Metaphorscape

Patterns for XP System Metaphors

Rilla Khaled, Pippin Barr, James Noble
Victoria University of Wellington, New Zealand
{rkhaled,chikken,kjx}@mcs.vuw.ac.nz

Robert Biddle
Carleton University, Canada
robert_biddle@carleton.ca

Abstract

System Metaphor is one of the key practices of Extreme Programming (XP). Unfortunately, the System Metaphor practice is poorly understood, and is the practice XP teams most commonly choose to ignore. In this paper, we provide a small collection of patterns that teams can use to develop metaphors for their systems, and for evaluating system metaphors. We hope these patterns will encourage Extreme Programming teams not to abandon system metaphors, but rather, to continue to use metaphors to strengthen their development practices.

Introduction

Historically, design has been an important concept in object-oriented software development methodology. It lays down the architectural foundations for how the resulting system will be structured, and therefore impacts virtually *every* subsequent architectural decision. Extreme Programming is a relatively new methodology that has done away with the design phase of development, and instead features a practice called the *System Metaphor*. Kent Beck defines it as:

> *A story that everyone – customers, programmers and managers – can tell about how the system works* [4].

The system metaphor is a means of communicating about the project in terms that both developers and customers will understand, and which does not require pre-existing familiarity with the problem domain [5]. The system metaphor guides the mental models that project members have of the system, and shapes a logical architecture for the system.

Pattern	Problem	Solution
1.1 Metaphorscape	How do you begin identifying potential metaphors for your project?	Shape a metaphorscape with the team.
1.2 Locations	How do you narrow down the field of focus on the metaphorscape?	Identify certain locations and build on the metaphorscape.
1.3 Merger	How do you develop a shared understanding of the metaphors?	Merge people's ideas and let new ideas emerge.
1.4 Inspection	How do you construct a clear view of the System Metaphor as a group?	Carry out an Inspection on the top three metaphors.
2.1 Goodness	How do you measure the goodness of a metaphor?	Check for programmable ideas, system representations and vocabulary.
2.2 Weakness	How do you measure the weakness of a metaphor?	Check for undescribed system characteristics.
2.3 Lies	How do you measure the misleading notions of a metaphor?	Check for non-existent system characteristics and overcomplication.

Table 1: Summary of the patterns

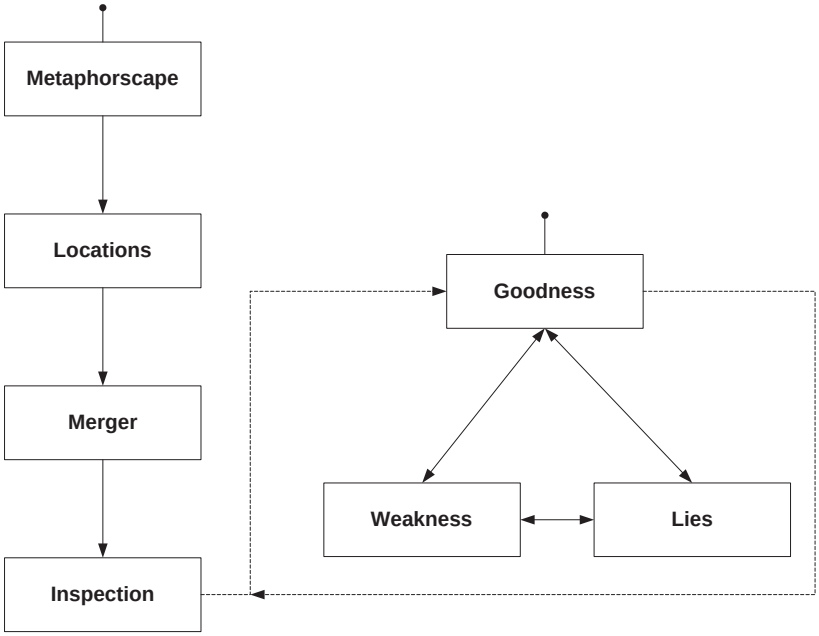

Figure 1: A relationship diagram for the Metaphorscape patterns. The directional arrows describe the path of pattern usage. Solid lines denote necessary paths while dotted lines denote optional paths. Starting points are represented by circle-topped lines.

After several years of using and teaching XP however, as well as observing the evolution of the XP community's beliefs and attitudes concerning XP, it was clear to us that the system metaphor practice seemed to be in trouble. We decided to investigate *why* people seemed to not use the system metaphor, as it is a key part of the XP methodology, and also scoured the existing literature on how to apply the Metaphor practice. Both endeavours pointed to the same conclusion: there was little information available on *what* exactly the metaphor could give to the project or *how* to begin applying it. In fact, experience with XP shows that the system metaphor practice is the most commonly dropped practice [13, 5, 12]. Yet at the same time, those who *have* managed to use metaphor tend to regard it quite highly [10, 9].

We hypothesise that this lack of knowledge about *what* a system metaphor adds to a project and *how* to apply it, is a root cause of why many XP practitioners seem to abandon the system metaphor practice. This paper describes two sets of patterns, aimed at addressing both of these issues. The first set described in Section 1 consists of patterns for establishing potential metaphors, while the second set in Section 2 contains patterns for evaluating potential and existing metaphors.

The patterns we present here form a part of a larger project we are currently working on that centres around the XP system metaphor practice. At the heart of the research is a structural model of system metaphor, based upon Peircean semiotics, giving a fundamental account of the way metaphors can contribute to a software system. The XP metaphor semiotics research came about as a follow up to an initial foray into the semiotics of user interface metaphors [3, 2].

Using what we had learned about metaphor from the user-interface metaphor work and from interviews with XP practitioners, academics, teachers and writers, we discovered that there was a sort of general process to how people went about establishing and applying metaphors. After some inspection, we divided the processes into semiotically-grounded activities which in turn gave way to these patterns. The patterns are also grounded in semiotics, although no experience or knowledge of semiotics is needed to understand and use them. We leverage the pattern format to introduce and explain any necessary theoretical background.

These patterns are intended for use by people interested in XP: this includes XP novices, up to seasoned XP practitioners. Our background research revealed that both groups are at times very confused about to make of the metaphor practice!

We have trialled all of the patterns both stand-alone, and in conjunction, on several system metaphors. We also gave the patterns to an XP team, who managed to use them effectively, and recently heard that the patterns were successfully used by 2 other teams elsewhere. We hope to test the patterns further on various groups to gain more insight on the system metaphor and we welcome any comments on the experimental usage of the patterns.

Each of the patterns features a **Motivation**, **Problem**, **Forces**, **Solution**, **Resolution of Forces** and **Related Patterns** sections. The **Solution** section for the patterns contained in Section 1 adhere to the following form: the first paragraph describes straightforward background set-up for applying the solution. The second paragraph explains the key concepts at the heart of the solution while any consequent paragraphs describe in detail how to carry out the solution. The patterns all make use of a common **Running Example**, explained in some detail in the following paragraph. The explanation of the application of the patterns

upon the **Running Example** is interleaved with the patterns, as the patterns (and the solutions they provide) largely carry on from one another. Table 1 summarises the patterns, and Figure 1 depicts the relationships between them.

Running Example Throughout the course of this paper, we will follow the progress of an XP team establishing a metaphor for an online networked music repository system. Music files that are part of the system are physically stored on separate machines but are retrieved and played by a main machine. Users may add and remove music tracks from a central playlist, and do *not* have to be using a computer storing music for the system in order to use the system. Tracks on the playlist are played over speakers broadcasting in a certain room.

1 Process Patterns

1.1 Metaphorscape

Motivation Lakoff & Johnson have defined metaphor as "*understanding . . . one thing in terms of another*" [11]. The System Metaphor practice is an application of this idea to XP systems: it is a way of explaining the logical architecture of a system by describing it in terms of something with which developers and customers are already familiar [6, 5]. A system metaphor facilitates discussion of the project in language that is accessible to both customers and developers, providing a shared vocabulary for discussing system problems and solutions [1, 13]. For developers, a system metaphor additionally supports consistency in naming elements of their programs, including subsystems, packages, classes, and methods [8].

The metaphor plays a role in shaping the "logical architecture" of the system. In *Extreme Programming Explained*, Beck gives explanations of how the metaphor shapes the architecture [4]:

> *The metaphor just helps everyone on the project understand the basic elements and their relationships. Words chosen to identify technical entities should be consistently taken from the chosen metaphor. As development proceeds and the metaphor matures, the whole team will find new inspiration from examining the metaphor.*

Problem How do you begin identifying potential metaphors for your project?

Forces

- The resulting metaphor should suggest a *useful* way of thinking about the major system components and interactions, that will influence the way code is written now *and* in the future. At the same time however, the amount of time spent establishing the metaphor should be kept to a minimum, as part of the XP mindset is avoiding investment in the future.
- The metaphor should be familiar to both the development team and client even though the client cannot be expected to possess the same level of technical understanding as the developers.
- Although the system metaphor is one of the 12 key practices of Extreme Programming, there is pressure not to use a System Metaphor, as many XP practitioners do not use it and say that it has not adversely affected them.

Solution A landscape can either describe an expanse of scenery or refer to an extensive mental view. We therefore use the word *metaphorscape* to describe *any and all* ideas for metaphors the team members have when they think of the project. These ideas do *not* need to be thought through extensively at this stage, they just represent possible metaphors,

Figure 2: Metaphorscaping.

each forming part of the metaphorscape. The potential metaphors should be recorded on a whiteboard by a metaphor session moderator, who can be anyone from the team. This activity should be bounded to a 10 minute time limit to prompt team members into suggesting ideas quickly.

Often potential metaphors will simply occur to team members. This is not always the case however, so to help inspire potential metaphors, we suggest thinking of a way to explain the system and its intended functionality to an audience of *non-experts*. Then try thinking of a "surrounding" metaphor that accounts for the explanation.

Taking the stereo system as an example, important features of the behaviour of the system include how it plays tracks from a playlist, which may be added from other computers. The music files corresponding to these tracks may be stored on any one of the machines that form part of the repository network, but only the main machine co-ordinates the "finding" and "activating" of the tracks. These behavioural features can be generalised into more abstract descriptions, e.g. making requests at a central venue, which are then executed under the control of one party. The party will carry out the task itself if it can, otherwise it will delegate the task to another party, as the original party has knowledge of what it, and every other party, is capable of achieving. Thinking about this type of control structure (still at quite a superficial level) with respect to the real world could point to surrounding metaphors such as a library, a job placement agency, a mail order company, and so forth.

This task will force team members to establish the major system components, their responsibilities and interactions, as well as a story that accounts for them - in other words, a first-pass metaphor.

Team members should try to vocalise any reasoning behind their metaphor suggestions during the group session, no matter how small, so that other team members may follow their chains of thought themselves.

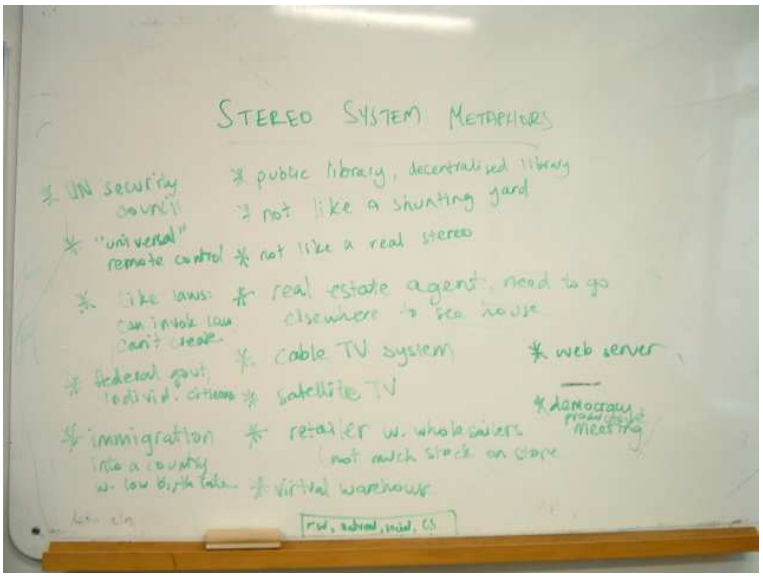

Figure 3: The metaphorscape.

Resolution of Forces Establishing a metaphorscape allows the team to come together and share any ideas, thus sparking new ideas from others. The aim is to develop a rich metaphorscape containing *many* metaphors, so the chance of it containing at least some useful ones is increased. Since the metaphorscaping stage does not require much justification to be supplied for each suggested metaphor and is also limited by an amount of time, time and effort expenditure are kept to a minimum.

Given that the metaphorscaping is done as a team, either it is the case that the metaphors suggested are already familiar, or team members can ask each other to clarify details. Since our metaphor-triggering tactic of thinking of a non-expert explanation caters specifically *for* non-experts, the client should not have trouble understanding the metaphor either.

Finally, in terms of both time and energy investment, this activity is *not* a costly one, and will help XP teams feel like they are staying "true" to the XP maxims. Even if teams *only* carry out this step then they will at least be much closer to potentially finding a suitable project metaphor than they would have been otherwise. In the worst case, even if no one can come up with *any* reasonable ideas, this activity is still advantageous in that it allows people to share their view of the system, thus leading to a shared system and project understanding.

Related Patterns The next step is to elaborate on some of the potential metaphors, which will involve using the **Locations** pattern.

Running Example The paragraph describing the stereo system featured in the introduction was read out to a team of 6 people, and one of the team members volunteered to take on the session moderator role. Initially, the team members were reasonably slow in suggesting ideas, of which the first few consisted of Public Decentralised Library, Not Like A Shunting Yard and Not Like A Real Stereo. At this point, the team agreed to stop suggesting "metaphors" of the form Not Like ... , as it is difficult to ascertain exactly

what information they yield since they effectively represent a negation of one entity from the universe! From this point on, ideas started to flow (and be voiced) more freely. The suggestions included, amongst others: Public Decentralised Library, Real Estate Agent, Immigration and Outsourcing, Retailer who deals with Wholesalers, and Phonebook. After about 10 minutes, the whiteboard had been sufficiently covered by ideas, and the team concluded the activity. Figure 2 shows the team in the process of suggesting metaphors and Figure 3 shows the complete list of suggestions.

1.2 Locations

Motivation The metaphorscape currently features a landscape of potential metaphors, which may or may not prove to be useful. None of the metaphors have been expanded much, and it is only possible to tell which metaphors are truly useful *after* expansion. Furthermore, although each potential metaphor was suggested by someone in the team, other team members may either dislike the suggestion or they may not even understand how it might work.

Problem How do you narrow down the field of focus on the metaphorscape?

Forces

- There are many metaphors within the metaphorscape, but it is desirable to pick a metaphor which is well-suited to the project.
- Focussing on many metaphors is a waste of time, and may make the decision process at the end more arduous, but focussing on only one metaphor may prove to be a waste of time if it turns out to be unsuitable for the project.
- Everyone in the team should have some input on the eventual system metaphor, but unfortunately often only the opinions of the dominant members are heard, as their opinions "drown out" the ideas of the quieter members.

Solution By group consensus choose what seem to be the three most promising potential metaphors from the list of suggested metaphors. Since it is likely that each person will have different ideas about what the metaphors suggest about the system, the session moderator should assign each one of the metaphors to every group member. Everyone should then work individually with their metaphor to establish what the metaphor means, or in other words, determine its *metaphorical entailments*. This metaphor expansion activity should take about 10 minutes.

The basic idea behind metaphorical entailments, introduced by Lakoff & Johnson [11], is as follows. Metaphors are often in the form A is B. To determine information about A, the thing or concept being described, we apply facts we know about B *to* A. For an XP system, A represents the system, so to determine metaphorical entailments of an XP metaphor, we apply what we know about B to the system. These entailments provide ways of thinking

Figure 4: The team, busy at work expanding metaphors.

about system components and interactions, and directly influence code, so it is important that lots of possible entailments are considered.

Sometimes it can be hard to think up entailments, so we suggest focussing on one part of the system, whether this part be a component or some specific functionality, then trying to map it to the metaphor in some way. Another suggestion is to reverse this technique, thinking about an entailment of the metaphor and finding parallels between the entailment and components and operations. In fact, entailments are not just related to the metaphor, they relate to each other as well. In other words, entailments are generative, and existing entailments might point to new entailments.

Resolution of Forces The metaphorscape contains a multitude of ideas, but by choosing the top three metaphors, it is possible for the group members to focus on the best metaphors in reasonable depth and concentrate on what the metaphors *mean*. By coming up with metaphorical entailments, it is much easier to identify in a somewhat concrete manner, what each metaphor may mean with regards to the system and also how well it will suit the system.

Furthermore, since each group member only looks at *one* metaphor, the time spent carrying out this activity is relatively small. Additionally, since three metaphors are expanded, the likelihood of at least one of them being useful is increased, therefore time wastage is less of a worry.

With regards to the difficulties encountered in group work situations, individual work on the metaphors ensures that each person gets the opportunity to think through his or her metaphor in a way which is uninfluenced by the opinions of others. In turn, this maximises the chance of getting metaphor entailments from multiple perspectives.

Related Patterns After the individual thinking sessions, use the **Merger** pattern to combine everyone's ideas.

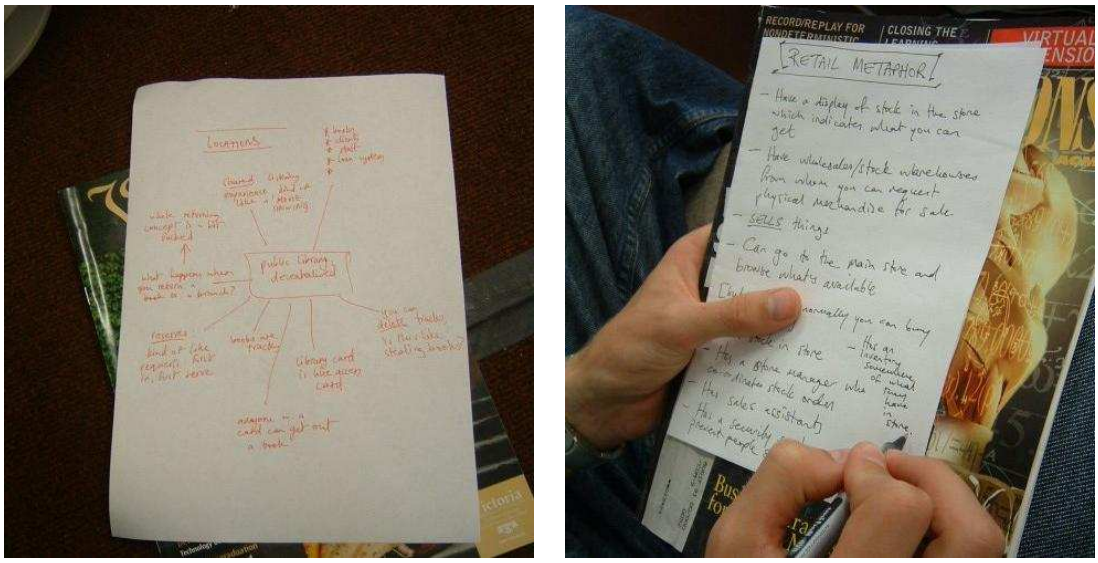

Figure 5: Entailments for different metaphors.

Running Example Faced with a board covered in ideas, but a requirement for three candidate metaphors only, the next task for the team was to decide which three metaphors immediately seemed most suitable. One team member said that he quite liked the Public Decentralised Library metaphor, and all of the other team members said that they agreed with him. Consequently, the Public Decentralised Library metaphor was made the first candidate metaphor. In similar fashion, the Phone System and Retailer Who Deals With Wholesalers metaphors were picked as the other two candidates, each time with either all or most of the team members agreeing on the suitability of each candidate. Having completed the selection, each of the three metaphors was allocated to two team members at random. Figure 4 shows the team expanding their metaphors, while Figure 5 shows two different sets of entailments. Notice that one of the entailments pages featured in Figure 5 contains a typical brainstorm diagram, while the other entailments page in Figure 5 contains a straightforward list of entailments. Different people have different methods and styles of coming up with ideas: we suggest people use whatever method they feel comfortable with. The team spent between 5 and 10 minutes on this activity - basically until they had stopped easily coming up with new ideas.

1.3 Merger

Motivation The team members have now individually expanded one of the three metaphors. They have not yet seen the entailments thought up by others either of the metaphor they were allocated, or the other two metaphors.

Problem How do you develop a shared understanding of the metaphors?

Forces

- While there are many potential metaphor entailments, only those that are compatible with the current understanding of the system should be used.
- Team members will have an understanding of the metaphor that they expanded, but this understanding may be vastly different from that of other team members who expanded the same metaphor.
- Team members will have an understanding of the metaphor that they expanded, but will potentially have little or no understanding of the metaphors they did not expand.

Solution The team should reconvene and share their individual entailment suggestions for the three metaphors, or any other ideas they have relating to the metaphors. The suggestions should be recorded by the session moderator in the form of three lists somewhere that everyone can see, such as on a whiteboard. The suggestions of the entire group are likely to be quite varied.

If we return to Lakoff and Johnson's definition of a metaphor, "*understanding one thing in terms of another*" [11], it becomes feasible that there is an infinite number of possible interpretations (not that all of these interpretations are necessarily *useful*). Metaphors are intrinsically generative. Consequently, although the team members may have "run out of steam" with their individual entailments lists, it is likely that exposure to the other suggestions will prompt new ideas in some people. The session moderator should record these new ideas also.

Take for example a hypothetical XP team that has chosen Mail Order Company as one of the candidate metaphors for the stereo system. After applying the **Locations** pattern, team member X has established the following list of entailments:

- The mail order central office is like the stereo system main machine.
- Products are like tracks.
- The mail order company probably obtains its goods from wholesalers, just as the stereo system obtains tracks from storage machines.
- To obtain anything from the company, a mail order form needs to be sent to the company, just as music requests need to be added to the playlist.

Team member Y's list consists of the following:

- Products = tracks.
- Company probably obtains goods off-site.
- Mailing time (round trip time from and to customer) akin to song waiting time.
- To match with stereo, only one order able to be delivered at a time? Probably not how mail order companies work...

- Mail order company would have security protocols, just as stereo system presumably has some sort of authentication protocol with storage machines.
- No face to face interaction (apart from delivery people) - like with stereo system.
- Mail order office may change physical premises, but this will have no effect on customers provided that their mail gets redirected to the new office. Similarly, physical location of main machine is irrelevant to stereo system users, who make requests from online interface.

After the team has regrouped, X and Y share their entailment lists with everyone, and a joint list is made of both lists. Both X and Y agree that products are like tracks and that wholesalers can represent storage machines. Y's entailment about one-at-a-time delivery gets everyone thinking however, and team member Z suggests that perhaps the mail order company takes care of its *own* delivery (as opposed to outsourcing to a delivery company). Furthermore, maybe it has just one delivery truck and can consequently only deliver to one location at a time. X points out that perhaps the entailment of products being like tracks is not a good one, because people placing orders from one location may order multiple items, but unless a very inefficient system of delivery is used, they will all be delivered simultaneously.

This type of discussion can be very fruitful in letting all of the team explore the potential metaphors in certain directions to scope out how, where and why the metaphors are both strong *and* weak.

Figure 6: The recombined team, going through the Library metaphor.

Resolution of Forces By developing all three lists in front of the whole team, everyone has an opportunity to contribute and discuss their entailments with regards to what they know about the system. In turn, this strengthens the team understanding of the system

A book is like a track.	Just as tracks from the music repository may be played, one can borrow books (and other unit-style items) from the library.
Access to the library requires membership of some type.	Access to the playlist requires having access to the machines that are able to manipulate the playlist. This implies some degree of "membership" to the graduate computer science labs.
Reserving a book is like making a request.	When a playlist request is added, a track gets added to the list, for the purpose of being played at some point in the future. Likewise, when a book is requested from the library, it is for the purpose of someone eventually being able to borrow that book.
First in, first served.	For basic purposes, all users of the stereo system have equal influence over the playlist. In other words, there are no special users whose requests are immediately jumped to the top of the list by the system. Similarly, the library (hopefully!) runs on such a basis, where people get to borrow books according to the time order in which they either found the book on the shelf, or placed a reserve.
Deleting a reserve is like removing a track from the playlist.	Removing a track from the playlist means that it will not be played at some point in the future. Likewise, deleting a book reserve means that the person who initially wanted to borrow the book will not be notified when it becomes available for loan.
The listening experience is shared, like a shared "Talking Book" session.	At a "Talking Book" session, everyone listens to a recorded book reading. In the same way, listeners of the stereo system all listen to the same music track.
Reading a book blurb is a type of content preview.	Reading a book blurb yields some fast, basic information about the book, typically related to its plot. Listening to a track preview may give the listener some idea of what the entire track will be like, and reading its file information will provide information about its duration in time, perhaps also its title, artist and distribution.
A request to the library can go to other branches if the book isn't available.	If the central machine does not store the requested track itself, it will retrieve the location of the track from the machine that *does* store the track, prior to playing it.
Some libraries don't have shelves: everything comes from a hidden stack.	"Browsing" can make up a large part of the library experience, i.e. looking at a book without necessarily intending to borrow it. It is unclear how browsing relates to the stereo system, which seems more akin to the hidden stack-style library.
Group environments changed by replacing silence with music.	Libraries typically have silence rules, so that everyone can read or study peacefully. The stereo system on the other hand serves to provide sound for an audience, therefore in this respect it is the opposite of a library.

Table 2: Entailments for the Public Decentralised Library metaphor.

Like a monopsony[a], only one buyer.	The stereo system has a main machine that co-ordinates the finding and playing of the tracks. In a retail scenario, in lots of cases it is possible to purchase the same product from a number of retailers, but this is not mirrored in the stereo system. Therefore the monopsony condition in the retail metaphor is necessary.
Like Argos in the UK, no stock on outside shelves.	Having no stock on the outside shelves implies that a request for the stock needs to be made by the customer before it can be obtained. This corresponds to how tracks must be placed on the playlist before they can be played.
Wholesalers can go bust or new ones can come up.	Storage machines store the music files, just as wholesalers supply stock. Just as wholesalers are not necessarily permanent, and there are usually a few of them, storage machines come and go from the network.
Retailer collates catalogue from wholesalers.	The retailer knows what each wholesaler is able supply. This is similar to the main machine keeping tabs on what the storage machines have, and it is from this that the main machine collates a play index that stereo system users view to make music requests.
3 roles: customer, retailer, wholesaler.	The three roles relate to the stereo system as follows: the customer is the system user, and is able to make and remove requests, as well as "listen". The retailer is the main machine, which co-ordinates finding and playing of tracks. The wholesalers are the storage machines, as they effectively "supply" the music files to the main machine.
Security systems on retailers and wholesaler premises.	The central and storage machines also have security protocols between them, to ensure "safe" system exchanges. Security in this sense does not however extend to prevention of outsiders breaking into the labs and requesting pumpin' beats!
Retailers send stock requests and get sent back stock.	The main machine and storage machines have a similar relationship: the main machine will reach a certain point in the playlist where a certain track is called for. The main machine then finds the storage machine that stores the track, retrieves the track from the storage machine into a temporary buffer and plays the track.
A standing order is like the playlist on loop.	A standing order is an order that repeats over a certain time frame. When the playlist is on continuous loop, the same sequence of tracks will get played repetitively unless someone changes the listing.

[a]The Oxford English Dictionary defines a *monopsony* as: "A state of the market in which there is effectively a single buyer or consumer for a particular product, who is therefore in a position to influence its price; a consumer in this position" [7].

Table 3: Entailments for the Retailer who deals with Wholesalers metaphor.

A phone number is a machine: multiple users per machine/number.	Each machine can be logged onto by various people, or conversely, one machine may store files for various people. Likewise, there may be a group of people who are within proximity to a certain phone and can therefore be reached by it.
Phonebook is only updateable at one location, updating your copy at home has no effect on someone else's copy.	Phone books are typically published yearly, containing new numbers for some parties. People can also choose to be unlisted, so these numbers will not be listed in the new phonebook. While it is possible for you to have the number of an unlisted friend, people will not be able to use the phonebook to *find* the number of the friend. Likewise for the system, just because you may have some tracks floating around your storage space, unless someone requests the tracks on the system playlist and supposing the stereo system is the only means of listening to the track, no one will be unable to listen to them.
Someone who uses the phonebook does not need to be listed in the phonebook.	In order to make requests on the playlist, users must log into the MCS (Mathematical and Computing Science) system at Victoria University, which means that the system potentially has a wide user base. In contrast, the number of storage machines is quite small. Considering phonebooks, the Wellington Phonebook contains numbers of people and parties contained within the greater Wellington region, which has an estimated population of up to 500 000. Anyone in the world however, can use the Wellington Phonebook to call someone in Wellington (provided they can access a copy of it), so the Wellington Phonebook user base is potentially 5 billion.
The phonebook is the public location of all of the numbers.	Likewise, the main server knows the location of all the public files, which are listed by name in a central play index.
Phone numbers are like tracks: users effectively "dial" songs.	Dialling a phone number and waiting for the person on the opposite end to answer is somewhat similar to making a request for a track and waiting for the track to be played over the stereo system. Of course this similarity rests on the slightly absurd assumption that only one set of parties on the phone network can communicate at a time.

Table 4: The entailments thought up for the PHONEBOOK metaphor.

Phonebook is "distributed" via WAP and the web.	Typically phonebooks are distributed to residences, companies and organisations at particular times in the year. They can also be found in public phone booths (provided they haven't been stolen or vandalised beyond use.) The stereo system has an online interface (also accessible via WAP), from which the playlist can be manipulated.
Phone is always on speakerphone or party line setting.	The broadcast nature of the stereo system means not only that the audience usually consists of a group, it is *intended* for a group audience. The typical phone call however is conducted between two parties only, unless it is a speakerphone or party line in which case many people can communicate with many people. The speakerphone/party line condition makes the metaphor more suitable for the system.
Continued looping of track like repeated ringing: telemarketers?	Resting on the interpretation of phone numbers being like tracks, telemarketers will often repetitively try phoning people at a certain number until they manage to speak to someone who they want to speak to/wants to speak to them. Looping phone requests may be considered similar to this scenario.

Table 5: Continuation of the entailments thought up for the PHONEBOOK metaphor.

and increases the likelihood of the entailments "fitting" the system. Some of the resulting entailments will be more useful than others, in terms of applicability to the system. Establishing which entailments are useful and which are not may require some discussion within the team. As long all of the entailments are recorded together however, it will be easier for people to mentally juxtapose entailments to determine for themselves which entailments are compatible, and which are not.

As well as strengthening understanding, this task serves as an iteration upon previous stages. Since everyone has worked on one of the three metaphors, seeing how other people interpreted the same metaphors may trigger in them new ideas about how the metaphor works. Perhaps the interpretations of the metaphor were the same, or perhaps they were different and can be merged together. If no such merge is possible, at least this opportunity allows team members to get together and discuss how they came up with entailments.

Having the whole team present during this task ensures that everyone gains an understanding, and as a consequence an appreciation, for each of the three metaphors, not just the ones they worked on. This is essential, as it will eventually be up to everyone to decide which metaphor will get chosen for actual use.

Related Patterns To fine-tune the metaphor lists and to eventually pick a system metaphor, use the **Inspection** pattern.

Running Example The team regrouped and looked at the PUBLIC DECENTRALISED LIBRARY metaphor first. As the two team members who had worked on that particular metaphor read out their entailments, the person in the moderator role re-iterated each entailment while writing it on the board, and placed it into the context of the project, in "thinking out loud" style. This process invited commentary, discussion and new entailments from all of the team members, and helped ensure that everyone had a similar understanding of what each entailment meant. For the remaining two metaphors, a similar "thinking out loud" process took place during write up. Tables 2, 3, 4, and 5 contain some of the entailments suggested for the PUBLIC DECENTRALISED LIBRARY, RETAILER WHO DEALS WITH WHOLESALERS, and PHONE SYSTEM metaphors, along with summaries of the accompanying discussions. Figure 6 shows a team member writing up the entailments thought up for the PUBLIC DECENTRALISED LIBRARY metaphor.

1.4 Inspection

Motivation An important aspect of the System Metaphor is that everyone must understand *how* it works for the system as well as *how far* it works for the system. All the team members must share an approximately identical view of the metaphor in order to benefit from its use, but unless the "meaning" the team has ascribed to the metaphor is coherent, uncontradictory and largely free of holes, it will not be possible to have a shared understanding.

Furthermore, ideally there should be just one system metaphor.

Problem How do you construct a clear, coherent view of the System Metaphor as a group?

Forces

- Some of the entailments will be well suited to the system, but others will not fit in with the current understandings, while others still will be plain incorrect.
- Only the suitable entailments should be kept, but some team members may feel defensive about the possible abandonment of their suggested entailments.
- There are three metaphors under consideration, but ideally only one metaphor should be chosen as the system metaphor.
- One metaphor should be chosen, but perhaps all three of the candidate metaphors are poorly suited to the project.

Solution Examine the lists as a group, and by consensus, cull the lists for correctness, consistency and coherency by getting the moderator to cross off unsuitable entailments from the whiteboard. Unsuitable entailments are those that are incorrect with respect to the workings of the system, inconsistent with each other or focus on irrelevant details.

While the entailments may initially act as simple ways of explaining system behaviour and interaction, as the system becomes more complex the entailments may *suggest* ways in which

existing/new code should act. Therefore, as a group, identify major system components and interactions, and make sure a lot of them are "covered" by the suggested entailments. Furthermore, only keep entailments which support and are consistent with the known workings of the system. This decreases the likelihood of future confusion suffered by team members regarding how an entailment adds understanding to the system. It is important to bear in mind that the remaining entailments *work together* as well, as it relates to the shaping force the metaphor will exert upon the project. Establish as a team which entailments deserve to be kept and which should be culled, so that everyone has the same understanding of how the metaphors works *for* the system, as well as *how far* the metaphors works. This activity may take only 10 minutes, or it may take up to half an hour. It is important that team members get to thoroughly discuss the entailments as the understanding gained during this process will strongly influence their understanding of the metaphor for the duration of its lifetime in the project.

By now it should be clearer how good the three metaphors are with respect to providing a useful vocabulary for the system and explaining the known system components and functionality. It should also be clearer whether the metaphors imply non-existent behaviour. Team members should vote for their top choice of metaphor, and if there is one clear winner, make this the system metaphor. If there is no clear winner and all of the metaphors have their strengths and their weaknesses, use a combined metaphor for the system.

Resolution of Forces By allowing everyone to partake in the culling process, not only are the entailments more likely to be reviewed objectively, but everyone will be clear about which entailments are kept and which are rejected.

Since the decision to keep or discard entailments is carried out as a group, there is likely to be good discussion about why or why not entailments should remain. While certain team members may feel strongly about suitable and unsuitable metaphors, since the decisions are all made by group consensus, their opinions will be treated equally alongside those of others.

After a group examination of the three metaphors, people will be in a position to make an informed decision about which metaphor seems most suited. In the straightforward case, this can be achieved, and suddenly the project has a System Metaphor!

If it seems that all three metaphors are weak, then either it is the case that the *wrong* metaphors were chosen to focus on, or it is the more likely case that the alternatives to the current three were even worse. Sometimes this is unavoidable, so other than doing another iteration of the entire process, it may be useful to use a *combined metaphor*, which features the best characteristics of all three metaphors and ignores the weaker characteristics.

Related Patterns This pattern essentially relies on using gut instinct to choose the best of the three metaphors. An optional more structured approach to evaluating metaphors is described in the **Goodness**, **Weakness**, and **Lies** patterns.

Running Example When the team members reached this stage, it became apparent that they had all clearly thought out their entailments quite well because it was difficult to establish where the metaphors fell apart based on the suggestions on the whiteboards. It

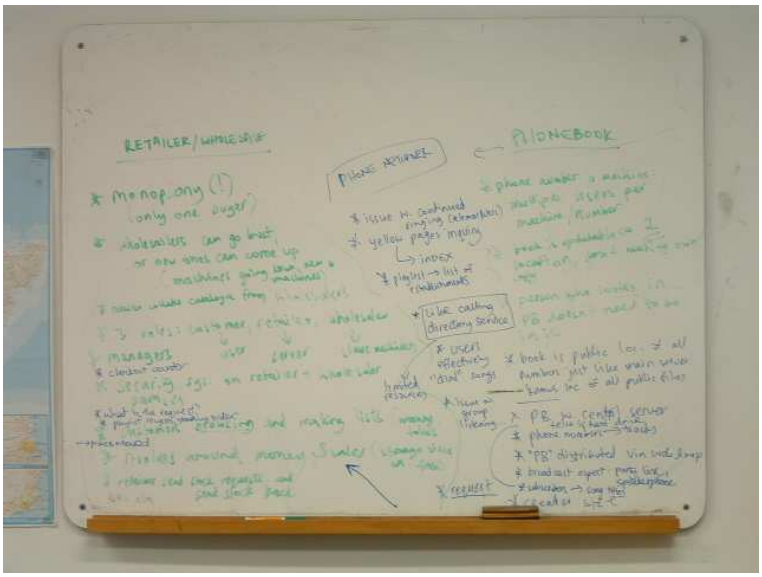

Figure 7: Two metaphors undergoing **Inspection** on the whiteboard.

seemed that the team members simply did not suggest entailments that they felt might be problematic, or if they did mention these types of entailments, they would invite others into a discussion of its justification during the **Merger** pattern. We believe it is important to find out whether the metaphor does match the system or if not, *how* it fails, so we would emphasise the need to put up at least some silly suggestions!

The team started off by "inspecting" the PUBLIC DECENTRALISED LIBRARY metaphor. A limitation of this metaphor that was broached is that while listening to the stereo is often a shared experience, in most cases, items taken out from libraries are viewed or experienced by an individual. Furthermore, with libraries you can often see a book on a shelf that you want and get it out immediately, you do not necessarily have to reserve it. Also, people do not usually get the same books out over and over again, but people seem happy enough to leave the stereo playlist on loop. Although the entailments listed for the metaphor seem to work together, the team expressed that they were unconvinced that it served as a good explanation for the stereo.

The team then moved on to the PHONE SYSTEM metaphor entailments list, which contained inconsistencies as both team members who had worked on the metaphor had interpreted it differently. After some debate between these members, the metaphor seemed to morph into PHONE NETWORK, and eventually into DIRECTORY ASSISTANCE, whereby a user phones a central information number to be connected to *another* phone number, therefore the people manning central information number act as the central server, and the numbers they connect to are like music tracks. This metaphor started becoming fiddley when the team established that only one central information helper was available and that since only one user could be connected at a time, users would have to leave messages about who they would like to be connected to in the near future. Eventually when team members started quibbling about the issue of repetitive ringing, this metaphor was also abandoned.

The last metaphor the team was left with was the RETAILER WHO DEALS WITH WHOLE-

SALERS metaphor, for which the entailments list was already internally consistent. Admittedly this metaphor shared a few of the problems suffered by the earlier metaphors. For example, music listening can be a shared experience, while an ordered item is usually only of benefit to the customer who ordered it, as opposed to all customers of the retailer. But with this metaphor, issues like only one client being serviced at a time seemed more reasonable, as this could well be the case depending on the mode of delivery the retailer uses to transfer goods between the wholesaler and the client. The team felt happiest with this metaphor, so they chose it as the system metaphor.

2 Evaluation Patterns

2.1 Goodness

Motivation For an XP team to adopt a certain metaphor, ideally they want to feel convinced that the metaphor they are choosing is well-suited to their project. Unfortunately it is not fully possible to evaluate how well-suited a metaphor is for the project until *after* it has featured in the project, and has either succeeded, failed, or performed somewhere in between.

Problem How do you measure the goodness of a metaphor?

Forces

- You want to pick the best metaphor for your project, but different metaphors are useful in different ways.
- The metaphor may seem good for the project in its current incarnation but may become unsuitable later.

Solution Since each potential metaphor probably evokes a different understanding of the system, it is necessary to start mapping out what this understanding might be. Questions to ask in determining the goodness of the system are:

- **do the entailments of the metaphor contain programmable ideas?**
  Coding ideas inspired by a metaphor should be consistent. This ensures the overall coherency of the metaphor and consequently increase the likelihood of a *shared* understanding of it. In turn, the shared understanding should lead to more coherent, consistent and simple code. If more than one metaphor is used, the consistency applies to *each* metaphor individually.

  Running Example Applying this pattern to the Retailer who deals with Wholesalers metaphor, some potential coding ideas are the following:

  - Adding a track to the playlist from the play index is like placing a stock order from the catalogue.
  - The notion of an order could be construed as a communications protocol.
  - Just as an order can be placed and cancelled, tracks can be added and deleted from the playlist.
  - Just as there is one central server, the retailer is a monopsony.
  - Wholesalers are like slave machines in that they can go down (go out of business) or come up (new wholesalers can appear on the scene).

  - Related to the previous idea, a specific wholesaler's stock can appear on the catalogue of items when the wholesaler is available for trade or disappear when the wholesaler is not available for trade, just as certain music tracks are available on certain machines only.
  - Certain security measures need to be taken on both the wholesaler and retailer premises, similar to security measures on slave and central machines.

- **does the metaphor addresses the major system components and their known functionality?**
  This can be checked by examining the entailments resulting from the metaphor to see whether entailments exist to describe major components and also important functionality. Entailments may need to be broken down into smaller entailments before this can be checked. This criterion is similar to the first in that it also probes the relationship between the metaphor and the system, but it differentiates itself from the first in the level of detail at which the metaphor *represents* the system. From another perspective, while the first criterion is examining an overall "story" that the metaphor suggests, this criterion focusses on "characters".

  Running Example Considering the Retailer who deals with Wholesalers metaphor, major system components and known functionality can be mapped to the metaphor as follows:

  - Central server: retailer.
  - Slave machine/storage machine: wholesaler.
  - System user: client.
  - Play index: catalogue of goods.
  - Ordered playlist: list of prioritised orders.
  - Track from play index: catalogue item.
  - Track from playlist: item on order list.
  - Adding/removing tracks to/from playlist: placing an order/cancelling an order.
  - Track information: item information.
  - Slave machine's stored tracks: wholesaler stock list.
  - Retailer security: central server security.

  Clearly, there is some cross-over between the results yielded by this question and the last. Having a structured way of thinking about the metaphor however enables it to be explored potentially in a more in-depth manner than it would be otherwise.

- **do the metaphor entailments provide a vocabulary with which to describe the system?**
  Sometimes a metaphor can still be helpful even if it does not fully address either of

the above concerns. While entailments of a certain metaphor may not necessarily deal with concrete programming issues, they may provide a way of describing the workings of the system, which is useful for communication amongst the *entire* development team, which is to say developers *and* customers.

Running Example Some useful vocabulary we have already established with the RETAILER WHO DEALS WITH WHOLESALERS metaphor includes: **retailer, wholesaler, client, catalogue of goods, list of prioritised orders, catalogue item, item on order list, placing an order/cancelling an order, item information, central server security, monopsony, wholesaler bankruptcy, new wholesaler** and **wholesaler stock list**.

This task should take approximately 20 minutes.

Resolution of Forces In the ideal scenario, it would be easy to find a perfect metaphor for the project. Unfortunately, this is usually not the case and it is necessary to make do with what *is* currently "available" (in terms of ideas). By using three different ways of thinking about how a metaphor may be good, it becomes easier to evaluate what each metaphor may bring to the project, be it an overall story, actual system component representations or a useful vocabulary. A truly good metaphor will probably meet all three criteria, but metaphors which meet only one or two of the criteria may still prove to be quite useful.

The measure of the metaphor's goodness supplies a reasonably fast way to evaluate how useful the metaphor is in the short term. Admittedly however, it cannot predict how the project will change over time however, as change is simply part of the expected XP project lifecycle.

Related Patterns While a metaphor may seem reasonably good, within it there may lurk misleading notions about the system, or completely unaddressed system parts. Use the **Weakness** and **Lies** patterns to gain an overall impression of a metaphor.

2.2 Weakness

Motivation A metaphor may *seem* to be suitable on first inspection, but a good performance under certain criteria does not exclude the possibility of a bad performance under *other* criteria.

Additionally, sometimes it seems impossible to come up with even a single metaphor for a project. The suggested strategy for dealing with this problem is to use a combined metaphor in the place of an all-encompassing metaphor. Yet in order to choose *which* metaphors to combine, some amount of attention should be paid to the disadvantages of each metaphor, as presumably they are all flawed in various ways.

Problem How do you measure the weakness of a metaphor?

Forces

- You want to *avoid* picking an unsuitable metaphor for your project, but different metaphors are unsuitable in different ways.
- If it has already been identified how a metaphor is *not* suitable, the inclination is to discard it, as many XP projects forge ahead without any system metaphor. On the other hand, these metaphors may still yield a useful insight into the system.
- A metaphor which may seem unsuitable prior to usage may or may not become *worse* after it has been used for a while. Without actually experiencing its use however, it is hard to determine what will happen.

Solution At a basic level, the metaphor exists to explain the system. Disregarding for the time being the varying levels of the *quality* of explanation the metaphor provides, to determine the weakness of a metaphor, it pays to consider:

- **which known system components and interactions are left undescribed by the metaphor?**
  This question is effectively asking *to what extent* the metaphor is not achieving its basic goal of explaining the system. To avoid confusion within the team regarding holes within the metaphor, each undescribed component or required functionality should be addressed either by another metaphor or by means of explicit description. A metaphor that is not contributing much may need to be dropped, as the existence of too many metaphors complicates the shared understanding of the system.

  Running Example Applying this pattern to the RETAILER WHO DEALS WITH WHOLESALERS metaphor, system components and behaviour/interactions that remain undescribed include the following:

  - Tracks can be viewed as continuous, throughout the duration of their playing. Orders seem to be more like discrete objects.
  - The playlist always loops, but orders are usually not standing orders.
  - Playlist items can be paused, causing music playing to completely cease for that moment. It is unclear what pausing an order might be, certainly it should not affect all of the other clients who have orders placed behind the current order.
  - Any user of the system can jump to any track on the playlist to cause it to play immediately, but clients cannot (usually!) force the retailer to ignore the orders of others that came earlier and prioritise their own orders.
  - Users of the system can delete tracks from the playlist they did not add, but clients cannot usually cancel the orders of other clients.
  - While anyone sitting near the speakers can "benefit" from someone's requested track by being able to listen to it, this is not the case for the retailer's clients, i.e. a client does not immediately "share" her purchase with other clients.

– The system can potentially become lagged or crash, but it is unclear that the reasons why a retailer might suddenly close shop are relevant for explaining a system crash.

This task should take between 10 — 20 minutes.

Resolution of Forces This pattern provides a way of gauging how a metaphor may fail to provide a suitable story for the system. Applying this pattern to a group of metaphors which seem weak will shed light on how the various metaphors both cater and fail to cater for the system in question.

Using the pattern can also shed light on positive aspects of the metaphor, namely whether or not the metaphor is "salvageable", i.e. if with minor patches, it can still be used. It should not be underestimated how much technical jargon confuses non-programmers, therefore as much as possible, the team should strive to able to communicate project concepts in a manner that is not overly technical.

The pattern also supplies a way to evaluate how well the metaphor explains the knowledge *currently* known about the system, as the very nature of XP asserts that it is both a waste of time and energy to worry about long-term changes and additions.

Related Patterns A weak metaphor may still have redeeming qualities, which may be found using the **Goodness** pattern. Equally, it may be misleading, so also try using the **Lies** pattern.

2.3 Lies

Motivation In contrast to poor metaphors, some metaphors describe too much. They may fit all the criteria for a "good" metaphor, but they will also imply that the system works or should work in a manner which is different from the simplest solution, which would be the XP way to address the problem.

Effectively, this is a problem that is opposite to the problem of metaphor weakness: the metaphors this pattern identifies are overly strong and misleading.

Problem How do you measure the misleading notions of a metaphor?

Forces

- You want to *avoid* picking a misleading metaphor for your project, but different metaphors are misleading in different ways.
- If it has already been identified how a metaphor is misleading, the inclination is to discard it, as many XP projects forge ahead without any system metaphor. On the other hand, these metaphors may still yield a useful insight into the system.

- A metaphor which is misleading prior to usage may or may not become even *more* misleading after it has been used for a while. Without actually experiencing its use however, it is hard to determine what will happen.

Solution Misleading metaphors can be identified by asking one of the following questions and obtaining a positive response:

- **do the metaphorical entailments imply non-existent system components or non-existent behaviour?**
  While it is almost impossible to avoid this to some extent, the metaphor should not imply *too much* that is incorrect. As every incorrect entailment is excluded from the metaphor, progressively the *intuitability* of the metaphor decreases, where intuitability describes the extent to which behaviour and components can be guessed or predicted. In turn this makes the metaphor less helpful.

  Running Example Considering the RETAILER WHO DEALS WITH WHOLESALERS metaphor, non-existent system components and/or behaviour include the following:

  - The notion of customer service is one that is integral to many retailers. However, the system does not try to be helpful to the user by inquiring about how his day has been, whether he is looking for something in particular or whether he wants some recommendations.
  - Retailers usually have hours of business, that the system does not have.
  - Retailers are sometimes unable to get hold of certain goods, and may suggest or provide the closest matching product, which is perhaps a similar item made by a different brand. The system does not do anything like this.
  - The metaphor centres around commerce and the exchange of money, whereas the stereo system has no concept of money and anyone (who is a graduate computer science student at Victoria University!) can use it free of charge.
  - Retailers add a mark-up charge to the items they provide. The central server does no such thing.
  - The slave machines store one copy of each track, but wholesalers stock *many* items of a specific type, and sell different ones each time, as opposed to "selling" the same copy each time.

- **do the metaphorical entailments make the system more complicated than it needs to be**
  Metaphors exist to *enhance* system understanding, not to obscure it. The heart of XP is a simple and flexible system architecture, therefore an overly complicated metaphor should be avoided. If the metaphor is *more* confusing than the naïve metaphor, where the system stands for itself, adopt the naïve metaphor.

Running Example Considering the RETAILER WHO DEALS WITH WHOLESALERS metaphor, system complications the metaphor potentially introduces are the following:

- Items need to be delivered to the retailer somehow, probably by a delivery company, who also get paid. This is probably unnecessary for the system.
- Currently the metaphor assumes that the retailer and wholesalers work together, linearly proceeding their way through he list of orders. In reality the retailer is much more likely to consist of a group of people, who would conduct various dealings in parallel. Furthermore, this group would probably be hierarchically structured, and the people the clients deal with are not necessarily the same ones that place the order with the wholesalers.

Despite having found various flaws with the RETAILER WHO DEALS WITH WHOLESALERS metaphor, we still think it fits the stereo system reasonably well. Rarely will it be possible to find a perfect metaphor, but it will be much more likely that a reasonably well-fitting metaphor will be found that may prove to be almost as useful, provided everyone is *aware* of its limitations.

This task should take approximately 20 minutes.

Resolution of Forces By identifying either the non-existent behaviour and components, or the complications added by any given metaphor, the team can be clear about what traps to avoid. Applying this pattern to multiple metaphors which seem misleading in various ways will allow the team to compare where each metaphor deviates from the current system understanding, and weigh up which metaphor will be easier to work with.

In the case where a metaphor is quite useful but also slightly misleading, the team may choose to record the bounding limitations on the metaphor, i.e. at what point the metaphor *ceases* to represent the system. In this way, they can make the most out of a metaphor they may have discarded in favour of having *no* metaphor, which should be avoided for the sake of the customer.

This pattern provides a way to evaluate how much the metaphor deviates from the knowledge *currently* known about the system. But similar to the **Goodness** and **Weakness** patterns, it cannot really be determined how misleading a metaphor will eventually become until *after* the project has been going for some time.

Related Patterns Each metaphor needs to be evaluated considering various characteristics. Imagining a continuum along which this pattern and the last two patterns lie, **Weakness** would feature on one end of the continuum, this pattern would sit on the other end and **Goodness** would feature in the middle.

Acknowledgements

I would like to thank everyone who helped with this paper in some way. In particular I want to thank the other co-authors James Noble, Robert Biddle, and Pippin Barr for contributing your ideas (and words!). I also want to thank my shepherd Paul Dyson, for your helpful constructive criticism, and our PC member Andy Longshaw for keeping an eye on us. Finally, I am grateful to those who took part in the experiment, namely Jennifer Ferreira, Donald Gordon, Stuart Marshall, and Robbie Morrison.

References

[1] Auer, K., and Miller, R., Eds. *Extreme Programming Applied*. Addison-Wesley, 2002, ch. 23: Overtime Is Not the Answer.

[2] Barr, P. A Semiotic Model of User-Interface Metaphor. In *Virtual, Distributed and Flexible Organisations: Studies in Organisational Semiotics - 3* (2003), The 6th International Workshop on Organisational Semiotics: Virtual, Distributed and Flexible Organisations.

[3] Barr, P. User-Interface Metaphors in Theory and Practice. Master's thesis, School of Mathematical and Computing Science, Victoria University of Wellington, 2003.

[4] Beck, K. *Extreme Programming Explained*. The XP Series. Addison-Wesley, 2000, ch. Glossary.

[5] Beck, K. The Metaphor Metaphor. Invited speaker at OOPSLA, 2002.

[6] Beck, K., Cockburn, A., and Bossavit, L. System Metaphor. `http://c2.com/cgi/wiki?SystemMetaphor`, 2003. (Various other anonymous authors.).

[7] Brown, L., Ed. *The New Shorter Oxford English Dictionary*, vol. I (A-M). Oxford University Press, 1993.

[8] Cunningham, W. System Of Names. `http://c2.com/cgi/wiki?SystemOfNames`, 2003.

[9] Fowler, M. *Extreme Programming Explained*. The XP Series. Addison-Wesley, 2001, ch. 1: Is Design Dead?

[10] Garzaniti, R., Haungs, J., and Hendrickson, C. Everything I Need to Know I Learned from the Chrysler Payroll Project. In *SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (Addendum)* (1997), ACM Press, pp. 33–38.

[11] Lakoff, G., and Johnson, M. *Metaphors We Live By*. The University of Chicago Press, 1980.

[12] Tomayko, J., and Herbsleb, J. How Useful Is the Metaphor Component of Agile Methods? A Preliminary Study. Tech. rep., Carnegie Mellon University, 2003.

[13] Wake, W. C. *Extreme Programming Explored*. The XP Series. Addison-Wesley, 2002, ch. 6: What is the System Metaphor?