Customer Interaction Patterns

Linda Rising
risingl@agcs.com
AG Communication Systems
2500 W. Utopia Road
Phoenix, AZ 85027-4129
602.581.4699

Abstract

With the move to self-directed work teams and the increasing emphasis on business awareness in the software development community, individual developers are asked to play a more active role in interfacing with customers. This new role poses a challenge for many engineers who may need guidance to improve their effectiveness in customer interaction. These patterns are the beginning of a pattern language to address that need.

Keywords: customer interaction, communication, negotiation, personal relationships

Introduction

Most of the Customer Interaction Patterns were mined from a presentation by David Saar, Senior Product Planning Manager at AG Communication Systems. Although product development teams in our company have had customer interaction experience, this was the first time a product development team had been formally prepared for their first interaction with the customer, in this case, GTE. David's primary message was "Build Trust." The customer relationship is important to developers throughout the development life cycle. Many team members would be involved one-on-one with a specific customer representative to coordinate the flow of information from the customer to our development team and from resident subject matter experts to the customer.

After capturing these patterns, I remembered a guideline from Jim McCarthy's *Dynamics of Software Development* that seemed to fit with the collection from David's presentation. I translated it to pattern form. I also translated a story from David Armstrong's *Managing By Storying Around*. These patterns present different but important aspects of the development engineer-customer relationship.

Pattern names are written in italics and pattern names are part of the narrative, following two patterns from a pattern language for writing patterns by Gerard Meszaros and Jim Double.

Common Context

These patterns share a common context. With the move to self-directed work teams and the increasing emphasis on business awareness in the software development community, individual developers are asked to play a more active role in interfacing with customers. This new role poses a challenge for many engineers who may need guidance to improve their effectiveness in customer interaction. These patterns help address that need. As such, they are to be used by developers. Other customer interaction patterns could define users who are in marketing or sales, for instance.

Bruce Whitenack's *Customer Rapport* (Develop a rapport with the customer) and Jim Coplien's *Engage Customers* (Closely couple the customer role to the Developer) define the context for these patterns.

Common Forces

These patterns share a set of common forces. Individual patterns may place more emphasis on specific forces or add forces that are not important to the pattern language as a whole. Forces for an individual pattern are given where they are emphasized, modified, or added.

Most of these forces are elements of risk management. The patterns evaluate these risks to find an acceptable, intelligent balance among cost, time, content, and quality. The common forces:

* Customers want understanding.
* We want to anticipate customer wishes.
* We want to protect our own interests.
* We want to delight our customers.
* We don't want to make promises we can't keep.
* Customers may have unrealistic expectations and demands.
* We have finite resources.
* Customers want results.
* We are all, developers and customers alike, under time pressure.

Known Uses

The patterns have many known uses. Usually David would say "Life!" when I asked him for a specific known use. Obviously, these patterns can be applied in settings other than customer interaction. Every human interaction can make use of them!

The Patterns

* *Active Listening*
* *Be Aware of Boundaries*
* *Build Trust*
* *It's A Relationship, Not A Sale*
* *Mind Your Manners*
* *Perfunctory Meetings*

Build Trust and *It's A Relationship, Not A Sale* are higher-level patterns that work together (reference each other) and define the context for the five lower-level patterns: *Active Listening, Be Aware of Boundaries, Mind Your Manners*, and *Perfunctory Meetings*.

Some of the lower-level patterns reference each other. *Active Listening* references *Be Aware of Boundaries* and *Mind Your Manners*).

References

Coplien, J.O., "A Generative Development-Process Pattern Language," in *Pattern Languages of Program Design*, eds. J.O. Coplien and D.C. Schmidt, Addison-Wesley Publishing Co., Reading, MA, 1995, pp. 184-237.

Meszaros, G. and J. Doble, "A Pattern Language for Pattern Writing," in *Pattern Languages of Program Design 3*, to be published.

Whitenack, Bruce, “RAPPel: A Requirements-Analysis-Process Pattern Language for Object-Oriented Development,” *Pattern Languages of Program Design*, eds. J.O. Coplien and D.C. Schmidt, Addison-Wesley Publishing Co., Reading, MA, 1995, pp. 260-291.

Pattern Name: *Active Listening*

Problem: Sometimes developers go to meetings and tell customers what the customer needs are instead of listening to what the customer really wants.

Context: Developers are trying to *Build Trust* and understand that *It’s A Relationship, Not A Sale.*

Solution: Listen to the customers’ concerns. Pay attention when they’re talking, don’t just use the time to prepare your response. Don’t interrupt. Don’t go off on tangents. Follow their agenda.

Pick up information. Don’t pass up the opportunity to learn what the customer is thinking.

Portray an agreeable, winning attitude. Be flexible and positive. Ask probing questions. Take action items or issues. Work with the customer.

“Be slow to speak and quick to listen!”

“Better to remain silent and be thought a fool than to speak out and remove all doubt!” Sometimes if we’re too anxious to please, we may speak out of turn. Let others talk. Give them room.

Resulting Context: The customer will feel valued, that concerns are being heard, and issues addressed. More than this, the customers’ needs really will be heard since we really are listening.

Developers should *Be Aware of Boundaries* while practicing this pattern and remember the cautions in *Mind Your Manners*.

Rationale: The real purpose of customer meetings is to interact with the customer, *Build Trust*, and share concerns.

The following story is from David Armstrong’s *Managing by Storying Around* referenced below.

A sales manager at Armstrong International wanted to add an obsolete feature to the division’s new fish finder. This approach contradicted the company’s strategy of always providing the latest and greatest technological advance. The sales manager wanted to add a simple flasher to a product that already provided information on the location and size of the fish. No one could understand why the simple indicator that a fish was nearby would be useful. The sales manager pointed out that many longtime customers were not comfortable with the new, computerized technology and wanted the simple interface they were used to.

The feature was added. Customer response was great and the product is still on the market.

The moral of the story is: listen, listen, listen, and if you can’t spend time with your customers, listen to your sales people.

References

Armstrong, David, *Managing by Storying Around: A New Method of Leadership*, Doubleday Publishing Company, 1992.

Author: David Saar, as told to Linda Rising.

Pattern Name: *Be Aware of Boundaries*

Alias: *Stay Within the Lines!*

Problem: Developers are interacting with a customer and may be in a position where it is easy to become engrossed in issues and/or problems. Sometimes an engineer may offer opinions or commitments that could damage relationships with customers. There can be wider impacts from simple discussions that can result in broken promises or incorrect information.

Context: Developers are trying to *Build Trust,* understand that *It's A Relationship Not A Sale*, and may be using *Active Listening*.

Solution: Be aware of boundaries. There may be commercial implications of technical discussions, e.g., real-time and memory sizing. Treat every discussion as part of a negotiation. Get the customer interested, take questions, and get back with answers.

Don't give away data or make instant judgements. Don't say, "Oh, that's easy!" or "That's impossible!" without realizing the lasting impact on the customer and current and future negotiations.

Don't treat commercial considerations, e.g., price, cost, schedule, content. Forward these to the Business Team.

Resulting Context: The customer will feel that concerns are being heard and issues addressed but no commitments are made that might later have to be broken. More than this, the customer's real concerns will be heard and the company's, as well as the customer's, interests will be protected.

Rationale: Boundaries are good for people. They set limits on actions and make it easier to act. Deferring to a higher authority before any commitment is empowering. We must be aware that everyone negotiates for the company and can impact current, future, and even past negotiations. As a result, we may be setting up the customer for disappointment.

It's easy to get carried away in customer interaction especially when trying to *Build Trust* and using *Active Listening*. Interactions with the customer can change not only customer perceptions but also dollar amounts in negotiations.

Author: David Saar, as told to Linda Rising.

Pattern Name: *Build Trust*

Alias: *Build Relationships*

Problem: Developers need to interface with their counterparts in the customer organization to address issues that arise as development proceeds. In addition, customers need contacts in our organization. How can both sides be satisfied?

Forces

* Developers want their questions answered quickly.
* Developers need solutions.
* People are reluctant to spend time with anyone they don't know.

Solution: Invest the time to build trust with the customer.

Resulting Context: As a trusting relationship is established, customer interaction is easier, developers' and customers' questions can be answered, problems solved, and progress made.

A trusting relationship enables a better understanding of customer priorities, which can be valuable if trade-offs with schedule and functionality must be made.

Rationale: Every contact with the customer is a chance to Build Trust. Take advantage of it. Don't just spend time with friends who attended the meeting.

Stephen Covey has observed, "If I make deposits into an Emotional Bank Account with you through courtesy, kindness, honesty, and keeping my commitments to you, I build up a reserve. Your trust toward me becomes higher, and I can call upon that trust many times if I need to. I can even make mistakes and that trust level, that emotional reserve, will compensate for it. My communication may not be clear, but you'll get my meaning anyway. You won't make me "an offender for a word." When the trust account is high, communication is easy, instant, and effective."

Treat each face-to-face encounter as a valuable opportunity to add to your Emotional Bank Account.

According to Scott Hunter, clients prefer to do business with people they like, with people who seem genuinely interested in them, who deal with their concerns. The worst customer interaction mistake is to get right down to business when first meeting the customer. The most critical result produced during an initial meeting is to begin to Build Trust.

As Mike Reynolds has observed, "People buy products from people."

Related Patterns: *It's a Relationship, Not a Sale* works with this pattern to implement Coplien's *Engage Customers* and Whitenack's *Customer Rapport*.

References

Covey, Stephen R., *The 7 Habits of Highly Effective People*, Simon & Schuster Inc., 1989.

Hunter, Scott, "Establishing and Maintaining Good Client Relations," http://www.thehost.com/hunter/clientr.htm

Author: David Saar, as told to Linda Rising.

Pattern Name: *It's A Relationship, Not A Sale*

Alias: *It's the relationship, stupid!*

Problem: How should customers be treated?

Solution: Gain an understanding of the customer, and then express that understanding in your product.

Resulting Context: No relationship ever disintegrated because of too much honest communication! Acting on feedback from your customers will encourage them to stay with you. Your customers will sense that you are responsive and that you are going somewhere together. Customers will feel they're buying into a relationship, not just buying a product.

As in a love affair, should you neglect the beloved for too long, or otherwise signal your disinterest or rejection, the course of love won't run smoothly. Your neglect will cause the customer to feel betrayed, hurt, angry, and punitive. Don't be inconstant with your customers.

Customer contacts are valuable sources of information as product development proceeds. Continuing effort is required to develop a good working relationship with customers.

Rationale: The relationship with the customer is like a dance, or a love affair. You take steps (your releases and messages), and they take steps in response, and then you take more steps. You must be focused on the flow of transactions, on the overall pattern and direction, not on merely the latest transaction.

Relationships are not instantly formed but develop slowly and evolve over time.

The following story is from Dan Behymer, director of quality systems at a manufacturing facility in Cincinnati. It appeared in a letter to the editor of *Quality Digest*, May 1997.

The quality-satisfaction gap is not about products and services. It is about feeling. In a culture where you are bombarded every day with advertisements, objectives and incentives, where someone is always after your hard-earned money, you just want to know that if you buy their goods, they will care once the sale is over. We want someone who cares and will take action. Caring can't come from a total quality improvement team, reengineering, just-in-time or any formula, objectives, or consultants. Customers are human, companies are collections of humans.

The following story is from David Armstrong's *Managing by Storying Around* referenced below.

The vice-president got a copy of a letter from a customer stating that things were going well except for one small detail: the indicator lights were burned out on 25% of the models. The vice-president looked into the problem and found that the defect had been present since the day the product was released and that it had been on the market for more than 10 years. He asked why nothing had been done about it and found that the attention had been directed to "more serious problems."

The vice-president stated that nothing could send a louder message of poor quality than burned out indicator lights on the front of the product. The customer sees this daily and it is a constant reminder that no one is interested or willing to do something about it. It shows the manufacturer thinks some things are too small to be concerned about.

When it comes to quality, nothing is too small to be ignored. To a customer, there is no such thing as a small problem. If the customer has a problem — no matter how minor it may seem — you can bet it is a big deal to him. Treat it that way.

The following is from an article by Rob Thomsett referenced below.

The question “What are your requirements?” is the wrong question. The right question is, “What is your world?” Once we have begun to understand our customers’ organization, their concerns, and their way of working, we can begin to get a clearer idea of them, and then it becomes much easier to understand their requirements.

To understand their needs, we must understand their culture, their dreams, and their expectations. Doing this requires you to get to know your customers as people first and customers second. Understand your customers and you’ll understand their expectations.

Related Patterns: *Build Trust* works with this pattern to implement Coplien’s *Engage Customers* and Whitenack’s *Customer Rapport*.

References

This pattern was adapted from #17, “It’s a relationship, not a sale,” in *Dynamics of Software Development*, pages 71-73, Microsoft Press, 1995.

Armstrong, David, *Managing by Storying Around: A New Method of Leadership*, Doubleday Publishing Company, 1992.

Thomsett, Rob, “It’s the Expectations, Stupid!” *American Programmer*, April 1997, pp. 30-33.

Author: Jim McCarthy, adapted by Linda Rising.

Pattern Name: *Mind Your Manners*

Problem: Developers are interacting with customers and may not consider etiquette, dress, behavior, and language.

Context: Developers are trying to *Build Trust* and understand that *It’s A Relationship, Not A Sale.*

Solution: Mind your manners. Be polite. The customer always goes first.

Be aware of body language.

Dress appropriately. Be sure to verify appropriateness with the sales or business contact. Sometimes business casual is appropriate, but other times a suit is required, while at other times, jeans are acceptable.

Resulting Context: Customers will feel that we are concerned about all aspects of our business interaction and are ready to share their concerns and issues with our products.

Rationale: Common courtesy is so uncommon! Simple, thoughtful acts convey a concern for the other person’s welfare that is essential in any business interaction. Our workplace environment may be very casual and we may not always be aware of pleasantries when interacting with our team, but these are important in customer interaction.

Author: David Saar, as told to Linda Rising.

Pattern Name: *Perfunctory Meetings*

Problem: Developers and customers don't get together often. Since the meetings usually convey routine information, there is a tendency to give the meetings short shrift, to arrive just as they begin and leave as soon as they are over.

Context: Meetings are held with the customer during product development to ensure a common understanding of progress being made. Developers are trying to *Build Trust* and understand *It's A Relationship, Not A Sale*.

Forces

* Developers want customers to be aware of the current status of the product.
* Social interaction can seem a waste of time to some technical people.

Solution: Arrive at the meeting location with enough time to meet the other attendees and spend time socializing. After the meeting is over, allow plenty of time to talk with others who might have common business interests. Often a post-meeting gathering is where the real work is done.

Resulting Context: A perfunctory meeting will become a more positive experience to help *Build Trust* and solve real problems. The extensions before and after the actual meeting more than justify the rather routine occurrence of the meeting itself, which becomes a shared experience instead of an obligation. The time after the meeting becomes more worthwhile as an extension of the earlier socializing and the more productive meeting.

Rationale: There are many meetings whose real purpose is to get concerned parties together. The announced purpose may be, for example, to hear status information, but the true benefit is the personal exchange that happens around the meetings.

Spending time socializing before the meeting allows everyone to come to the meeting with a sense of camaraderie.

Author: David Saar, as told to Linda Rising.

Acknowledgements

Thanks to David Saar for the initial inspiration for these patterns, for spending time reviewing the patterns I captured during his presentation, and providing the quote from Mike Reynolds. Mike is our Division Vice President of Telecom Products. Mike has read the patterns and said he liked them!

Thanks to Greg Gibson for the aliases and the positive feedback. Thanks to Paul Bramble for the improvements to *Active Listening.*

Thanks to LIzette Velazquez for helping me eliminate the recursion from this pattern language and make the patterns stronger.