

Chapter XI

Ten Lessons from Finance for Commercial Sharing of IT Resources

Giorgos Cheliotis, IBM Research GmbH, Switzerland*

Chris Kenyon, IBM Research GmbH, Switzerland

Rajkumar Buyya, University of Melbourne, Australia

Abstract

Sharing IT resources within and among organizations is an attractive value proposition in terms of efficiency and flexibility, but despite this, commercial practice is limited. In contrast, financial and commodity markets have proved very successful at dynamic allocation of different resource types to many different organizations. Thus to understand how the potential benefits of sharing IT resources may be promoted in practice, we analyze enabling factors in successful markets. We present 10 basic lessons for IT resource sharing derived from a financial perspective and modified by considering the nature and context of IT resources. From each lesson we derive the

required software or process capability required to support it. We then evaluate the maturity of the respective capabilities within the peer-to-peer and grid environments using a simple framework based on the standard Capability Maturity Model approach. We conclude with a description of the largest capability gaps and the lowest hanging fruit for making IT resource sharing a more viable business proposition.

Introduction

Sharing IT resources within and among companies is an attractive value proposition for many organizations in terms of efficiency and flexibility, but despite this, commercial practice is limited. The scope of potentially sharable IT resources includes computation, storage, and data. Network bandwidth has been shared for some time, but this is typically done without a defined quality of service. It will also be necessary to create appropriate packages of the different resources to be shared, but beyond the scope of this article. The context we are considering is large-scale sharing among separate budget entities, for example, within a large life-sciences company, an oil company, or a financial institution, or indeed, among them all. Common technical paradigms for enabling resource sharing have been established in terms of the peer-to-peer (P2P), the cycle-harvesting, and more generally, the whole grid movement. Whilst the value proposition for resource sharing may be compelling in the abstract, sharing is still at a rudimentary stage in practice.

Although business executives and IT managers would surely welcome the ability to extract more value from available resources, especially in light of shrinking IT budgets, they have been slow to adopt such practices, presumably because many of them are not yet convinced that these new sharing paradigms can deliver in practice. Technically, it is indeed possible to allocate resources as needed and to change this allocation on very short time scales. However, the ability to dynamically align resource allocations with changing business objectives is largely absent. Thus the principal reason for the slow commercial adoption of P2P and related technologies is that although such technologies enable sharing, they do not help an organization decide how to best allocate the resources it owns. In contrast, financial and commodity markets have proved very successful in terms of both scale and scope regarding the dynamic sharing and allocation of many different types of resources among many organizations.

Thus, to understand how the potential benefits of sharing IT resources may be realized in practice, we consider 10 lessons learned from the financial sector. From each lesson we derive one or more software or process capabilities

required to support it. We then evaluate the maturity of the respective capabilities within the P2P and grid environments using a simple framework based on the standard Capability Maturity Model (CMM) approach from the Software Engineering Institute (2004).

We do not claim that the 10 lessons considered here and the respective capabilities we derive are exhaustive and definitive but rather that they have been fundamental for enabling commercial efficiency and flexibility in other resource-sharing environments. Thus, these lessons are important for management and engineers who aim to promote efficient and flexible usage and sharing of IT resources in a commercial context.

These lessons provide management with a checklist of the relevant business issues and required capabilities for the successful implementation of appropriate sharing strategies in their organization. Practitioners will appreciate the scope of the challenge remaining for business alignment of sharing technologies and be able to identify which issues to prioritize in practice.

The objective of this chapter is not to derive an academic research agenda, although there are clearly many opportunities in this field.

Grid, P2P, and Cycle-Harvesting: Three Converging Paradigms

Resource sharing can be implemented in many different ways, and several technical approaches, each with its own community, literature, software, and so forth (e.g., Forster, Kesselman, Nick, & Tuecke, 2002; Kamkar, Shahmehri, Graham, & Caronni, 2003; gnutella2.com; Thain, Tannenbaum, & Livny, 2002). Of these approaches, we identify three main movements: grid computing, P2P, and cycle-harvesting. These three movements have large areas of overlap, but given that all three terms are frequently used to describe similar but not identical technologies, we briefly discuss and compare them to set the stage for the rest of the chapter. A summary of this high-level comparison is shown in Table 1.

We compare resource-sharing paradigms according to the following categories: organization, function, enabling technology, user expectations, user sophistication, access requirements, and commercialization of enabling technology and of relevant resources. The category that deserves particular attention in the context of this chapter is the last, commercialization. As the other categories serve to illustrate that these technologies may be different but in essence refer and lead to the same objective, that is, large-scale resource sharing, we will rarely distinguish between them in this chapter.

Under the term *commercialization*, we consider the commercial success of companies selling enabling technologies, and separately the success of compa-

*Table 1. Comparison of common paradigms for IT resource sharing***IT RESOURCE SHARING**

Feature	Sharing Paradigm		
	Grid Computing	Peer-to-Peer	Cycle Harvesting
• Organization	• Any	• Peers (by definition)	• Classes of peers: job originators; harvested resources
• Main purpose	• Resource sharing – Computation – Storage – Data	• Content sharing – Data	• Cycle sharing – Computation
• Core enabling technology	• Resource virtualization	• Distributed data search and retrieval	• Resource virtualization
• Expectations for: Security / Reliability	• Medium / Medium	• Low / Low	• Medium / Low
• Sophistication of: Implementers / Users	• High / Medium	• Low / Low	• High / Medium
• Access Requirements	• Read/Write/Execute	• Read	• Read/Write/Execute
• Standardization	• Formal process led by the Global Grid Forum (GGF)	• De-facto standards from successful software; some movement to more formal process through GGF	• Proprietary solutions; now moving to more formal process through GGF
• Commercialization – Of enabling technology – Of resource usage	• Underway – Many companies – Some	• Problematic – Limited – Failed	• Problematic – Many companies – Failed

nies selling the resources themselves (internally or externally). Whilst some P2P software is or has been pervasive, for example, Napster or Kazaa, the business success of the companies supporting the software is less clear. In contrast, there are many companies successfully selling grid and cycle-harvesting software or offering a significant services business around these technologies, for example, Platform Computing, Entropia, United Devices, Sun, IBM, and others. In terms of sales of the resources themselves for money, there are some internal company examples in the grid space (although we cannot cite company names at this point). There have been a number of attempts to commercialize resources on a P2P or cycle-harvesting basis, but we are unaware of any significant successes.

From the comparisons in Table 1, it is clear that in terms of delivering resource-sharing capabilities, P2P and cycle-harvesting are functional subsets of the grid paradigm although they do not usually share the Global Grid Forum's technical standards at this point. The paradigms are often directed toward different market segments and there is little technical commonality in their most successful implementations (e.g., content sharing for P2P versus CPU/storage use by applications for grid). However, the high-level business propositions are basically the same and we can expect the functionality to merge with increasing commercial interest and further standardization. It follows that the 10 lessons we discuss are relevant for all three sharing paradigms.

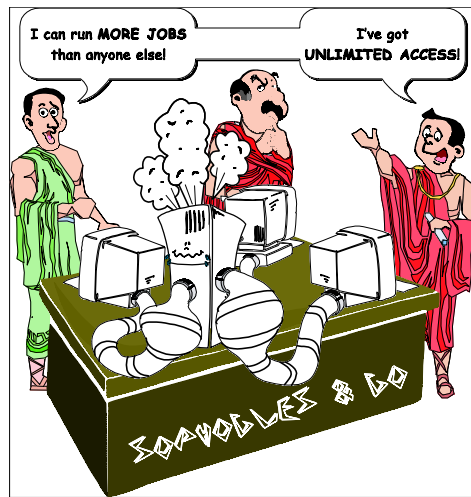
Ten Lessons from Finance

In this section, we describe each of the 10 lessons and the capability required to derive the benefit from each. In the following section, we assess the maturity of the respective capabilities in IT sharing systems.

Avoid the Tragedy of the Commons

Usage increases whenever there is an increased need, that is, when the marginal benefit to the user of utilizing an extra unit exceeds the cost of consuming that unit. In an uncontrolled or free-for-all situation this marginal cost may be very low. This is especially true in the context of IT resources (see Shapiro & Varian [1999]), and is potentially a source of trouble. If companies consider only individual user's budgets and preferences in determining resource value, they may neglect a very important factor: what economists call (network) externalities. An externality can be defined as the impact of one person's actions on the well-being of a bystander and it can be positive or negative.

Figure 1. *The tragedy of the commons*



This socioeconomic phenomenon whereby the individually “rational” actions of members of a population have a negative impact on the entire population is often called the tragedy of the commons (see Figure 1). Common recipes for dealing with this issue target the internalization of negative externalities into every individual’s decision process. Mankiw (1997) and Shapiro and Varian (1999) state that this can be achieved by taxation, regulation (e.g., TCP congestion control), private solutions, or prices for access rights, for example, permits.

Shared IT infrastructure is particularly prone to negative externalities because there is currently no scalable and dynamic standard mechanism for limiting system (ab)use. Local priority rules are efficient only in returning grids to their pre-grid, that is, nonshared, state whilst free-access spaces suffer from the tragedy of the commons. Static policies are particularly inappropriate for dynamic virtual organizations and do not scale well as the number of participating entities increase. Pricing access to IT resources and permitting resale is a direct and scalable way to preclude such a tragedy of the commons for grid deployments that deserves serious consideration.

Discover Dynamic Value

Given even the most cursory awareness of conventional resources and commodities such as copper, electricity, and petrol (gas), it is clear that resource value at the wholesale level is dynamic. What is perhaps less clear to some casual observers is that resources on grids have dynamic value.

Value derives from a combination of need and scarcity. User needs are not constant; they change over time, and the changes also depend on the time scale and granularity of observation. During a project life cycle, a single user working on that project will have varying workloads in different phases of development. The number of projects that a user is involved in also changes with time. Needs are also driven by external and irregular events, for example, reactions to advertising campaigns, seasonality, requests for bids that require data analysis. Variations in user needs change resource value very little if the resources are not scarce, that is, if the capacity of the shared infrastructure is never exhausted. However this happy state is rarely present for users with computationally heavy applications.

After recognizing that the value of any given resource changes with time, the obvious question is how to discover this dynamic value at any point in time. One approach is to use a dynamic “price formation” mechanism. The mapping of needs to prices, called price formation, has no single solution, but there is an extensive body of work precisely on this topic: auctions (see Klemperer [1999]). Whilst prices must be fed back to users (see next lesson), there is no corresponding need for the price formation mechanism to be visible to users. This can be handled for the most part by automated software, but a mechanism is still required and there are significant design challenges for it.

The lesson from auction theory and practice is that effective price discovery is difficult: the choice of price formation mechanism can either promote market efficiency or hamper it. Generally, it is difficult to achieve a balance between the needs of producers and consumers. Recent examples that illustrate this difficulty very well are the 3G mobile telephony spectrum auctions of Klemperer (2002) and von Weizsäcker (2003). High-profile auctions for 3G licenses have been carried out in many European countries. Two distinct problems arose in these auctions: bidder busts (“winner’s curse”) and auctioneer flops. 3G auctions in Germany and the UK yielded enormous profits for the local authorities at the expense of the bidders, whereas in Switzerland, The Netherlands, Italy, and Austria, prices remained well below expectations, disappointing the respective auctioneers.

In IT resource sharing, we want to avoid the winner’s curse. Moreover, these resources are perishable (capacity not used now is worthless in the next moment), needs are dynamic and applications require bundles with multiple units of items (CPU, RAM, permanent storage, network bandwidth). Krishna (2002) states that with these conditions in mind, potentially suitable auction models for IT resources include continuous double auctions, Vickrey, Dutch, multiunit, and multi-item (or combinatorial) auctions. However, individually these approaches alone do not offer a comprehensive and precise price formation solution. The optimality of an auction mechanism will always depend on the particular deployment environment; there are no one-size-fits-all solutions.

Communicate Dynamic Value

Should dynamic value, that is, price, be communicated to users, or should value only be used internally by the resource-sharing system to produce allocations of resources to users? Should users (or their respective managers) pay only a fixed subscription fee for accessing a shared resource space? Isolating users from price dynamics makes sense when users never see—or cause—scarcity, that is when they have low and uncorrelated needs. For example, bread price dynamics at supermarkets have little relation to corn futures markets. On the other hand, electricity companies seek methods to pass intraday price dynamics on to consumers because of the enormous loads consumers produce through correlated responses to events (e.g., extremes of temperature) even though each individual consumes little relative to the capacity of an electricity generator.

Most users of grid infrastructures are heavy resource consumers almost by definition, so dynamic prices must be communicated at some level. Fixed pricing may only make sense in a limited number of cases, for example, in pure P2P file sharing environment with a large population of uncorrelated user demands.

Use Real Money

An issue that concerns the grid community is the definition of a grid currency (see Barmouta & Buyya [2003]). This issue is generally more important for commercial IT-sharing environments that cover many different budget entities than for earlier distributed systems that did not have a clear notion of, or connection with, budget entities. In addition, managers will face the issue of whether they should buy resources on accessible shared spaces or boxes. Managers will also need to decide whether, and how, to make their boxes available to the shared spaces to which their organization is linked. Shared IT resources are typically heterogeneous and potentially of arbitrary scale. Scale and heterogeneity are exactly the drivers which led to the establishment of standard monetary units and currency exchange rates in the real economy.

The administration of a particular shared space may choose to introduce prices for a local artificial currency. The administration must then act as a national bank by guaranteeing the convertibility of the currency into units of value, that is, resources or real money. Now who sets the exchange rates and to which unit of value? A currency board? A fixed exchange rate? IT administrations should quickly choose to skip the intermediate step of an artificial currency with its trust and convertibility problems and use real money straight away. Using a real currency for shared resources additionally brings the following benefits: buy/build/lease or upgrade/retire decisions are simplified and the allocation of IT budgets is directly meaningful.

Guarantee Property Rights

What quality of service (QoS) is required for tradable value and convertibility? Most IT systems today do not support hard QoS guarantees, that is, they do not guarantee specific properties of a service to the user. Often best-effort service is provided. Approaches that go beyond best-effort typically introduce job/packet marking so that different priorities can be assigned to different tasks (Blazewicz, Eaker, Pesch, Schmidt, & Weglarz, 1996; Ferguson & Huston, 1998). How much better the service will be for differentiated service classes is generally hard to determine in advance for large-scale heterogeneous systems and even harder to characterize in absolute terms.

Despite the difficulties of guaranteeing QoS (especially end-to-end), commercialization of shared IT resources requires guaranteed property rights at the level at which pricing is done. Best-effort service has near-zero economic value. In fact the value would be exactly zero if it were not for the assumption that there is a common understanding between the buyer and seller of the service on the quality level to be delivered (see Figure 2).

Advocates of IT resource sharing envision dynamic near-real-time negotiation and provisioning of distributed resources (Benatallah, Dumas, Sheng, & Ngu, 2003). This vision may appear very ambitious at first sight, but it is actually very similar to existing financial and commodity markets. Such markets typically operate at electronic speed and rely on the use of extremely detailed processes and contracts to determine the allocations of large, heterogeneous sets of resources to an equally large and heterogeneous population of users. Complexity is no barrier to value for a good. The definitions of some resources traded on the

Figure 2. Best effort



Chicago Mercantile Exchange (CME) run for many pages, and even then, reference external tests and standards (see *The Chicago Mercantile Exchange*, 2004; *The Chicago Mercantile Exchange Rulebook*, 2004).

Computers and applications may be complex but they also have unambiguous definitions. A high level of detail in contract specifications and a hard guarantee regarding these specifications are necessary elements to create the appropriate confidence among users of a highly distributed cross-organizational system that what they get is exactly what they expected to receive. In some cases, a tradable asset must be described in statistical terms, but it is still feasible to provide hard guarantees in this sense. This has been applied to cycle-harvesting (see Kenyon & Cheliotis [2003]).

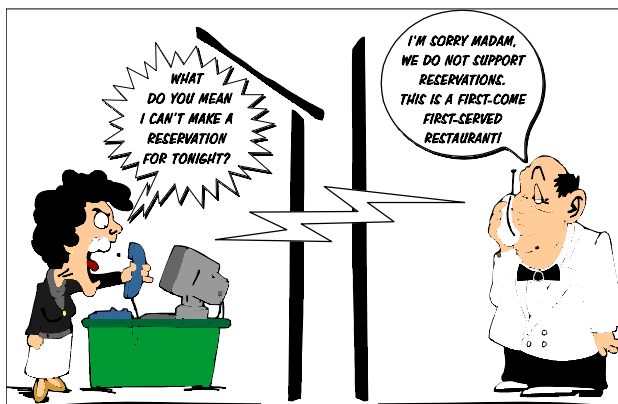
Use Futures Markets

IT resources are generally not storable, in the sense that capacity not used today cannot be put aside for future use. Since the resources cannot be stored, there need be no link between the price for a resource now and the price that the resource (if available) will fetch at any future time (even in 1 second!). Given that it is impossible to build up inventories to smooth out the differences between supply and demand, prices can be arbitrarily volatile (this has been observed in practice for other nonstorable commodities by Pilopovi'c [1998]). To avoid this price volatility and to enable planning and risk management, conventional nonstorable commodities have developed futures markets, that is, markets for reservations.

The most significant non-IT commodity that is also nonstorable is electrical power (with the notable exceptions of hydroelectric and pumped storage). In electricity markets, as in several others for nonstorables (e.g., live cattle, interest rates), contracts for future delivery (forward or futures contracts) are the most used and have much higher trading volumes than those for immediate delivery (spot contracts). The notable electricity market failure in California was directly tied to poor use of forward contracts (see California ISO [2002]; see Hull [2003] for an introduction to futures markets). The experience of electricity markets is clear (e.g., California, UK) and has led to their redesign with the aim to move everything possible off the spot market and onto the futures markets.

IT resources—like any other resource—have dynamically changing value. Given that they are not storable, availability cannot be guaranteed without a formal mechanism, that is, reservations. This availability guarantee acts as a substitute for inventories and helps to prevent unwanted excessive value fluctuations.

Figure 3. No reservations?



The fact that there is work within the Global Grid Forum for supporting advance reservations is encouraging (Roy, 2002; see also Figure 3) since this capability is vital in commercial practice for nonstorable commodities. Some broadly used schedulers also have reservation (or deadline) capability (e.g., Maui’s Advanced Reservation interface) and reservations are common practice in supercomputer environments. However, linking reservations to value and exchangeable reservations between users is largely missing.

Create Incentives

In the period 1999–2001, more than 1,000 new Internet-based business-to-business (B2B) exchanges were set up, according to IDC. Almost all of them failed, not because markets have poor theoretical properties, but because their specific value proposition was unconvincing. This was made manifest in “low liquidity,” that is, no activity.

The success stories in B2B markets are primarily in specialized national or regional exchanges for electricity (e.g., NordPool). These usually have something in common: regulation accompanying persuasion to get people to sign up. Some of the results may be good for everyone, but the startup adoption costs must still be paid. The other area of outstanding B2B exchange success is the traditional financial and commodity markets with their vast turnover. Here the startup costs were paid long ago.

Where does IT sharing and exchange fit into this spectrum of experience? A detailed answer to that question is beyond the scope of this chapter, but certainly the value proposition of cost savings and greater flexibility is generally accepted. Commoditization is also accepted: there are many fewer computer flavors than there are different companies on, say, the New York Stock Exchange. In addition to stocks, a wide variety of standard instruments are traded on commodity, FX, and derivatives exchanges.

A company can decide to migrate to an internal market in the same way that it can decide to outsource. This is an executive decision to be made on business grounds: units may protest for whatever reason, but the needs of the business are the deciding factor. This is equivalent to regulation in the case of electricity markets mentioned above.

Public IT resources exchanges are unlikely in the short to medium term because of complexity of implementation and of the required changes in business processes. Within a single company or a closed group, the prospects for having appropriate incentives to overcome the startup costs and general inertia are much brighter.

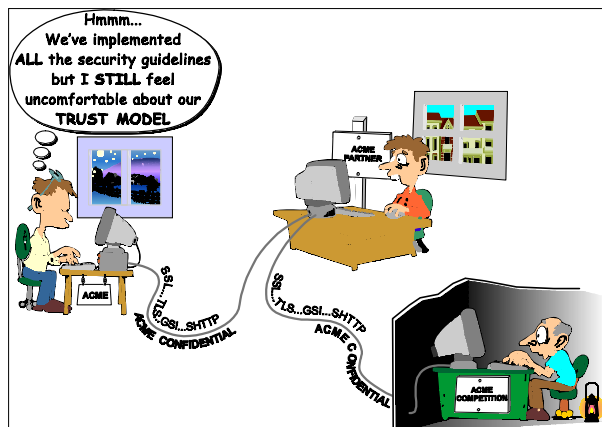
From the examples above, it is clear that practical incentives are vital to promote resource sharing and that incentives are a meta-capability. That is, softwares and systems can support incentive mechanisms but incentive design occurs at a higher business level: the strategic level.

Ensure Trust

What is a good trust model for IT resource sharing? We note that trust is different from security and we are concerned here with trust. Security is just one enabler of trust (see Figure 4).

A good trust model for an online bookstore is not the same as a good trust model for a financial exchange. In fact, online bookstores effectively outsource their trust model to credit card companies for the most part. All the bookstore has to do is provide a basic level of security.

A financial exchange such as the CME has a more complex trust model. First, all transactions on the exchange between different parties are guaranteed by the exchange, not by the individual parties. Thus, the traders only need to trust the exchange to do business, not each other. On the other hand, the exchange trusts the traders because it monitors their actions and requires them to provide a (cash-equivalent) deposit, which is a function of the risk that each trader represents to the exchange for default. That is, the exchange trusts the traders because it has their money. All other people wishing to trade must do so via the traders. Thus, we see a two-tier trust model with distributed risk.

Figure 4. *Trusted or just ... secure?*

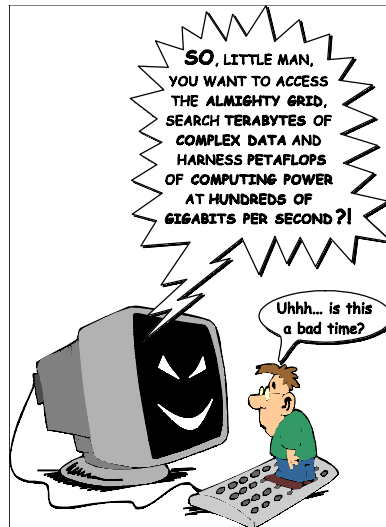
Systems without proportional consequences do not engender trust: this is why contracts exist and are specific and clear methods to invoke financial and legal penalties for improper actions. IT-sharing paradigms require trust models adapted to their environments (e.g., single company, group, etc.). The capability required to deliver trust is a system of proportional consequences for breaches of trust, both in terms of magnitude and, more important, time scale.

Support Process Tools

A typical portfolio directed by a fund manager can easily run to hundreds of stocks selected according to maximizing a specific objective and limited by equally precise constraints on number of stocks to hold, position limits, cash flow obligations that must be met on specific dates, hedging against worst-case scenarios, and so forth. Financial firms do not optimize their resource allocations (portfolios) by hand. They use sophisticated processes to support their decisions; there is an extensive literature that has grown up since the original work on portfolio theory more than 40 years ago by Markowitz (1959) and Birge and Louveaux (1997).

The dynamic system enabled and embodied by resource-sharing paradigms for a typical large company is a significant challenge for users to be able to exploit efficiently and economically. Without sophisticated tools users will find that the more potentially useful and flexible the system is, the less practically usable it will be (see Figure 5). The scarcest resource for many users is attention: they already

Figure 5. User in need of process tools



have jobs and do not want to be bothered with unnecessary details of resource allocation systems.

The process tools for resource allocation must enable users to express their resource needs in a simple way together with the users' uncertainties about these needs over time. They should also enable resource trading (buying and selling of blocks of time and capacity) and capture the effective price dynamics of both spot and futures prices together with changing availabilities. However, the users should not be bothered by these details. Building such tools which integrate budgets and business objectives with resource allocation decisions may seem overly ambitious but it is a situation that is tackled every day for fund managers balancing opportunities and obligations over time. The technical area that these tools build from is multistage stochastic optimization that is generally applicable to large-scale resource systems whether the underlying resources are financial or IT (see Neftci [2000]; Markowitz [1959]; Birge & Louveaux [1997]).

Design for Value

When making a business case for the adoption of a resource-sharing technology, one commonly involves the following arguments: increased utilization, cost savings, greater allocation flexibility, feasibility of previously impossible computational tasks, and so forth. These benefits may be theoretically possible, but to what extent can an organization practically realize these potential values?











































To achieve economic objectives, *laissez-faire* alone is not enough. As mentioned in the previous section on futures markets, a market and resource product structures require engineering to achieve results. Economic engineering for resource sharing covers two main areas: market design and product design. The following questions are just a small selection of the economically significant market design issues. Should the IT department of an organization be operated as a profit or as a cost center? How much reselling of resources should be allowed? Should short-selling be allowed? Is speculation permitted? What level of financial and project risk are users and departments permitted to take?

Product design is just as important as market design in practical systems. For simplicity, it can be important for most of the usual user and manager needs to be available directly as products rather than having to be built each time they are needed. Spot and forward contracts (reservations) can be useful decompositions for describing and controlling the theoretical basis of value. Alternatively these contracts can be automatically assembled into products to match user, application, and department requirement profiles using process tools. For example, a requirement profile could be described using a stochastic process, thus capturing both variability over time and the uncertainty of this variability. Birge & Louveaux (1997) state that sets of resource requirements could be expressed as a portfolio of liability profiles and resources allocated through the application of standard multistage stochastic portfolio optimization techniques. However the markets and the products are designed, they need to be crafted to ensure that the maximum value is realized from the underlying resources and business context. This design capability is vital to deliver confidence to executive decision makers for implementation and subsequent improvements to their bottom line in practice.

Maturity of Proposed Capabilities

We qualitatively assess the maturity of the capabilities required to benefit from each of the lessons using a five-stage framework derived from the CMM (see Software Engineering Institute [2004]). This is not a formal quantification but rather the personal appreciation of the authors derived from our experience and discussions around the tutorial on this subject we gave at GGF5 (see Cheliotis & Kenyon [2002]). There are two differences between a typical use of CMM and our approach: (a) in practice, CMM analysis often focuses on the capabilities of a particular organization whereas we assess maturity levels in the broader field of IT sharing and (b) CMM levels typically refer to the maturity of software development processes whereas we focus on the maturity of IT-sharing processes/capabilities derived from our 10 lessons. We, therefore, need to redefine these levels for the context of this chapter. CMM defines five levels

Table 2. Framework for assessing capability levels

CMM-BASED ANALYSIS			 Common practice  No significant evidence of existence				
Lessons	Required Capabilities		Assessment of maturity level				
	Finance View	Engineering View	Initial	Repeatable	Defined	Managed	Optimizing
• Avoid the tragedy of the commons	• Charging and accounting for use of resources	• Congestion control					
• Discover dynamic value	• Auctions, dynamic pricing	• Real time system monitoring with feedback to users					
• Communicate dynamic value							
• Use real money	• Link to accounting systems	• Usage credits linked to financial accounting system					
• Guarantee property rights	• Firm contracts	• Guaranteed QoS					
• Use futures markets for non-storable assets	• Forward contracts, futures markets	• Advance reservations, exchange mechanism					
• Create incentives	• n/a	• n/a					
• Ensure trust with proportional consequences	• Financial guarantees, active risk monitoring	• Fail-safe trust mechanism					
• Support process tools for users	• Stochastic optimization and pricing	• User friendly advanced task scheduling					
• Design for value	• Market design and simulation	• System simulation and design					

which we adopt, label, and (re)define below: initial, repeatable, defined, managed, and optimizing.

1. Initial: Basic elements of the capability have been observed in practice.
2. Repeatable: it is possible to reliably repeat the implementation of the capability, and there is a common body of work supporting the understanding of the capability.
3. Defined: There is a defined process for implementing the capability and industry or de facto standards exist.
4. Managed: The processes of implementing and maintaining the capability can be managed in terms of time-varying desired objectives and requirements.
5. Optimizing: The processes of implementing and maintaining the capability can be continuously optimized in terms of the desired objectives and requirements.

Our assessment is given in Table 2, which can be used to identify gaps in current capabilities, as a management checklist for implementation features, and to help practitioners prioritize new technical developments. We will give brief comments on each of the capability assessments but not an in-depth review, as this would be beyond the scope of the current chapter.

- **Avoid the tragedy of the commons.** Congestion of commonly accessible resources is well recognized, and current grid and cycle-harvesting software have support for increasingly sophisticated policies covering resource usage (e.g., in Sun's Grid Engine or in Platform Computing's LSF products). However, there is still very little support for policy design in terms of desired outcome versus stochastic demand models (which are not supported either). In this sense sharing software is well behind, say, network design tools.
- **Discover and communicate dynamic value.** Price discovery mechanisms are almost absent from the IT resource sharing space. Whilst top-down allocation is possible via policies, the ability for users to express their urgency and for systems to provide real-time feedback has only been seen in some research demonstrations.
- **Use real money.** The authors are aware of only a very small number of resource-sharing systems where resource usage (or reservation) is directly tied to accounting systems (although we cannot mention company names at this point), but this is a long way from common practice. Selling resources, for example, supercomputer time, for money is a well-established practice, as is monitor-and-bill for variably priced outsourcing contracts. This knowledge and practice is not yet integrated into resource-sharing software.
- **Guarantee property rights.** Guaranteed QoS for allocated resources is present on very few systems, mostly mainframes and supercomputers (where this is provided in the operating systems, for example, IBM zOS for z900 series, or alternatively provided at a machine-level granularity). Policy-based systems can generally support a basic version of property rights for certain privileged users. Management of rights in order to achieve objectives is largely absent.
- **Use futures markets.** The first operational futures market in computer time was a manual system in 1,968 states (Sutherland, 1968), but little has been implemented in practice since then apart from certain research demonstrations. Ernemann and Yahyapour (2004) state that there is some understanding of how to run futures markets for IT resources but this is far from common practice.

- **Create incentives.** The incentive structure around resource sharing is, in our view, a meta-capability in that it is implemented based on the careful design of a business model. Incentives are not inherent in software/system mechanisms as the many hundreds of failed business-to-business and business-to-consumer auction sites demonstrated in the dotcom era.
- **Ensure trust.** In a commercial environment, trust is based on legal and financial consequences, both immediate (penalties) and deferred (reputation, litigation). Whilst there is a body of work around reputation in agent economies, for example, Xiong and Liu (2003), this has yet to be applied to resource sharing except in some limited P2P examples emphasizing fairness. There is no equivalent of the real-time, risk-adjusted margin accounting of financial exchanges (see Section “Ensure Trust”).
- **Support process tools.** The need for user-friendly interfaces is well understood in all IT-sharing paradigms with a range of answers from portals to dedicated software. However, the process tools contained in the interfaces are generally limited, especially in terms of achieving business objectives.
- **Design for value.** Both IT professional and financial experts understand the need to design systems to achieve financial goals and there are some, generally proprietary, tools to assess the return on investment from installation of different IT-sharing technologies (e.g., from IBM and Platform Computing). However there is no generally accepted process for this assessment and certainly no tools available for designing how a sharing system should run in terms of financial objectives.

Summary and Conclusion

Past developments in the IT resource-sharing field have been primarily technology driven. In those cases where needs were the driving force, it has been mostly the needs of scientists (grids) or home users (P2P). This modus operandi has produced a plethora of inventive-sharing mechanisms, but corporate IS managers still do not know how their organizations can derive value from these technologies. This should not come as a surprise, as resource-sharing mechanisms per se are indeed of little business value in the absence of tools that will assist companies with the implementation of a suitable sharing strategy.

From our analysis of IT-sharing software and systems in Table 2, it is clear that there is a large gap between current practice and the potential for value in this area. By value we mean the realization of potential business benefits from the efficient use and dynamic allocation of resources. The biggest gaps are around

the quantification of the value and designing systems so that maximum value is achieved in practice. That both are missing together is not surprising: it is hard to maximize value if you have no means of making it visible and quantifying it. Thus, the lowest-hanging fruit is in making value visible to users and allowing users to express their urgency back. Once value can be seen, then it can be maximized, and the other lessons regarding design (i.e., use of reservations) and the use of real money can be applied in a meaningful way, assuming that a viable incentive structure is in place.

References

- Barmouta, A., & Buyya, R. (2003). Gridbank: A grid accounting services architecture (gasa) for distributed systems sharing and integration. *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS'03)*, 245.
- Benatallah, B., Dumas, M., Sheng, Q.Z., & Ngu, A.N.H. (2002). Declarative composition and peer-to-peer provisioning of dynamic web services. *Proceedings of 18th International Conference on Data Engineering*, 297.
- Birge, J. & Louveaux, F. (1997). *Introduction to stochastic programming*. New York: Springer.
- Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., & Weglarz, J. (1996). *Scheduling computer and manufacturing processes*. Springer-Verlag.
- California ISO. Amendment 44 Docket Nos. EL00-95-001 and ER02-1656-000 (Market Design 2002). Retrieved March 4, 2004, from <http://www.caiso.com/docs/2002/05/29/200205290858531076.html>
- Cheliotis, G., & Kenyon, C. (2002, July 25). Grid economics. *Tutorial, Global Grid Forum 5, Edinburgh*. Retrieved March 4, 2004, from <http://www.zurich.ibm.com/grideconomics/refs.html>
- The Chicago Mercantile Exchange. Retrieved March 4, 2004, from <http://www.cme.com>
- The Chicago Mercantile Exchange Rulebook. Retrieved March 4, 2004, from <http://www.cmerulebook.com>
- Ernemann, C., & Yahyapour, R. (2004). Applying economic scheduling methods to grid environments. In J. Nabrzyski, J. Schoph, & J. Weglarz (Eds.), *Grid resource management: State of the art and future trends* (pp. 491–507). Kluwer.

- Ferguson, P., & Huston, G. (1998). Quality of service on the Internet: Fact, fiction or compromise? *Proceedings of Inet 98*. Retrieved March 4, 2004, from http://www.isoc.org/inet98/proceedings/6e/6e_1.htm
- Foster, I., Kesselman, C., Nick, J., & Tuecke, S. (2002, June 22). The physiology of the grid: An open grid services architecture for distributed systems integration. Retrieved March 4, 2004, from <http://www.globus.org/research/papers.html#OGSA>
- gnutella2.com. The Gnutella 2 specification. Retrieved March 4, 2004, from [http://www.gnutella2.com/tiki-index.php?\(page=Gnutella2\)](http://www.gnutella2.com/tiki-index.php?(page=Gnutella2))
- Hull, J. (2003). *Options, futures, and other derivatives* (5th ed.). Prentice Hall.
- Kamkar, M., Shahmehri, N., Graham, R.L., & Caronni, G. (2003). Third international conference on peer-to-peer computing (P2P'03).
- Kenyon, C., & Cheliotis, G. (2003). Creating services with hard guarantees from cycle-harvesting systems. *Proceedings of CCGrid 2003*.
- Klemperer, P. (1999). Auction theory: A guide to the literature. *Journal of Economic Surveys*, 13(3), 227–286.
- Klemperer, P. (2002). How (not) to run auctions: The European 3G telecom auctions. *European Economic Review*. Retrieved March 4, 2004, from <http://www.paulklemperer.org>
- Krishna, V. (2002). *Auction theory*. New York: Academic Press.
- Mankiw, N. (1997). *Principles of economics*. Dryden Press.
- Markowitz, H. (1959). *Portfolio selection: Efficient diversification of investments*. New York: John Wiley & Sons.
- Neftci, S. (2000). *An introduction to the mathematics of financial derivatives* (2nd ed.). New York: Academic Press.
- Pilopović, D. (1998). *Energy risk: Valuing and managing energy derivatives*. New York: McGraw-Hill.
- Roy, A., & Sander, V. (2003, April 30). Advance reservation API. GGF draft. Retrieved March 4, 2004, from <http://www.gridforum.org>
- Shapiro, C., & Varian, H.R. (1999). *Information rules: A strategic guide to the network economy*. Boston: Harvard Business School Press.
- Software Engineering Institute. Capability maturity model. Retrieved March 4, 2004, from <http://www.sei.cmu.edu/about/about.html>
- Sutherland, L. (1968). A futures market in computer time. *Communications of the ACM*, 11(6).
- Thain, D., Tannenbaum, T., & Livny, M. (2002). Condor and the grid. In F. Berman, G. Fox, & T. Hey (Eds.), *Grid computing: Making the global infrastructure a reality*. New York: John Wiley & Sons.

- Vishnumurthy, V., Chandrakumar, S., & Sirer, E.G. (2003). Karma: A secure economic framework for peer-to-peer resource sharing. *Workshop on Economics of Peer-to-Peer Systems. Berkeley, CA*. Retrieved March 6, 2004 from, <http://www.sims.berkeley.edu/research/conferences/p2pecon/papers/s5-vishnumurthy.pdf>
- von Weizsäcker, C.C. (2003). The Swiss UMTS auction flop: Bad luck or bad design? In H. Nutzinger (Ed), *Regulation, competition, and the market economy* (pp. 281–293). Göttingen, Germany: Vandenhoeck & Ruprecht.
- Xiong, L., & Liu, L. (2003). A reputation-based trust model for peer-to-peer e-commerce communities. *Proceedings of IEEE Conference on E-Commerce (CEC'03)*.

Author Note

- * McKinsey & Company, Business Technology Office, Switzerland