CHARTERS AND CHARTERING:
IMMUNIZATION AGAINST FORESEEABLE PROJECT FAILURE

This is the pre-edit draft of a Chartering article written by III. The edited version was published in the January 2001 edition of STQE Magazine.

When does a project really begin? Where do the stewards of organizational resources declare the effects they hope to achieve when all the resources are gone? How can external observers determine after the fact that the project was successful? Who gets to say that the work is good and the effort may continue? Why do so many projects create so much difficulty for so many people and organizations?

A project is a finite undertaking with intentions and limited resources. It is different from a service or a product or a discipline or an occasion. It serves as a rallying point for participants from two very distinct groups of folks. The two groups are those who have responsibility for organizational intentions and resources, and those who practice their craft creating information capability to further the organization's ultimate purpose. In this discussion, I shall refer to the first group as GoldOwners (based on the 21st-century version of the Golden Rule: Whoever Has The Gold Makes The Rules) and to the second group as Developers. In every system effort, the developers are spending money that belongs to the GoldOwners (or, perhaps, the GoalDonors). In return for the use of resources, the developers are obliged to return some sort of value-added consequence to the organization. Chartering provides a mechanism for negotiating and formalizing the understanding between the two groups, and the Charter is the official container for the shared contractual agreement between both sides.

I dream of the day when senior business managers will walk into the office every Monday morning or so with fully-formed Charters already in hand. They will have sorted through the research, pondering and decision-making to focus the attention of the organization and the development crews toward establishing expectations, aligning the perspectives of all involved players, and securing commitment for the successful completion of important work.

When I wake up, the situation is quite different. Work often gets underway in the absence of an explicit contract, with the result that people launch off in various directions, understandings are diverse and changing, time and money drift away while informal chats try to determine the next best thing to do, various individuals step up and do their best from a narrow perspective to keep everyone else's head above water, and results diverge from expectations.

In far too many unchartered projects, the developers wind up in the gunsights of the money-spenders, taking the blame for any and all hiccups that unfold during the life of the project.

In my experience, a primary cause of the difficulty in getting started competently arises from a galling diversity in the perceptions around the topic of "requirements". Study after study points out that a majority of project (and system) failures come from problems with "requirements". I'm going to take a stab at that issue as a basis for carving out some space for the Charter concepts, and for establishing a more workable taxonomy for the statements we make to each other during project deliberations.

Quirements abound. I'm using a made-up word to characterize the total set of statements representing the wishes, hopes and desires of interested parties when they talk about their view of a system to be built. In the first rounds of discussions, business people get lots of encouragement to say everything they can think of that might give guidance to developers about the content and operation of the system. In most cases, these declarations become "the requirements", and constitute the reference for all subsequent development work. The underlying rationale for this approach holds that if it came from a business person's mouth, it must be gospel. Project members capture the statements as narrative bullet-points, number them, load them into a requirements management spreadsheet and then spend the rest of the life of the system trying to figure out what they mean.

The first dilemma emerges from the use of natural language as the primary medium of communication in this endeavor. Natural language is wonderful for love poems and political speeches, where there is value in the richness of nuance and ambiguity. If, on the other hand, we are trying to be specific about very particular kinds of work and flow that must proceed correctly in order to support the enterprise, we have a problem. What we need is a much more precise style for saying what must be true about our efforts, and about the systems that those efforts produce.

The next challenge is to figure out a way to package this information in a way that allows the container to suggest the content. In the cases where I have dealt with a collection of these "requirements" statements, I have conducted my own private parsing of the bullet points. The variety of categories is often breath-taking. If you try this yourself someday soon, I suspect you will find most of the following:

- Organizational values
- Political guidance
- Shop best practices
- Intermediate project scheduling concerns
- Mandates about pre-selected technology
- System objectives
- Project objectives

- Organizational objectives
- Resource availability/constraints
- Bright Ideas
- Business justifications
- References to the insides of other systems
- User procedures
- Vendor relationship issues
- Methodology dictates
- Formats and layouts
- Mission statements
- Protocols about system operations
- Risk avoidance strategies and tactics
- ???

Certainly, all of these concerns are real. Yes, it's important that the team know about them. Of course, we want to honor the willingness of the business people who have worked so hard to share with us what they know, think and feel. What would help here is a better way to organize and save this shared knowledge and perspective so that the appropriate portions of it are readily available to the people who need it, when they need it, in a form that will fully support the work they have to do.

Quirements is based on the Latin verb *quaerere,* which means to ask or inquire. When we add the two-letter prefix "re" to the front of the ask word, the intensity changes. In this instance, the prefix means "again". What I propose is that we limit our use of the term **re**quirement to those policies and rules that hold true over time, over and over again, within the boundary of our system. With this approach, technology and its symptoms have to find another container, because every time we think the latest and greatest box or language or operating system is the right answer to our implementation challenges, a newer something comes along to take its place. A more insidious consequence of including technology aspects in a **re**quirements model is the drastic reduction in shelf-life of the model. I hold a personal metric that suggests that over the entire lifetime of a system, roughly 70% of the gross resources expended are used to discover and clarify true requirements. That's a huge investment. If it were just a phase, it might seem more tractable; instead, I'm talking about the entire duration from blue-sky idea through development, operations, changes and ultimate retirement. In order to extend the payback period of the outlay, the result should last for a while. If technology details reside in the model, then a change in the technology will dictate a change in the requirements view; and the requirements work starts all over again. I suspect that modelers instinctively realize this is true, and are then generally reluctant to be rigorous and thorough in detailing requirements, because they know that (with design facts included in the approach) they're just going to have to be constantly revising the models of both requirements and design. The upshot is a brief pass through requirements before settling in with design work, where most of the exhaustive work on organization policies emerges in the details of the design models, fragmented and brightly wrapped in implementation pieces of the system.

Based on timing and location, we can explicitly and reliably distinguish objectives from requirements from implementation. In order to talk about how we can accomplish this partitioning, I need to offer definitions of some terms. Along with these definitions, I will suggest examples of each from the history of the Cold War, to show how they complement each other.

Every organization has an interest in its Mission, Objectives, Requirements and Design. I have been to so many shops where these terms are loosely used, with some groups making them seem almost synonymous with each other. In fact, they are very different, and serve very different purposes for different roles in the organization.

<u>Mission</u> - From the Latin verb *mittere* , to send, comes the word mission. I define it as a one-liner that sets a direction for the work to follow. It should be both comprehensive and compelling. If it goes on for paragraphs or pages, it loses its impact, and drops off everyone's radar screen. A well-formed mission statement should suggest everything involved in its pursuit, almost in the sense of a calling. Missions persist; they are always in force (as opposed to the military slant on this term, where a mission is a single journey out and back to do one thing), and they endure from project to project and from system to system. In the Cold War example, a mission of the Free World was to demonstrate the superiority of our system and values versus those of the Communists. (A strategic choice then directed our national energies to the arena of space adventures to play this out)

<u>Objective</u> - This one derives from the Latin verb *jacere*, to throw. I like the flavor this brings to the word, in the sense that you toss an objective out there in front of your future, and then manage the present through time to catch up with the measure described in the objective. A good technique for discovering objectives is to interrogate "quirements" with the question that every four-year-old keeps asking: "WHY?". If the quirement is about activities (work that takes place inside the system boundary) or deliverables (work products that leave boundary and head for the outside world). the four-year-old's inquiry will ultimately lead to an objective: a statement of external, measurable, policy-based effect, as of an instant in future time. The objective should be framed in a way that makes its assessment binary: we either achieved it or we failed. In the Cold War example, President Kennedy announced in May of 1961 (following Sputnik and Yuri Gagarin) the objective of sending a person to the moon and returning them safely by the end of the decade. And we did it. Objectives create alignment among participants by making clear how we are going to measure success without having to describe what happens between now and then. When everyone knows what the game is, they are individually empowered through knowing whether their local actions or decisions will help or hinder the measurable effect by the assessment date.

Objectives come and go. After the success of the Apollo 11 moon-landing, that objective was evaluated and retired. (The mission persisted, and gave rise to a host of additional objectives, one of which even included a successful space rendezvous with the Soviets in the Soyuz program.) The location and timing distinction between objectives and requirements makes it easier to distinguish them from each other: the assessment of the success of objectives takes place outside the boundary at a specific point in time, whereas the set of requirements holds true inside the boundary over time. For each approved objective, there is an inevitable set of requirements which will have to continuously behave properly in order for the measurable effect to occur by the appointed time. You can always work from the objectives to the requirements; I have yet to confidently approach it in the other direction.

Requirement - The iteration prefix "re"attached to the ask word suggests the definition: a statement of organizational policy, generally expressed in terms of information, that holds true over time. The most effective way for me to get a grip on requirements is to indulge a fantasy of Perfect Technology. What if, for a given system boundary, I had absolutely perfect technology with which to implement everything inside. The characteristics of Perfect Technology are: infinite storage capacity, clean and transparent interfaces, processing power faster than the speed of dark, a location only at a point in three-dimensional space, always up, zero cost, implementation before I finish thinking about it, even faster changes, and the universe would pay me to use it. With the fantasy in place, and with clear objectives to work from, all I have left to think about are the true requirements. A good way to organize that thinking is to partition by Key Events, which provides a framework for capturing the details of content, transformation, behavior changes and data manipulation and retention. Requirements are free. There is zero cost associated with a GoldOwner declaring that employees are described by a set of attributes or that the following algorithm will set the values for commands issued to external devices. (Certainly there is the nominal cost of discovering and recording those policies; the policies themselves are free). Whereas an objective leads to an inevitable and finite set of requirements, a requirement is open to an infinite number of design and implementation possibilities. In our Cold War story, with the mission to show that our way was better, and the objective to return safely from the moon within the decade, a whole host of requirements emerged. For instance, there were ongoing requirements about communication between the ground and the Apollo vehicle:

Key Events for messages between the two described the content, fidelity, cycle time and frequency obligations necessary for successful accomplishment of the objective. (As a practical matter, the design choice for implementing the communications requirements turned out to be radio, even though there were lapses in fulfilling the requirement during passages behind the moon and re-entry into the Earth's atmosphere. Because the objective was clear, the Apollo managers reasoned that they could provide workarounds or gracefully absorb the downtime without ultimately jeopardizing the objective.) Another requirement, ongoing, was the support for and monitoring of the pulmonary or breathing activities of the passengers. The specifics had to do with respiration rate and O_2 concentration in the blood. Lots of different design options could have supported this requirement. The Apollo 1 iteration, intended to rehearse the assembly of the vehicle, the transfer to the launch pad, the connections for fueling and communication, the access available for the astronauts and their comfort and effectiveness in the capsule, made a design choice of pure oxygen in the command module. The tragic deaths of Chaffee, Grissom and White pointed out the error of that choice. The requirement was still true, and subsequent design thinking yielded a blend of elements to mix with the oxygen in order to fulfill the requirement and have a much better chance of accomplishing the objective.

By this definition of requirements, all levels of detail come into play. Some shops hold the view that requirements are only high level, overview considerations, and that moving toward finer granularity is inherently the subject of design. It turns out that there are both generalities and details in the requirements, as well as generalities and details in the design. I suspect that the source of this misapprehension is the confusion of objectives with requirements, along with the traditional use of the "what/how" guideline as a way to distinguish the organizational view from the technology view. The real distinction that emerges from the what/how spectrum is one of differing detail rather than of different subject matter. There are requirements "whats" along with the details of their "hows"just as there are design "whats" supported by technology-specific "hows".

Design - The Latin noun *signum* means a mark or a sign. This suggests a representation of some kind that stands for the tangible stuff that will make the system operate properly in a physical world. I define design as a blueprint that shows the use of, interactions among and control over components of technology selected to fulfill requirements. Any discussion during system development that refers to things that cost money or take time or can be ordered from a catalog or you can bust your knuckles on is a design discussion.

With these definitions in place, we can now turn our attention to the contents of a Charter.

I suggest that a project is the consequence of a Charter. Prior to the formal ceremony of agreeing and committing to the stipulations of the Charter, we lack a project budget with authorized funds with which to pay for early discussion and exploration. I see this early work as management R+D, as part of their ongoing obligation to explore possibilities and opportunities for the organization. One amazing benefit of Charter exploration is the dismissal of lots of Bright Ideas that would truly waste time, money and good will if someone tried to follow through on them. If there is major difficulty in forming and agreeing on a Charter, imagine the grief that would emerge from plowing ahead anyway.

The basic charter has four sections:

- Objectives
- Boundary Exhibits
- Committed Resources
- Authorizing Players

Every shop that has worked on Charters with me has inevitably added additional content they felt was necessary in order to gain wider acceptance. I have generally counseled against doing so, because of the dilution effect that results from trying to cram everything we know into a single presentation. There are generally other more appropriate containers for the additional content; I leave it to practitioners to make their own choices about where the other material might best find a home.

The charter objectives come in at least two flavors: External Objectives, which the GoldOwners crave and evaluate; and, Internal Objectives, which usually have to do with intentions for the development shop, that need to be pursued within the same time frame as the External Objectives, utilizing GoldOwner resources. For example, a powerful External Objective might be an improvement in the turnaround of missed payment notifications from four days to twenty minutes by the end of next quarter; at the same time, there might be a significant need for the shop to extend the percentage of re-use in code for new systems from 10% to 60%, and to have that in place by the end of the year. The GoldOwners are paying for the Internal Objectives anyway, it makes good political sense to be open with them about how their resources are improving capabilities as well as effects

Objectives are hard things. I've been focusing on them for the last two decades, and I still find it challenging to crystallize organizational intentions in a way that creates focus for the system community in a way that highlights outcomes while retaining a fair amount of internal autonomy. One approach that helps me is to imagine explaining to a visitor from the starship what it is we are all about, and how they could go see for themselves that we have done what we set out to do. The temptation is to look at the work we do as objectives, or point to the deliverables we create as objectives. At the end of the day, GoldOwners are far less enthusiastic about projects or systems than they are about external business effects that tie in to the larger strategic view they build for the organization. They would be perfectly happy to dispense with projects and systems forever if they could still make the difference they want to make in the outside world. Another caution: in environments where management has historically demonstrated success by their own pronouncements rather than by hard measurements, there will be enormous resistance to agreeing on well-formed objectives. In some cases, it may be the first time in their careers that anyone has had a way to hold them accountable for meaningful measures of their work. The remedy for this is to demonstrate that hard evidence of success is in their best interests, and to suggest that their careful thought and consideration at the start of the effort will shortstop a host of pitfalls that would otherwise occur much later, when the solutions are much more painful. I hold another metric that seems conservative and that I would love to prove somehow: for every person-hour spent on diligent and clear-headed consideration of the objectives at the start of a project, there is a savings of roughly one person-month in reduced wheel-spinning, chasing down blind rabbit-holes and reframing the shared community perspective concerning the work at hand. While working on objectives, it is useful to stay in touch with any mission statement based on the same boundary, both for inspiration and preservation of sanity.

Boundary is everything. In each and every discussion about the workings of a system, statements have meaning only in relation to the boundary. In the general description of how to edit or filter a transaction, the location of the boundary will determine whether the internal technology does the job, or the filtering takes place outside the boundary before becoming input to the system. I use the term boundary where others might talk about scope or context. When I look up scope in the dictionary, it has to do with seeing; context speaks to the outlying items that surround a thing; what I need is a word that clearly designates a limit or an edge so that I can clearly tell what's in and what's out. The sense of boundary is really one of authority and accountability. For a project, there must be a territory within which the team has control and responsibility, and outside of which they must take questions and decisions to some other source for answers and choices. The traditional method for declaring the size of a system has been to list "functions" included and excluded. That might work if all hands shared the same understanding of just what a "function" is, and if there were any rational way to assure shared understanding of the inclusions and exclusions of a particular "function". I find that a much clearer way to sustain confirmable views of system size is to use two exhibits: the classic Context Diagram, from Structured Analysis, with fully-fleshed-out content definitions for all of the named pipelines in and out and descriptive names for all the pieces of the outside world with which our system shares traffic, along with a roster or census of all Key Events for which the system must have a planned response. Key Events originate outside of the boundary, and can arise from choices or timing considerations. They reflect the most comfortable way that users appreciate their systems: as black boxes with mysterious insides that can provide useful service when prompted or triggered by information or commands. A rigorous portrayal of Key Events would show, in tabular form, for each Event: the external occurrence; the named pipeline through which the outside world informs the system that the Event has taken place; a phrase (or phrases) in verb-noun format that describe the work the system must do when it finds out about the Event; (for each response phrase) a listing of all information items inside the system that the response activity must interrogate or modify or compose or remove; and a classification of the response work as data manipulation or behavior change. The pairing of these two exhibits makes exquisitely precise the capabilities of the system. It can only be triggered by the Events on the list, and it can only work with the data available to it. Over time, as changes occur in the amount or type of work the system covers, those changes will usually result in revising the Diagram or the Events list, or both.

The third section of the Charter, Committed Resources, really gets to the heart of the matter. Here the GoldOwners have a chance to reckon the value of achieving these objectives consistent with this boundary. We can move beyond the classic brouhaha that emerges when an idea about a new system floats in, the developers are burdened with saying how much it will cost the organization to build and deliver, managers respond by saying "that's the wrong answer", developers are then in a position to low-ball the projection in order to get any approval to do anything, the real costs far exceed the agreed amounts, and the developers get to be the bad guys. Comedy is never pretty.

In a Chartering organization, the GoldOwners make business choices that determine what it is worth to accomplish the effects, in line with other competing interests, guided by larger concerns portrayed in the mission and strategies. In an extreme example, the GoldOwners might call for a complete overhaul (as an objective), of the entire universe (as a boundary), and be willing to invest $1.32 and from now until next Thursday to get it done. In the negotiations, I hope the developers will invoke a lovely two-letter Anglo-Saxon word --NO!--and counter with a proposal to accomplish a partial overhaul of a portion of the universe for $2.79 and two weeks from Friday. Part of the intent here is to encourage and allow the GoldOwners to see systems as a business rather than a technical priority.

There are a number of flavors of Committed Resources:

- Dollars
- People Time (as contrasted with the day of assessment for the achievement of the Objectives)
- Tools (for the developers as well as for folks who interact with the installed system)
- Training (as above)
- Access, to places/sources of necessary knowledge, and to authoritative decision-making when the occasion demands
- Working Environment, where developers have enough privacy and space to do high-quantity, high-quality work
- Permission To Iterate, rather than have their first-cut results seized upon and installed as working systems

A necessary aspect of this concept relies on the development shop to have sufficient metrics history so that they can partition a very early model of the proposed system into meaningful units which can then serve as a basis for projecting the historical delivery cost per unit and come up with a reasonably confident basis for accepting or declining the GoldOwners' offer. In the absence of this history, everything is just a guess anyway.

Anyone who has been to a car dealership to purchase a new vehicle remembers that the first meaningful dialogue centered on how much they were willing to spend. Only with that number in hand does the rest of the conversation make any sense. If a transaction as crass as buying a car can start out that way, it seems reasonable to expect that the negotiations over a new system can have at least as much information to begin with.

Once having committed to the resources, there are now expectations and obligations on both sides, along with a much more valuable circumstance: there is a balance between the desired accomplishments and the resources available to allow them to come to completion.

The final chunk of the Charter lists the Authorized Players. We specify the names of flesh-and-blood human beings who are authoritative witnesses for the following two simple questions:

- Is the work to date acceptable?
- May we continue?

It is hollow to make reference to policy documents, or job titles or departments from the organization chart when what we really need are individuals who have a stake in the outcome and sufficient preparation to be helpful in evaluating progress and plans. It is a show-stopper for a Charter to be without real persons as Authorizing Players.

A challenge for Chartermakers is to keep the document as concise as possible. Excess is the enemy of clarity. I often use the mission statement as a preamble to the Charter; it helps keep the community in tune with the larger picture while focusing on the current effort.

Once the Charter carries the signatures of GoldOwners and development leaders, it then becomes the reference point for response to any change in project circumstances. If resources get diverted, or objectives change, or new Events enter the mix, or additional traffic needs to cross the system boundary, everyone declares an immediate (and usually brief) time-out while the parties reconvene to re-establish the balance between intentions and resources. Failure to come to a new agreement shuts down the project.

Here are some tests to help you certify that the Charter is well-formed:

Objectives-

– Have we stated each objective as a measurable external effect with an associated day of reckoning (either discrete or recurring)??

- Can we show that all objectives support the strategic plans of the larger organization??
- Could the success of any objective prevent the success of any others??
- Can we reasonably hold the circumstance inside the boundary to account for the success or failure of each objective??
- Is there a balance between the Objectives and the Committed Resources??
- Is there at least one Authorizing Player who will speak for each Objective??
- Is there a credible and feasible way to actually assess the conditions of each Objective at the appointed time??

Boundary-
- Have we declared (and defined with rigorous content definitions) all of the cross-Boundary traffic, along with the identity of the sources and destinations of those flows??
- Have we listed all of the external triggering occurrences (Key Events), including temporal conditions and other-system-generated prompts, that oblige some sort of behavior within our Boundary??
- Are all Key Events reasonably detectable??

Committed Resources-
- Have we clearly itemized all of the necessary and appropriate resources from these categories:
  - Dollars
  - People Time
  - Tools
  - Training (for both developers and users)
  - Access (to information <u>and</u> to timely decision-making)
  - supportive Working Environment
  - Permission to Iterate (rather than have preliminary results be embraced as final products)??
- Have we secured agreement to re-negotiate when there is any failure to make the Committed Resources available??
- Is the set of Resources sufficient to meet the objectives within the Boundary??

Authorizing Players-
- Do the Players truly have organizational sanction to make the commitments they have signed??
- Have we secured agreement to re-negotiate when there are any changes in the cast of Authorizing Players??

The Charter is offered to the developers as a proposal, with the developers either accepting or declining the offer. With acceptance, there is a formal ceremony of agreement and a project is launched; the other possibility is for the developers to suggest a counter-proposal that speaks more realistically to the capabilities and energies of the team. The counter-proposal is a starting point for additional negotiation, where the two parties work on a balance between intentions and resources. They either agree or let it go. In the absence of agreement, the business folks are free to engage in a search for other players interested in taking on work that is poorly targeted, under-provisioned, or vaguely bounded.

It is time in the history of computing to complete the loop. We turned our first efforts to mastering code, over half a century ago. With the emergence of faster, cheaper, hairier technology we set our sights on bigger challenges and opportunities during the 60's and 70's. That created such a mess that we discovered and refined the practice of design, in order to make software more manageable and accommodating to change. With increasingly elegant solutions to what turned out to be the wrong problems, we spent the 80's and 90's coming to grips with effective methods and presentations for agreeing on requirements. Now we see more urgency than ever for taking positive steps to fully align the interests of the organizations we serve with the rapidly expanding system capabilities at our collective disposal. Charters and Chartering offer a basis for fulfilling that potential.