Coherence and responsiveness

Jared Harris — Scalable Conversations — jed@jive.com
Austin Henderson — Scalable Conversations — henderson@rivcons.com

This article presents a model of the trade-offs between responsiveness and coherence often found in designing and managing systems. The model also describes how both responsiveness and coherence often decline as a system grows. The authors argue that designers need not accept a zero-sum or least-bad choice but rather should seek platform improvements and collaboration tools that increase both responsiveness and coherence.
— Hugh Dubberly, Editor

The world we live in is full of systems: phone systems, legal systems, air traffic control systems, educational systems, banking systems, digital communication systems (such as the Internet), computer operating systems, purchasing systems, HR systems, healthcare systems. Systems are designed and evolved; they are built, maintained, modified, and replaced. Systems are made up of people and things, rules and practices, options and constraints.

Systems pattern activity in their domain. They help individual users get their work done more easily. Even better, systems can help users in their interactions with one another.

Each system we create embodies a tension: The world is diverse and dynamic; different users at different times have different needs and expectations. At the same time, users affect each other, so a system must provide coherence.

As designers and users, we would like each *part* of a system to be *responsive* to local circumstances and also the system *as a whole* to be *coherent*.

Choosing a Balance Point

Systems are therefore always engaged in an interplay of *responsiveness* and *coherence*: The more a system's parts are responsive to the diversity and dynamism of the world, giving people the ability to meet their needs, the less we can know about how the whole system will behave. The more a system drives toward coherence, the stronger the relationships between its parts, and the less freedom each part has to adapt to its circumstances in unexpected ways.[1]

System designers often see themselves as confronted with a zero-sum choice: Increase responsiveness to local needs or opportunities and you must reduce the system's ability to conform to preferred patterns, and vice versa (see Figure 1). This is even stated as a binary choice—order or anarchy—though no real system can ever achieve those extremes.

So system designers often find themselves trying to pick the "least worst" point on this spectrum, regretfully sacrificing some responsiveness or some coherence.

Figure 1 Tension between responsiveness and coherence (1 dimension).

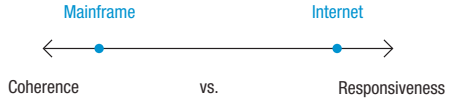


Table 1
Similar dichotomies.

Coherence	Responsiveness
Top down (designed from principles)	Bottom up (designed from activity)
Walled garden (authorization required)	Open source development (anyone can play)
Cathedral (designed by a small group)	Bazaar (designed by a crowd)[5]
Tested distributions (quality controlled)	Open source repository (quality depends on recent events)
Consistent replication (assembly-line produced)	Customized one-offs (locally grown)

Figure 2 Tension between responsiveness and coherence (2 dimensions).

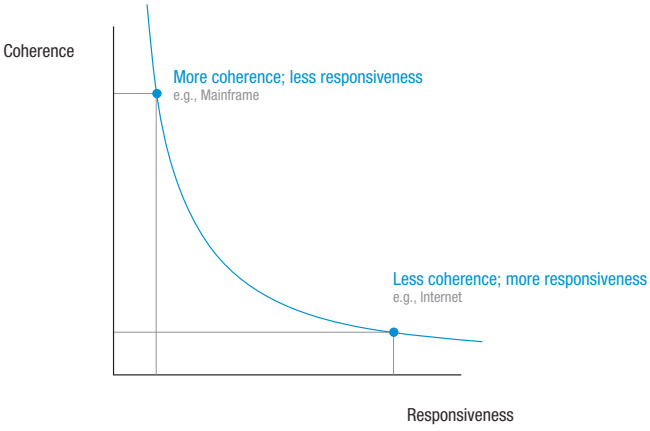
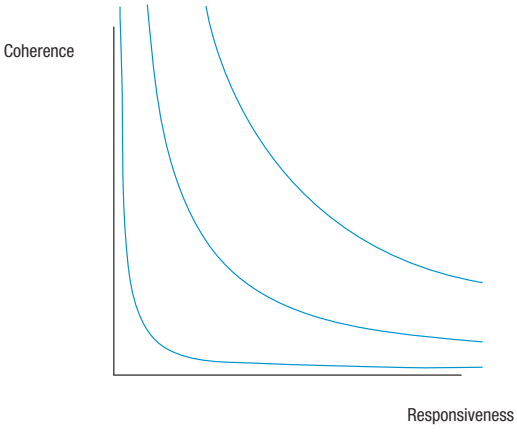


Figure 3 A family of trade-off curves.



However, there are many similar dichotomies, embodying the same tension, that seem to require the designer or policy maker to choose a balance. Some of these are listed in Table 1.

We find coherence and responsiveness to be the most general of these dichotomies, but all point in a similar direction and all raise similar issues.

Breaking out of Zero Sum

There is another way to see the tension in these pairs. Instead of trying to pick a "right place" on the single dimension between the two poles, we can see responsiveness and coherence as separate, more independent dimensions of our systems.

Moving to two dimensions lets us explore the relationship between the two poles. Viewed in this way, we can easily see the relationship is not simply zero sum.

When we shift to two dimensions, our one-dimensional, zero-sum line becomes a *trade-off curve*. And as long as we stay on this curve, the dimensions are still in an antagonistic relationship: If we do better on one dimension, we do worse on the other (see Figure 2).

But now we can see it is possible to leave this trade-off curve. On the negative side, just because we do worse on one dimension, we won't automatically do better on the other; that is, we can do *worse* than zero

sum. For example, a bad system can be both unreliable and stubbornly ignorant of actual usage patterns. More generally, systems can be *both* incoherent and unresponsive; both top-down and bottom-up processes can function poorly; and so on.

More optimistically, we can move off a trade-off curve on the positive side: We can do *better* than zero sum. By improving our institutions, technology, and practices, we may move to a higher trade-off curve—further from the origin, better on *both* dimensions. (We discuss some examples below.)

Thinking About Trade-off Curves

Trade-off curves are often used in engineering complex systems, such as factories. They are also common in economic theory. These disciplines work with maps of a family of trade-off curves (see Figure 3).

The ways we move from one trade-off curve to another vary from field to field.

- In manufacturing, we can move to higher trade-off curves by redesigning processes to be more efficient, finding better materials, designing products to be easier to manufacture, and so on. Conversely, if a factory is poorly maintained, processes are hacked up in inefficient ways, and so on, it can slide down to lower trade-off curves.

- In economics, we can move to higher trade-off curves by increasing productivity, reducing wasteful activities, and eliminating "friction" and the like (often summed up as "technology"). Conversely, corruption, monopoly, cronyism, and so on can move us to lower trade-off curves.

Factories, economies, and many other systems described with these trade-off families are socio-technical systems. But the disciplines involved typically abstract away the human interaction to focus on simpler formal properties of the systems.

In contrast, our interest is in systems where human interaction is central. Thus our interest is in trade-off families where we move between curves by changing the ways people interact—with machines and with each other.

The biggest effects come from changes in how multiple parts of the system interact. For example, as the patchwork of walled email gardens, bulletin boards, and specialized information services was absorbed by the Internet and then the Web, communication via computers changed dramatically. When you had to navigate multiple email conventions, addressing mechanisms, and so forth, and didn't know for sure whether you could reach someone, it often wasn't worth the trouble. Once this jumble was unified, the pressure on everyone to get on email was intense, and the system as a whole moved to a dramatically higher trade-off curve.

One of the big pressures that tends to move systems toward lower trade-off curves is *increasing scale* (see Figures 4 and 5). As systems grow bigger, the tension between responsiveness and coherence tends to become more severe. The more people a system needs to accommodate and manage, the harder it gets to maintain coherence, and the less responsive the system tends to be to individual needs, outliers, and misfits.

Another big pressure is *sunk costs*. Systems require investments—in hardware, software, training, and so on. Once this investment has been made, it often becomes an anchor that inhibits changes. It is easy to see how this tends to reduce responsiveness; interestingly, it also tends to reduce coherence. As the world diversifies and changes, the system must be patched and extended to deal with unanticipated circumstances, and in consequence drifts further and further from a comprehensible or maintainable design.

Figure 4 Moving to a less desirable trade-off curve.

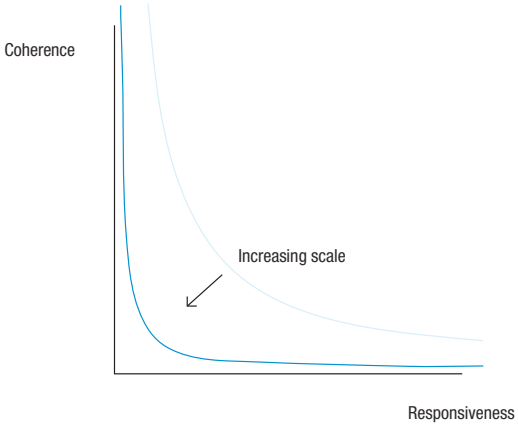
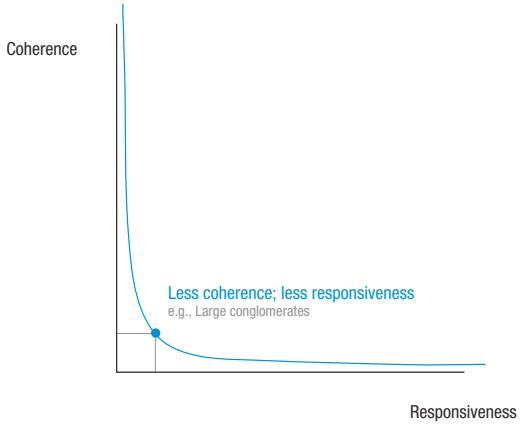


Figure 5 E.g., as organizations grow, getting things done may become more difficult and take longer.



Moving Systems to Higher Trade-off Curves

Recent history shows us ways to dramatically improve the trade-off between coherence and responsiveness (see Figures 6 and 7). Improvements are of two types: local and non-local adjustments.

When local needs don't fit a system's design, the drive for coherence can make local work inefficient and unpleasant. In response, people do whatever they can to adapt the system to their needs: They augment the system with personal files, attach memos to reshape the meaning of forms, and use "illegal values" in fields to extend the system to meet unanticipated circumstances. The resulting extended socio-technical system may be less coherent, but it is often on a higher trade-off curve.

Such "local fixes" are required in the deployment of real systems, and are *essential* for enabling systems to respond to diverse and changing worlds. Users usually keep them below the radar, because they are often thought to offend system developers, who may see them as criticism for not getting the system right.

In contrast, when system developers accept the inevitable need for such "fixes," a further move can be made: The technical systems can be designed to support this work of extension, providing more responsiveness with less loss of coherence, less burden on users, and often very little technical effort.

For example, margins on forms support going beyond the frame of the form; supporting "illegal values" empowers users to better express real business conditions; and "multi-lane" systems support out-of-band (human) handling of special cases, letting *routine* usage remain unencumbered by *exceptions*. Such extensions can support an individual making notes to themselves, groups maintaining alignment, or even large-scale drifts in the usage and value of a system.

As an additional advantage, when kept within the technology itself, these records of extensions can inform ongoing development.[2,3]

These local fixes maintain or raise the level of the trade-off curve. Now consider some examples of system-wide changes in interaction, which can have—as we said earlier—a profound impact on moving to higher trade-off curves. Their diversity suggests that many more will be forthcoming.

Design languages Tools for supporting domain-specific design are often stuck trying to find the right balance between generality and particularity in providing ways to describe the domain. Designers of the Trillium design environment for photocopier user interfaces saw this tension as due to forcing all design into a single language (complete coherence).[4] Instead, the language of description itself was recognized as part of the ongoing design activity, and therefore was made a part of Trillium's subject matter. New concepts were created for new product families, new products, and new designs (very responsive). For balance, coherence was maintained socially, through the ability to easily share and extend design concepts, daily use of email, and twice-yearly meetings.

Allowing language to evolve in use is a powerful means of managing the trade-off curve for a growing space of products.

Web search Early navigation of the Internet was supported by hand-built "maps," such as the old Yahoo catalog. But the rapid growth and change of the Internet quickly made comprehensive maps impossible, while early search utilities were not good enough to replace human mapping.

Larry Page and Sergei Brin solved this problem with the PageRank algorithm, which aggregates the local knowledge implicit in the network of references between pages. Since then Google and others have

Figure 6 Moving to a more desirable trade-off curve.

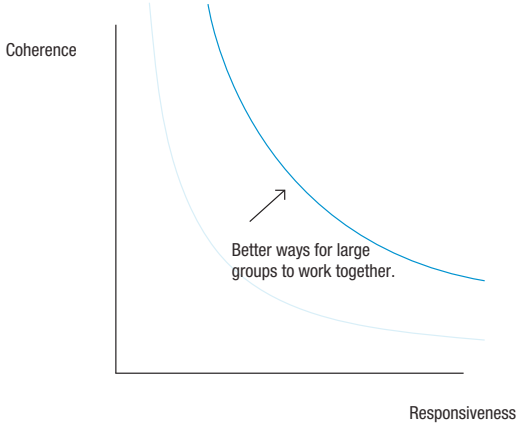
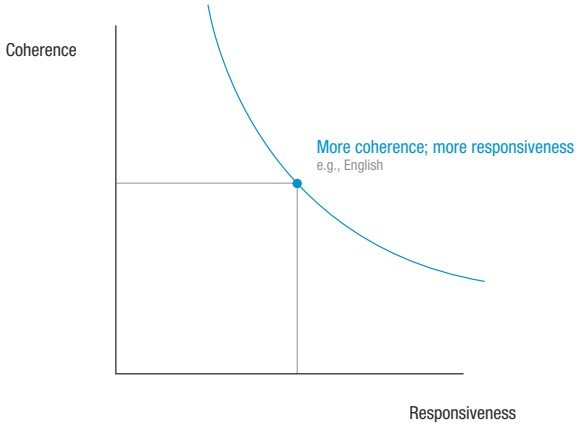


Figure 7 E.g., Google's PageRank search algorithm sits on a higher trade-off curve than early Internet navigation systems such as Yahoo's original directory.



evolved increasingly sophisticated ways to aggregate local knowledge.

PageRank can adapt to almost unlimited changes in the content and uses of the Web without needing to change the core algorithm at all. So although the algorithm is rigid, it can power a service that is both coherent and responsive, because it is "permeable" to human concerns.

Review aggregators Consider Yelp, an online service that aggregates, curates, and helps users search reviews of local businesses (e.g., restaurants, dry cleaners, mechanics).

Prior to such systems, knowledge about local businesses could be obtained in three major ways:

- Patchy but useful local interaction, which doesn't work well in a new area or when you don't have friends with the specific knowledge you need (lots of local responsiveness, but not much coherence).

- Recognizable brands, such as chain restaurants or department stores—often mediocre, but reliable (quite coherent, but not responsive).

- A guidebook, perhaps even one you picked because it matched your own preferences. Results are dependent on the taste and knowledge of the writer, and only as timely as their last edition, at best (in the middle: less coherent than large reliable brands, and less responsive to local conditions than well-informed residents).

Review aggregators like Yelp move access to local knowledge to a much higher trade-off curve. While Yelp is focused on local businesses, similar services aggregate other knowledge (e.g., books, appliances, electronics, games), giving us a much higher level of both coherence *and* responsiveness.

Again, these systems succeed because they are "permeable" to user preferences and judgments, while still filtering and organizing them. Review aggregators need more active curation than Internet search engines. However, they have delegated some of that curation to users, by aggregating feedback on reviews as well.

Online collaboration systems The Internet has catalyzed the emergence of very large open working groups, such as the Linux development community, the Wikipedia authoring community, and various fan and support groups. There have been open working groups in the past—in some sense any academic discipline is such a collaboration—but these new groups are larger, more open, and work more quickly.

These groups are possible due only to collaboration mechanisms such as mailing lists and their archives, wikis, ticketing systems, and version management. These help maintain both the coherence and the responsiveness of groups that are too large or dispersed for older modes of coordination.

Consider version management. It is interesting because it has evolved considerably in the recent past, is still evolving fairly quickly, and its role in moving to higher trade-off curves has been explicitly discussed by its users.

For example, Wikipedia invites any visitor to edit most Wikipedia pages. This is only sustainable because the version management built into the software platform (Mediawiki) allows rapid reversion (undo) of inappropriate edits, helping a huge collaborative editing project sustain both high responsiveness and high coherence.

A more complex example is source management for Linux. It evolved incrementally over 21 years, driven by increasingly rapid change in the code base and changing structure of the collaboration itself. The history of this evolution is interesting but too complex to recount here. To brutally oversimplify, the project adopted a sequence of tools for version management; it eventually wrote its own tool, which has since become very popular for other projects as well. Without major innovation in version management, Linux would have been crushed by the need to increase responsiveness to bugs, new features, and many versions for different circumstances, while maintaining coherence.

In both examples, quite simple infrastructure can manage very complex coordination. This infrastructure is highly permeable and permits a very high level of control by users, while still giving the collaboration as a whole the ability to maintain coherence.

Conclusion

As these examples show, even very large systems can be both coherent and responsive. Furthermore, in many cases they can achieve both apparently conflicting goals using a relatively simple and slowly evolving service platform.

One theme is that the technical systems are permeable to human meaning, values, and choices—they encourage communication between their users, coordinated by the infrastructure.

A second theme is that each service is focused on relatively simple ways of handling a relatively small set of functions. Building any feature into a service requires assumptions about how people will use it; the fewer such assumptions, the wider the range of users and uses that can be accommodated gracefully.

We believe there are many ways to simultaneously increase the coherence, responsiveness, and scalability of systems, and this quest has enormous potential to improve our lives.

Endnotes

1
Harris, J. and Henderson, A. A better mythology for system design. *Proc. of CHI '99*. ACM, New York, 1999, 88-95. DOI=10.1145/302979.303003; http://doi.acm.org/10.1145/302979.303003

2
Yates, J. *Control through Communication: The Rise of System in American Management*. The Johns Hopkins University Press, 1993.

3
Henderson, A. The 100% solution: What is a user to do, and how are we helping? *Proc. of the 15th European Conference on Cognitive Ergonomics: the Ergonomics of Cool Interaction*. J. Abascal, I. Fajardo, and I. Oakley, eds. ACM, New York, 2008. doi>10.1145/1473018.1473021

4
Henderson, Jr., D.A. The Trillium user interface design environment. *Proc. of CHI '86*. M. Mantei and P. Orbeton, eds. ACM, New York, 1986, 221-227. DOI=10.1145/22627.22375; http://doi.acm.org/10.1145/22627.22375

5
Raymond, E. *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media, 2001.

About the authors

Jed Harris started out exploring cultural anthropology, linguistics, philosophy of science and artificial intelligence research, and then spent forty years in research and development at SRI, Stanford, Xerox PARC, Data General, Intel and Apple. He's now happily meshing technology with the human sciences.

Austin Henderson's 45-year career in HCI includes research, design, architecture, product development, and consulting at Lincoln Laboratory, BBN, Xerox (PARC and EuroPARC), Fitch, Apple, and Pitney-Bowes. He focuses on technology in conversations in a rich and changing world.

About this column

Models help bridge the gap between observing and making—especially when systems are involved (as in designing for interaction, service, and evolution). This forum introduces new models, links them to existing models, and describes their histories and why they matter.
— Hugh Dubberly, Editor