Work as a Service

Daniel V. Oppenheim, Lav R. Varshney, and Yi-Min Chee

IBM Thomas J. Watson Research Center, Hawthorne NY 10532, USA
{music,lrvarshn,ymchee}@us.ibm.com

Abstract. Improving work within and among enterprises is of pressing importance. We take a services-oriented view of both doing and coordinating work by treating *work as a service*. We discuss how large work engagements can be decomposed into a set of smaller interconnected service requests and conversely how larger engagements can be built up from smaller ones. Encapsulating units of work into service requests enables assignment to any organization qualified to service the work, and naturally lends itself to ongoing optimization of the overall engagement.

A service request contains two distinct parts: coordination information for coordinating work and payload information for doing work. Coordination information deals with business concerns such as risk, cost, schedule, and value co-creation. On the other hand, payload information defines the deliverables and provides what is needed to do the work, such as designs or use-cases. This general two-part decomposition leads to a paradigm of work as a two-way information flow between service systems, rather than as a business process that needs to be implemented or integrated between two organizations.

Treating work as information flow allows us to leverage extant understanding of information systems and facilitates information technology support for work using mainstream service-oriented architectures (SOA). Significant benefits from this approach include agility in setting up large engagements to be carried out by distributed organizations, visibility into operations without violating providers' privacy or requiring changes to internal processes, responsiveness to unpredictability and change, and ongoing optimizations over competing business objectives.

Keywords: Work, encapsulation, service, decoupling, information flow.

1 Introduction

Differences among labor pools globally, rapid proliferation of capacious information technology infrastructures, and increased churn due to a millennial generation that is project-based rather than jobs-based [6] has disrupted the nature of work in many institutions, causing increased decentralization of workforces and increased leverage of communities, networks, and ecosystems of people and of firms to do work. These business, technological, and social trends have intensified interest in general ways of structuring the coordination and doing of work.

The fundamental problem of doing work is to transform inputs into outputs to meet specified requirements. For human tasks, work systems can be individuals

G. Kappel, Z. Maamar, H.R. Motahari-Nezhad (Eds.): ICSOC 2011, LNCS 7084, pp. 669–678, 2011.

or groups that may be distributed within or among organizations. But most work required by businesses is complex. The problem then becomes how to translate a business need, perhaps expressed as a service request, into an optimal decomposition of units of work and how to optimally *coordinate* its execution.

The basic problem of coordinating work is to decompose a service request into units that can be assigned to a set of work systems, provide the necessary inputs and requirements to the work systems, and aggregate their outputs, while continuously responding to changing conditions. Often there are dependencies among work assigned to different work systems. The work systems may have conflicting local objectives and may perform work with differing costs, schedules, and reliabilities. Optimal coordination must take these factors into account.

In this paper we treat *work as a service* (WaaS). Doing of work is encapsulated as a service request and coordination of work involves routing service requests to work systems. Within the WaaS paradigm, large work engagements can be decomposed into a set of smaller interconnected service requests and conversely larger work engagements can be built up from smaller ones.

An encapsulated service request contains two distinct parts: coordination information for coordinating work and payload information for doing work. Coordination information deals with business concerns such as risk, cost, schedule, and value. Payload information defines deliverables and provides what is needed to do the work, such as designs or use-cases. This general two-part decomposition leads to a paradigm of work as a two-way information flow between work systems, rather than as a business process that needs to be implemented or integrated between two organizations.

Encapsulation eliminates search inefficiencies on inputs, requirements, and formats for work systems. But more importantly, it enables assignment of work requests to any qualified work system, leading naturally to ongoing optimization of the overall engagement in response to unpredictable system dynamics. Coordination becomes a problem of dynamically routing information flow.

By treating work as information flow, several patterns of and organizational structures for doing work can be treated in a common framework.

Since work is encapsulated as service requests, mainstream service-oriented architectures (SOA) can be used to provide information technology support.

As demonstrated in the sequel, significant benefits from this approach include agility in setting up large engagements to be carried out by distributed work systems, visibility into operations without violating providers' privacy or requiring changes to internal processes, responsiveness to unpredictability and change, and ongoing optimizations over competing system-level business objectives.

2 The Changing Nature of Work and Workforce

The combined force of the trends described earlier has been the emergence of new models of work including: globally dispersed teams in firms [18], outsourcing, crowdsourcing [20], information factories [8], virtual enterprises [13], cross-enterprise collaborations [17], open source, social production [5], and asset reuse [2].

With these new models, there is greater division of labor and workforce specialization, but traditional coordination mechanisms such as mutual adjustment through informal communication [14] are no longer effective [11]. The trade-off between specialization benefits and coordination costs are well known [4], but formal mechanisms may reduce these costs without reducing benefits from specialization. The goal of the WaaS paradigm is precisely this.

As will be evident, encapsulation of work makes it procedurally equivalent to plug in any work system, whether a crowd, partner organization or combination of several work systems. This is in contrast to business process management (BPM) approaches, where recombining the doing of work requires a new business process to connect the pieces together, a provably complex undertaking [15]. Further BPM models do not lend themselves to many optimizations [25], whereas the information flow paradigm herein readily supports optimization.

Notwithstanding, the work for a single encapsulated work request may be carried out using BPM. Further, it is possible to use fulfillment of work requests as signals to transition between states in business process models.

3 Work as a Service (WaaS) Encapsulation

In this section, we describe the essential aspects of the WaaS encapsulation and the resultant information flow paradigm.

As depicted schematically in Fig. 1(a), a work engagement consists of essentially three parts:

1. A *requestor*, which is a service system that requests work to be done, provides inputs, and specifies the requirements.
2. A *provider*, which is a service system charged with fulfilling the work request to meet requirements.
3. An *encapsulated service request*, which captures the interaction between the requestor and provider, the two-way information flow among them.

There are three aspects of a work task that must be established between the requestor and the provider. First, a so-called service level agreement (SLA) must be reached, which specifies business-level properties, such as cost and schedule. Second, as work is ongoing, there may be monitoring by the requestor of partial results and checkpoints achieved by the provider, so as to have appropriate visibility. Third, at the conclusion of a work engagement, the provider sends deliverables to the requestor who either confirms or rejects the delivered work. Note that all three involve two-way flows of information.

These two-way communications that arise should all be captured in the encapsulated work request. We define a general two-part decomposition of work into business concerns and domain concerns. These two parts are called *coordination information* and *payload information*, respectively, and are depicted schematically in Fig. 1(b). Coordination mechanisms can restrict attention to the former part whereas work systems can restrict attention to the latter.

(a) (b)

Fig. 1. (a) Work as an encapsulated service request with requestor R and provider P. (b) The encapsulated request partitioned into coordination and payload information.

In detailing the contents of coordination and payload information, it is easiest to first consider an *atomic service request*: an encapsulation of a unit of work so small that it cannot be broken into pieces. We will later see how to combine and recombine atomic service requests into *molecular service requests*.

3.1 Coordination Information

What information about an atomic service request is needed for coordination? Since the goal of coordination is to satisfy concerns, we enumerate several possible business concerns that arise in work. Note that coordination may be done by the requestors and providers themselves or by an external agent.

A first consideration is schedule: how long will it take for an atomic piece of work to be done. Note that this varies across different work systems and is also potentially stochastic. A second consideration is cost: how much money will the provider charge the requestor to do work and whether there are bonuses or penalties associated with speed or quality [3]. Although the encapsulation formalism is eminently amenable to outcomes-based pricing [21] rather than effort-based pricing, the cost may still have some variability as well. A third consideration is quality: how good will the deliverable be with respect to the requirements. One way to certify the quality of a work system is through the use of CMMI level—higher levels imply more stringent process and quality control.

Each of these concerns can be captured as a mapping from the Cartesian product of the possible set of work systems and the possible set of work tasks to the space of random variables that represent time, cost, or quality. As milestones are reached and partial results are achieved, the probability distributions can be updated with new information that is furnished by the provider. Exogenous perturbations to the system such as natural disasters might also change these distributions, as can changes in requirements imposed by the requestor.

In reality, schedule, cost, and quality are very much intertwined. For example, loosening schedules may reduce costs. These competing business concerns can be balanced through the notion of *value*, which is what should be optimized. The value-dominant logic brings this point out even further [23], and indeed we feel that the encapsulated service request is the seat of value co-creation.

For molecular service requests, coordination information needs to also contain the interdependencies among its atomic constituents. This includes not only things like the fact that one piece of work needs to be done before another, but also inertia effects and other factors, cf. [26].

3.2 Payload Information

We now ask what information is needed by a work system to do work. Broadly, payload information should include the inputs that are to be transformed into outputs and the requirements that specify what is to be done. This should be the minimal sufficient information for doing work.

More specifically, for software development, payload information may include APIs, architectural diagrams, and requirements documents; for engine design, it may include specifications of mechanical, hydraulic, and electrical interfaces, performance requirements and CAD language. The WaaS encapsulation is designed to be general, supporting the needs of any specific domain.

As the lifecycle of the work request proceeds, payload information is updated based on partial results and milestones. Technological developments that impact the doing of work, or changes in requirements would can evolve payload information. Fixes for errors made in execution may also be incorporated.

3.3 Information Flow

The WaaS paradigm may be interpreted as an information flow description. One can think of the encapsulated service request as a multidimensional variable that captures the current state of things, including the value being generated. As things happen, information flows to the service request for it to be updated. Updates to both payload and coordination information happen continuously, capturing both business and domain concerns.

Coordination mechanisms can also be thought of in informational terms as routing. Essentially, coordination involves connecting requestor and provider together to interact through an encapsulated service request. Moreover, as in Sec. 4, large work engagements can be constructed by routing several service requests within a work ecosystem. Coordination must then consider governance issues such as accountability, responsibility, and decision-making rights. There may be different organizational structures for coordination. See Sec. 5.

4 Patterns and Structures

Now we discuss the structural building blocks that enable composition and decomposition of encapsulated service requests to form large work engagements. Several canonical patterns emerge, which are tied to various organizational structures that arise in businesses; cf. Malone's notions of flow, sharing, and fit [12, p. 140]. This demonstrates that the WaaS paradigm applies to its targeted problem space of complex work within or between organizations.

First consider delegating work from one work system to another, as in Fig. 2. This may be done, e.g. if the original provider is overloaded. The original provider becomes a requestor (delegator) for the downstream provider. Payload information and the interdependency portion of coordination information are copied

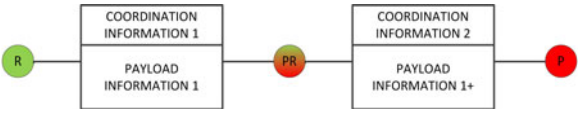


Fig. 2. Delegation of work by re-routing an encapsulated service request. The original provider becomes a requestor for the downstream provider.

essentially unchanged to the new re-routed service request, perhaps adding information useful to the new provider. The remaining coordination information, however, is written anew to capture the business concerns of the delegator and to hide the business concerns of the original requestor, which are not of direct relevance to the new provider. The delegator remains accountable and responsible to the original requestor, but the new provider is only responsible and accountable to the delegator. Note that the original request may specify that when delegated some coordination information must also flow downstream.

Another possible pattern of work is to tear a molecular service request into pieces and re-route them to several producers. Payload information is partitioned into (possibly overlapping) pieces with minimal sufficient information required to do the newly reconstituted work. The interdependency portion of the coordination information is also partitioned, and remaining coordination information is written anew. Recursive hierarchical tearing can also be done.

Tearing should be done for specialization gains, or when work systems can be leveraged in parallel. For example, if a general service provider has several sourcing channels such as a crowd and a factory, different pieces might be routed to different places. Note that tearing requires PR to monitor and eventually integrate or aggregate the completed work so as to be able to respond to R.

A third pattern is to merge several service requests (from one or more requestors) into one. The payload information of the merged request is simply the union of the payloads of the requests being merged. Coordination interdependencies must be combined with any new interdependencies that arise. Remaining coordination information is written afresh, typically meeting the minimum specifications of the original requests. Hierarchical merging can also occur.

Economies of scale are a prime motivator for merging. For example consider several requests to perform environmental testing for electronics where a single cold room could be used simultaneously for all the requests.

Another kind of re-routing that can arise is to withdraw a service request from one provider and assign it to another. This could happen if changing conditions prevent the original provider from completing the service request.

The basic operations described above can be used in combination to generate other structures. As an example, consider a coordination hub [16], a centralized authority charged with coordinating work to derive maximal value, by taking work from several requesters, tearing and merging, and then delegating to several providers, while responding to changing conditions. A hub can be thought of as

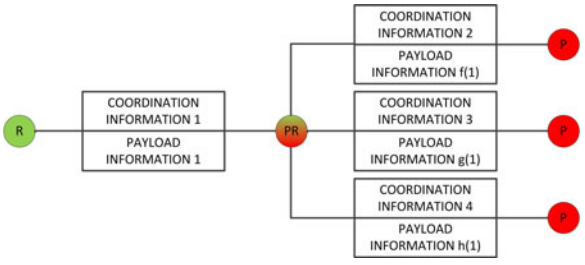


Fig. 3. Tearing and delegating work by re-routing encapsulated service requests. The original provider aggregates and becomes a requestor for downstream providers.

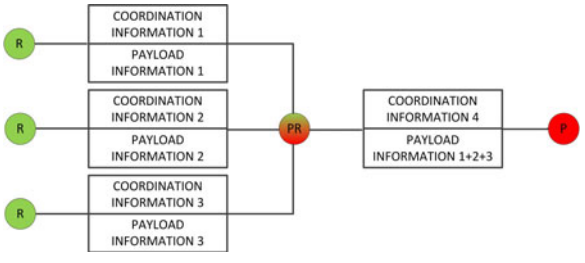


Fig. 4. Merging and delegating work by re-routing encapsulated service requests. The original provider becomes a requestor for the downstream provider.

a kind of intermediate delegator. Cross-enterprise collaboration, when several organizations partner to do work, may take the form of a hub [3], as may robust supply chain collaboration [19].

Formal proof aside, it should be clear that arbitrary topologies can be constructed using the basic building blocks and operations we have defined.

5 Coordination and Governance

We have discussed how service requests can be combined and recombined in various patterns, but it is still unclear how to initially make the plan for doing so or in response to change. That is the purpose of a coordination mechanism.

Initial coordination involves appropriately tearing and merging requests, and then determining routes for work to assign it to qualified producers; unlike communication networks the source and destination are not pre-specified.

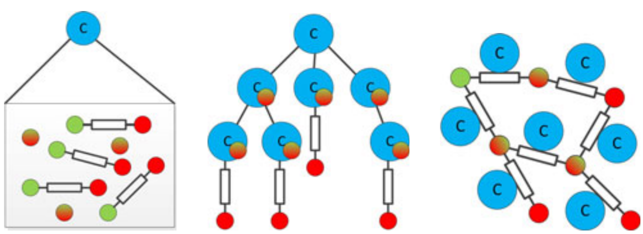
Over time as more information becomes available, the work plan may need to be modified and requests re-routed, whether due to environmental events, or updates in the work lifecycle itself or in interdependent tasks and systems.

Coordination mechanisms may be implemented with an automatic program in SOA built on a protocol like WS-coordination, a human program manager or program executive, or a governance council in cross-enterprise collaboration.

Any of these, however, require both access to the coordination information in the encapsulated work requests and the ability to make re-routing decisions, which are matters of governance and organization. By requiring coordination information to be freshly written, the WaaS paradigm naturally limits information to the requester and provider who clearly need it. But when required, governance policies may provide a window to other work systems.

Many possible visibility and governance policies exist, as in Fig. 5. In a centralized hub with complete visibility, responsibility, and decision-making authority [17], a globally optimal coordination scheme can be used [3]. If there is hierarchical authority, as in a globally integrated enterprise [18], visibility and decision-making authority is restricted to one level depth in the tree and coordination mechanisms must respect this. In a fully decentralized governance structure, each pair of service systems is responsible for their own coordination.

Due to differing visibility and authority for re-routing, these various forms of coordination have different abilities to react when conditions change.



(a) (b) (c)

Fig. 5. Coordination, with coordinators C. (a) central coordinator with full visibility and authority into all encapsulated service requests. (b) globally integrated enterprise with hierarchical visibility and authority. (c) decentralized organization with localized visibility and authority.

6 Agility, Optimization, and Innovation

Having discussed the doing of work and the coordination of work within the WaaS framework, we discuss some beneficial attributes of this paradigm.

Due to the block-building nature of WaaS, there is agility in setting up large engagements involving work systems that may be globally distributed within or across enterprises. Since service requests contain sufficient information for work, any admissible work system of any kind, whether a crowd or a virtual enterprise, can be plugged in as a provider. Due to encapsulation, how work is done matters little; only what work is done and the collaboration induced between the requestor and provider. As such, business processes need not be integrated nor internal processes changed. Moreover, agility extends to response to change; reconfiguration by re-routing is as easy as initial setup. The recombinant nature of the encapsulated service requests provides exponential flexibility.

A second thing to note about the WaaS paradigm is that visibility is provided into operations without violating providers' privacy. Since the payload information in the encapsulated service request is the *minimal sufficient statistic* needed to do work, by the data processing inequality in information theory [9], it minimizes information leakage. When service requests are delegated, coordination information is not passed to the new provider, again preserving privacy.

Within the WaaS paradigm, optimizing value becomes a partitioning and routing question, but with destination also subject to choice. In particular, determining how to restructure and then re-route service requests is rather similar to the routing problem faced by packet-switched communication networks like the internet. Centralized hub optimizations are essentially equivalent to nonlinear multicommodity flow problems with a further optimization for balancing destinations, cf. [3], for which optimal flows can be found in polynomial time [7]. When performing distributed coordination, as in Fig. 5(c), efficient routing algorithms developed for internet protocols can be adapted.

When BPM approaches are used, optimal coordination becomes a scheduling problem rather than a routing problem; optimal scheduling problems are often NP-hard [22]. Although routing and scheduling are rather similar, the computational complexity of finding an optimal solution can be different.

When design complexity is low, as with small loosely coupled service requests, innovation is easier due to increased opportunities to experiment [1].

7 Concluding Remarks

We presented a new way for describing work as an information flow and then defined the underlying formalisms that enable the decomposition of requests into fine-grained units that can be coordinated and optimized over competing business, customer, provider, and resource objectives. We further demonstrated how this model generalizes over disparate models of work and can be utilized to support different patterns of organization, business, and governance.

Our approach in this paper has been to describe the basic structural elements and decompositions in the WaaS paradigm. Moving forward, detailed study of optimal coordination mechanisms that really make WaaS go is necessary. The role of uncertainty should take greater prominence [24]; after all, "Uncertainty is what typifies projects. It's the nature of the beast" [10].

One of the biggest revolutions in the evolution of multi-cellular organisms occurred when neurons emerged. Before neurons, cells had to be very close to each other to coordinate their functions. After neurons, cells could communicate from a distance and ongoing two-way flow of information became central to complex life. This allowed cells to be rearranged and assigned different functions in a wide variety of life forms. The information flow paradigm for work developed herein may similarly allow an expansion in the variety of economic and organizational forms that are then able to efficiently fill a wide variety of niches.

References

1. Auerswald, P., Kauffman, S., Lobo, J., Shell, K.: The production recipes approach to modeling technological innovation: An application to learning by doing. J. Econ. Dyn. Control 24, 389–450 (2000)
2. Bacon, D.F., Bokelberg, E., Chen, Y., Kash, I.A., Parkes, D.C., Rao, M., Sridharan, M.: Software economies. In: Proc. FSE/SDP Workshop Future Softw. Eng. Research (FoSER), pp. 7–12 (2010)
3. Bagheri, S., Oppenheim, D.V.: Optimizing cross enterprise collaboration using a coordination hub. In: Proc. SRII 2011 Global Conf. (2011)
4. Becker, G.S., Murphy, K.M.: The division of labor, coordination costs, and knowledge. Quart. J. Econ. 107, 1137–1160 (1992)
5. Benkler, Y.: The Wealth of Networks. Yale University Press, New Haven (2006)
6. Bollier, D.: The Future of Work. Aspen Inst., Washington (2011)
7. Cantor, D.G., Gerla, M.: Optimal routing in a packet-switched computer network. Comput. C-23, 1062–1069 (1974)
8. Chaar, J.K., et al.: Work packet delegation in a software factory, Patent Application Publication US 2010/0031226 A1 (2010)
9. Cover, T.M., Thomas, J.A.: Elements of Information Theory. John Wiley & Sons, New York (1991)
10. Goldratt, E.M.: Critical Chain. North River Press (1997)
11. Gumm, D.C.: Distribution dimensions in software development projects: A taxonomy. IEEE Softw. 23, 45–51 (2006)
12. Malone, T.W.: The Future of Work. Harvard Business School Press (2004)
13. Mehandjiev, N., Grefen, P.: Dynamic Business Process Formation for Instant Virtual Enterprises. Springer, London (2010)
14. Mintzberg, H.: Mintzberg on Management. Free Press, New York (1989)
15. Norta, A.H.: Exploring Dynamic Inter-Organizational Business Process Collaboration. Ph.D. thesis, TU-Eindhoven (2007)
16. Oppenheim, D., Bagheri, S., Ratakonda, K., Chee, Y.M.: Coordinating Distributed Operations. In: Maximilien, E.M., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6568, pp. 213–224. Springer, Heidelberg (2011)
17. Oppenheim, D.V., Bagheri, S., Ratakonda, K., Chee, Y.M.: Agility of enterprise operations across distributed organizations: a model of cross enterprise collaboration. In: Proc. SRII 2011 Global Conf. (2011)
18. Palmisano, S.J.: The globally integrated enterprise. Foreign Aff. 85, 127–136 (2006)
19. Tang, C.S.: Robust strategies for mitigating supply chain disruptions. Int. J. Logist. 9, 33–45 (2006)
20. Tapscott, D., Williams, A.D.: Wikinomics. Portfolio Penguin, New York (2006)
21. Tiwana, A.: Does technological modularity substitute for control? a study of alliance performance in software outsourcing. Strateg. Manage. J. 29, 769–780 (2008)
22. Ullman, J.D.: NP-complete scheduling problems. J. Comput. Syst. Sci. 10, 384–393 (1975)
23. Vargo, S.L., Lusch, R.F.: Evolving to a new dominant logic for marketing. J. Mark 68, 1–17 (2004)
24. Varshney, L.R., Oppenheim, D.V.: Coordinating global service delivery in the presence of uncertainty. In: Proc. 12th Int. Research Symp. Service Excellence Manage, QUIS12 (2011)
25. Vergidis, K., Tiwari, A., Majeed, B.: Business process analysis and optimization: Beyond reengineering. IEEE Trans. Syst., Man, Cybern C 38, 69–82 (2008)
26. Wiredu, G.O.: A framework for the analysis of coordination in global software development. In: Proc. Int. Wksp. Global Softw. Dev. Practitioner, pp. 38–44 (2006)