A lightweight and flexible process for designing intuitive error handling and effective error messages

Sarah Packowski

IBM Canada Lab

Abstract

Unintuitive error handling and ineffective messages result in lost revenue, wasted time, and unsatisfied customers. Yet, error conditions are often considered edge cases. As a result, a focus on error conditions is usually left until late in the software design and development cycle and is sometimes limited to just resolving unexpected test case failures. This paper outlines a process that enables software development teams to collaborate more effectively to produce intuitive error handling and useful error messages. The process is structured: there are artifacts to produce and rituals to step through. However, the process is also lightweight and flexible. In addition, the process scales well for small and large projects. This paper also describes the benefits that the IBM® DB2® for Linux®, UNIX®, and Windows® (henceforth referred to as IBM DB2 LUW) software development team discovered after adopting this process, including better quality messages, a shorter and easier-to-manage translation cycle, and improved integration between error messages and related product documentation.

1 Process overview

1.1 Process terminology

Throughout this paper, the term *message* refers to text that is returned to a user when there is an event. The message might be returned to the user in a pop-up window, on a command line, or printed in a log. The event might be the successful completion of a process or the failure of a component. Alternatively, the event might be a change of state that requires an information message.

Throughout this paper, reference is also made to *feature teams* or simply *teams*. A feature team is composed of eight to twelve people, on average. Each team is composed of architects, developers, testers, writers, and sometimes others, such as user-centered design specialists.

On each feature team, one person is identified as the *messages focal point*. The messages focal point drives the messages process for the feature team. For example, members of the IBM DB2 LUW Information Development (ID) team act as messages focal points for the feature teams of which they are a part.

1.2 Process steps

The main steps of the messages process are as follows:

1. Identify potential error conditions

The messages focal point reads all internal design documentation looking for potential error conditions.

Details about identifying potential error conditions are described in subsection 4.1.

2. Record potential error conditions in a messages scratch pad

The messages focal point creates a messages scratch pad to record notes about the potential error conditions.

Details about the messages scratch pad are described in subsection 3.1.

3. Have a messages meeting

The messages focal point schedules a messages meeting to step through the potential error conditions listed in the message scratch pad.

Details about the messages meeting are described in subsection 4.2.

After the messages meeting, the messages focal point updates the messages scratch pad with notes from the meeting and then proposes message text additions and changes. Another messages meeting might be needed, depending on the needs of the feature team and the amount of error handling and messages work.

4. Implement messages

The details of this step vary for different development organizations. After the feature team has reached consensus about the general content of their messages, then whoever is responsible for updating the message files would update them with the new or changed message text. For example, someone must update the files that return messages at run time. If messages appear in the external documentation, those files must be updated too.

5. Formally review messages

The messages focal point collects all the messages that are related to the team's feature and presents those messages together in a formal review. By reviewing the messages together, a team can more easily see whether the messages that a user might see while working with the feature are consistent.

Details about the formal message review are in subsection 4.3.

A team can step through this process once or multiple times throughout the development cycle. Details about incorporating this messages process in an Agile development environment are described in section 6.

2 Roles

2.1 Messages architect

In 2005, members of the User-Centered Design (UCD) team and the ID team at the IBM Canada Lab drove an error message improvement initiative. This small team improved the text of 20 messages that appeared in a disproportionately large percentage of user problem records. After these message improvements were delivered in the next release of the product, the incidence of the targeted error conditions and related messages in user problem records was reduced by 35%. For details of that initiative, see Appendix A.

As a result of the success of that initiative, the role of *messages architect* was created to drive further improvements to existing messages and to enable teams to design better error handling and write more effective messages.

The messages architect's responsibilities include the following tasks:

- Delivering message text changes to message source files
- Driving messages architectural changes
- Creating internal guidelines and technical and process documents
- Facilitating education sessions and workshops to help all members of the development organization write better quality messages
- Developing tools, tests, and automation to streamline processes
- Collaborating with other messages architects to drive industry standards

For example, the IBM DB2 LUW messages architect collaborated with several people to create the process described in this paper and worked with members of the ID infrastructure team to create the necessary supporting tools and infrastructure. With the assistance of writers from the ID team, she also created guidelines and process documents and drove education presentations and workshops to give the messages focal points the information that they needed to carry out their tasks. Throughout the development cycle, the messages architect participated in messages meetings, which are described in subsection 4.2, contributed to messages scratch pads, which are described in subsection 3.1, and generally sup-

ported teams and messages focal points though the messages process. She was responsible for propagating the message changes to the product engine code and updating any regression tests that failed as a result of message text changes. Finally, she assisted teams that managed translation and translation verification testing.

Having one person who is familiar with all aspects of messages and who can work with the different teams from the beginning of design to the end of translation verification testing is very beneficial. If one team works on message files, and then hands those files to a different team to process them, the members of the first team might not have a good understanding of the processes of the second team. When that happens, the first team might make architectural, design, or process decisions that negatively affect the second team. Also, if the second team has problems with the files, much time might be wasted if it is difficult for members of the first team to help with trouble-shooting because they do not understand the processes and tools that the second team uses.

2.2 Messages focal point

As described in section 1.1, a feature team's messages focal point drives the messages process for the feature team.

Although the messages focal point can be anyone on the feature team, there are advantages to having the writer who is responsible for the external documentation for the feature be the messages focal point for that feature team. For example, the writer is constantly thinking about aspects of the behavior of the feature relative to the user's context and about how to explain the feature in the user's terms. Effective error handling and messages must be created from that perspective too. Also, messages must integrate well with the external documentation. For example, to help a user find information about a given event, a message must use terminology that is consistent with the terminology in the external documentation to facilitate keyword searching, and the message might even direct the user to specific topics.

3 Artifacts

At various stages throughout the messages process described in this paper, certain artifacts are created. *Artifacts* are objects that you can see, read, edit, touch, or hold in your hand. Artifacts are not necessarily the final product. For example, an XML message file that is used internally is an artifact but is not delivered to users. However, a generated XHTML message file that is used in online documentation is an artifact, and it is a final product that is delivered to users.

3.1 Messages scratch pad

Unlike a traditional formal design document or specification that might "freeze" at some point in the design and development cycle, the *messages scratch pad* is a living document that continues to be used and changed throughout the development cycle. It is a place to record rough ideas and design decisions about error conditions and messages. Team members can collaborate by editing the messages in the scratch pad. Status notes such as "These three messages were delivered on March 19th" can be recorded in the messages scratch pad for future reference.

Although a feature team's messages focal point creates the scratch pad, everyone on the team must be able to read and edit the messages scratch pad. That is how the bulk of the collaboration and record-keeping is facilitated.

How the messages scratch pad is stored depends on the needs of the team using it. For example, some IBM DB2 LUW teams use an internal wiki page as a messages scratch pad. Other teams use a document in a database. It is important that the messages scratch pad be located somewhere permanent for reference throughout and beyond the development cycle.

The general layout and contents of the messages scratch pad also depend on the needs of the feature team using it. One strategy that the IBM DB2 LUW development teams find effective is creating a table to organize information about error conditions. Commonly, the table contains one row in the table for each error condition, and three columns: a description of the error condition; a draft of the related message; and associated notes. For an example of such a table, see Appendix B.

In the messages scratch pad, it is important to include or refer to new messages that are being created for the feature that the team is working on, existing messages that are being updated so they can be used with the feature, and existing messages that are related to the feature but are not being updated. Even if a team decides that a mes-

sage is not being updated for the feature, it is still valuable for the team to have the conversation about whether that message needs to be updated and record the decision. The conversation might spark an idea or identify a design point that was not considered. Also, the record of the decision might help resolve future confusion about whether a message was overlooked or intentionally left unchanged.

The messages scratch pad is an excellent place to record information about design choices and to collaborate on the text of messages. However, the messages scratch pad should not be used to store the final versions of error messages. As described in the subsection 3.2, it is best to store the latest version of messages in the message source files.

Finally, the messages scratch pad is meant to be lightweight, incurring minimal overhead. Find a balance between taking the time necessary to include enough information in the messages scratch pad to serve the needs of the team, and wasting time excessively formatting or reworking the messages scratch pad, which might not benefit the team.

3.2 Message source files

The format in which message source files are stored varies for different development organizations. For example, if the messages are included in the external documentation, there might be HTML files or PDF files that contain the text of messages. For the messages to be returned to users at run time, those messages must also be implemented in source files in the product, such as in Java .properties files or in files that use a proprietary message format.

Each IBM DB2 LUW message is stored in a single file that is in an XML-based format called Darwin Information Typing Architecture (DITA). Those DITA message source files are processed to create both XHTML files for the external documentation and run time message files of various types, such as Java™ .properties files. Implementing messages in a single source like this simplifies development, translation, and maintenance.

The messages scratch pad is important for driving error handling design discussions, recording notes about design choices, and for collaborating on message text. However, teams should create message source files as early as possible and make subsequent revisions of message text in the message source files directly, rather than in the messages scratch pad. It is best for the message source files to contain the most up-to-date version of the messages. For example, if multiple teams are working on error conditions that return the same message and each of those teams stores a detailed version of the message in their own messages scratch pad, a merge will be required when the teams try to deliver their changes to the message source file.

Infrastructure and automation can help teams stay organized and synchronized. For example, as mentioned, the IBM DB2 LUW messages are included in the product's external documentation. This documentation is in the form of an information center. The IBM DB2 LUW ID infrastructure team created tools that automatically build the IBM DB2 LUW Information Center each night, based on the latest version of the DITA files in the source file repository. Feature teams can therefore view the latest version of a message in the nightly build of the Information Center.

4 Rituals

At various stages throughout the messages process described in this paper, certain rituals are performed. These *rituals* are formalized actions or meetings that must happen to produce the artifacts, drive collaboration, and deliver the final products.

4.1 Identifying potential error conditions

As early in the design and development cycle as possible, the messages focal point reads through internal design information identifying potential error conditions and recording them in a messages scratch pad.

Internal design documentation can include e-mail, meeting minutes, proof-of-concept presentations, stakeholder presentations, high-level design documents, specifications, and other sources of information. Potential error conditions to look for include extreme conditions such as running out of memory, unmet prerequisites, incompatible components being used together, hardware or software component failures, and user errors. As described in subsection 1.1, messages are used for successful or failed events and changes of state, so successful events and changes of state that might

require an information message should be identified too.

The strength of this ritual is that the messages focal point is reading the internal design documentation with "messages glasses on." That means that the messages focal point is looking only for potential error conditions. This activity is very focused and efficient. Contrast this with a general review where the reader might stop to make a comment on each poorly constructed section, inconsistent description, or vague requirement. Many interruptions can cause the general review activity to take a very long time and make it harder to follow the flow of the document, potentially causing the reviewer to miss important points. By reviewing the design material with a tight focus, looking only for potential error conditions, the messages focal point is able to extract key information efficiently.

4.2 Messages meetings

After the messages focal point identifies potential error conditions and records them in a messages scratch pad, the messages focal point drives one or more messages meetings.

To prepare for a messages meeting, the messages focal point must have the messages scratch pad in good order: potential conditions must be listed, proposed new or updated messages must be written out, and comments and questions must be included.

The wider the range of experiences and perspectives of the messages meeting attendees, the more thorough and comprehensive the conversations will be. Ideally, the meeting includes the feature owner or architect, developers, testers, writers, user-centered design experts, and someone with a customer support background.

During the messages meeting, the messages focal point walks the group through the potential error conditions, asking questions and clarifying expected behavior of the product when the conditions occur. Updating the messages scratch pad throughout the meeting is a good way to keep track of the decisions and design points discussed.

When discussing particular error conditions, the messages focal point should do more than get consensus on the grammar of messages. The messages focal point must also direct the team's attention to questions such as these:

- What *precisely* is the error condition?
- What component or agent encountered or reported the error?
- What external behavior does the user see when this error condition is encountered?
- In what context is a user likely to encounter this condition?
- Why do we handle this error condition this way? Could we handle this in a different way, and possibly not return a message at all? Is the way that we are handling this condition intuitive for users?
- How should the user respond?
- What related external documentation would be helpful for users to review if this error condition occurs?

Through the resulting conversations, the team comes to fully understand the expected behavior of the feature and fills in any error-handling design gaps.

When the IBM DB2 LUW development organization adopted the messages process described in this paper, driving these messages meetings was initially somewhat uncomfortable for some of the writers who acted as messages focal points. Some writers were not used to questioning the design of features that they had been assigned to document. However, the writers ended up really enjoying these messages meetings. They felt satisfied because they were contributing to design choices more than they had before. Writers felt empowered because during the messages meetings, their teams inevitably uncovered some important aspects of their features that had been previously overlooked. The writers also indicated that they had a much more thorough understanding of the features that they were documenting, relative to previous releases, which made writing the external documentation easier and faster.

4.3 Formal messages review

After a number of messages have been created, the messages focal point drives a formal messages review. The messages focal point for a feature team identifies all the messages that are related to the feature the team is working on, including new messages, updated messages, and messages that are related to the feature but not updated, and presents all of those messages for review together.

Everyone on a feature team participates in the formal messages review. There might also be

people from other feature teams, product architects, or other stakeholders who that should be involved in the formal messages review. Even if a product architect does not have time to be involved in every messages meeting or e-mail chain, the product architect might have time to participate in the formal messages review. Also, even if a reviewer was not involved throughout the development of the feature, the reviewer can gain a really good overview of the feature just from reviewing all the related messages.

The tools and infrastructure used for the formal messages review can make a big difference to the experience that reviewers have. For example, if it is difficult to view multiple messages side by side for comparison, some of the potential benefit of the formal messages review would be lost. In general, the document that is used to review a collection of messages for a specific feature should meet the following criteria:

- Reviewers must be able to read and easily submit comments about the messages.
- Reviewers must be able to easily compare the contents of related messages, to assess whether the terminology in the messages is consistent.
- Reviewers must be able to search the collection of messages for key words or phrases.
- Reviewers must be able to easily navigate to related external documentation for reference when reviewing a message.

The messages focal point should schedule a formal messages review meeting for the end of the review so that reviewers can provide any additional comments that they did not submit using the review tools.

As the error conditions are discussed in messages meetings and message text is finalized through e-mail or updates in the messages scratch pad, everyone on the team should have a chance to provide input. However, while the team is in the middle of creating the messages, it is easy for someone to focus too much on one or two messages of interest. Sometimes it is not until all the messages are presented together in a formal review that inconsistencies or global mistakes become apparent.

5 Benefits

5.1 Assessing the benefits of the messages process

The messages that were created after the IBM DB2 LUW development organization adopted the messages process described in this paper were not yet generally available to users when this paper was written. Therefore, user data that indicates the impact that the messages process had on the quality of the product messages was limited when this paper was written. However, there are several other indirect metrics that can be used to assess the impact that adopting this messages process had on the quality of the product messages. These other metrics include efficiency of delivering changes, message quality evaluations, ease of translation, and the degree of integration between messages and product documentation.

5.2 Previous messages process

To understand the impact on the IBM DB2 LUW development organization of adopting the messages process described in this paper, it is important to understand the earlier messages process.

Before the IBM DB2 LUW development organization adopted the messages process described in this paper, developers used *defects* to request message changes or additions. A defect is an artifact that has a unique identifier, an owner, a target date for completion, and fields that contain information about the changes that are needed. There was a *messages team*, consisting of one person who spent 100% of his time working on messages and up to four other people who spent 50% - 100% of their time working on messages. This team of people responded to messages defects by manually making changes to the message source files, which were stored in multiple places and in multiple formats. Messages were sometimes included in formal design documents. When they were included, the entire text of messages was copied into the design documents.

5.3 Increased volume of message text changes

In the year before the IBM DB2 LUW development organization adopted the messages process described in this paper, the messages team

delivered 966 message content changes. In the year following the adoption of this messages process, the messages focal points collectively delivered 2010 message content changes. For details about how these numbers were obtained, see Appendix C.

Even though more than twice as many changes were delivered, no one person was over loaded with delivering message content changes, and feature teams waited less time for someone to respond to their message requests. Messages focal points were in constant contact with their teams, and because of supporting tools, they could deliver a message change in minutes.

5.4 Improved message quality

The quality of the messages that were created by messages focal points was significantly improved over messages created before the IBM DB2 LUW development organization adopted the messages process described in this paper.

A subset of the messages that were created before the new messages process was adopted was evaluated according to a list of quality metrics. A subset of the messages that were created after the new messages process was adopted was also evaluated according to the same quality metrics. The average quality rating of the messages added before the messages process was adopted was 10.5 out of 19, or 55%. The average quality rating of the messages added after the messages process was adopted was 15.4 out of 19, or 81%. That is a 26% improvement. For details about how the messages were selected and evaluated, see Appendix D.

5.5 Fewer translation problems

As described in subsection 5.2, before the new messages process was adopted, all message changes were delivered by a small number of people who were specialists in the slightly unusual format of the message source files, the DITA messages specialization. After the new messages process was adopted, messages focal points used Web-based tools to make changes to messages without having to know the underlying format of the message source files. The automated testing of the Web-based tools, as well as more thorough message content discussions and formal reviews described earlier in this paper, resulted in fewer errors in the message files.

During the translation cycle after the new messages process was adopted, the amount of errors reported by translators, including spelling mistakes, typographical errors, and DITA markup errors, was 32% less than projected, based on the volume of changes to be translated, relative to the previous release.

After the new messages process was adopted, because the task of delivering message content changes was distributed to messages focal points, the messages architect and infrastructure team members could focus their efforts on tools and automation. As a result of this infrastructure work, the amount of infrastructure problems reported by translators was 49% less than projected, based on the volume of changes to be translated, relative to the previous release.

For details about the number of translator questions and problems, see Appendix E.

5.6 Shorter, easier-to-manage translation cycle

After the new messages process was adopted, the messages translation cycle was 26% shorter than the translation cycle during the previous release. Also, more messages changes were delivered to translators earlier in the translation cycle.

Delivering a higher percentage of message changes to translation early makes it easier to adjust translation resources if the volume of changes is significantly different than anticipated. Also, if more messages are translated earlier, then translation testing can start earlier and will produce more meaningful results earlier, both of which reduce the amount of translation-related product defects that must be fixed late in the development cycle. Conversely, when a large volume of message changes is delivered to translators very late in the translation cycle, there is little time to adjust translation resources, and little time to respond to translation-related defects that result from those late changes.

In the release prior to adopting the messages process, at 67 days into the messages translation cycle, 85.9% of message changes had been sent to translation. After the messages process was adopted, at 43 days into the messages translation cycle, 96.9% of messages changes had been sent to translation.

In the release after adopting the messages process described in this paper, the percentage of message changes that was delivered to translation

in the final days of the translation cycle was reduced by nearly 10 times. In the release prior to adopting the message process, 1.9% of message changes were delivered to translation in the final 14 days of the translation cycle. In the release after adopting the messages process, 0.2% of message changes were delivered to translation in the final 12 days of the translation cycle.

See Appendix F for details about the translation cycle before and after adopting the messages process described in this paper.

5.7 Improved integration with product documentation

One advantage of having the messages focal point for a given feature team be the person who is writing the external documentation is improved integration between error messages and related external documentation.

After the messages process described in this paper was adopted by the IBM DB2 LUW development organization, the number of references to specific topics in the Information Center in IBM DB2 LUW messages more than tripled. For details about how this was measured, see Appendix G.

Having messages referring to specific topics in the external documentation is just one example of how product messages can be integrated with the external documentation. The following examples show some of the connections between error messages and related external documentation. If the related documentation does not exist, the messages focal point can make a note in the messages scratch pad to create the documentation.

- If a user encounters an error condition because the user did not meet a prerequisite, the message should direct the user to the topic that lists the prerequisites for the feature.
- If a user encounters an error condition because the user misunderstood a fundamental concept of a feature, the message should direct the user to the topics that outline those concepts.
- If a pattern of potential user errors emerges while the feature team is working through all the messages for a feature, the messages focal point can create some documentation to guide users through those pitfalls.
- If the "user response" text for one or more messages is long, maybe that information belongs in a task topic and the message should direct users to that task topic.

While a writer is working through the errors listed in a messages scratch pad, and while a writer is having conversations about those error conditions with the feature team in messages meetings, the writer is planning documentation that users will need if they encounter these error conditions. In this way, the messages process can even drive the development of the external documentation. Some argue that those topics that arise as a result of messages discussions are the very topics that are most likely to be sought out by users and should receive the greatest focus.

6 Agile

The messages process described here was designed to be flexible and fit well in different software development environments, including Agile. In fact, some IBM DB2 LUW feature teams found that incorporating some of the elements of this messages process helped make their team's transition to Agile easier.

6.1 Conveying information using conversations instead of documents

One of the principles behind the Agile Manifesto is using conversations to convey information. [1]

Before the IBM DB2 LUW development organization adopted the messages process described in this paper, details about which messages would be returned for specific error conditions and messages text were sometimes included in formal design documents. Feature teams used defects to request that members of the messages team deliver message changes to the message source files. The requested changes would be written in the defect or the defect would refer to the design document.

After adopting the new messages process, feature teams had conversations about error conditions in messages meetings. Feature teams were able to communicate their message requirements to the messages focal points much more clearly than they had been able to communicate their

messages requirements to members of the messages team using documents.

6.2 Deferring commitment

Deferring commitment is an important principle of Lean software development [2].

As described in subsection 6.1, before the IBM DB2 LUW development organization adopted the messages process described in this paper, details about which messages would be returned for specific error conditions and messages text were sometimes included in formal design documents. Those design documents were written so early in the design and development cycle that they very quickly became out-of-date. Teams would inevitably discover error conditions during development and testing that had not been included in the design documents. Message text in particular is very volatile: testers or translators often recommend changes to messages text.

The design documents would also "freeze" early in the development cycle. This meant that if a feature team wanted to update the messages information in the design documents, they had to navigate a change management process.

Moving the messages information out of the formal design documents and into messages scratch pads allowed feature teams to defer committing to details such as messages text or which messages identifier would be returned for specific error conditions.

6.3 Eliminating waste

Eliminating waste is another important principle from Lean software development [3].

In a release of IBM DB2 LUW before the new messages process was adopted, when messages information was included in design documents, the complete text of the messages was copied into the design document. Copying the messages to the design documents was time-consuming and tedious, and the messages consumed a great deal of space in the design documents, relative to other design information.

In the release before the new messages process was adopted, messages information accounted for approximately 4% of the total formal internal design documentation. However, as described in subsection 6.2, much of that information quickly became out of date and could therefore be considered waste. For details about how the number 4% was calculated, see Appendix H.

After adopting the new messages process, feature teams reduced the information about messages in their formal internal design documentation to a single sentence or paragraph directing readers to the messages scratch pad for the most current information. That represents a significant amount of waste eliminated.

6.4 "Whole team" and contributing as generalists

An important element of an agile team is the concept of "whole team" where team members contribute in any way they can [4, 5, 6].

Information developers, in particular, wonder what their role is in an agile development environment. For example, they wonder whether contribute as a generalist means that they must contribute C++ code to the product. The messages process described in this paper gives writers an opportunity to use the specialized skills that they already have to contribute in more ways than before.

In their traditional role, writers consume internal, sometimes disparate, sometimes complex sources of design information for a feature, and they produce a clear, simple explanation of the feature for external documentation. When acting as messages focal points for their feature teams, writers are doing exactly the same thing, but they are leveraging the clear, simple explanation to help their team be more focused, work more efficiently, and create a better quality product.

By acting as the messages focal point for their feature teams, writers contribute as generalists. They contribute to the design of their team's feature and to the test plans, and they reduce the potential for wasted development and test time by clarifying the vision for how the feature behaves in various error conditions.

Acknowledgements

The author would like to thank Serge Boivin, former IBM DB2 LUW messages architect, for his tireless patience and guidance. The author would also like to thank Cindy Stuy, manager of the IBM DB2 LUW ID team, for her support and help as well as Nina Kolunovsky and Farzana Anwar, both members of the IBM DB2 LUW ID

team, and Leslie McDonald, an editor with IBM DB2 LUW, for their review and editorial help with this paper.

About the Author

Sarah Packowski joined the IBM Canada Lab in 2000. She is currently the messages architect for IBM DB2 LUW. She is also a member of the Agile @ IBM community and an Agile workshop facilitator. Her Internet address is spackows@ca.ibm.com.

References

[1] The Agile Alliance, "Twelve Principles of Agile Software," 2001, <http://agilemanifesto.org/principles.html> (1 June 2009)

[2] Mary Poppendieck and Tom Poppendieck. *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley, 2006. p 32

[3] Mary Poppendieck and Tom Poppendieck. *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley, 2006. p 23

[4] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change, 2nd Edition*. Addison-Wesley, 2004

[5] Scott Ambler. Agile Developers: Generalists or Specialists? *Dr. Dobb's Journal* February 12, 2002

[6] James Shore and Shane Warden. *The Art of Agile Development Pragmatic guide to agile software development*. O'Reilly, 2007

[7] E. Wilde and R. Colvin. Editing for quality: how we're doing it. In *Professional Communication (IPCC) Conference Proceedings*, pages 296-303, Saratoga Springs, USA, 1996.

Appendix A: Error message improvement initiative

In 2005, members of the UCD team and the ID team at the IBM Canada Lab drove an error message improvement initiative.

The team created a Java application to analyze user problem records that IBM technical support collected while serving IBM DB2 LUW users. First, the Java application searched for the unique identifiers of messages in the user problem records. Then, the application produced a list of messages that it found in the user problem records and ranked those messages according to two criteria:

- The number of problem records in which the specific messages appeared
- The number of hours that IBM support spent resolving the user problems in which the messages appeared

The team identified 20 top-ranked messages out of approximately 8000 messages in one version of IBM DB2 LUW. These 20 messages appeared in user problem records that cumulatively accounted for 19.7% of the total hours that IBM support spent assisting users of that version of the product.

The team then collaborated with developers and writers to improve the text of the 20 top-ranked messages. No significant changes were made to the underlying error handling.

After the improved messages were released in the next version of the product, the team used the Java application again to analyze user problem records for the new version. The incidence of the 20 messages that were improved was reduced in the user problem records for the newer version. Those 20 messages appeared in user problem records that cumulatively accounted for 12.6% of the total hours that IBM support spent assisting users of the newer version. That is a reduction of approximately 36%.

This message improvement initiative shows that improving the text of error messages alone, without changing the behavior of the software, can reduce user problems and the amount of related support costs. The success of that initiative convinced project managers to allocate more resources for further error handling and messages improvements, including the creation of the role of messages architect and the creation of the messages process described in this paper.

For more information about the error messages improvement initiative described in this appendix, contact Rick Sobiesiak (ricks@ca.ibm.com), Serge Boivin (sboivin@ca.ibm.com), or Maciej Mazur (mmazur@ca.ibm.com) at the IBM Canada Lab.

Appendix B: A sample error conditions table for a messages scratch pad

In Figure 1 below, there is an example of a table of potential error conditions that might be included in a messages scratch pad.

Here are some elements of the table in Figure 1 that are of particular interest:

- The first column in the table in Figure 1 is labeled "Error condition." As described in subsection 1.1, the messages process described in this paper is concerned with successful or failed events and with changes of state. So even though that column label says "Error condition" successful events and changes of state that might require an information message should be listed here too.
- The second column contains the text of the proposed message. Including a snapshot of the message text is valuable for initiating discussion. However, as described in subsections 3.1 and 3.2, the messages scratch pad is not a good place to record the final version of a message.
- The notes in the third column are invaluable for clarifying understanding, driving information design, and recording why the team made certain design choices. For example, in the third column of Figure 1, you can see that the messages focal point has drawn out information from the team about the error condition, information that was not in the specification. Also, you can see a note reminding the writer to create a particular topic in the documentation.

Error condition	**Message**	**Notes**
A user tries to run command X of feature Y without having met prerequisite A.	DBX1234B Command X failed because prerequisite A has not been met. Explanation You can perform tasks 1, 2, and 3 by running command X of the feature Y. Before you can use feature Y, prerequisite A must be met. For more information about the prerequisites of feature Y, refer to the topic called "Prerequisites of feature Y" in the DB2 Information Center. User response To meet prerequisite A, perform the following steps: 1. .. 2. .. After satisfying prerequisite A, rerun command X.	[ID] Q: In the design documents, it says that A is a prerequisite, but it doesn't say what happens if a user tries to run command X without meeting prerequisite A. Does the database crash? [Dev. Team] A: We are not sure. We just know it is a prerequisite. Make sure to include a topic called "Prerequisites for feature Y" in the external documentation. [Tester] I have the environment set up, so I could try running command X without having met prerequisite A and let you know what happens.. [ID] Thanks! I will create the topic you suggested. One more question: If a user interrupts their work to meet prerequisite A when they get this message, do they need to restart the database? …
…	…	…

Figure 1: A sample error conditions table.

- If the format of the document used for the messages scratch pad supports it, using colors to distinguish contributions from different team members or to highlight certain contributions to the discussion can make the messages scratch pad easier to use. For example, in the third column of the table in Figure 1, you can see questions posed by the messages focal point and answers from developers. Using different colors for the questions and answers might have made reading that column easier.

Appendix C: Total number of message content changes

IBM DB2 LUW has more than 8000 messages. During a given development cycle, it is common for the messages architect or the ID infrastructure team to use scripts to deliver bulk changes to message files that do not affect the text of the messages. For example, during the development cycle just prior to adopting the messages process described in this paper, a large number of bulk changes were delivered because the format of message source files had just been changed from SGML to XML DITA. Just after the messages process described in this paper was adopted by the IBM DB2 LUW development organization, there were DITA markup changes delivered, using scripts, to most messages files to improve search results in the online version of the external documentation. In both examples, those changes did not affect the contents or the meaning of any messages and did not require he message text to b retranslated.

In order to make a meaningful comparison between the messages changes that were delivered by the messages team prior to the adoption of the new messages process and the message changes that were delivered by the messages focal points after the new messages process was adopted, a distinction must be made between *content changes* and *non-content changes*. Content changes include those changes that affect the text of a message and must be translated. Non-content changes include bulk changes to file formatting or DITA markup that do not affect the text of messages and do not require new translated text for the affect messages.

During the year before the new messages process was adopted a total of 9857 file changes were delivered to message source files. Using comments that were recorded by the file repository and management system when updates were delivered, it was determined that 966 of those 9857 changes represented content changes.

During the year after the new messages process was adopted a total of 9938 changes were delivered to the message source files. Using comments that were recorded by the file repository and management system when updates were delivered, it was determined that 2010 of those 9938 changes represented content changes.

Appendix D: Evaluating message quality

Michelle C Carey, of the IBM Silicon Valley Lab, created a technique for measuring the quality of a message that she calls *message scoring*. The technique involves creating a list of quality indicators and awarding a message a pass or fail for each of the indicators. The *score* that the message receives is the number of quality indicators for which the message received a pass. Some of the quality indicators that Michelle selected for her scoring process align with an existing quality assessment technique called Editing for Quality (EFQ.)[7] For more information about her messages scoring technique, contact Michelle C Carey at mcarey1@us.ibm.com.

To measure the impact that adopting the messages process described in this paper had on the quality of messages that the IBM DB2 LUW development team delivered, the following method was used:

1. **A list of 19 quality metrics was created.**

   Here are some examples:

   - The *severity indicator* seems appropriate for the error condition.

     The severity indicator is a letter at the end of the message identifier that indicates the severity of the error condition. For example, “E” represents “error” and “W” represents “warning.”

   - The “Explanation” section provides context and answers users’ questions such as “Why does this matter?” and “What are the implications or consequences?”

- □ The message is specific, as opposed to vaguely saying something such as "an unknown error occurred" or "reasons that this error might happen include" without indicating which of the reasons applies in the current situation.

2. **A subset of IBM DB2 LUW messages was selected.**

   There are a number of message groups in IBM DB2 LUW, distinguished by component prefixes such as "ADM," "SQL," "DBI," or "CLI." Each message group is used in different error conditions. For example, the DBI group of messages is commonly used for error conditions that are related to installation. The SQL group of messages is commonly used for error conditions that are related to running SQL statements. The majority of messages in the SQL group have the additional requirement that they must be consistent across IBM DB2 products.

   The ADM message group was chosen to be evaluated. This group was chosen because it is not constrained by external factors like the SQL message group is and because a significant number of new messages were added to the ADM group before and after the new messages process was adopted.

3. **The selected messages were evaluated.**

   All ADM messages that were added to the version of IBM DB2 LUW before the messages process described in this paper was adopted were scored. Number of messages scored: 27. Average score: 10.5 out of 19.

   All ADM messages that were added to the version of IBM DB2 LUW after the new messages process was adopted were also scored. Number of messages scored: 47. Average score: 15.4 out of 19.

Appendix E: Number of translation questions and problems

For any large software product, significant infrastructure is required to efficiently translate external documentation, messages, pop-up help, and GUI panel content. The translation infrastructure for the IBM DB2 LUW development organization includes a Web-based tool that translators can use to post questions. Translators use this tool to clarify a message to translate it, if they discover an error such as a spelling error or DITA markup error, or if they have trouble processing the message files with the translation tools. Archived articles in this Web-based tool were the source of translator questions and problems referred to in this appendix.

As described in Appendix B, during the release before the new messages process was adopted, there were 966 message content changes delivered. In the release after the new messages process was adopted, there were 2010 message content changes, more than double the volume in the previous release. This ratio was used to compare the amount of translation questions or problems before the messages process was adopted with the amount after. For example, if a certain number of translation problems were reported during the translation cycle before the new messages process was adopted, it would be reasonable to expect about twice as many such problems to be reported during the translation cycle in the release after the messages process was adopted. If there were fewer than twice as many translator questions or problems during translation after the messages process was adopted, that was considered an improvement.

In the release before the new messages process was adopted, translators reported 17 incidents of message file errors using the Web-based translation tool described earlier in this appendix. For the purposes of this paper, *file errors* include invalid DITA markup, spelling mistakes, and typographical errors. Given that the volume of changes translated doubled in the release after the new messages process was adopted, it would be reasonable to expect translators to report twice as many errors, 34, while translating the messages for the later release. However, the actual number of file errors reported in the later release was 23. That is 32% less than the expected number of file errors.

In the release before the new messages process was adopted, translators reported 18 incidents of *infrastructure problems*. For the purposes of this paper, infrastructure problems include not understanding the translation instructions that are sent with English message files, being unable to open the English message files in the translation tools, and being unable to generate the XHTML

or run time files from the translated DITA source. Given that the volume of changes translated doubled in the release after the new messages process was adopted, the expected number of infrastructure problems in the later release was 36. The actual number of infrastructure problems reported by translators was 19, which is 47% less than expected.

Appendix F: Messages translation cycle

The IBM DB2 LUW message files are first sent to translation for a given release near the middle of the overall development cycle. The message files are then resent to translation iteratively throughout the remainder of the development cycle. For example there were four messages translation shipments in both the release before the new messages process was adopted and the release after the new messages process was adopted.

For the purposes of this paper, the *length of the translation cycle* is the number of days that pass after the first time messages are sent to translation until the final time that messages are sent to translation. Before the messages process described in this paper was adopted by the IBM DB2 LUW development organization, the length of the messages translation cycle was 128 days. After the new messages process was adopted, the length of the messages translation cycle was 94 days.

When translators process the English files that they receive to be translated, they use tools to count the number of changed words. In this paper, the number that the translation tools calculate is referred to as *changed word count*. This number is somewhat abstract because a change to a single word in an English sentence might require the entire translated version of the sentence to be rewritten, depending on the language. However, for the purposes of this paper, the exact meaning of the number that the translation tools calculate is not as important as the fact that this number is positively correlated to content changes: the larger the changed word count calculated, the greater the amount of content changes.

Table 1 shows a comparison of the distribution of changed word count delivered to translation throughout the translation cycle for the release before the new messages process was adopted and the release after the new messages process was adopted.

Days since first translation shipment	Accumulated percent of changed word count shipped to translation	
	Before	After
0	53.9%	63.2%
43		96.9%
67	85.9%	
80		99.8%
94		100%
116	98.1%	
128	100%	

Table 1: Changed word count for messages delivered to translation before and after the new messages process was adopted.

In Table 1, you can see that the first messages translation shipment for both releases was on day 0. For the release before the new messages process was adopted, the second shipment happened 67 days after the first shipment, the third shipment happened 116 days after the first shipment, and the fourth shipment happened 128 days after the first shipment. For the release after the new messages process was adopted, the second shipment happened 43 days after the first shipment, the third shipment happened 80 days after the first shipment, and the fourth shipment happened 94 days after the first shipment.

Appendix G: Number of messages that refer to the documentation

To calculate the number of references from messages to specific topics in the IBM DB2 LUW Information Center, the message source files were searched for the text “Information Center.” This search would identify occurrences of statements such as “Refer to the topic called ‘X’ in the DB2 Information Center.” This search would not identify statements such as “Refer to product documentation for more information.”

In the release before the new messages process described in this paper was adopted by the IBM DB2 LUW development organization, the number of references to “Information Center” in message sources files was 66. In the release after

this messages process was adopted, the number of references to “Information Center” in message sources files was 226.

Appendix H: Calculating the percentage of total internal, formal design documentation devoted to messages

The IBM DB2 LUW development organization stores internal design documents for each release of the product in a central repository.

For a recent release, there were 572 design documents in the repository. Of those 572 documents, 125 documents of different types were sampled for analysis.

In the 125 sampled design documents, there were a total of 4644 pages. Of those 4644 pages, 246 pages were devoted to messages design information, which is approximately 5.3%.

There were several different classes of design documents in the 125 sampled documents. Each class of document had a different profile, and each class of document contained a different amount of messages information.

Based on the number of pages devoted to messages in the different classes of documents in the 125 sampled documents, and based on the number of the different classes of documents in the complete collection of 572 design documents, the amount of pages devoted to messages in the complete collection of 572 documents was estimated. Estimated total number of pages: 27248. Estimated total number of pages for messages: 1164, which is approximately 4.3%.