

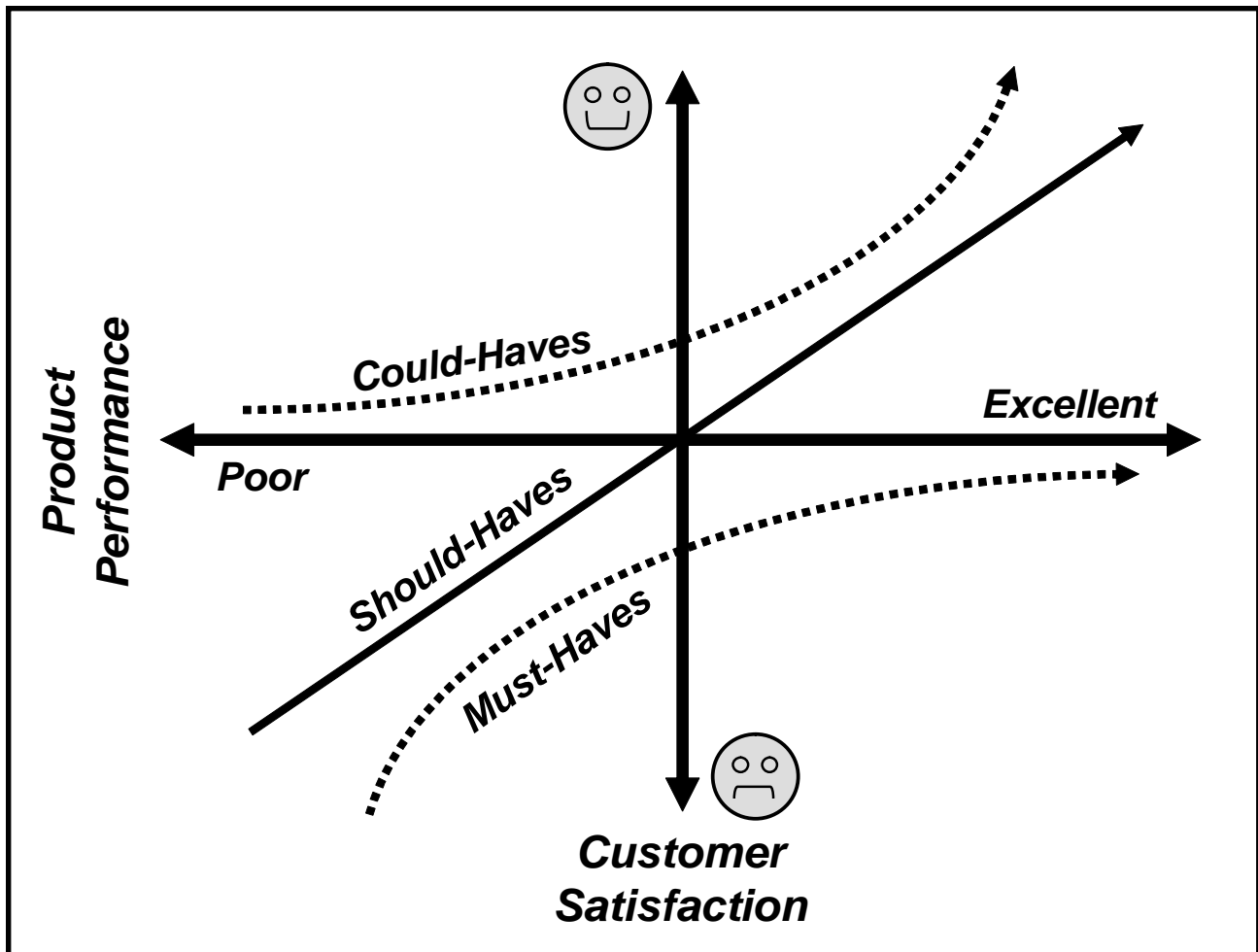
## *Must / Should / Could Prioritization*

One of the most powerful ways to reduce waste and accelerate new product development is to consistently apply your team's efforts to the highest priority work at hand. Unfortunately, it is often difficult to determine which tasks should take precedence, which features are the most important, or which tests should be run first. Even if clear priorities are known, how can they be easily communicated to the design team? The lean method described in this section is one of the most widely adopted and successful tools covered in this guidebook. It provides a simple and effective way to both categorize project priorities and communicate them in a straightforward and unambiguous manner. Although the genesis of this technique is rooted in how customers prioritize product attributes, the method can be applied to essentially every aspect of the development environment, and even extended into your personal life.

### *The “Kano Model” of Customer Perception*

A number of years ago, one of the gurus of the total quality management (TQM) movement, Noriaki Kano, created a model for customers' perception of product value. The Kano Model has become widely accepted as an insightful and accurate representation of how product attributes influence customer satisfaction. Before we go further, however, I must apologize to purists everywhere, including Kano himself. I have shamelessly co-opted this model to allow a far broader application of the underlying concept, as shown in Figure 4.5. In the original version, as in the one I've presented, customer satisfaction is plotted against product performance. However I've given Kano's curves somewhat different names. First allow me to make amends by telling the story of this model in the way its creator intended. I will then explain the liberties that I have taken.

Imagine that you are about to purchase a new automobile. Assuming that you have established your budget in advance, what factors would you consider when making your buying decision? With price held constant, you would likely compare mileage, horsepower, safety, legroom, styling – the usual differentiators that we are all familiar with. It is highly unlikely, however, that you would crawl under each of your candidate cars and check out their mufflers. The radiator would also be ignored, along with U-joints, fan belts, and so on. Why do we ignore these items? Because we assume that if the car is produced by a reputable manufacturer, these basic components will work acceptably well. In fact, there is little an automobile designer could do to enhance these elements that would influence our buying decision. The only time such basic components are noticed at all is if they fail. If



**Figure 4.5:** The “Kano Model” of customer perception identifies three categories of product attributes: “dissatisfiers” (which I’ve renamed “must-haves”), “satisfiers” (which I refer to as “should-haves”), and “delighters” (which I call “could-haves”). These categories can be used to effectively set priorities for development teams.

that should occur, we would be *dissatisfied*. The lowest curve in the figure was originally identified in just this way; attributes that follow this behavior were referred to by Kano as “*dissatisfiers*”. They must be good enough to never become a negative, but there is little or no opportunity to delight the customer through enhancement of these design elements.

The center curve in the figure was originally titled “*satisfiers*,” and was intended to represent those attributes that directly impact customer satisfaction. In our automobile example, characteristics such as mileage, horsepower, styling, and so on, would fall into this category. Finally, Kano made perhaps his most insightful observation. The top curve reflects how customers react when they are offered a new and highly beneficial innovation. Kano refers to these attributes as “*delighters*”; you could also think of them as providing a “wow factor”. One of the key take-aways from Kano’s model is that the greatest oppor-

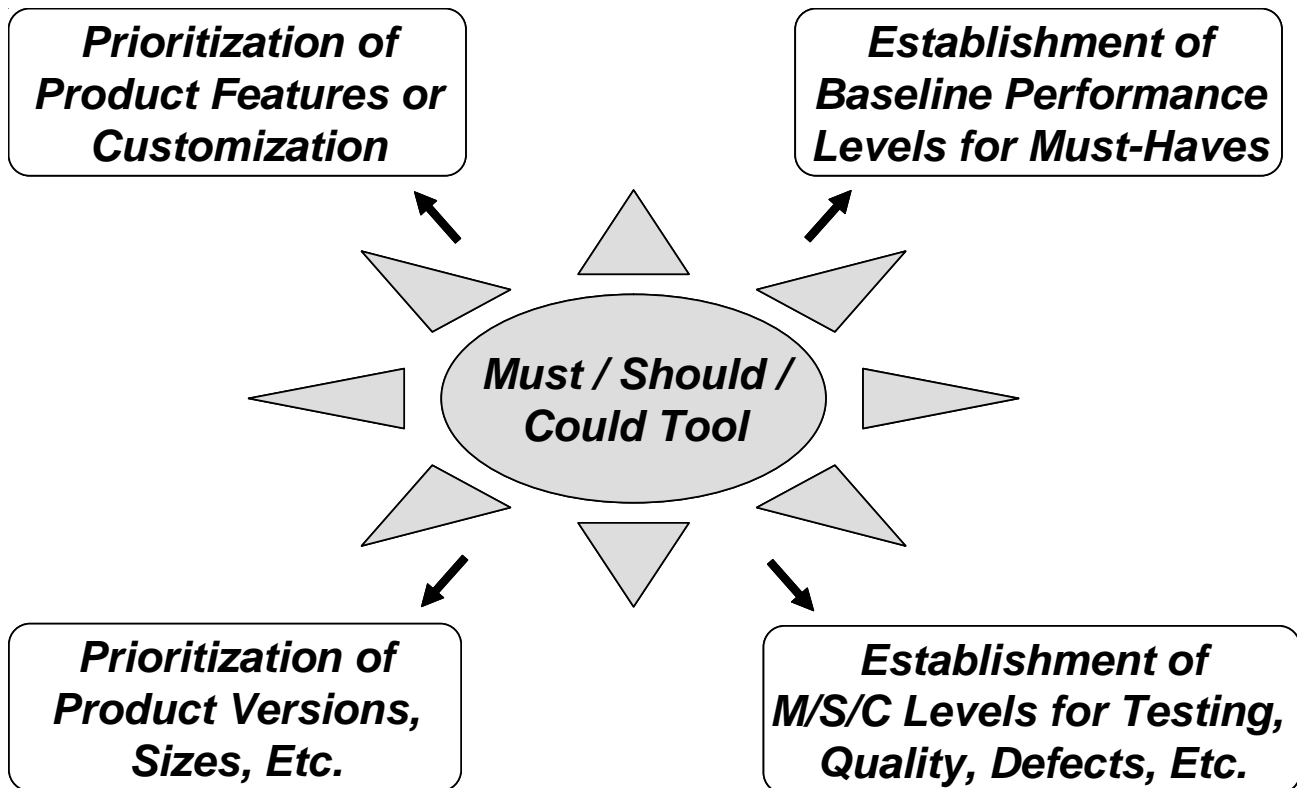
tunities for product success lie in identifying delighters. Unfortunately, there is often significant risk involved. It would be far safer to have your development teams spend their time enhancing satisfier attributes, since they are a known quantity. Delighters require some level of breakthrough innovation, and may prove to be of little interest to the marketplace. Going back to our car-purchase example, a delighter (as of this writing) might be something like side-impact airbags, dashboard control through voice recognition, and most certainly, the hybrid engine. These features generate excitement, and can often make the difference in a purchase decision among roughly equivalent products. Yet many new improvements to automobiles have received poor marks from customers and have been relegated to the scrap heap. When was the last time you saw a new car with those automated seat belts that pull across your chest when you start the engine? After several million hot cups of coffee spilled onto an equal number of drivers' laps, the "wow factor" seems like a distant memory.

Although Kano's terminology works well in the context of customer satisfaction, I prefer to broaden the applicability of his insights to virtually every aspect of product development work, as shown in Figure 4.6. The result is a simple, three-category prioritization tool: must-haves (dissatisfiers), should-haves (satisfiers), and could-haves (delighters). The must / should / could approach to prioritizing customer needs is somewhat more coarse than quality function deployment. (See Section 2.1 of *The Lean Design Guidebook* for a complete description of this voice-of-the-customer methodology.) Yet what it lacks in specifics, it makes up for in ease of communication. In the following subsections, I will describe several opportunities to use must / should / could (M/S/C) within the context of product development work. Don't restrict yourself to this arena, however. A consultant friend recently mentioned that she is using M/S/C to help her son prioritize his homework. *All* of our time is valuable, not just time spent in the workplace.

### ***Application #1 – Product Features and Requirements***

The most obvious application of M/S/C is in the prioritization of product features based on their impact on customer satisfaction. Much as Kano observed, must-haves represent features or performance levels that are both basic and essential; falling short in this area will have a negative impact on the perceived value and market acceptance of a new product. It follows that should-haves are those attributes of a design that directly address known customer wants and needs, and will have a reasonably predictable impact on price and value. Finally, the could-haves represent new innovations or unique design elements that may capture the imagination of the marketplace, but may also prove to be of little or no benefit if the market fails to respond. Naturally, the best source of insight into which attribute falls into which category would be direct feedback from representative customers. The M/S/C methodology can be used as an informal voice-of-the-customer tool, either through surveys or through individual or group interviews. Again, ease of communication plays a powerful role; it is often difficult to get customers to sit through a Lean QFD (i.e., quality function deployment) exercise, for example, but M/S/C feedback can be gathered in seconds.

From a project execution standpoint, a development team should ensure that all must-haves have been bedded down in the early stages of a project, and then spend most of



**Figure 4.6:** The must / should / could (M/S/C) prioritization tool can be applied to every aspect of new product development, from the features to be included in a product to the time spent on testing and qualification.

their efforts optimizing the should-haves. If time and money permit, the could-haves can be pursued, but only consistent with meeting the product's intended launch date – these aspects of the design might be trimmed to avoid excessive delays in product shipment. The only time this logic fails is if the new product is intended to be a breakthrough design, in which case the could-haves may represent the primary reason for investing in its development.

I recommend adding a column to your product specification document that explicitly captures the priority of each important requirement, as shown in Figure 4.7. Basic functionality and baseline performance would be assigned must-have status, with the shoulds and coulds falling into place as appropriate. The assignment of M/S/C priorities can trigger vital negotiations among marketing, engineering, and executive management. Conducting these negotiations at the beginning of development can save designers from wasting time on nonessential aspects of a new product, before core functionality has been addressed. Note that it is possible, and indeed beneficial, to identify must, should, and could levels of performance for a *single* product attribute. For example, the specification for an automobile might state that thirty miles-per-gallon is a must-have, thirty-five mpg is a should-have, and forty mpg is a could-have. In this context, the could-have becomes a “stretch goal” for the design team, while the should-have level represents the actual target

for designers. The must-have baseline serves as a last resort; if the product is significantly delayed, the could and shoulds may be compromised somewhat, but the product cannot enter the market without meeting the must-haves.

Continuing with the last point, I have found it valuable in some cases to create a separate must-have specification at the inception of a development project, as shown in Figure 4.8. The idea is to provide the design team with a baseline for every critical specification, below which the team cannot go without invalidating the product's business case. The inputs to this must-have spec might be based on achieving parity with competing products in the marketplace, or may be derived from an understanding of market expectations and essential quality thresholds. However the requirements are selected, they must truly represent a "no-go" threshold of performance or features. Why is this beneficial to a design team? As the development process proceeds, it is often necessary to perform multidimensional tradeoffs to achieve the best possible mix of desirable attributes. When performing these trade studies, it is essential that designers understand the full space in which they may work. The must-have specification represents a firm lower boundary for any performance tradeoff, thereby enabling the team to avoid wasting time chasing possibilities that are simply not viable. In my experience, compromises to the initial product specification are generally made at the end of the development process, triggered by unacceptable slips in schedule or excessive cost. It is far better to proactively determine possible lower limits up front, so the team can use this information at those times in the process where it can provide the most benefit.

<i>Priority</i>	<i>Requirement</i>
<div style="border: 1px solid black; padding: 2px; display: inline-block; margin-bottom: 5px;"> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">M</span> <span style="border: 1px solid black; padding: 2px; margin-left: 5px;">S</span> <span style="border: 1px solid black; padding: 2px; margin-left: 5px;">C</span> </div> <div style="border: 1px solid black; border-radius: 15px; padding: 5px; width: fit-content; margin-bottom: 5px;"> <i>Each spec is categorized at the beginning of product design</i> </div> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> <span style="border: 1px solid black; padding: 2px; margin-right: 5px;">M</span> <span style="border: 1px solid black; padding: 2px; margin-right: 5px;">S</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">C</span> </div>	<p data-bbox="521 1167 1024 1205"><b>4.3.8 Enclosure Dimensions</b></p> <p data-bbox="615 1253 1170 1289"><b>Enclosure dimensions shall be:</b></p> <p data-bbox="615 1295 1078 1331"><b>Height = 27.5 inches +- .01</b></p> <p data-bbox="615 1337 1078 1373"><b>Width = 35.4 inches +- .01</b></p> <p data-bbox="615 1379 1078 1415"><b>Depth = 22.5 inches +- .01</b></p>
<div style="border: 1px solid black; padding: 2px; display: inline-block;"> <span style="border: 1px solid black; padding: 2px; margin-right: 5px;">M</span> <span style="border: 1px solid black; padding: 2px; margin-right: 5px;">S</span> <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">C</span> </div>	<p data-bbox="521 1463 992 1501"><b>4.3.9 Enclosure Insulation</b></p> <p data-bbox="615 1549 1451 1669"><b>Enclosure shall be insulated with 0.5 inch thick Styrofoam to reduce heat transfer and ambient equipment noise.</b></p>

**Figure 4.7:** The must / should / could priority of each product requirement can be shown explicitly within an engineering specification document. In some cases, a single requirement might have a must-have level of performance, along with should- and could-have levels that represent challenges for the development team.

<b>Priority</b>	<b>Requirement</b>
<b>M</b> S C	<b>1.0 Maximum acceptable tolerances</b>
<b>M</b> S C	<b>2.0 Essential functionality</b>
<b>M</b> S C	<b>3.0 Minimum feature /option set</b>
<b>M</b> S C	<b>4.0 Mandatory cosmetics and ergonomics</b>
<b>M</b> S C	<b>5.0 Required testing and approvals</b>
<b>M</b> S C	<b>6.0 Market-driven performance levels</b>

**Figure 4.8:** It is often useful to create a “must-have specification” for a product at the inception of product development. This baseline can then be used throughout the process as a touchstone for the team when conducting performance tradeoffs.

### **Application #2 - Prioritizing Tasks Within a Development Project**

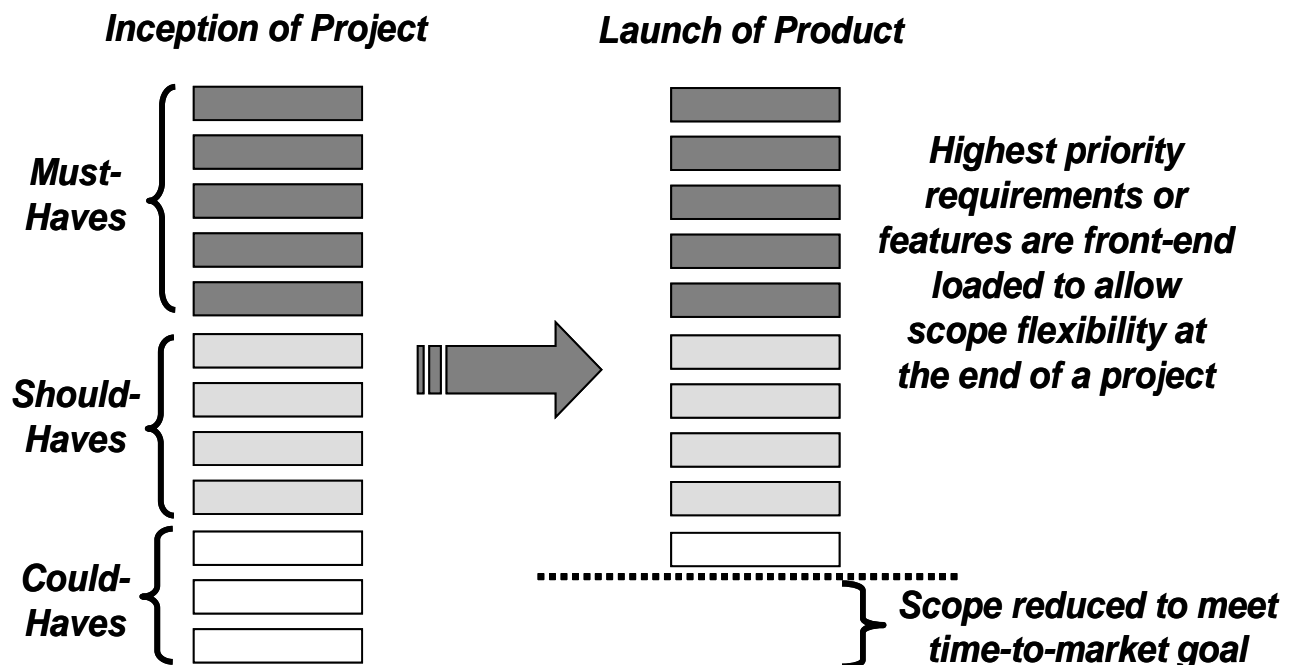
Although the application described above comes the closest to Kano’s original intent, it is the prioritization of tasks within a product development project that will provide design teams with the greatest schedule benefit. The M/S/C concept can be used in this context, both as an initial planning tool and as an ongoing means of communicating the priority of project tasks. Initially, those activities that are on or near the critical path should be given must-have importance. Should-have tasks would be those that involve significant technical or quality-related issues; clearly important but not necessarily on the critical path. Finally, any task or activity that can be postponed until later in the project (or perhaps until after the first units are shipped) would be assigned could-have priority. As the project progresses, these priorities should be modified as needed to reflect the team’s current situation. A task which was on the critical path at the beginning of development, for example, might move to a lower priority due to some unforeseen change in project direction.

Keeping all team members focused on either schedule-critical or technically challenging aspects of a project is a powerful way to ensure the highest levels of productivity and efficiency. Moreover, using M/S/C can enable far greater control of project schedule, as shown in Figure 4.9. The could-haves represent a “shock absorber” for time-to-market, allowing the team leader to pull the project schedule back into line if slips occur. Recall

the three-legged-stool analogy used in Section 3.3 to illustrate the impact of change on a project: if one leg is too long, the stool (and by reference, the project) is unbalanced. If M/S/C is employed and the schedule leg of the stool becomes longer than desired, the scope leg of the stool can be adjusted by shaving could-haves until the project is back in good graces. In a sense, the trimming of could-have requirements or features gives the team leader an adjustable leg on the project stool. What tasks might this include, you ask? Virtually every aspect of a project is a candidate: design niceties, low-priority documentation, sizes of the new product that have low sales potential, and even selected qualification tests that can be performed either during or after product launch. If you've ever been on a product design team, it is likely that you have seen this approach in action...or perhaps I should say, in *reaction*. When a project gets into schedule trouble, it is amazing how many essential activities become optional. Rather than being unprepared for the inevitable push at the end of a project, why not use the M/S/C tool to provide both the team and its leader clear priorities up front. This way, the project might not get into schedule trouble in the first place.

### ***Application #3 - Testing and Documentation***

Of all the activities that constitute new product development, the ones that get the least respect are testing and documentation. Since the outputs of these tasks are not directly observable by customers, it is always tempting to cut a few corners, particularly



**Figure 4.9:** By applying must / should / could prioritization throughout product development, it may be possible to downscope a project without significantly affecting either quality or market price. The could-haves can be used as a “shock absorber” to protect a project’s schedule.

when time (and your manager's temper) gets short. Having spent several years running a semiconductor testing laboratory, you might think that I would scold you for such heretical thoughts. Actually, just the opposite is the case; I recommend that you *plan* for cutting corners in testing and documentation. Perhaps I should explain myself.

What types of activities must be scheduled near the end of a development project? That's right, preproduction testing and product documentation. And when is time pressure the greatest? Very good, now you are starting to see my point. Although it is true that external customers are typically not aware of which tests you've run or how many levels of drawings you've released, the main reason why these activities get no respect is that they are in the wrong place at the wrong time. If the universe were to run backward, we would probably find ourselves happily performing test after test early in a project, but would be forced to shortchange innovation and conceptualization as the completion date approaches. In the absence of excellent planning and disciplined execution, whatever comes last will probably get short shrift.

So why should we *plan* for a crunch at the end of a project? Isn't this just admitting defeat? Rather, I believe that this is a matter of accepting reality. If you manage to pull off the "perfect storm" of ideal project scheduling and effective team leadership, than having spent a few hours putting together a contingency plan will be easily forgiven. On the other hand, if your project goes the way of most, knowing in advance which tests can be scaled back and which documents can be delayed or waived will avoid desperate flailing as your launch date approaches. How many preproduction units *must* be tested to have adequate confidence in a new product design prior to release? Which documents *must* be in place, because they impact vital areas such as production, quality, rework, and repair? If we are to fully protect our customers and our firm's reputation, what testing *should* we perform? For the benefit of future projects, which clever ideas or lessons learned *could* we capture? In my experience, applying M/S/C prioritization to activities that occur just prior to product commercialization is essential to launching a quality product. If appropriate priorities are followed, both your firm and its products will be protected from last-minute pressures to "get this thing out the door". Of course, our goal should be to execute a disciplined process that allows adequate time for the musts and shoulds, and even some of the coulds. When the going gets tough, however, and testing and documentation tasks end up on the chopping block, it will be a relief to know where and how deeply you can cut.

#### ***Application #4 - Daily Work Activities***

Like many of you, the first thing I do each morning is make up a list of what I'd like to accomplish that day. Also like many of you, at the end of the day there are an intimidating number of items that I've failed to complete. The difference is that *I always complete my must-have activities*, and usually make significant inroads into the should-haves. The next morning, my list gets updated. With great joy, I scratch through those items that I've completed, and add new tasks that have come up since the day before. I then reevaluate my priorities, often elevating the shoulds and coulds from the prior day to musts and shoulds. This may sound a bit anal-retentive, but setting priorities in this way has allowed me to grow my business and still have time to enjoy life.

This concept can easily be adapted to the day-to-day execution of product development projects, as shown in Figure 4.10. The template presented in the figure is just a strawman; you can modify its format in anyway you wish. I've found that looking two weeks ahead is about right for most projects, and it is important that the action items be assigned to responsible individuals. Beyond that, any format will work just fine. As with my personal example above, the "rule" for a design team should be to complete all must-have actions as soon as possible and apply significant effort toward moving the should-haves forward. Could-have activities are kept on the back burner, awaiting available resources. As the project progresses, completed actions are removed from the list and the priority of remaining items is reevaluated on a regular basis. I could go on about how effective this tool has been for the teams that I've managed, but enough about me. Try this application of M/S/C for yourself and you will almost certainly be sold.

#### ***Application #5 - Filtering Design Concepts***

This final application of M/S/C is really just an excuse to present one of my favorite innovation tools. In his excellent book, *Total Design*, Dr. Stuart Pugh describes a simple method for sorting through various design concepts and quickly identifying those

<b><i>Two-Week Action Tracking Sheet</i></b>				
<b><i>Action Item</i></b>	<b><i>Responsibility</i></b>	<b><i>Priority</i></b>		
		<b><i>M</i></b>	<b><i>S</i></b>	<b><i>C</i></b>
<i>Finish critical-path tasks</i>	<i>S. Carton</i>	<b>X</b>		
<i>Order long-lead parts</i>	<i>C. Darney</i>	<b>X</b>		
<i>Submit regulatory documents</i>	<i>L. Mannette</i>	<b>X</b>		
<i>Review and approve final drawings</i>	<i>D. Copperfield</i>		<b>X</b>	
<i>Complete and release test plan</i>	<i>O. Twist</i>		<b>X</b>	
<i>Finalize bill-of-materials</i>	<i>T. Tim</i>		<b>X</b>	
<i>Begin creating product sales literature</i>	<i>E. Scroog</i>			<b>X</b>
<i>Update software documentation</i>	<i>B. Cratchet</i>			<b>X</b>
<i>Do other non-schedule-critical stuff</i>	<i>C. Dickens</i>			<b>X</b>

**Figure 4.10:** Another example of how must / should / could prioritization can be applied to a wide range of project activities. Here it is used to help development team members prioritize their near-term actions.

candidates with the greatest potential for success. This so-called Pugh Method is shown in Figure 4.11. Pugh notes that one of the most difficult aspects of creative brainstorming is filtering out the good ideas from the not so good ones. He suggests that at the end of any brainstorming session, a two-dimensional matrix be created. Along the top of the matrix, various design concepts are listed. On the left side of the grid, he suggests identifying a number of must-have criteria for the future product. For each design concept, scores are chosen that reflect how well that alternative satisfies each of the must-have requirements. To illustrate this methodology, I'll describe a real-life application.

Several years ago, I facilitated a brainstorming event for a firm that produces vacuum cleaners. They were embarking on a major new project to design their next-generation portable canister vacuum. As with most mature products, profit margin was a significant issue, so each required function of the new vacuum was put under intense scrutiny. The goal was to identify opportunities to reduce production cost while maintaining high levels of customer satisfaction.

One of the attributes of the product that was addressed was the “allow-movement” function. We brainstormed on various ways to achieve portability, and came up with a wide range of ideas. These concepts were then organized as shown in Figure 4.11 (the drawings are optional, but they can help ensure that everyone visualizes the concepts correctly). The most familiar approach was selected as a “default” design; if no better idea came along, “four wheels” would be the logical choice. The group then generated a list of must-have filtering criteria. For example, the allow-movement solution would have to support the projected weight of the vacuum, move smoothly over various surfaces, and most important, cost less than the default design.

Once the matrix was constructed, the group went through each alternative concept and compared it to the default design. If an alternative was worse than the default with respect to a given must-have criterion, it received a minus (-) score. If it was better than the default, it received a plus (+) score. Finally, if there was no significant difference between an alternative and the default design, the alternative received an “S” for “same”. Upon completion of this ranking exercise, the group totaled up the pluses and minuses for each design concept (ignoring the “S” scores, since they are neutral). Any concept that showed several pluses but no minuses represented a likely improvement over the default design. In this example, the three-wheel approach proved to be a winner. If an alternative concept earns a large number of pluses but also several minuses, there may still be a chance to salvage it. Perhaps there is a way to mitigate the minuses so that all must-have criteria can be met. Often this can be accomplished by combining the concept with elements from other design ideas. In this example, the “air-floatation” option looks promising, but it was projected to have stability problems. This drawback might be eliminated by adding skids (from one of the other concepts) to provide stability, while the air floatation would make the product glide like a hovercraft.

As you might have guessed, the firm chose the three-wheel design over the hovercraft option. The good news is that in a very short time (about a day of work), the design team in this example was able to interrogate a number of product functions and identify significant cost savings. Using must-have filtering criteria was the key to rapidly sorting ideas, and easily communicating why some clever concepts should be left on the drawing board.

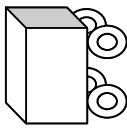
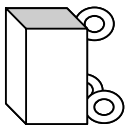
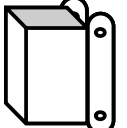
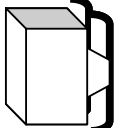
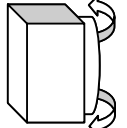
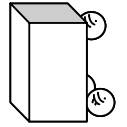
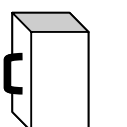
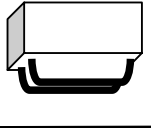
Design Alternatives	 4 Wheels	 3 Wheels	 Treads	 Skids	 Air Float	 Casters	 Handle	 Backpack
	Must-Have Criteria							
Support 10 lbs.		S	S	S	-	S	-	-
Minimal Friction		S	-	-	+	S	-	-
Turning Radius		S	S	S	+	S	+	+
Smooth Movement		S	S	-	+	S	-	-
Light Weight		+	-	-	+	S	-	-
Nice Appearance		S	S	-	S	+	S	-
Handle Stairs Easily		S	S	-	-	S	+	+
Easy Assembly		+	+	-	+	S	+	+
Lower Cost		+	+	-	+	-	+	S
High Reliability		S	S	-	+	S	+	+
Number of Parts		+	+	-	+	S	+	+
<b>Totals</b>	<b>+</b>	<b>0</b>	<b>4</b>	<b>0</b>	<b>8</b>	<b>1</b>	<b>6</b>	<b>5</b>
<b>Totals</b>	<b>-</b>	<b>0</b>	<b>0</b>	<b>10</b>	<b>2</b>	<b>1</b>	<b>4</b>	<b>5</b>

Figure 4.11: The “Pugh Method” for filtering ideas generated by a brainstorming session. Here we use must-have criteria to effectively eliminate concepts that don’t meet the product’s baseline requirements, and highlight those design alternatives that should be given further attention. The “allow-movement” function for a portable vacuum cleaner is provided as an example.

