
Additional Resources ................................. 102
References ............................................. 102
## PART III  PEOPLE MODELS

### Chapter 8  Org Chart  105

Org Chart Template ........................................... 107  
Example .......................................................... 108  
Creating Org Charts ........................................... 110  
  Locate Existing Org Charts .................................. 110  
  Determine the Right Level of Org Chart ................. 111  
  Complete the Org Chart .................................... 111  
Using Org Charts ............................................... 113  
  Identifying the People Who Have Requirements .......... 113  
  Identifying Internal Users .................................. 114  
  Identifying External Users .................................. 114  
  Identifying People Used in Other Models ................. 114  
Using Org Charts with Process Flows for Completeness .... 115  
  Deriving Requirements .................................... 116  
  When to Use .................................................. 116  
  When Not to Use ............................................ 116  
Common Mistakes ............................................. 116  
  Not Using Org Charts to Identify Stakeholders .......... 117  
  Only Including Project Team Members .................... 117  
Related Models ................................................. 117  
Exercise .......................................................... 118  
  Instructions .................................................. 118  
  Scenario ...................................................... 118  
Additional Resources ......................................... 118  
References ...................................................... 119

### Chapter 9  Process Flow  121

Process Flow Template ........................................ 123  
Example .......................................................... 126  
Creating Process Flows ....................................... 128  
  Create an L1 Process Flow .................................. 129
Create L2 Process Flows ..............................................130
Create L3 Process Flows As Necessary .........................132
Using Process Flows ..................................................133
Targeting the Audience with Different Levels of Detail ....133
Running Elicitation and Review Sessions ......................133
Driving Completeness ...............................................134
Deriving Requirements ..............................................134
When to Use ............................................................135
When Not to Use ......................................................135
Common Mistakes .....................................................136
Level of Detail Is Inconsistent Within a Flow ...............136
Reviewers Do Not Understand the Level of Detail ..........136
Reviewers Forget to Look at the Full Process Flow ........136
Process Flow Has Too Many Steps ...............................136
System Responses Are Mixed with User Actions ..........136
Processes Outside the Scope of the Project Are Not Included ....136
Related Models .........................................................137
Exercise .................................................................137
Instructions ............................................................137
Scenario ...............................................................138
Additional Resources .................................................138
References .............................................................138

Chapter 10 Use Case ..................................................139
Use Case Template ...................................................140
Example .................................................................141
Creating Use Cases ..................................................142
Identify Use Cases ...................................................143
Write Descriptions .................................................145
Capture Organizational Benefits ...............................145
Capture Frequency of Use .......................................146
Prioritize Use Cases ...............................................146
Complete the Remaining Header Fields ......................146
Write the Main Course ..............................................148
Chapter 11 Roles and Permissions Matrix

Roles and Permissions Matrix Template ........................................ 160
Example .................................................................................. 161
Creating Roles and Permissions Matrices ................................. 163
  Identify the Roles ................................................................. 164
  Identify the Operations ......................................................... 164
  Indicate Permissions ............................................................ 165
  A Word About When to Create the Matrix ......................... 168
Using Roles and Permissions Matrices ..................................... 168
  Deriving Requirements ......................................................... 168
PART IV  SYSTEMS MODELS

Chapter 12  Ecosystem Map  177

Ecosystem Map Template ................................. 179
Example ..................................................... 181
Creating Ecosystem Maps ............................... 183
   Identify Systems ....................................... 183
   Identify Interfaces ................................... 184
   Tie the Diagram Together ........................... 186
Using Ecosystem Maps ................................. 186
   Defining Scope with Ecosystem Maps ............ 186
   Deriving Requirements ............................... 186
   When to Use .......................................... 187
   When Not to Use ..................................... 187
Common Mistakes ............................ 187
   Showing Physical Systems .......................... 187
   Too Much Documentation ............................ 187
   Lack of Organization ................................. 187
Chapter 13 System Flow

System Flow Template ..................................................192
Example ......................................................................195
Creating System Flows ..................................................196
  Identify System Steps ..................................................197
  Write Steps ..................................................................198
Using System Flows.........................................................199
  Running System Flows Parallel to Process Flows ..........199
  Deriving Requirements ................................................200
  When to Use .................................................................200
  When Not to Use ..........................................................201
Common Mistakes ..........................................................201
Related Models .............................................................201
Exercise ....................................................................202
Instructions ..................................................................202
Scenario ....................................................................202
Additional Resources ....................................................202

Chapter 14 User Interface Flow

UI Flow Template..........................................................204
Example ......................................................................206
Creating UI Flows ..........................................................207
  Determine Scope of Screens .................................207
  Identify Screens ........................................................208
  Create Transitions .....................................................210
  Label Triggers ............................................................212
Using UI Flows ................................................................. 212
Identifying Navigation .................................................. 212
Validating Navigation .................................................... 212
Optimizing Usability ..................................................... 213
Developing Test Cases ................................................... 213
Deriving Requirements ................................................... 213
When to Use ................................................................. 213
When Not to Use ............................................................ 213
Common Mistakes .......................................................... 214
Including Too Much Detail ............................................... 214
Including Unimportant Details ......................................... 214
Not Using a UI Expert When Needed ............................... 214
Related Models .............................................................. 214
Exercise ....................................................................... 215
Instructions ................................................................. 215
Scenario ................................................................. 215
Additional Resources ..................................................... 215
References ................................................................. 215

Chapter 15 Display-Action-Response 217
DAR Model Template ..................................................... 219
UI Element Description .................................................. 221
UI Element Displays ..................................................... 221
UI Element Behaviors ................................................... 221
Example ................................................................. 222
Creating DAR Models ..................................................... 224
Prepare the Screen ......................................................... 224
Create a UI Element Description ...................................... 225
Create the UI Element Displays Section ........................... 226
Create the UI Element Behaviors Section ......................... 227
Guidelines for Creating Element Tables ............................ 227
Using DARs ................................................................. 228
Driving Completeness ................................................... 228
Deriving Requirements ................................................... 228
### Chapter 17 Decision Tree

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree Template</td>
<td>247</td>
</tr>
<tr>
<td>Example</td>
<td>248</td>
</tr>
<tr>
<td>Creating Decision Trees</td>
<td>250</td>
</tr>
<tr>
<td>Identify Decisions</td>
<td>250</td>
</tr>
<tr>
<td>Identify Choices</td>
<td>251</td>
</tr>
<tr>
<td>Identify Outcomes</td>
<td>252</td>
</tr>
<tr>
<td>Repeat Until Every Branch Ends in an Outcome</td>
<td>252</td>
</tr>
<tr>
<td>Simplify the Decision Tree</td>
<td>253</td>
</tr>
<tr>
<td>Using Decision Trees</td>
<td>253</td>
</tr>
<tr>
<td>Driving Completeness</td>
<td>253</td>
</tr>
<tr>
<td>Simplifying Logic</td>
<td>253</td>
</tr>
<tr>
<td>Modeling Nested &quot;If&quot; Statements</td>
<td>254</td>
</tr>
<tr>
<td>Training Users</td>
<td>254</td>
</tr>
<tr>
<td>Deriving Requirements</td>
<td>255</td>
</tr>
<tr>
<td>When to Use</td>
<td>255</td>
</tr>
<tr>
<td>When Not to Use</td>
<td>255</td>
</tr>
<tr>
<td>Common Mistakes</td>
<td>256</td>
</tr>
<tr>
<td>Modeling Process Steps</td>
<td>256</td>
</tr>
<tr>
<td>Making All Yes or No Choices</td>
<td>256</td>
</tr>
</tbody>
</table>
PART V  DATA MODELS

Chapter 19  Business Data Diagram  269

BDD Template ..................................................270
  Business Data Example Diagram Template .................272
Example ............................................................274
Creating BDDs .....................................................275
  Identify Business Data Objects  .........................276
  Relate Business Data Objects  .........................277
  Add Cardinalities ........................................277
Creating Business Data Example Diagrams ...............278
Using BDDs .........................................................279
  Understanding the High-Level Business Data Objects ...280
  Driving Completeness ................................281
  Identifying Processes ................................281
  Helping Technical Teams with Database Design ....282
Using Business Data Example Diagrams to Review BDDs .282
  Deriving Requirements .................................282
  When to Use .................................................282
  When Not to Use ........................................282
Common Mistakes ...............................................283
  Including Fields As Objects  .........................283
  Creating Middle-Man Objects  ....................283
  Thinking in Terms of a Database Design ........283
Related Models ...................................................283
Exercise ...........................................................284
  Instructions ...............................................284
  Scenario .................................................284
Additional Resources .........................................284
References .......................................................285
Chapter 20  Data Flow Diagram  287

DFD Template .................................................................288
Example .................................................................289
Creating DFDs .................................................................290
  Identify Business Data Objects ...................................290
  Identify Processes .......................................................291
  Identify External Entities .............................................291
  Tie the Diagram Together ...........................................292
Using DFDs .................................................................292
  Representing Data Used Across Multiple Processes ........293
  Using DFDs to Help with Readability .........................293
  Driving Completeness ................................................294
  Deriving Requirements ...............................................295
  When to Use ..............................................................295
  When Not to Use .......................................................295
Common Mistakes ........................................................295
  Trying to Articulate Order in a DFD .........................295
  Trying to Document Every Single Data Flow ...........295
Related Models .............................................................296
Exercise .................................................................296
  Instructions ..............................................................296
  Scenario .................................................................296
Additional Resources ....................................................297
References ...............................................................297

Chapter 21  Data Dictionary  299

Data Dictionary Template ..................................................300
  Properties List ..........................................................300
Example .................................................................304
Creating Data Dictionaries .............................................307
  Tailor Properties .......................................................307
  Identify Business Data Objects and Fields ..................308
  Populate Properties ..................................................308
  Supplement with Data Catalogs .................................309
Using Data Dictionaries ............................................. 309
  Promoting a Consistent Data Nomenclature ................. 310
  Driving Completeness ........................................ 310
  Deriving Requirements ....................................... 310
  When to Use .................................................. 311
  When Not to Use .............................................. 311
Common Mistakes .................................................. 311
  Becoming Overwhelmed by the Size ......................... 311
  Not Articulating Important Validation Rules ............. 312
Related Models ..................................................... 312
Exercise ............................................................ 312
  Instructions .................................................... 312
  Scenario ......................................................... 313
Additional Resources ............................................. 313
References .......................................................... 313

Chapter 22  State Table  ........................................ 315
State Table Template ............................................. 316
Example ............................................................ 317
Creating State Tables ............................................. 318
  Identify the Business Data Objects ......................... 319
  Identify the States .......................................... 319
  Analyze the Transitions .................................... 320
Using State Tables ................................................. 320
  Enhancing Readability .................................... 321
  Driving Completeness .................................... 321
  Deriving Requirements .................................... 322
  When to Use .................................................. 323
  When Not to Use .............................................. 323
Common Mistakes .................................................. 323
  States That Are Not States .................................. 323
  Missing States ................................................. 324
  Incorrect “No” Transitions .................................. 324
Creating Report Tables .................................................. 346
Identify Reports .......................................................... 346
Prioritize Reports ......................................................... 346
Complete Report Table Elements ................................. 346
Using Report Tables ....................................................... 349
Defining the Reports ...................................................... 349
Checking Completeness and Consistency Against Other
Models ............................................................................. 349
Deriving Requirements ................................................... 350
When to Use ................................................................. 350
When Not to Use .......................................................... 350
Common Mistakes .......................................................... 350
Not Relating Reports to Decisions Made ......................... 350
Documenting Unnecessary Reports .............................. 350
Related Models .............................................................. 351
Exercise ............................................................................ 351
Instructions ....................................................................... 351
Scenario ........................................................................... 352

PART VI MODELS IN THE BIG PICTURE

Chapter 25 Selecting Models for a Project 355
Selecting Models by Project Phases ............................... 355
Envision Phase ............................................................ 356
Plan Phase ........................................................................ 358
Develop Phase ............................................................... 358
Launch Phase ................................................................. 359
Measure Phase ............................................................... 359
Selecting Models by Project Characteristics .................. 360
Objectives Characteristics ............................................ 361
People Characteristics .................................................... 363
Systems Characteristics .................................................. 366
Data Characteristics ....................................................... 371
Project Examples .......................................................... 372
<table>
<thead>
<tr>
<th>Thinking About the Audience</th>
<th>374</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tailoring Models</td>
<td>375</td>
</tr>
<tr>
<td>Exercise</td>
<td>376</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Instructions</td>
<td>376</td>
</tr>
<tr>
<td>Scenario</td>
<td>376</td>
</tr>
</tbody>
</table>

### Chapter 26 Using Models Together

<table>
<thead>
<tr>
<th>Many Different Views</th>
<th>377</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using More Than One Model</td>
<td>378</td>
</tr>
<tr>
<td>Requirements Architecture</td>
<td>379</td>
</tr>
<tr>
<td>Relationships Between Models</td>
<td>380</td>
</tr>
<tr>
<td>Where the Artifacts Live</td>
<td>382</td>
</tr>
<tr>
<td>A Models Plan</td>
<td>383</td>
</tr>
<tr>
<td>Related Models</td>
<td>383</td>
</tr>
<tr>
<td>Exercise</td>
<td>394</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Instructions</td>
<td>394</td>
</tr>
<tr>
<td>Scenario</td>
<td>394</td>
</tr>
</tbody>
</table>

**Appendix A: Quick Lookup Models Grids**

**Appendix B: General Guidelines for Models**

**Appendix C: Exercise Answers**

**Glossary**

**Index**
The other day, I was trying to teach my friend how to play Texas Hold ‘em poker. In the game, there are only a few actions that you can take: call, raise, fold, or check; yet the game is very complex, and it isn’t at all obvious when you should take one of those actions.

I realized that there were just a few factors to consider, but that they created too many combinations to make me feel sure that I had covered all the important situations for my friend. Some of the factors going into the decision about your action are which seat you are in, how many other people have stayed in, how many have yet to act after you, the probability of improving your hand compared to the current pot size, and the type of each of the players.

For example, if the size of the pot compared to how much I have to put in is too small compared to my chances of improving my hand, then I usually fold regardless of any other factors. However, if everyone before me has folded and there are only one or two people after me, I might raise, unless I know that the players yet to act are the type to protect their antes (blinds). On the other hand, even if I have a good hand with good chances for improving it, if there are one or two really aggressive players still in, I might fold just to stay out of their way.

These kinds of multifactor decisions are very common in software development. The Decision Table is an RML systems model that helps you analyze all the permutations of complex logic in a comprehensive way. Decision Tables are used to answer the question, “Under what conditions will this outcome occur?” or “Given these conditions, what outcome should I choose?” The format of the model allows you to easily ensure that all possible conditions are being checked and acted upon properly. Decision Tables are used if there is no specific order to evaluating the decisions. If the decisions need to be made in any kind of order, a Decision Tree should be used instead (see Chapter 17, “Decision Tree”).
Decision Tables show all possible combinations of a set of conditions and their corresponding outcomes, represented in a grid. The Decision Table format allows you to ensure completeness. Because the full quantity of combinations is known, you can be 100 percent certain that you have looked at every permutation. It is easier for technical teams to use Decision Tables than tree structures because every option is shown in an organized format. A Decision Tree can get cumbersome when it attempts to show all possible conditions and outcomes.

The advantages of a Decision Table over a Decision Tree are visual traceability and speed. You can instantly trace an outcome back to the conditions that cause that particular outcome. You can also always trace all the potential outcomes from a condition. A table is faster to create than a Decision Tree, because with Decision Trees you have to rearrange your branches for every new branch that you add. If you decide to create both, it usually is best to create a Decision Table first, before you create your tree, so that you have an idea of what the layout should look like.

**Decision Table Template**

A Decision Table is represented as a grid, as shown in Figure 16-1. The top row contains labels for the business rules. The first column contains all the possible conditions and outcomes. Each of the remaining columns in the grid represents the outcomes valid for the specified choices. Collectively, the combination of choices and outcomes in a column make up each business rule.
A condition is an individual check such as “Lives in the United States - Yes or No” or “Marital status - single, married, divorced, or widowed.” We use the term “condition” instead of the more common term “decision” because “decision” implies that there is an order to the decision process, whereas “condition” does not.

The choices for a condition are the set of possible values for that condition. Choices can be binary, such as Yes or No or True or False; they can be multi-valued, such as ages 0-9, 10-21, 22-34, and 35+; or they can be a dash (-), meaning that the choice is irrelevant. The outcomes can have unique identifiers on them to help reference them from requirements.

Each outcome cell either contains an X, a number, a dash, or is left blank, as explained in Table 16-1.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Rule 1</th>
<th>Rule 2</th>
<th>Rule 3</th>
<th>Rule 4</th>
<th>Rule 5</th>
<th>Rule 6</th>
<th>Rule 7</th>
<th>Rule 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition 1</td>
<td>choice 1a</td>
<td>choice 1b</td>
<td>choice 1a</td>
<td>choice 1b</td>
<td>choice 1a</td>
<td>choice 1b</td>
<td>choice 1a</td>
<td>choice 1b</td>
</tr>
<tr>
<td>Condition 2</td>
<td>choice 2a</td>
<td>choice 2b</td>
<td>choice 2a</td>
<td>choice 2b</td>
<td>choice 2a</td>
<td>choice 2b</td>
<td>choice 2a</td>
<td>choice 2b</td>
</tr>
<tr>
<td>Condition 3</td>
<td>choice 3a</td>
<td>choice 3b</td>
<td>choice 3a</td>
<td>choice 3b</td>
<td>choice 3a</td>
<td>choice 3b</td>
<td>choice 3a</td>
<td>choice 3b</td>
</tr>
</tbody>
</table>

**FIGURE 16-1** The Decision Table template.

A rule is the particular set of choices for each condition that have to match for the outcomes to apply. For example, the condition “lives in the United States = yes” and the condition “Marital status = single” could form the choices that make up a rule that says a specific outcome applies. The rules don’t have names; they are just permutations of all the possible choices of all the conditions and valid outcomes. Rules eventually become business rules in your requirements. For example, in the template, if the choices choice 1a, choice 2a, and choice 3a are true, then outcomes 2, 3, and 4 apply.

**TABLE 16-1** Outcome and Choice Intersection Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Outcome applies when the choices are valid</td>
</tr>
<tr>
<td>Number</td>
<td>Outcome applies when the choices are valid; the outcomes should be executed in a specified order</td>
</tr>
<tr>
<td>-</td>
<td>Outcome is irrelevant (does not apply) when the choices are valid</td>
</tr>
<tr>
<td>Blank</td>
<td>Outcome is unknown; follow-up is needed</td>
</tr>
</tbody>
</table>

**Tool Tip** Decision Tables are typically created as Microsoft Excel or Microsoft Word tables or in a requirements management tool that allows you to create tables of requirements.
Example

An insurance company has a formalized process for determining which homeowner policies a customer is eligible for. Their decision process is modeled in the Decision Table in Figure 16-2. The model provides the assurance that all possible combinations are considered, because every combination of choice for every condition is included.

For example, the Decision Table tells us that if a customer has passed a credit check, has an existing policy, and has been a homeowner for 20 years, then he is eligible for policies A, B, and C. If the same customer did not have an existing policy, then he would be eligible for policies A and B.

Looking at the complete table, you can see where unnecessary columns can be pruned. For example, being a homeowner and having existing policies with the company are irrelevant if the person fails the credit check. Figure 16-3 shows the simplified table.

Creating Decision Tables

The high-level steps for creating a Decision Table are outlined in Figure 16-4. We suggest an order to the steps; however, you might do this differently, as discussed later in this chapter, in the “Identify Outcomes” section. To illustrate how to create Decision Tables, this section continues with the example scenario from the previous section.
Identify Conditions

Think about all the potential conditions that apply to the situation and list them in the first column of your table. Conditions can be decisions that people make, data attributes that trigger various business rules, or any other factor (Gottesdiener 2005). Each condition needs its own row in the table. If you combined multiple conditions, it would be very hard to check for completeness. Figure 16-5 shows the first part of the table with just the possible conditions. A condition will often reference specific data fields from a Data Dictionary. In those cases, use the `<object.field> notation, as described in Chapter 21, “Data Dictionary.”

<table>
<thead>
<tr>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passed credit check</td>
</tr>
<tr>
<td>Existing policy</td>
</tr>
<tr>
<td>Total years as homeowner</td>
</tr>
</tbody>
</table>

FIGURE 16-5 The conditions.

Identify Choices

When you identify choices, you first need to determine what the choices are, then you might re-order your conditions in the table based on the choices, and then you enumerate the choices to build the rules columns.

Determining Valid Choices

After you have the conditions, consider the possible choices for each condition. Look at your Data Dictionary to identify valid values for the field. Some conditions will have binary choices of Yes and No or True and False, but the choices can be more complex, such as a range of numbers or matching words. There is no requirement that the conditions all be binary; in fact, the table can be simpler if a single multi-choice condition is used instead of several binary conditions. For example, if the condition is “Years as a homeowner,” you might have three potential choices for that condition that should be modeled: more than five years, one to five years, and less than one year.

Make sure that the choices you use reflect all possible choices for the condition. If you miss any possible choices, your table will be incomplete and you cannot guarantee that you have all of the requirements. For a particular condition, the choices must also be mutually exclusive, in that only one can exist at a time. A condition cannot have overlapping ranges such that each could simultaneously match because they share values. For example, the ranges 0–1 and 1–5 overlap because each includes the number 1. If you need to specify ranges like this, use the following type of range definitions instead: <1 and 1–5, or <=1 and >1–5.
Reordering Conditions
You do not need to list the conditions in any specific order; however, it is generally easier to make a table when they are listed in the order of number of choices. The condition that has the fewest choices should be the first condition. For example, if you have a rule such as, “If the customer failed the credit check, then automatically decline him,” then you can cut out half of your permutations. You don’t need to do any work to evaluate the rest of the conditions; they just don’t matter. The condition that has the most choices should be the last condition listed. This order puts the most constraining conditions at the top of the list, which aids you in reviewing and simplifying the table.

Enumerating Choices
Now you add rules (columns) to enumerate all possible combinations of choices for all conditions. If you multiply the number of choices together for each condition, you can determine how many columns you need. If you have just one condition with two choices, you will need only two columns. If you have three conditions with two choices each, you will need eight columns (2 x 2 x 2 = 8). In this example, there are two choices for the first condition (Y or N), two for the second condition (Y or N), and three for the third condition (>5, 1–5, or<1), so there are 12 columns (2 x 2 x 3 = 12).

In addition to each choice being mutually exclusive, each rule needs to be mutually exclusive. This means that you should have no columns with the same set of choices.

The order of the rules should make it easy to recognize that all possible choices of conditions are being examined. In the first row, group all the like choices together. In the example, for the first condition, half will be Y and half will be N, so make the first four columns Y and the second four N, as in Figure 16-6.

![Figure 16-6](image)
The first condition with choices.

For the second condition, half of the Ys from the first condition will be Y and half will be N. This results in the pattern in Figure 16-7.

![Figure 16-7](image)
The second condition with choices.

Finally, in the third condition, alternate the three choices again to get the pattern in Figure 16-8. When you complete the conditions and choices of a Decision Table, you will get a result that looks like this. Remember that you will have more columns if you have more conditions or valid choices for conditions.
CHAPTER 16 Decision Table

239

FIGURE 16-8 All conditions with all choices enumerated.

Identify Outcomes

The combinations of choices lead to one or more outcomes. Outcomes are the decisions, conclusions, or actions that occur when the choices are valid. Add the outcomes to the first column of your table, below all of the conditions. Figure 16-9 shows our same scenario with outcomes added.

FIGURE 16-9 Outcomes.

You might find it easier to identify the outcomes first. If you know your outcomes and want to determine the conditions under which they are valid, start with the outcomes part of the table and then identify conditions and choices. It is possible that you might start with a few business rules, so creating the table with just a few rules filled in can quickly show where there are gaps in your information.

Label Valid Outcomes by Choice Combinations

Each column now represents a possible combination of choices for the conditions. Label the valid outcomes under the combination of choices. If you know that an outcome is not valid, mark it with a dash, and if you are not sure, leave it blank to follow up later. Figure 16-10 shows the resulting Decision Table for the scenario.

FIGURE 16-10 The complete Decision Table.
Simplify the Decision Table

Because so many columns are necessary even with just a few conditions, it is best to try to simplify your table as early in the process as feasible. If there are specific rules that are only dependent on a few of the conditions, that means that you do not need to evaluate other conditions. You should combine the rules to remove the ones that differ in conditions and not in outcomes.

When a condition is irrelevant to the outcome, put a dash in the cell for that condition. Do not just haphazardly delete columns, though! You must ensure that all combinations are included in the table, using the dash to represent those conditions whose choices do not matter. For example, you know that when a customer fails the credit check, she is declined, so you do not need to evaluate the existing-policy or years-as-a-homeowner conditions. Figure 16-11 shows the example after it has been simplified.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Rule 1</th>
<th>Rule 2</th>
<th>Rule 3</th>
<th>Rule 4</th>
<th>Rule 5</th>
<th>Rule 6</th>
<th>Rule 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passed credit check</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Existing policy</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Total years as homeowner</td>
<td>× 5</td>
<td>1 - 8</td>
<td>&lt; 1</td>
<td>× 5</td>
<td>1 - 8</td>
<td>&lt; 1</td>
<td>-</td>
</tr>
</tbody>
</table>

**FIGURE 16-11** The simplified Decision Table.

In simplifying your table, you might discover that some conditions are always irrelevant. You can remove those conditions and further condense your table. Finally, if, after you complete your table, you see that certain outcomes are always executed in tandem, you can combine those outcomes into a single outcome.

Using Decision Tables

Decision Tables can make very complex decisions appear orderly and complete because the table communicates a lot of information in a very compact format.

Making Decisions

A business can use a Decision Tree for training users how to make decisions, but a Decision Table is hard to read for these purposes. More commonly, a Decision Table is implemented in the system to make the decisions automatically.
Driving Completeness
The value provided by Decision Tables stems from the ability to walk through all possible combinations of choices. Assuming that every outcome, condition, and possible choice is identified, you can literally see that all the potential choices have been considered, and therefore your requirements for the decision scenario can be considered complete.

When you are analyzing Decision Tables, if you have any cells you do not know about, leave them blank and track them for follow-up.

Using Decision Tables with Decision Trees
Decision Tables are used to identify all possible conditions and choice combinations and their resulting outcomes. You can also use them to eliminate combinations that are not relevant. Finally, you can use the Decision Table to build a corresponding Decision Tree to better visualize the decisions and look for additional opportunities to simplify the logic. Generally, if you plan to create an ordered Decision Tree, you should order the conditions in the Decision Table by using the same order that you use to make the decisions in the ordered Decision Tree.

Deriving Requirements
A complete Decision Table indicates that you have modeled all of the requirements and business rules related to a particular set of decisions. You might also need to write out individual statements from your table for developers and testers to work from. Each complete column of the Decision Table represents one business rule you need to write; the rule describes the conditions under which the outcomes occur. Decision Tables can sometimes act as stand-alone requirements for developers and testers.

When to Use
Decision Tables are best when the order of the decisions does not matter. If the order does matter, you can use Decision Tables to identify all combinations of conditions and simplify them before you create a Decision Tree, but you still must create an ordered Decision Tree.

If you are in an elicitation session and have no time to prepare in advance, use a Decision Table with your business stakeholders to elicit an initial draft of the logic, because you can create Decision Tables very quickly.

Modeling Complex Logic
Decision Tables are usually supplements to System Flows (see Chapter 13, “System Flow”), UI Flows (see Chapter 14, “User Interface Flow”), Process Flows (see Chapter 9, “Process Flow”), or Use Cases (see Chapter 10, “Use Case”). The Decision Table model is used to simplify any of these models by removing the complex decision logic from the main flow. This allows the audience to focus on the overall picture of the flow in those models without getting lost in the details.
It is very useful to use Decision Tables alongside these other models when there are validation steps. For example, in a Process Flow to create an order, there might be complex logic to check that the proper fields are completed and formatted correctly, with different error paths if they are not. These validation steps can be represented in a Process Flow, but if there are many of them, it would be easier to read both the Process Flow and the logic steps by putting the validation into a Decision Table. Instead of putting them all in the Process Flow, the analyst can include one simple decision box in the Process Flow, such as, “Are order fields valid?” and can then reference the Decision Table, where that logic is further analyzed.

A Decision Table should be created any time you have a scenario with a series of nested “if” statements. If you find you need to write two to three “if” statements in a row in a Process Flow or Use Case, you probably should use a Decision Table to model them instead.

Particularly if the order of the decisions is not relevant, a Decision Table is usually the best choice, especially if each combination of choices leads to a different set of outcomes. The visual structure of a Decision Table allows you to quickly see the logic represented by your decisions. A five-by-five table is much easier to understand and review than the 25 separate “if” statements that would have to be written up to describe the information in the table.

**When Not to Use**

Decision Tables are not ideal for documenting decisions that cause you to move around in your decision hierarchy. If you need to show any order to your decision-making process, you simply cannot use a Decision Table as your only model, because these models do not show order. Similarly, you cannot use a Decision Table if you want to show any loops in your logic. You can use a Decision Table to do your initial analysis to prune the tree, and then follow up with the Decision Tree to show the order of the decisions.

**Common Mistakes**

The following represent the most common mistakes we have seen with Decision Tables.

**Missing Permutations**

Make sure you calculate the number of possible rules to ensure that you gather them all.

**Overlapping Choice Ranges**

Ensure that your boundary conditions for ranges don’t overlap; be explicit by using >=, <=, >, and < symbols.
Not Combining Rules
Decision Tables can easily become large and unwieldy if you don’t reduce the total number of rules. Make sure that you have identified any conditions that are not important for a rule, and then remove the rules that vary by that condition and not by outcome.

Modeling a Sequence of Decisions
Decision Tables should not be used if you are trying to model a sequence of decisions that occur in order; a Decision Tree is more appropriate.

Related Models
Decision Tables outside of RML sometimes call the conditions factors and the outcomes actions. Most Decision Table nomenclature allows you to have a blank cell when an outcome does not apply, but we recommend that you do not leave cells blank, because you will not be able to tell whether blank means “not valid” or “I haven’t determined whether this is valid.”

The following list briefly describes the most important models that influence or are enhanced by Decision Tables. Chapter 26, “Using Models Together,” contains a more thorough discussion about all related models.

- **Decision Trees** These are used to visually show the decision logic in a tree structure. Also, the Decision Table can help you to create a Decision Tree.

- **Process Flows, System Flows, Use Cases, and User Interface (UI) Flows** Decision Tables are used to model complex logic found in them.

- **Data Dictionaries** The valid choices for conditions are based on data in the Data Dictionary and should use the <object.field> notation.

Exercise
The following exercise is intended to help you to gain a better understanding of how to use this model. The exercise is open ended, and therefore the answer you come up with could be substantially different than the answer that we have provided. There are potentially many correct solutions. The answer provides an explanation of how we arrived at our solution. You will gain the most out of the exercise by attempting to do it yourself before looking at the solution. The answers for the exercises can be found in Appendix C.
Instructions
Create a Decision Table for the following scenario.

Scenario
You are on a project to launch a new eStore to sell flamingos and other lawn decorations. You are capturing the rules for handling payment information. You decide a Decision Table will be the best model to capture the rules the system must implement to handle the different forms of payment. You know that the customer can store credit cards in his profile, but he can also choose to pay with a new credit card, check, or gift card. If he pays with a gift card, he will have to pay any balance with another payment form if there isn’t enough money available on the gift card. Use this information and your general knowledge about online payment options to create a Decision Table.

Additional Resources

- *The Software Requirements Memory Jogger* has an overview of Decision Tables with a good example in Section 4.11 (Gottesdiener 2005).
- Wieger’s *Software Requirements* has an example of a Decision Tree and Decision Table (Wiegers 2003).
- Chapter 4 of Davis’s *Software Requirements Revision* has a description of Decision Trees and Decision Tables (Davis 1993).

References

CHAPTER 25

Selecting Models for a Project

When I decide to work on a project around my house, I start by selecting the tools to use. I first consider what type of project I’m doing and then consider what step in the overall project I’m working on, so that I can pick the right tool. For example, if I’m cooking, I use tools such as a mixer, measuring spoons, a rolling pin, a frying pan, and a spatula. I can further narrow my tool selection by the type of cooking I’m doing. If I’m baking, I would more likely use a mixer than a frying pan. Finally, I pick the tools I need based on which steps in my cooking project I’m executing. Initially, I use measuring spoons and cups, then I use a mixer, then I might use a spatula or rolling pin, and finally I might use a baking dish and cooling rack near the end.

On the other hand, if my task is to build a bookshelf for the house, I would select different tools than those I use for cooking. I could use a hammer, a screwdriver, a saw, nails, and screws. Similarly, I can further narrow my tool selection at any point based on the step I’m at in the project. Early in the project, I might use a saw to cut the wood, but later I would use the hammer and nails to assemble the bookshelf.

To select tools for projects around my house, I determine what I’m trying to accomplish and then decide which tools best help me meet those goals. There are many RML models, and selecting the right models to use for specifying a solution is similar to selecting tools for home projects. The task can be overwhelming if all you have is a list of models without a framework to help you narrow the list.

This chapter will help you select models for your project based on the project phase you are executing and the project characteristics. For each of these factors, we overlay the model categories to ensure that you consider all types of models.

Selecting Models by Project Phases

Models are a key component of every stage of the software process. It’s important not to lose sight of the overall project approach and become bogged down in the minutiae of creating models, but rather to understand how the models fit into the approach to ensure a complete set of requirements during the entire development cycle. Figure 25-1 shows a generic requirements process and the activities that occur within each phase. The phases are envision, plan, develop, launch, and measure. Within each phase there are requirements-based activities. These process phases map to phases in virtually any development approach (such as waterfall, iterative, agile, and custom approaches). The focus is on the activities within the phases rather than how the phases are executed. The key is that no matter what your development approach is, within each phase there are activities that benefit from using requirements models.
Using the appropriate models helps all stakeholders understand the requirements so that nothing important is missed and only the requirements that are important are implemented. The following sections further discuss the role of models in each phase of the requirements process. Although these are described in an order similar to a waterfall approach, they can be applied in any development methodology.

The information from the following sections is summarized in a table in Appendix A, “Quick Lookup Models Grids.”

**FIGURE 25-1** A requirements process.

**Envision Phase**

The Envision phase occurs before a project is actually chartered. Executives who own corporate strategy and manage a portfolio, program, or product roadmap have to determine which projects to fund based on the value they add to the organization. This phase determines how the project supports the corporate or program strategy, what value the business stakeholders will receive from it, and what high-level features the stakeholders need to achieve that value.
During the Envision phase, a business analyst helps executives explore the business problems, determine the business objectives that all requirements eventually map back to, develop the business case for the project, and set a broad scope for the project. The business problems and objectives are captured in a Business Objectives Model, and the highest-level features are captured in a Feature Tree. The business objectives enable business analysts to determine the project’s priorities, which will be used throughout the entire project to help the team make decisions when cutting scope or when trying to decide which features to develop first.

You should also create initial high-level models such as Org Charts and Ecosystem Maps. Org Charts are a great starting point for creating people models, because they offer the opportunity to consider every single person or role that might affect or use the system. Though Ecosystem Maps do not identify specific requirements, they do identify the interfaces for which to elicit requirements and the systems that might be affected by the project. In addition to models, you should create an issues list to log every outstanding requirements-related unresolved issue or question. If you are not using a requirements management tool, then typical documents that you might produce during this phase might have names such as “business case,” “business requirements document,” “marketing requirements document,” “product backlog,” and “vision scope document.”

Models in Agile

Many organizations try to depict agile methods as mini-waterfalls. Even the requirements process diagram in Figure 25-1 could be interpreted as a mini-waterfall. However, it should not be interpreted in that way. The diagram is simply intended to classify the activities that a business analyst would perform regardless of the development approach. In a waterfall approach, each phase would require signoffs before the project could enter the next phase. In an agile project, all of the phases could be executed simultaneously within a particular sprint. However, even when you are using an agile approach, at some level (perhaps at the individual story level) you have to figure out the value of the story first (Envision); then the details of the story, such as acceptance criteria (Plan); then your developers build it (Develop), potentially updating the story as you gain a better understanding of the project; and then you test and deploy what you have just built (Launch). Within a sprint these phases might be occurring every single day, simultaneously.

Even when you are using an agile approach, on very large projects with many teams that have to synchronize their work, develop some of the models up front before any development begins. This is the best way to ensure that each team has a shared understanding of the overall business goals and system requirements. Models such as the Business Objectives Model and the Objective Chain help to ensure that all teams really understand the value of the project. Org Charts, Ecosystem Maps, Business Data Diagrams, and Process Flows ensure that all teams have a shared understanding of how users will need to use the system and the environment that they are working in. A product backlog is great for managing the project, but it provides no framework for determining which things need to be in the backlog in the first place. Models can be used to populate the backlog and determine any detailed requirements and business rules within each sprint.
Plan Phase
During the Plan phase, you are trying to determine how the software needs to work so that it will achieve the value expected by the business. By listing out all the requirements, including the functions the software needs, the business rules that influence the functions, and the non-functional qualities, you will create a complete list of requirements that your business stakeholders, developers, and testers can use to build, configure, or test the system to ensure that the business value is achieved. Identify which models are needed based on the characteristics of the project, and define a requirements architecture (described in Chapter 26, “Using Models Together”). Finally, you might select tools to create and store models that support the requirements architecture, and create a requirements plan that outlines when each of the models will be created in relation to the others.

You should test out your requirements architecture and process on a portion of the project, adjusting it as you determine what works best for your organization. Project priorities can change; analysts might find that the choice of models is not sufficiently capturing requirements, or a variety of other reasons can force the team to alter the requirements architecture. In these scenarios, it’s important to review the already created requirements and models to ensure that no additional artifacts need to be created or existing ones altered; changes to the requirements architecture often affect prior work.

This phase is where you will do most of the elicitation and analysis to complete models, working from a high level to more detailed models (see Chapter 26 for more about this). You also should derive requirements from models. In addition, you should continue to update your requirements issues list as you determine parts of models that you cannot yet complete.

During this phase, you might create a Key Performance Indicator Model (KPIM) to articulate how the project will improve or at least maintain the throughput of the business. You will also create Objective Chains to determine the scope of the requirements. You should create Business Data Diagrams (BDDs) and Process Flows to define and bound the project scope. The remainder of the models you need on the project will typically be created in this phase.

Further, you can use the Business Objectives Model to prioritize the analysis work you have to complete. If you are not using a requirements management tool, examples of documents that you might create during this phase include business requirements documents, system requirements specifications, functional requirements specifications, sprint backlogs, and user stories.

Develop Phase
After the models have been created, validated, and verified, the next step is to ensure that the development and testing teams understand what they need to build. Developers and testers will use models and corresponding requirements and business rules to build code, configure existing systems, and develop test cases. During your model creation, you should have derived requirements for most of the models. This is the phase for which it is most important to have done that step, because developers and testers have an easier time knowing that their job is done when they have a list of functional requirements and business rules that they can use as a “checklist” rather than just a model to build and test from.
During this phase it is important to fully explain the usage of the models to these teams; otherwise, the models might be interpreted and used in a manner for which they were not intended. For example, developers might use only the Use Cases and not look at the requirements if they do not understand how the two fit together. Further, they might look at only the requirements and not the models, missing the important context you set with the models.

It is common for this phase to also include steps for updating models that are now in a “locked” state as changes occur. This means that all changes to requirements and models must be approved and communicated to teams downstream of the business analysts because those teams might have created artifacts using those requirements and models.

During this phase, you should be maintaining your issues list as well as updating all documents and models as necessary. You should be engaged with the developers as they build the solution, and with testers as they ensure that the system is built properly. Your role is to clarify how the software should work, update priorities, and ensure that what is being built works the right way. No matter how well you document the requirements, there will always need to be verbal clarifications, especially as you get into the details of the business rules, usability, and detailed functionality of the system. As you explain the functionality, you will most often be using models to help other team members understand the context. For example, you could use a Process Flow to help the developers understand what the user is trying to accomplish, then use a BDD to explain how the business thinks about the data elements. During the development phase, you will be updating your existing documents and might create documents such as “user acceptance tests” or versions of previous documents tailored to specific development teams.

Launch Phase

During the Launch phase, the business stakeholders confirm that the solution meets their needs. They can use Process Flows and Use Cases to create user acceptance tests to perform this verification. When the system has stabilized to the point where the business evaluates and accepts the solution, it is ready to be deployed. Process Flows, Use Cases, Roles and Permissions Matrices, and Display-Action-Response (DAR) models can be used to create training materials for the new system. Typical documents created during this phase could include training manuals, user guides, help files, and any other materials that will ensure that users have high satisfaction and adoption rates.

Measure Phase

The Measure phase takes place after a new system is live and the users have adopted the system for their use. During this phase, analysts can measure the return on investment of the solution’s business objectives and use key performance indicators (KPIs) to truly determine the value that the project brought to the organization. By using real data and a live measurement, it is possible to confirm that the business objectives set at the start of a project in the Business Objectives Model were actually achieved. Furthermore, the organization can measure individual business processes to ensure that they have met KPI targets described in KPIMs. Documents created during this phase might include presentations to executives describing the return, lessons learned documents, or retrospectives.
Selecting Models by Project Characteristics

In addition to the project phase, you should take into account the characteristics of the project when selecting which models to use. Typically, the first questions to ask are:

- Is this system being custom built from scratch, or will it be purchased from a vendor?
- Will the system replace an existing system entirely, result in a new implementation, or enhance an existing system?

There are several common project characteristics that can be used as guidelines to help determine which models to use. The list of project characteristics in the sections that follow is not meant to be comprehensive but is meant to give you a starting point for determining which models might be appropriate for your project. Project characteristics are not mutually exclusive, so you probably will have multiple characteristics that apply. For example, a project to replace an existing system can also be a cloud implementation. A system that will exist in a large ecosystem might also be an analytics system.

A Model to Select Models

Notice that we actually use a model to communicate how to select models by project characteristics. We use a grid that has all project characteristics down the left side and all models across the top. However, leaving the grid in that form would have given us 20 items in the list of characteristics, which is far more than 7+/-2 items. To make the grid more consumable, we further divided the characteristics into Objectives, People, Systems, and Data categories. Each project characteristic section in this chapter shows the relevant row of that grid. The complete grid is in Appendix A, for your quick reference later.

To select models using this section, first decide which project characteristics apply to your project, and then consider the suggested models for those characteristics (shown in rows in the grid in each section) to determine which models will be helpful on your project. The recommended models grid shown in each project characteristic uses the key in Table 25-1.

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Cell Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likely to be needed</td>
<td>L</td>
</tr>
<tr>
<td>Might be needed</td>
<td>M</td>
</tr>
<tr>
<td>Not needed based on this characteristic alone</td>
<td>blank</td>
</tr>
</tbody>
</table>

The meaning of blank cells is tricky and is important to understand so that you can interpret this information correctly. If a cell is blank, it means that the particular characteristic alone does not indicate the need for that particular model. The project might still need the model, though, because of another relevant project characteristic. Conversely, if the cell is filled in, it means the characteristic alone is sufficient to indicate that the model is needed for that type of project. For example, you
might notice that the Business Objectives Model is not indicated for most of the characteristics. That's because the use of a Business Objectives Model is dependent on a project having one of only a few of the characteristics; however, those characteristics are actually pretty common, and at least one of them will apply to almost all projects. Also, a project for implementing enhancements to replace existing systems functionality might need a KPIM, but it needs the KPIM because the project is replacing an existing system, not because it is an enhancement.

Objectives Characteristics

The following objectives characteristics help determine which objectives models are needed based on the type of project implementation.

Greenfield Projects

A greenfield project is one in which a brand new system is custom built from scratch because no system currently exists to provide the needed functionality. A primary consideration in these projects is scope, because it can easily balloon out of control. Many future users of these systems will offer input on features and requirements that they have for the system, and it is important to prioritize these requests in the context of a Business Objectives Model and Objective Chains. You should create a Feature Tree to show all planned features, and then tie each feature to a business objective so that the organization only builds the features that have the most value. A Requirements Mapping Matrix (RMM) is important to ensure that requirements ultimately map back to business objectives through other models.

Commercial Off the Shelf (COTS) Projects

The goal of a COTS project is to evaluate and select a third-party solution to solve a business problem and then implement it. We have broken the COTS project characteristic into two aspects, selection and implementation, because in many organizations they exist as completely separate efforts with differing model needs. Often the selection phase proceeds, but then COTS implementation does not occur because the team decides to build a new system or improve the existing system.

COTS selection The selection phase involves qualifying vendors, determining what the primary business objectives are, and ultimately selecting a system that meets an organization's needs. Creating a Business Objectives Model and Objective Chain might be important to ensure that the selection
process focuses on the systems that provide the greatest return on investment (ROI) as defined by the business objectives.

During the selection phase of a COTS system, many organizations create a list of features that the organization needs. This method is extremely weak because it does not address how the software satisfies the business process. To resolve this, you should use Process Flows and KPIMs to prioritize the Process Flows and determine the features that support the most critical business processes. You should use an Org Chart to ensure that you are talking to all of the right people about their Process Flows. During interviews with the vendor, you can use the highest-priority Process Flows to have the vendor demonstrate exactly how those processes and KPIs would be satisfied by using their software. You might still want to create a Feature Tree to help you quickly summarize the features deemed to be most important. An RMM helps you keep your features prioritized. In addition, a BDD should be used to ensure that there are no major discrepancies in data models between the software and the business needs. A Data Dictionary might not be needed because the individual fields typically do not require major changes. Finally, an Ecosystem Map should be used to ensure that there is a good understanding of the integrations that the COTS system will need to support. The vendor should be prepared to address gaps between the models and the COTS system.

<table>
<thead>
<tr>
<th>COTS selection</th>
<th>Objectives</th>
<th>Business Objectives Model</th>
<th>Key Performance Indicator Model</th>
<th>Feature Tree</th>
<th>Requirements Mapping Matrix</th>
<th>Org Chart</th>
<th>Process Flow</th>
<th>Use Case</th>
<th>Roles and Permissions Matrix</th>
<th>System Flow</th>
<th>Display/Action Response Model</th>
<th>Decision Table</th>
<th>Decision Tree</th>
<th>System Interface Table</th>
<th>Data</th>
<th>Business Data Diagram</th>
<th>Data Dictionary</th>
<th>Data Table</th>
<th>State Diagram</th>
<th>Report Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**COTS implementation** The implementation phase of a COTS project could be part of an existing system replacement project, but it could also represent installation of a completely new system. If the COTS system is replacing an existing system, and there is little customization, KPIMs are better because they help ensure that business throughput is maintained at desired levels. If there is significant customization, then there are features that you can map to business objectives by using a Business Objectives Model and Objective Chain. If the COTS system is not replacing an existing system or is introducing a significant amount of new functionality, defining the business objectives and their relationship to features by using a Business Objectives Model and Objective Chain is crucial to prioritizing features.

Org Charts can help ensure that all existing users are represented. Process Flows ensure that those users’ functionality needs are understood. The RMM is helpful for ensuring that all requirements of the business processes are implemented in the new system. A Roles and Permissions Matrix is useful because many COTS systems allow you to configure roles out of the box, so this helps you decide who should have those roles and what permissions those roles should have. Ecosystem Maps and System Flows are important if the COTS system will be deployed in an existing ecosystem, to help identify
integration points. BDDs and Data Dictionaries help ensure that data used in existing systems or processes is considered, whether it is to be converted to a new type of data or not. Most COTS software comes with standard reports, so Report Tables are important to define how those are deployed. Use Cases might be used to describe how users will interact with the COTS software directly. DAR models might be useful because tables can be created for each element in the UI that is configurable, to help capture the configuration requirements.

Enhancement Projects
An enhancement project makes a major change to an existing system by adding new functionality to the system’s capabilities. Because enhancement projects tend to primarily focus on new capabilities, focusing on mapping features to business objectives is paramount to minimize gold plating. The Business Objectives Model and Objective Chain are extremely important, because they allow you specifically to restrict the solution’s scope to the appropriate people, systems, and data. The RMM will help you continue the traceability by mapping requirements to other prioritized models to control scope. The Feature Tree can provide a quick view of all features that are in scope for implementation.

Because an existing system is being modified, often the data model is not changing. In these cases, a BDD is not necessary. By the same token, if the new features do not require additional integration, then an Ecosystem Map might not be necessary either.

People Characteristics
The following project characteristics are related to the users who use the system.
Systems with Extensive User Interaction

Systems with extensive user interaction are defined as those that have many users performing many types of operations in the system. You should focus on people models first. Typically, Process Flows will be used in a project that is heavily driven by the user interface (UI). Use Cases might be helpful to further describe user interactions. Roles and Permissions Matrices are helpful if you have a limited number of roles or types of user and need to implement a security model in the user interface. UI Flows and Display-Action-Response (DAR) models will be two of the most important models, because they illustrate visual aspects of the solution that Process Flows and Use Cases cannot capture. Each step in the Process Flows can then be linked to a DAR model to illustrate the UI in the level of detail needed.

If you are replacing an existing system or automating a process, KPIMs will probably be helpful because of the increased focus on end-user throughput and completion of tasks. If your project is an enhancement with an extensive UI, these models will be key in order to allow you to keep a consistent look and feel with the current software.

Customer-Facing Systems

Customer-facing systems primarily have users that are external to the organization that is implementing the system. There are typically a few internal roles as well, such as administrative roles, but most users are from outside the organization.

Because the users are external, Org Charts are typically not helpful. Roles and Permissions Matrices might be used to define the type of permissions necessary for security in the system. Process Flows and Use Cases might be important to describe how the customers will use the system. UI Flow and DAR models should be created to ensure that the user interface is easily navigated and used by external users. This is important even if there aren’t very many customer-facing screens.
Business Process Automation Projects

Business process automation projects either fully or partially implement the business’s processes in a system. These projects use KPIMs because the performance of existing manual processes can be measured to ensure that the performance levels are either maintained or improved in the new system. Org Charts are important to ensure that you talk to all existing business stakeholders who execute the process. Process Flows can be used to document existing business processes that will be performed using the new system. Use Cases might help describe how those activities will occur in the system. Furthermore, future-state Process Flows and Use Cases can illustrate how the system should function and provide a basis for new training guides.

Roles and Permissions Matrices might be important for putting a security model in place. BDDs and Data Dictionaries will be necessary to describe the business data objects and fields used and manipulated in the process.

Workflow Automation Projects

A workflow is a specific type of business process that has a heavy emphasis on approvals and routing of information between groups. A project that automates a workflow typically requires a Process Flow to describe context for the workflow. These projects usually have a BDD to show the business data objects that are manipulated during the workflow. State Tables and State Diagrams can help show how the objects change state during the workflow. Typically, these projects have security needs related to who can perform functions at different steps in the workflow, so Roles and Permissions Matrices are helpful.
Systems Characteristics

The following project characteristics are related to the type of system being worked on.

System Replacement Projects

A system replacement project replaces an obsolete solution with either a custom-built system or a COTS system. When an existing system is being replaced, the business objectives are often related to goals such as improving throughput, performance, reduction in license fees, or reduction in maintenance costs. These goals are often not achieved through implementation of specific new features. Even if there are new features, a majority of the functionality simply needs to be maintained as compared to the legacy system. The Business Objectives Model and Objective Chains aren’t as useful because their purpose is to map the value (for instance, return on investment) of features to business objectives. Unlike projects that trace their value up to business objectives, existing system conversion projects should use KPIMs for business processes to prioritize requirements and business rules. At a minimum, the new solution must be able to maintain the KPIs at their current levels—there should be no degradation in overall efficiency from switching to a new solution.

The KPIM is one of the most critical models because it helps analysts demonstrate to business stakeholders that even if the new system behaves differently, the business outcomes will be the same or better. One common challenge with existing system conversions is that new software might cause a reduction in the KPIs of one group while improving the KPIs of another group. Even though overall the throughput and business value are positive, the group that is negatively affected might not approve the system unless they understand that the negative impact is in the context of an overall improvement to the business. You should use KPIMs to reassure the business stakeholders that the new system, though different, will still let them get their jobs done.

With the use of KPIMs, you will also need Process Flows against which to map the KPIMs. Org Charts, Ecosystem Maps, and BDDs are all valuable to an existing system conversion project as well, because they help you understand all current users, the existing system integrations that might need to be replaced, and the full set of data the business cares about. Process Flows are needed to describe the activities that users perform in the existing system. Report Tables are needed because existing systems almost always have reports that need to be converted to the new system.
Real-Time and Embedded Systems

Real-time and embedded systems have significantly smaller or more primitive user interfaces than most user-facing systems. The goal of these types of projects might be to implement automation or controller systems. In real-time and embedded systems, System Flows are the dominant models. Ecosystem Maps and System Interface Tables might be helpful if the real-time system has interfaces with many other systems. Most people models will not be helpful, because a majority of the effort will focus on the steps within the system. Real-time and embedded systems often have very simple data models, so a BDD and a Data Dictionary might not be necessary. Although the system will obviously be dealing with data, it is most likely doing so at a technical level, so the business stakeholders are not concerned about the details of the data itself. State Tables and State Diagrams are commonly used, because these types of systems often have complex state changes that trigger behaviors.

Large-Ecosystem Projects

Projects with large ecosystems have many existing systems that interact. You should focus on the systems first by starting with an Ecosystem Map. System Interface Tables might be needed to describe the interface requirements between systems. Identify the business data objects to create BDDs, and then use Data Flow Diagrams (DFDs) to show the flow of data between the systems. Data Dictionaries will be necessary to describe the fields and rules for the data.

Internal IT Systems

Internal IT systems are systems in which all (or most) of the users are internal to the organization. These systems are deployed in one organization’s environment. Org Charts will certainly be used because the users are internal, and Process Flows are necessary because they define how the business will use the
internal system. Roles and Permissions Matrices will probably be used to describe the security model for users. Ecosystem Maps are helpful to show how the system fits in with other existing systems in the IT organization, and System Interface Tables might be helpful if the interface requirements are important to the business stakeholders.

### Hardware and Software

Systems that have both hardware and software components to be implemented typically have many inputs and outputs to be considered. Although people and data models are important, system models are the most critical to create for this characteristic. An Ecosystem Map shows how the components are related, System Flows show how the hardware and software interact, and System Interface Tables describe the inputs and outputs between each component. Keep in mind that although these models are similar to technical models, the purpose of these models is to derive the requirements—whatever the business stakeholders need. Leave the technical documentation to the technical team.

### Packaged Software

Packaged software is software that is sold as stand-alone software. Packaged software should heavily use people and data models and will probably use few system models. Process Flows and Use Cases are helpful for showing how the users will interact with the software. Org Charts are not useful because the users are not common to one environment. Feature Trees are useful for actually building the "packaging" for the software. RMMs help control scope by mapping the requirements to Process Flow steps to ensure that extra features that are not anticipated to create significant value for the users are excluded. UI Flows and DAR models are important for modeling the user interface.
Cloud Implementation Projects

In cloud implementation projects, you are implementing a cloud solution to solve a business problem. The fact that the project is a cloud implementation does not influence the selection of requirements models as much as other the other characteristics do. For instance, if the cloud implementation is part of a large ecosystem, then an Ecosystem Map will be useful to show how the cloud part of the solution interacts with other parts of the system, a System Flow can help describe the system steps, and System Interface Tables might be necessary to describe the interfaces. The Org Chart might be useful in identifying users, and Roles and Permissions Matrices can define security access in the cloud for the user types. Process Flows can be used to describe how users will interact with the cloud. Also, state models are often useful because cloud implementations typically are based on user states, such as logged on, logged off, online, or offline.

Web App Projects

Web apps expose functionality and display data to users through a web interface. Because the web app is communicating to a back-end server, an Ecosystem Map is helpful for showing the architecture, but System Flows are most important for describing those interactions. Data models are important for showing what data exists in the system and is passed between the server and client, so you should create a BDD and a Data Dictionary. UI Flows and DAR models are often helpful for building the web interface, to ensure that it is usable.
Mobile Systems

Systems with mobile capabilities are intended to be deployed at least partially on mobile devices. Mobile systems should have Use Cases that precisely describe how users will interact with the mobile device, and they might use Process Flows to describe users’ goals with the device. Ecosystem Maps and System Flows might be helpful to show the interfaces and interactions between the mobile system and the servers it communicates with. Because mobile devices have limited screen size and sometimes slow interaction times, UI Flows and DAR models are helpful to ensure that the screens on the mobile device are designed efficiently and can be easily used.

Projects with Complex Decision Logic

Projects with complex decision logic automate a decision process. These projects typically have other characteristics that drive model selection. You should use Decision Tables and Decision Trees to model the complex logic. State Tables and State Diagrams might be needed because many decisions are based on states in the system.
Data Characteristics
The following project characteristics are related to the data needs of the system.

Analytics and Reporting Components
Systems that have analytics and reporting components are typically used in business intelligence to help people make decisions based on large data sets. In fact, these projects can be identified by their business strategy—any project that involves getting information to make a decision has significant data requirements.

Projects that involve significant volumes and use of data need several data models to accurately document the requirements. You can use the BDD to determine which types of data are involved in the project, DFDs to describe the flow of the data, and Data Dictionaries to further describe the data. Report Tables are a necessity if you are creating reporting requirements.

Often Process Flows and the other people models aren’t needed at all for a pure analytics project. However, keep in mind that Report Tables include decisions that need to be made. For a very large business intelligence project, prioritizing reports might require Process Flows for determining which reports support the most important processes and for articulating the decisions that need to be made. Also, decision models might be helpful on analytics projects.

Database Back-End Components
Many projects will have a database back-end component. These systems contain data that is used by and stored in the system. For these projects, you will need to identify all of the business data objects in BDDs and define their flow between processes, systems, and storage components in DFDs. You should also define the actual field-level details in Data Dictionaries. Keep in mind that you do not need to document the database schema or physical architecture of the database servers. Instead, focus on documenting how the business stakeholders think about the data.
Data Warehouse Components

Data-heavy systems contain many business data objects and pass a large volume of data between systems. You will probably use many non-data models, but you should focus on data first, by identifying business data objects to create BDDs, then completing DFDs. You can create Data Dictionaries to provide an increased level of detail for the data requirements.

Project Examples

An existing financial services system is to be replaced with a heavily customized COTS product that will integrate to several other existing systems. The system will be used by hundreds of thousands of customers every day. The team works with several departments that handle the various regions that are transitioning to the new system. A majority of the functionality will be maintained. Figure 25-2 shows the appropriate characteristics and most useful models for this project.

Another project is to build a single-user game for a mobile device. Figure 25-3 shows the appropriate characteristics and most useful models for this project.

A final project is to select and implement COTS software to manage a loan approval process. Figure 25-4 shows the appropriate characteristics and most useful models for this project.
### CHAPTER 25
Selecting Models for a Project

<table>
<thead>
<tr>
<th>Objectives</th>
<th>Business Objectives Model</th>
<th>Objective Chain</th>
<th>Key Performance Indicator Model</th>
<th>Feature Tree</th>
<th>Requirements Mapping Matrix</th>
<th>People</th>
<th>Chart</th>
<th>Process Flow</th>
<th>Use Case</th>
<th>Roles and Permissions Matrix</th>
<th>Ecosystem Map</th>
<th>System Flow</th>
<th>UI Flow</th>
<th>Display Action Response Model</th>
<th>Decision Table</th>
<th>Decision Tree</th>
<th>System Interface Table</th>
<th>Data</th>
<th>Business Data Diagram</th>
<th>Data Dictionary</th>
<th>State Table</th>
<th>State Diagram</th>
<th>Report Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>COTS implementation</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extensive user interaction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer-facing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>M</td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System replacement</td>
<td></td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large ecosystem</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>M</td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database back-end components</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td>M</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selected models for scenario</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 25-2** Models for a financial services example project.

<table>
<thead>
<tr>
<th>Objectives</th>
<th>Business Objectives Model</th>
<th>Objective Chain</th>
<th>Key Performance Indicator Model</th>
<th>Feature Tree</th>
<th>Requirements Mapping Matrix</th>
<th>People</th>
<th>Chart</th>
<th>Process Flow</th>
<th>Use Case</th>
<th>Roles and Permissions Matrix</th>
<th>Ecosystem Map</th>
<th>System Flow</th>
<th>UI Flow</th>
<th>Display Action Response Model</th>
<th>Decision Table</th>
<th>Decision Tree</th>
<th>System Interface Table</th>
<th>Data</th>
<th>Business Data Diagram</th>
<th>Data Dictionary</th>
<th>State Table</th>
<th>State Diagram</th>
<th>Report Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer-facing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>M</td>
<td></td>
<td>M</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobile</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>M</td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex decision logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>M</td>
<td></td>
<td>M</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selected models for scenario</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 25-3** Models for an example mobile game project.

<table>
<thead>
<tr>
<th>Objectives</th>
<th>Business Objectives Model</th>
<th>Objective Chain</th>
<th>Key Performance Indicator Model</th>
<th>Feature Tree</th>
<th>Requirements Mapping Matrix</th>
<th>People</th>
<th>Chart</th>
<th>Process Flow</th>
<th>Use Case</th>
<th>Roles and Permissions Matrix</th>
<th>Ecosystem Map</th>
<th>System Flow</th>
<th>UI Flow</th>
<th>Display Action Response Model</th>
<th>Decision Table</th>
<th>Decision Tree</th>
<th>System Interface Table</th>
<th>Data</th>
<th>Business Data Diagram</th>
<th>Data Dictionary</th>
<th>State Table</th>
<th>State Diagram</th>
<th>Report Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>COTS selection</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COTS implementation</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workflow automation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex decision logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selected models for scenario</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 25-4** Models for an example loan approval project.
Thinking About the Audience

Consider who the audience is when you select your models. All RML models are designed to be understood and used by all audiences. However, when you ask someone to review or use models, you should still select models that are most appropriate to them.

- Asking a vice president to review models that are very detailed might not be a good use of her time. Conversely, creating only very high-level models for developers and testers to use will not provide them with enough information to do their jobs.

- You might find that business stakeholders have a hard time telling you all of the systems and their integrations, whereas architects might not be very familiar with business processes or how the product manager intends users to use the system.

Regardless of who creates, reviews, and uses the models, you should make sure that your development teams are aware of the full set of models and requirements that you do create. Models linked to requirements provide additional information beyond just the checklist of requirements to be developed. Although models represent a way to organize and present information, it is still necessary to verbally communicate with all stakeholders to ensure that they understand the material. Handing over models without discussing them with the technical teams is a recipe for failure. You will never be able to capture every iota of information in the models.

Table 25-2 describes the most common stakeholder audience scenarios. The types of stakeholders who will directly help with creating a model are marked with a C, those who will be more likely to only review what you give them are marked with R, and those who probably won’t use it at all are blank. The analyst is not listed in this table because it is assumed that he will help create and review all models.

**TABLE 25-2  Model Use by Audience Type**

<table>
<thead>
<tr>
<th>Model</th>
<th>Business</th>
<th>Technical</th>
<th>Executive</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objectives Models</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business Objectives Model</td>
<td>C</td>
<td>R</td>
<td>C</td>
</tr>
<tr>
<td>Objective Chain</td>
<td>C</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>Key Performance Indicator Model (KPIM)</td>
<td>C</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>Feature Tree</td>
<td>C</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>Requirements Mapping Matrix (RMM)</td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td><strong>People Models</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Org Chart</td>
<td>C</td>
<td>C</td>
<td>R</td>
</tr>
<tr>
<td>Process Flow</td>
<td>C</td>
<td>C</td>
<td>R</td>
</tr>
<tr>
<td>Use Case</td>
<td>C</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Roles and Permissions Matrix</td>
<td>C</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>
Tailoring Models

When you are selecting requirements models, it is possible to get stuck trying to fit information into a structure that is not appropriate. It might be necessary to make small adaptations to the structure of some models based on the specific needs of a project. Each requirement model is flexible enough that the components can be tailored to the specific needs of the project.

The most common types of customization we see are the addition of coloring to highlight particular types of elements or the addition of elements, such as fields to a Data Dictionary or a particular shape from Business Process Modeling Notation (BPMN) to a Process Flow.

Avoid modifying the models unless it’s absolutely necessary to communicate additional information that the model does not normally contain. If you do need to modify them, do it in such a way that an untrained user can still consume them easily. If a model must be modified, the tailoring should take place before the start of the project rather than during the requirements process, to reduce the amount of rework and to avoid inconsistency in the requirements documentation. Often you might not recognize that a model needs to be modified until you have done a significant amount of work. In those cases, you will have to use your best judgment as to whether it is worth it to go back and fix the prior work.
Exercise

The following exercise is intended to help you to gain a better understanding of how to use the information in this chapter. The exercise is open ended, and therefore the answer you come up with could be substantially different than the answer that we have provided. There are potentially many correct solutions. The answer provides an explanation of how we arrived at our solution. You will gain the most out of the exercise by attempting to do it yourself before looking at the solution. The answers for the exercises can be found in Appendix C.

Instructions
Identify the project characteristics and select the appropriate models for the scenario.

Scenario
You are on a project to launch a brand new eStore to sell flamingos and assorted lawn decorations, and you have to document all of the requirements. Currently, orders are only taken over the phone and manually entered into the order system by sales representatives. You expect to have thousands of customers visiting the website every day. You know that there will be servers that push catalog data to the website and send the orders on to fulfillment.

As you explore the high-level features, you also learn that the eStore orders will have various states, such as “New,” “Received,” “Packaged,” “Billed,” “Shipped,” and “Returned.” The president of Wide World Importers wants to view reports that show him metrics such as sales volume, inventory volume, and inventory costs by month. In addition, the training team wants to be sure that the process of shopping is well documented from the point at which a customer arrives at the website to when he receives an order confirmation after checkout.