Jini Community Pattern Language

Richard P. Gabriel & Ron Goldman

This document presents a pattern language to be used to create the initial Jini Community and to evolve it over time. The concept of a pattern language was created in the late 1960's by the architect Christopher Alexander. His pattern language for designing cities, towns, buildings, houses, and parks was intended to be a set of patterns that worked together to evolve over time these known things in the built world in such a way that they had an extra component which can be called *life* or *wholeness*—something he called "The Quality Without a Name."

Context

The dominant architecture for connecting computers and electronic devices is currently to cluster dumb devices around a computer and then network the computers together. Advances in microelectronics now make it possible for devices to contain significant computing power. Recognizing this, Sun Microsystems developed Jini technology to facilitate communication between a world of intelligent devices. In this new model devices offer their services to any applications or other devices that wish to make use of them. This new architecture of digital services is highly distributed as opposed to the centralized nature of the current computer-centric model.

To fully realize the potential market for Jini technology requires Sun to work with a large number of partners. As a first step, Sun has made available the Jini source code under a modified Open Source license and many thousands of individuals, universities and companies have downloaded the code. However the Open Source model of software development is not sufficient; a community around Jini technology needs to be developed to create the broad range of Jini services that are needed for a large, thriving marketplace to emerge.

What form this community should take is very unclear. This is a new way of doing business and there are no existing organizations that can be used as a model. The decentralized, non-hierarchical nature of Jini technology will need to be reflected in whatever organization the Jini community chooses.

Our strategy is to work with members of the Jini community to develop a set of community principles that will allow the community to self-organize and develop Jini services. These principles will provide a framework within which to design the community and its organization. Examples would be:

- Community activities should be *open*. Any community member should be able to follow electronic discussions and attend meetings.
- Community activities should be *fair* for all community members. We do not want an organization where all decisions are made by a few large companies, nor do we want one where a few small members can block all action.
- We need to *encourage diversity* so Jini services range from experimental, innovative ones to those that are more mature and whose specifications are considered as standards.

We have chosen to express these principles in the form of a pattern language. Pattern languages have proven to be an excellent form in which to talk about how a set of principles and forces in a constructed artifact—a house, a city, a software system, or a development organization—combine to create an effective and livable instance of that artifact. In a pattern, you can talk about how a set of existing forces can be resolved according to a rule (or pattern) that becomes a principle in the organization.

In the patterns that follow, you will see a set of principles explained in pattern format. Some of the patterns will talk about specific organizational entities. This does not mean that such entities are required to exist in the organization, but the pattern explains what forces would give rise to the need for such an entity. Thus, the pattern language is a sort of handbook for those creating the Jini Community to refer to while building and evolving the community, just as Christopher Alexander's book, **A Pattern Language**, is used every day by people working with architects and builders to make their homes or remodel them.

Use of Piecemeal Growth

The process we are using to create the Jini Community relies on the concept of *piecemeal growth*. Rather than try to design the overall structure to be used, we are working with smaller segments of the community as they emerge around common interests. For example, a group of companies that manufactures printers has coalesced and we are working with them to define the structures they need to allow them to define Jini services.

As other groups form they will need to determine what organization is most appropriate for them. We imagine that different groups will have different needs. Using the patterns below they will generate the organizational structures best suited for the activities they need to perform. Other patterns will help generate how they interact with the rest of the Jini Community.

Finally we expect that both the Jini Community and the pattern language will evolve over time.

Summary of the Language

The patterns are presented so those with related principles are grouped together. The current patterns fall into five rough areas: organizational process, the marketplace, fairness, product development, and quality and stability.

The first group focuses on some very general principles about the process of organizing the Jini Community. The patterns in this group are:

- Gathering Requirements
- Evolve the Community
- Organization Follows Activities
- Adaptable Organization
- On-The-Fly Adaptation
- Organization Follows Geography
- Decision-Makers Near the Action
- Decision-Makers with Broad Oversight
- Go Whole Hog

The next group of patterns focus on how the organization will need to create a new marketplace and what the value added might be for companies.

- Dangerous Waterhole
- Safe Place to Share
- Grow the Market
- Encourage Diversity in Emerging Marketplaces
- Let the Market Decide
- Short Time to Market
- Coalitions Form Markets
- Strengthen the Brand
- Make the Most of the Least
- Drop Barriers
- The Heft of Quality
- Give Away the Least Expected
- Anchor Stores

How to achieve fairness in the functioning of the Jini Community is the focus of the next set of patterns.

- Fair Processes
- Checks and Balances
- Protect the Weak
- Right to Appeal
- No Distinguished Members
- Appeals to Supermajority
- Information Flows Everywhere
- Ratification/Community Approval
- Proportional Votes
- Senatorial Votes

Next are several patterns concerned with various ways community members will develop products.

- Cut and Run
- Microcosm
- Make a Standard
- Single-Source Specification Creation
- Multiple-Source Specification Creation
- Compelling Idea

Finally there are the patterns concerned with building quality and stability into Jini services. These include a number of patterns that generate Open Source and Community Source development.

- Levels of Maturity
- Reward Stability
- Peers Ensure Quality
- Peer Review
- Community Ensures Completeness
- Shepherds Ensure Architecture

- Shepherd Panel
- Grow your Shepherds
- Shepherd Code of Ethics
- Community Reviews Shepherds
- Community Controls Development
- Compatibility Logo
- Give Away All You Can
- Every Bug is Trivial to Someone
- Survival of the Fittest
- Scratch Your Own Itch
- Work in Your Own Medium
- Continuous Releases

The concept of a pattern language is hard to grasp in the abstract, but any example is quite easy to understand. We recommend that you just start reading these patterns, and you'll see how they relate to each other, what they mean in isolation, and how they relate to the task of designing the Jini Community.

A Sequence to Create a Service

When a company decides that it wishes to create a Jini service, that company first will see that it needs to approach or create a **Dangerous Waterhole** which will be occupied by its competitors. On closer inspection the company might determine that there are reasons to cooperate and that it is a **Safe Place to Share**. One of the most compelling reasons to cooperate is that creating a service alone might mean duplicating work that the community has already done and is continuing to do. Moreover, a service defined in a vacuum might find no real uses. By joining with the community's efforts, the company can concentrate on whatever value it can add, and it can **Make the Most of the Least**.

Next is the question of how the new service will be used. Is it a **Microcosm** only to be used internally by its creators or is it to be made available to the larger community? If the latter, then will it be developed independently in a **Cut and Run** manner or in concert with other community members to **Make a Standard**?

The development of a Jini service involves the creation of specifications for interfaces and their correct implementation. To improve the quality of the resulting service, the company needs to decide whether to make use of peer review so that **Peers Ensure Quality** or to involve a community shepherd so that **Shepherds Ensure Architecture** (or both). In either case, the final service must pass the community specified compatibility test if it is to receive the **Compatibility Logo**.

Finally, the service creator should **Give Away All You Can** by contributing as much of source code as possible back to the community in order to make the best use of community resources to further its development. When deciding this they should **Go Whole Hog**.

A Note on Form

The patterns in this sequence are in one of two forms. The skeletal form was used to quickly write down the pattern so we could look at it in the context of the pattern language. Later, as we grew comfortable with the pattern, we expanded it into Alexandrian form.

Patterns are a literary form consisting of several distinct sections. First the pattern has a short name to remind you of its essence. Then there is a description of the *context* that the pattern occurs in, possibly referring to a larger pattern that this pattern helps to complete. This is followed by a statement of the *problem* that the pattern is trying to solve. Next is a discussion of the various *forces* that are present and must be considered. Then comes instructions to help you generate a *solution* to the problem. Finally there may be some concluding *remarks* relating the solution to other patterns.

This is very much a work in progress. In the patterns that follow you will note that many are mere skeletons, written in outline form. These need to be fleshed out. You will also notice that the pattern language is nowhere near complete. When you notice a gap or have an idea for a pattern that should be included, please let us know. The more community participation in the creation of the pattern language, the better a community we can create.

Gathering Requirements

Context: The beginning of a community dedicated to developing a new marketplace.

Problem: What is the first step in the process of defining the community organization?

Force: There are large, conservative members of the community.

Force: There are small, fast-moving members of the community.

Force: Some members are interested in sustaining technology.

Force: Some members are interested in disruptive technology.

Force: There is a desire for a common good to be part of the community charter.

Therefore: Take time to gather requirements from the community members about what the right organization would be.

Evolve the Community

Context: Any community and attendant organization

Problem: How can the community adapt to change?

Force: The community has an initial organization.

Force: The needs of the community change over time, perhaps because its underlying marketplace is changing.

Force: The community wishes to keep up with change.

Therefore: Plan to diagnose and repair the organization as conditions change. Put in place a process for such change. Exercise that process when needed.

Organization Follows Activities

. . . There are many ways to structure an organization.

* * *

In any organization there are a set of activities its members are engaged in. In most organizations, members are permitted to act only within components of the organization. In most organizations, it is strictly specified what each component may do and which components are permitted to interact.

To get work done, the right people need to be able to work together directly when needed. Sometimes such interaction is frequent, and sometimes it's infrequent.

Therefore,

In designing an organization, discover the activities the organization will engage in. For each such activity, design a component or type of component for that activity or type of activity. Make sure that membership in that component is determined by who needs to do the work of that activity. If two activities are related, arrange for the two corresponding organizational components to be able to interact with each other.

* * *

It is important to lay out these hypothetical structures beforehand in order to minimize organizational disruption when a new substructure is put into place and to obviate design on the fly.

Adaptable Organization

. . . There are many ways to structure an organization. When the organization is always in flux, it is particularly important to anticipate structural evolution.

* * *

In an organization whose primary activity is to achieve a set of ever-changing goals, a static organizational structure is sometimes too brittle or too unadaptable. On the other hand, constantly inventing new organizational substructures and fitting them within an existing structure can drain energy from the organization that should be directed toward achieving goals.

In many such fluid organizations, the range of types of activities is usually not very large and can be predicted, though there may be an unexpected type of activity every now and then. If such activities are force-fitted into an existing static structure, the disruption could cause more problems than are solved by making the fit. A totally ad hoc organization might have too little structure to maintain organizational cohesion.

Therefore,

While designing the organization, lay out the types of activities that may come and go. Design the right sort of organizational entity for each type of activity. For each such organizational entity, determine how it would fit into an existing organization with other such organizational entities. As each new activity comes up, create the organizational entity according to the set of anticipated activities and associated organizational entities.

* * *

It is important to lay out beforehand as many as possible of these hypothetical structures in order to minimize organizational disruption when a new substructure is put into place and to obviate design on the fly. When an entirely new type of activity arises, a more lengthy process of organizational design must take place, but it is important that there be a process for doing the design and adding it to the set of pre-existing organizational types. See **On-The-Fly Adaptation**.

On-The-Fly Adaptation

. . . a fluid organization is one that can adapt to change quickly. In any organization that tries to anticipate the potential variations, there will be the case where the unexpected turns up.

* * *

In an organization that has prepared a process for adding new organizational substructures by using a set of pre-defined types of substructures based perhaps on activities, there will always be the potential for an unforeseen type. It may not be feasible to ignore this new type of activity, but without a defined way to proceed, the organization could be deadlocked.

Long-lived and apparently static organizations like the United States Federal Government have mechanisms for changing themselves when needed, though often the process is difficult and not entirely suited to that task. It is not a good idea to change an organization very often.

On the other hand, being able to change the organization and its rules is one of the best ways for a tyrannical majority to oppress a weaker minority.

Therefore,

Arrange to have a pattern language available for designing new parts of an organization or alterations to existing parts. Put in place a process for implementing such additions or alterations. Make the process the most difficult to conclude within the organization.

* * *

See **Protect the Weak**, **Senatorial Votes**, **Checks and Balances**.

Organization Follows Geography

Context: Any scattered organization.

Problem: How can the interests of a region be ensured?

Force: The organization is scattered.

Force: Difference forces apply to different regions.

Force: There is an advantage to having strength in numbers within the organization.

Therefore: Arrange for regional suborganizations to be possible and tie the strength of the suborganization within the larger organization to the combined strengths of the regional members.

Decision-Makers Near the Action

Context: Any nontrivial organization.

Problem: Who makes decisions?

Force: Almost anyone in the organization may feel they have a right to assist making a decision.

Force: There are some individuals who will ultimately have to live with the consequences of any decision.

Therefore: Arrange it so that those who must live directly with any given decision are the primary decision-makers for that decision.

Decision-Makers with Broad Oversight

Context: Any nontrivial organization.

Problem: Who makes decisions?

Force: Almost anyone in the organization may feel they have a right to assist making a decision.

Force: There are some individuals who will ultimately have to live with the consequences of any decision.

Force: The decision affects the organization as a whole.

Therefore: Arrange it so that those who have broad oversight and responsibility for the organization are involved in making the decision.

Go Whole Hog*

Context: A world where organizations judge people.

Problem: People often know what is needed, but if it is a radical departure from normal practice they are reluctant to do it all and instead try to water it down to a small step in the right direction.

Force: The unknown is scary and hard to sell.

Force: People avoid change whenever possible.

Force: Being conventional is rarely punished.

Force: People want to cover their asses.

Force: You know what the right thing to do is.

Force: Half measures rarely work.

Therefore: Do the right thing to the maximum extent possible. Encourage others to do the same by supporting innovation and not penalizing failure.

Remarks: The world is full of trade-offs, but when a course of action is clearly superior we should take it even (and especially) if it makes us uncomfortable because it seems too radical.

Known Uses: Noyce and Sanders, founders of Fairchild Conductor in the 1960's, selling transistors at $1.05 each (the price of the competitive tube equivalent) when it cost $100 to make them.

Dangerous Waterhole*

Context: A community is trying to create a new marketplace.

Problem: Competitors don't drink simultaneously for fear of attack.

Force: Competitors don't trust each other.

Force: There are no compelling reasons for the unformed marketplace to be able to be created nor for the result to have any value.

Force: No company can succeed alone: The marketplace—the range of possible markets—is too large.

Force: The community needs competitors both to demonstrate the existence and value of the potential marketplace and to fan out in a competition net to find the sweet spots. That is, you need natural selection to work for you.

Therefore: Make it so the waterhole is not too dangerous; make it safe for companies to cooperate; make the water cool and sweet. Recognize that competitors have secrets and want to protect them, but are willing to share secrets for a larger return. Know that competitors are predators and protect the marketplace from sabotage.

Remarks: Patterns that will make the waterhole less dangerous include: **Safe Place to Share**, **Give Away the Least Expected**, **Strengthen the Brand**, **Compatibility Logo**, **Fair Processes**, **Information Flows Everywhere**, and **Shepherd Code of Ethics**.

Known Uses: Open Source projects, industry consortia, NATO and UN peacekeeping groups, the courtroom, summer camp, the schoolroom, writers workshops, Visa International, and singles bars.

Safe Place to Share

Context: Any software development situation at a **Dangerous Waterhole** or a company or individual trying to **Make the Most of the Least**.

Problem: How can a community of competitor companies share those parts of the software that is crucial for the community's success while keeping private those parts on which competitiveness depends?

Force: There is a core of the software whose proper and effective operation is of value to all members of a potential community.

Force: Control of that core is not essential to any particular organization.

Force: Organizations do not want their competitors in control of that core.

Force: Intellectual property rights and/or processes have been already established or negotiated.

Force: Software is challenging—it requires careful, coordinated thought and planning.

Force: Software takes a lot of work—every development staff has much more work it wants to do than it can.

Force: Organizations do not want to share anything with their competition.

Therefore: Create a place or mechanism within which the software and its development can be shared, with the community in charge of future development. Each software-contributing organization needs to **Go Whole Hog** on giving up control.

Remarks: Patterns that will make it safe to share include: **Make the Most of the Least**, **Compatibility Logo**, **The Heft of Quality**, **Fair Processes**, and **Information Flows Everywhere**. This pattern directly resolves the "software takes a lot of work" force; more difficult is the "software is challenging" force, which can be resolved, perhaps, by this and other patterns that create a wealth of available programming gazorch that does not require as much planning and coordination which is solely required to minimize wasted effort and mistakes. In some sense, a development organization can make progress on a set of tough problems either by thinking and planning or by trying things out—we could say "by typing **`delete`** rapidly." In an Open–Source-like situation, there are more people to think and to type **`delete`**.

Known Uses: Open Source projects, Visa International, coffee clatches, the board room.

Grow the Market

Context: Any emerging marketplace and a community created to promote it.

Problem: What can the community do that is beneficial to all?

Force: Members compete with each other sometimes.

Force: Members cooperate with each other sometimes.

Therefore: The community should focus on how to grow the market.

Encourage Diversity in Emerging Marketplaces

. . . an emerging marketplace is difficult to predict. Attempts to control the development of such a marketplace are likely to backfire by killing the market or severely limiting its growth.

* * *

An emerging marketplace forms through the interaction between companies trying to address what they perceive as the wants and needs of their customers and customers trying to figure out their needs and desires. It is natural to wish to control the emergence of such a marketplace—for a competitor company to ensure market share or for a technology company to reinforce its leadership position—but such control can have the opposite effect of what's desired.

Consider a technology company that has developed new technology to spawn a new marketplace. As the acknowledged expert in the new technology, the company may naturally assume it understands how the technology will be used in the new marketplace. Moreover, the company might believe that the marketplace will be best off if it uses that technology in a particular way. But, customers like to think for themselves, and how they use the technology might be surprising and even frightening to the technology leader.

But, even the technology leader whose vision is wrong reaps larger benefits if the marketplace grows as large as it can as rapidly as it can than if the market stagnates.

Therefore,

In the early stages of an unformed marketplace, create a governing organization that encourages as much diversity as possible in the range of products or services available for sale, and in the ways that any underlying technology is used. Be careful as you alter the organization to be aware of whether you are trying to encourage diversity or standardization.

Let the Market Decide

Context: Any emerging marketplace and an entity related to a product in that marketplace.

Problem: How can the strength of the entity be measured?

Force: Political force is not permitted to be part of the equation or makes no sense in the context.

Force: There is a need to gauge the strength of the entity.

Therefore: Gauge the strength of the entity by its relative success in the marketplace.

Short Time to Market

Context: An emerging marketplace and a community dedicated to growing it.

Problem: How should the community approach its own organization with respect to the emerging marketplace?

Force: An new marketplace is evolving rapidly.

Force: Some companies are taking it slowly.

Force: Some other companies are moving very fast.

Force: There are no established players in the new marketplace.

Therefore: Arrange the community organization to enable companies to get to market quickly.

Coalitions Form Markets

Context: Any thriving marketplace.

Problem: How are markets formed?

Force: No company has a dominant position.

Force: Products in the marketplace have greater value if they work together.

Therefore: A coalition of companies can create a marketplace by making their products work together.

Strengthen the Brand

Context: A marketplace that is being promoted by a community. Products in the marketplace are marked by a brand or logo.

Problem: How can the community increase the value and size of the marketplace?

Force: Some members of the community are competitors to other members in the community.

Force: Some members of the community cooperate with other members in the community.

Force: Members of the community share the brand and perhaps some related technology.

Therefore: The community should cooperate to strengthen the brand.

Make the Most of the Least*

Context: A business in which effort or expense is required to earn money (for example, building automobiles rather than loaning money).

Problem: Fools and nice guys easily fail at business even when they have every good intention to please the customer.

Force: Many people believe, "if a little is good, more is better."

Force: Many people prefer to reinvent/reimplement things rather than use something created by an outside organization.

Force: Work is the wrong way to make money. "No one ever made money by typing."

Force: Maybe Tom Sawyer was right.

Therefore: Choose carefully the value you bring to the table, and figure out how to produce what remains for nothing or close to it.

Remarks: Safe Place to Share might be a way to complete this pattern.

Known Uses: Tom Sawyer, Microsoft, Linux.

Drop Barriers*

Context: Any emerging marketplace.

Problem: Some business situations seem impossible. One is that you need to have a lot of something—call it the *enabler*—in customers' hands already so you can sell your product. This happens when you are trying to use the network effect. For example, you want to sell razor blades, so you need people to already have the razors.

Force: The enabler is probably only part of a solution, so it's not worth much by itself.

Force: But it's worth something.

Force: The enabler may be useful for some other purpose that is worth nothing to you.

Therefore: Give it away. Make it impossible to do anything but give it away.

Remarks: This seems like a special case of a more general pattern that could be called Tom Sawyer. In such a pattern, you find what would make your target desire what is otherwise undesired and provide that along with the foul thing. This makes what you're spreading act like a virus.

The Heft of Quality*

Context: Sometimes in a free-source situation you have a developer or group of developers that needs to be paid. For example, the product is not done or is in an early release.

Problem: How can you charge more for something than a customer will pay?

Force: There is nothing you can add to the product to add value.

Force: You need value to keep making the product.

Force: Your product is only part of the solution to your customer's real problem.

Force: Customers will pay more if you solve more of their real problem.

Therefore: Recast your solution to a larger context. Create a logo or trademark for the product. Associate higher quality, guaranteed compatibility, or cachet with the logo and trademark. Charge extra for the trademark and the larger solution. Engage partners/competitors to help out by creating a Dangerous Waterhole.

Remarks: This may seem like a scam, but it usually isn't. For instance, customers are willing to pay for a logo if it implies consistent or reliable behavior or other qualities. People will pay to obtain something of known quality. People will pay for a solution to a larger problem.

Give Away the Least Expected*

Context: You are trying to create a new marketplace based on new technology. You've got a development group that needs to be paid, a logo program to pay for that and to help create the market, and you have competitors gathered at the watering hole eyeballing each other and you.

Problem: Caged animals will attack without warning. Control of the wild is nonexistent.

Force: Companies and individuals don't like to be controlled.

Force: People won't both pay and contribute work.

Force: Control is difficult to swallow in the free-market world.

Therefore: Create a real community for the competitors you need at your **Dangerous Waterhole**. Turn over control of your technology development group to the community. Provide real self-determination.

Remarks: The community will build your marketplace for you. They will license the logo, they will work for free because it's in their own self interest.

Anchor Stores (finish this...)

Context: a mall

Problem: solid draw

Force: why go

Force: attractive

Therefore: make sure there is an well known store to act as an anchor

Remarks:

Fair Processes

Context: A community with common goals and a charter including a common good.

Problem: How can the activities of the community be kept fair?

Force: The community is diverse and scattered.

Force: The community is organized around activities.

Force: There is a need to keep the organization healthy and functioning well.

Force: Some members of the community are competitors of other members.

Therefore: Arrange for all actions and activities within the community to be subject to processes that all agree are fair from the outset.

Checks and Balances

. . . any organization ruled by voting contains both powerful and weak entities. The concept of *common good* may require mechanisms for fairness.

* * *

An organization ruled by voting has entities arrayed in a range defined by power and influence. In a pure democracy, the majority has total control, and this usually means that the weak are at the mercy of the strong. If there is a concept of the common good in the charter of the organization, then pure democracy cannot be an alternative, because in a pure democracy the only common good is the good of the strong.

Building a new marketplace is an example of building an organization that might be ruled by voting but which has a concept of the common good in its charter. A new marketplace based on new technology, for example, requires some degree of planning to create common infrastructure. Typically standards (de facto or otherwise) are used to define commonality for the purpose of growing the market while providing a base from which to compete.

Despite having the common goal of growing the marketplace, an organization is still susceptible to political action, and stronger entities within the organization can act against their long-term good by taking advantage of weaker entities for short-term gain.

Therefore,

When setting up a voting-based organization, make sure there is a system of checks and balances which ensure that the organizational rights of the weak are not infringed by the strength of the strong.

* * *

If there is **Proportional Voting** to initiate action in the community, make the acceptance voting a **Senatorial Vote**. Perhaps implement a **Right to Appeal**. Build several independent suborganizations with jurisdiction over different aspects of the organization's common activities and design a process that moves actions between the suborganizations; organize the suborganizations with different political structure in such a way that there is competition between the suborganizations or that different power structures exist in different suborganizations.

Protect the Weak

. . . in an organization built around interacting entities where the entities are of varying sizes or degrees of influence, there is a tendency for the powerful to dominate the weak.

* * *

Organizations that are built on top of a population of entities of varying sizes or strengths can favor the strong over the weak, perhaps because of proportional voting or because of influence over other entities in the organization.

The large or powerful in most cases have earned their size or power—perhaps through market leadership and success. In some situations the large or powerful deserve to be rewarded for their success. However, in an organization that promotes the common good, weak or small entities should not be damaged in the real world by actions of the organization, because that violates the essential principles behind the concept of a common good.

The weak can be taken advantage of effectively in any system in which voting is a primary way to make decisions.

Therefore,

Put into place mechanisms that would protect the weak in situations where their small size or power would otherwise cause actions in the real world damaging to them.

* * *

For example, use **Senatorial Vote** to balance the strength of the strong with the weak in situations where there are **Proportional Votes**.

Right to Appeal

Context: Any political organization with a charter including the common good operating in a marketplace.

Problem: How can the rights of the weak or small be guaranteed?

Force: A political organization with voting (and especially **Proportional Voting**) can evolve to threaten the weak or small.

Force: Decisions that favor the strong are allowed and perhaps valued because the community has adopted **Let the Market Decide**.

Force: There is a strong ethic that states that no member of the organization should be unduly damaged solely by the actions and decisions of the community as a whole.

Therefore: Arrange it so that all decisions that can adversely effect individual members or a group of members can be appealed before an unrelated political entity.

No Distinguished Members

. . . the members of an organization need to know the rules and accept them in order for the organization to thrive.

* * *

An organization thrives when its rules are fair and accepted, but sometimes not all the members of the organization are strictly equal with the others. Perhaps due to popularity, size, or some other developed attribute, one member has more power than another.

In the United Nations, the 5 permanent Security Council members enjoy special powers and privileges based on a charter which names them. This is in contrast to the 10 temporary members of the Security Council elected by the General Assembly. Further, any country may join the General Assembly within which each country is treated equally. It is important to make a distinction between members of an organization who have earned power and influence in an organization and those who have been granted it based on their identity.

Therefore,

Arrange for there to be no members of the community, its associated organization, or organizational subentities or subcomponents who are granted special power, status, or influence based solely on their identity or uniquely identifying descriptions.

Appeals to Supermajority

Context: Any political organization with a charter including the common good operating in a marketplace.

Problem: How can the rights of the weak or small be guaranteed?

Force: A political organization with voting (and especially **Proportional Voting**) can evolve to threaten the weak or small.

Force: Decisions that favor the strong are allowed and perhaps valued because the community has adopted **Let the Market Decide**.

Force: There is a strong ethic that states that no member of the organization should be unduly damaged solely by the actions and decisions as the community as a whole.

Therefore: Arrange for all decisions that can adversely effect individual members or a group of members to require a supermajority vote or greater.

Remarks: In an organization with **Right to Appeal**, appeals of the outcome of the appeal can be to a supermajority or greater.

Information Flows Everywhere

Context: A community with common goals and a charter including a common good.

Problem: How can the activities of the community be kept open?

Force: The community is diverse and scattered.

Force: The community is organized around activities.

Force: There is a need to keep the organization healthy and functioning well.

Therefore: Arrange for all actions and activities within the community to be reported without bias or unnecessary delay to all members.

Ratification/Community Approval

Context: Any community in which specifications are arranged according to **Levels of Maturity**.

Problem: How can a specification move up the ladder of maturity?

Force: Support for the idea is growing, both in the marketplace and within the community.

Force: There is no formal standards body available inside or outside the community for this specification.

Therefore: Through **Peer Review** or a **Shepherd Panel**, the specification can be ratified by the community and labelled with a higher maturity level.

Proportional Votes

. . . in an organization built around interacting entities where the entities are of varying sizes or degrees of influence, there are occasions when voting is required as part of a process of ensuring fairness.

* * *

Decisions made by voting can be unfair in a number of ways when the size or degree of influence of the entities vary. The needs of a large or powerful entity can be blocked by a small or weak entity that is acting to weaken the position of a larger or more powerful competitor.

The large or powerful in most cases have earned their size or power—perhaps through market leadership and success. In some situations the large or powerful deserve to be rewarded for their success. For example, because a large company may represent a large user community, the large company perhaps should be able to propose for standardization features and facilities useful to its large user population.

Votes are sometimes required to initiate action, and other votes are required to accept proposals. Proposal votes are a good place to aim to reward the large or powerful.

Therefore,

Arrange for proposal votes to count the vote of the large or powerful entity proportionally to its size or power.

Senatorial Votes

. . . in an organization built around interacting entities where the entities are of varying sizes or degrees of influence, there are occasions when voting is required as part of a process of ensuring fairness.

* * *

Decisions made by voting can be unfair in a number of ways when the size or degree of influence of the entities vary. A small or weak entity can easily be overwhelmed by the large and powerful.

The large or powerful in most cases have earned their size or power—perhaps through market leadership and success. In some situations the large or powerful deserve to be rewarded for their success. However, in situations where the fate of the small or weak lies in the hands of a vote, it would be unfair for the large or powerful to have more voting power than the small or weak.

Votes are sometimes required to initiate action, and other votes are required to accept proposals. Proposal votes start action and may be subject to proportional voting, while acceptance votes should aim to protect the small or weak.

For example, in the United States federal government, representation in the House of Representatives is by population—states with large populations have more representatives than states with small populations—and, among other things, the House initiates revenue-spending bills. The other house of Congress, called the Senate, approves or accepts revenue-spending bills passed by the House. The US Senate has 2 senators per state, and so each state gets the same vote.

Therefore,

Arrange for the final acceptance votes to count the vote of the large or powerful entity in equal measure to the vote of the small or weak.

Cut and Run

Context: A member of the Jini Community developing a new service.

Problem: How does a community member move quickly with a product idea?

Force: A member has a great idea for a service.

Force: The member doesn't want to go through the hassle of a standards-like process or doesn't think it's necessary.

Force: The risk/reward equation appears to the member to favor being the first mover.

Force: The requirements specified by the SCSL on all community members.

Therefore: The member does the minimal amount of specification and testing code required by the SCSL and works the market alone or in conjunction with a couple of partners.

Known Uses: Normal business practice.

Microcosm

Context: A member of the Jini Community developing a new service.

Problem: How can a member create a microcosm of services that talk privately to each other?

Force: The member believes there is a market for an interrelated set of services perhaps on a variety of devices.

Force: The member does not want to share the idea.

Force: The requirements specified by the Sun Community Source License (SCSL) on all community members to publish openly any specification intended to be shared with any third party.

Therefore: Create a private interface. Do not share it with a third party and implement it privately. Form any coalition around subcontractor relationships.

Known Uses: Drivetrains.

Make a Standard

Context: A member of the Jini Community developing a new service.

Problem: How does a community member pursue a product idea that requires some degree of standardization to succeed?

Force: A member or coalition of members decide that there might be a market for a new service.

Force: The member or coalition believes that the creation of the market requires a standards-like ratification process to help convince other companies to create complementary products.

Force: The requirements specified by the SCSL on all community members.

Therefore: The member or coalition engages the community process or creates one for ratifying specifications to some level of community standard.

Known Uses: Normal business practice, IETF, Visa International.

Single-Source Specification Creation

Context: The Jini Community

Problem: How can a member gain support for a product idea?

Force: A member of the community has a product idea that would benefit from support within the marketplace.

Force: There is no existing support or supporters disagree with the original product idea.

Therefore: The member writes the specification alone and publishes it to the community.

Multiple-Source Specification Creation

Context: The Jini Community

Problem: How can a group of members gain support for a product idea?

Force: A group of members has an idea for a product that would benefit from support in the marketplace.

Force: The product idea is not ready for standardization or ratification within the community.

Force: There are a number of members who believe in the idea.

Therefore: The group writes the specification together and publishes it to the community

Compelling Idea

Context: A situation where a member of the community has a good idea for a specification though that member has no intention to implement the specification.

Problem: What status does a compelling idea with no implementation have in the community?

Force: Compelling ideas can sometimes make a large positive difference in shaping a community.

Force: A specification without an implementation is likely to be less effective than those with an implementation.

Therefore: Arrange for one of the lower levels of maturity to contain specifications based on compelling ideas which have not been implemented.

Levels of Maturity

. . . in an emerging marketplace there will exist a range of products from novel offerings attempting to grow the market to variations on established products. Both extremes are necessary for the health of the marketplace.

* * *

In an emerging marketplace there is a tension between the need to grow the marketplace by developing new products based on innovative and possibly competing specifications and the need to reduce uncertainty for established products by standardizing existing specifications.

Moreover, as products mature, the best interests of the marketplace may dictate that those products be subject to more stringent requirements that might have been considered oppressive if imposed when the product was first introduced.

Also, for those not very well attuned to the marketplace, it is important to know how much import a given specification has.

Therefore,

Identify several levels of maturity for specifications and institute a mechanism to label specifications as to their maturity level.

* * *

In a system where standards are marked by their levels of maturity, it can make sense to institute a process within the community to explicitly move standards up the ladder of maturity, perhaps in response to already made market decisions. See **Reward Stability**.

Reward Stability

. . . in an organization that is promoting a marketplace, specifications may emerge that enable companies to offer commonality to customers while providing places for competition. The more successful the specification is, the more it should be respected and followed.

* * *

Standards in a marketplace represent agreements about what is important to keep the same. In an emerging marketplace it is not initially clear which specifications will become good standards, and setting a premature standard can do more harm than good. As a specification matures and gains the force of a standard, it becomes more risky to ignore it.

A new specification perhaps has fewer adherents than an old one, and so some variations in its use may be permissible and, in fact, desirable, while an older specification may be treated as a standard and demand strict adherence.

This is true of both informally created and officially developed specifications. C existed as a programming language for a number of years before K&R C became a de facto standard. It was the de facto standard for many years— though with some machine-specific variations—before there was an ANSI C standard.

A specification can be ignored unless it is very well accepted. Ada, which is an ANSI standard, is largely ignored today outside the Department of Defense.

Once successful and stable specifications emerge, basing standards on them makes it easier and safer for companies to build new products based on them.

Therefore,

Promote specifications that are both successful and stable to the level of a standard so they are more resistant to change and companies will be more motivated to use them.

Peers Ensure Quality

Context: A community that is creating specifications for products in an emerging market.

Problem: How can the quality of specifications be guaranteed?

Force: No single member is able through its own expertise to make the highest quality specification.

Force: More eyes ensure the quality and coherence of a specification.

Force: The market is emerging, so that established standards are a long ways off.

Force: There are no external or objective measures of quality.

Therefore: Arrange for a system of peer review to ratify specifications. Make sure that the process of peer review includes steps that reduce the likelihood that good suggestions are ignored.

Remarks: Note that this solution can work alongside **Shepherds Ensure Architecture**.

Known Uses: Editorial review, writers workshops, IETF.

Peer Review

Context: A community where specifications are part of the community working material necessary for the growth of the market.

Problem: How are specifications ratified?

Force: More eyes ensure the quality and coherence of a specification.

Force: There is sometimes a need for members of the community to work together.

Force: There is a concept of "Levels of Maturity."

Therefore: Arrange for specifications to be peer reviewed before being labelled with a higher level of maturity.

Remarks: Note that this is an alternative to **Shepherds Ensure Architecture**.

Community Ensures Completeness

Context: A community where specifications are part of the community working material necessary for the growth of the market.

Problem: Who ensures that each specification is complete?

Force: More eyes ensure the quality and coherence of a specification.

Force: There is sometimes a need for members of the community to work together.

Force: No single member is able through its own expertise to make the highest quality specification.

Force: The market is emerging, so that established standards are a long ways off.

Force: There are no external or objective measures of completeness.

Force: There is a larger community using the specification than the members who are specifying it.

Therefore: Arrange for the community to comment on areas of the specification that might be incomplete, and perhaps arrange for there to be a process to respond to such comments.

Shepherds Ensure Architecture

. . . in a community where specifications are part of the community working material necessary for the growth of the marketplace and where there is an overriding architectural integrity that must be maintained, there are situations where writing a specification requires knowledge both of the domain of the specification and of the architecture in which the specification operates.

* * *

Without experience writing specifications within a context where architectural integrity is important, it is easy to lose sight of the needs of the overall architecture. In a high-maturity market, the proposed specification requires deliberation because there are established players in the area; in a low-maturity market, there might be a desire to plan the structure of this part of the marketplace ahead of time.

Writing a specification should primarily be done by an expert in the domain that the specification covers. Yet, in the early stages of a new marketplace based on a new architecture, there will likely be no individuals or very few who are expert in both the domain of the specification and the new architecture.

Therefore,

Form a body of shepherds who are expert at the underlying architecture. Arrange the organization so that whenever a domain expert wishes to write a specification, a shepherd is assigned to make sure that the underlying architectural integrity is maintained.

* * *

Like all good shepherds—such as the editors of technical journals, PLoP shepherds, and built-world architects—the architectural shepherds should put the interests of the domain experts ahead of their own. See **Community Reviews Shepherds**, **Grow Your Shepherds**, **Peer Review**, **Peers Ensure Quality**, **Appeals to Supermajority**, and **Right to Appeal**.

Shepherd Panel

Context: A system of ratification based on shepherds.

Problem: How does a specification make it to the highest level of maturity?

Force: Shepherds are experts at the underlying technology and specification writing.

Force: The members are experts at the marketplace.

Therefore: Arrange for the highest levels of ratification to require a panel of shepherds and members to make the final recommendation.

Grow your Shepherds

Context: A community where specifications are part of the community working material necessary for the growth of the market and where shepherding is an appropriate process

Problem: Where do shepherds come from?

Force: Shepherds should not come only from a distinguished class.

Force: Shepherds should be representative of the community.

Force: Shepherds should be trained by a mentoring or experience-based process.

Therefore: Arrange to grow shepherds by promoting individuals who have written successful specifications and who demonstrate excellent design and architectural taste.

Shepherd Code of Ethics

. . . shepherds must be trusted in order to be effective.

* * *

Effective shepherds act as knowledgeable guides. Domain experts who are writing specifications alongside a shepherd must be able to trust that the shepherd has the welfare of the specification and community uppermost in his or her mind.

Review processes can go a long ways toward alleviating problems with shepherds having too much power, but in day-to-day activities, the shepherd is working shoulder-to-shoulder with a domain expert or a group of them. The level of trust in each of these relationships needs to be high.

Therefore,

Arrange for the shepherds to adopt a code of ethics which proscribes their behavior. Make the code visible and part of any review process, both self-review and community review processes.

* * *

This pattern can be used in conjunction with **Community Reviews Shepherds** or with **Right to Appeal**.

Community Reviews Shepherds

. . . shepherds represent an authority within a community that writes specifications requiring both domain and architectural expertise. Shepherds must be trusted in order to be effective.

* * *

In an organization with shepherds, a shepherd can represent an authority figure. Effective shepherds, however, act as knowledgeable guides. As holders of special knowledge, it is natural for a shepherd to feel and act superior to a domain expert, and it sometimes happens that a shepherd will be suspected of harboring a hidden agenda.

In a purely cooperative environment, a code of ethics can govern the actions of a shepherd—for example, in a refereed journal, the goal of both the author and the editor is to produce the highest quality, best possible paper or essay. And even though there is little danger of a misuse of editorial power, the editor-in-chief can always be called in to arbitrate disputes. This works because the editor-in-chief serves the community of authors whose work supports the journal.

In a competitive environment, the need for checks and balances is even greater.

Therefore,

Arrange for shepherds to be reviewed by the community, either as individuals, as a group, or by a process that combines self-review with community review. Attach consequences to the result of the review process

* * *

If there is a group of shepherds, rotating shepherds through that group not only keeps the shepherds fresh but provides a natural place to insert a community review process. Community review may take place in conjunction with a group self-review process within the group of shepherds. See also **Shepherd Code of Ethics**.

Community Controls Development

Context: An open-source or community-source community with a group of underlying technology experts in a development organization; the goal of the community is to build a marketplace based on the technology.

Problem: Who controls development?

Force: The expert group has special knowledge and expertise focused in the underlying technology. The development group should retain the right to set day-to-day priorities and assign resources.

Force: No such expert group is in a good position to set the overall direction when the marketplace is unpredictable.

Force: The marketplace is unpredictable and unformed.

Force: There should be no distinguished members.

Therefore: Arrange for the community to control or direct the actions of the expert development organization in terms of its overall direction, perhaps by providing the set of non-maintenance projects from which the development group can choose.

Compatibility Logo

Context: A community needs to ensure compatibility among a set of independently developed products.

Problem: How can the community get a set of separate companies, usually competitors, to agree on compatibility.

Force: Competitors like to get an advantage over one another by making things different.

Force: The competitors would be better off being compatible, but there is no compelling argument in the form of existing forces.

Therefore: Create a logo that signifies to the customer the value of products in the new marketplace. Attach a compatibility requirement to license the logo. Charge a fee to pay for developing the logo (and building its value in the marketplace) and for any other development costs you need to bear.

Known Uses: Intel Inside, Underwriter's Laboratory, Visa International.

Give Away All You Can

Context: A development situation in a **Safe Place to Share**.

Problem: An organization has too much work to handle.

Force: Software is challenging—it requires careful, coordinated thought and planning.

Force: Software takes a lot of work—every development staff has much more work it wants to do than it can.

Force: Lots of the work to do is necessary, but does not add marketable value.

Force: Other organizations also need to do similar work that does not add marketable value.

Force: Lowering the barriers to entry might increase and/or mature the market more rapidly.

Force: People overvalue what they own and do not want to give it up.

Therefore: Go Whole Hog and give away as much of the work that does not add marketable value as possible. Turn over control of this work to the community, adding sweet water to the **Dangerous Waterhole**.

Remarks: This is a way to implement **Anchor Store**.

Known Uses: Netscape.

Every Bug is Trivial to Someone**

Context: Any software development situation.

Problem: A large corpus of software can be overwhelming for an individual or small group of developers. Just repairing defects can be a problem.

Force: The software is of value to all members of a large community.

Force: Control of that software is not essential to any particular organization.

Force: Software is very hard to do and every development staff is too small.

Force: With enough experienced developers with diverse backgrounds, any bug is trivial to one of them.

Therefore: Share responsibility for diagnosing and repairing the code with a large, diverse community.

Survival of the Fittest**

Context: A shared source community.

Problem: Several members of a community wish to work on the same part of the system.

Force: The software is of value to all members of a large community.

Force: Control of that software is not essential to any particular organization.

Force: Someone or some group produced the main idea for the system or developed the initial version of it and holds its vision.

Therefore: Allow the interested members to independently work on the same part of the system. Let the vision holder or the community decide which version is best.

Scratch Your Own Itch**

Context: A shared source community.

Problem: Once the main thrust of a piece of software is determined, completing the software can be overwhelming.

Force: Complex software has many parts all of which taken together is too confusing for any small group or an individual.

Force: With enough diverse members of the community, there are enough different parts of interest to cover the entire corpus of software.

Force: There is no central control, so coordinating repairs and completions is not possible.

Therefore: Let each member of the community complete or repair the part of interest to that member.

Work in Your Own Medium*

Context: Any situation where work is needed.

Problem: Some people work for money, some for fame, some for the love of it. And when the work is hard, the money, fame, or love must be in good supply.

Force: There is not much money or fame, and maybe not enough love.

Force: The work must get done.

Force: Volunteers are all that are available.

Therefore: Arrange it so that each volunteer can work naturally in his or her own medium. If you need art and have a painter, supply canvas and oils; if you need software to be fixed or extended and have a hacker, supply source code.

Remarks: This applies to intellectual property as a medium.

Continuous Releases*

Context: A shared-source community.

Problem: How can the best software be made available when a number of parties depend on the software and share control of it in a distributed organization?

Force: Someone needs the most recent error corrections and shared modifications.

Force: Someone needs a stable version.

Force: Changes are being made at unpredictable times.

Therefore: Arrange for relatively stable (tested) releases to be made as soon as is practicable. Label the releases accurately and permit members of the community to choose the proper one without penalty. Try to maintain backward compatibility.