A novel use of big data analytics for service *innovation harvesting*

Aditya K. Ghose and Evan Morrison and Yingzhi Gou
Decision Systems Laboratory
School of Computer Science and Software Engineering
University of Wollongong, Australia
Email: { aditya, edm92, yg452}@uow.edu.au

Abstract

Service innovation has assumed considerable significance with the growth of the services sectors of economies globally, yet progress has been slow in devising carefully formulated, systematic techniques to under pin service innovation. This paper argues that a novel approach to big data analytics offers interesting solutions in this space. The paper argues that the use of big data analytics for generating enterprise service insights is often ignored (while the extraction of insights about customers, the market and the enterprise context has received considerable attention). The paper offers a set of techniques (collectively referred to as innovation harvesting*) which leverage big data in various forms, including object state sensor data, behaviour logs as well large-scale sources of open data such as the web to mine service innovation insights. The paper also outlines how systematic search might help overcome the limitations of big data analytics in this space.*

1 Introduction

Service innovation has assumed considerable significance with the growth of the services sectors of economies globally [13]. However, in practice, innovation in the services sector has met with less success that in the manufacturing and technology sectors [2]. Many of the methodological bases for innovation that have been developed for physical goods, technologies and software systems do not easily port to services [1]. Service innovation is a particularly hard problem.

There are multiple ways in which one might decompose the service innovation problem. den Hertog concludes that service innovation can be considered in four dimensions, (1) service concept, (2) client interface, (3) service delivery system and (4) technology [6]. Alternatively, we might conceptualize service innovation in terms of process innovation, enterprise structure innovation, service interface innovation and contractual/business model innovation. In a similar vein, Miles argues that all of these forms of innovation require reorganization of the processes and procedures underpinning services [12].

Innovation therefore involves identifying new ways of doing things (know-how or process innovation), new ways of organizing the enterprise, new service interfaces and novel business models or contractual arrangements. These innovations can be of two kinds: those that have been deployed elsewhere (but not in the context of the enterprise in question), and those that have never before been identified or deployed.

This paper offers means of leveraging both categories of innovation. We will argue that the former category of innovation can be *harvested*, both from open-source data (such as the web) or from data collected via bespoke instrumentation. We shall refer to these as *harvested innovations*. The second category of innovation (which have not been identified before and/or never deployed) shall be referred to as *discovered innovations*.

We shall argue that harvesting innovations from open data sources is often feasible, and shall illustrate this by describing a simple system that harvests know-how from the textual content of the web. We shall then illustrate the use of data collected using bespoke instrumentation in mining process semantics, and in mining intent (the techniques for these are closely related).

The data being leveraged in all of these applications is big data. In the first instance, the data store being mined is the textual content of the web. In the second instance, the data sources are object state sensors. In mining intent, we use large repositories of behaviour logs as the data source. The final step in this account is the use of systematic search in *discovering* innovation, which helps overcome some of the innate limitations of big data analytics in supporting innovation.

2 Innovation harvesting: General principles

Our approach to service innovation is underpinned by the following 7 principles/observations.

Data to support service innovation can be found almost everywhere. Many of the data sources that we will discuss below will be situated within enterprise boundaries, and might not be available for use outside of the enterprise. Many other data sources, however, are increasingly becoming open, such as open government data. Still other data sources, such as the content of the web, are intrinsically open. Finally, we now have the wherewithal to instrument *public good* data collection infrastructures that can generate vast quantities of open data (e.g., government-mandated logs of B2B and B2C interactions to be maintained by all government contractors).

Consider the following examples:

- *Process logs:* There is a long history of the use of process support systems in enterprise contexts. We therefore have a large and growing repository of process logs to draw upon for innovation insights.

- *Interaction/message logs:* Organizations (often driven by compliance considerations) routinely maintain logs of internal messaging (emails, memos or even simpler semaphores) as well as external (B2B or B2C) interactions.

- *Object state monitors:* A range of objects are involved in the delivery of services. A range of objects are also impacted by services. The objects in question might be physical objects or business objects such as an insurance claim, a credit application etc. It is possible to instrument state monitors to log state changes of both types of objects.

- *Legacy artefacts:* Legacy artefacts such as UML models, textual artefacts etc. can also be a rich source of innovation insights.

- *The web:* The textual content of the web can serve as an important source of insights.

In the remainder of the paper we will offer examples of the use of each of these types of data sources in harvesting innovation.

Service innovation requires identifying novel enterprise architectures. Enterprise architectures offer answers to the questions of who, what and when, and offer an integrated view of the enterprise from multiple perspectives, including process models, role models, goals models etc. Service innovation must be underpinned by novel conceptualizations of these (such as novel process models, novel sets of strategic objectives and so on). One might then, tentatively, conflate service innovation with innovative conceptualizations of the enterprise as seen through the lenses of these models.

Legacy artefacts can be mined for components of an enterprise architecture. Novel enterprise architecture components can be mined from legacy artefacts such as the textual content of the web, or legacy models. A novel organization of roles within an enterprise might be extracted from text on the web, or from a legacy UML sequence diagram. A novel process design can similarly be extracted from web-based text, or a legacy collection of UML state diagrams.

Bespoke instrumentation can be leveraged for mining the formal semantics for key components of an enterprise architecture. The need for formal semantics for the components of an enterprise architecture is well recognized. For instance, formal semantics enable consistency and completeness analysis of goals. Formal semantics annotation of process models enables sophisticated compliance management. Formal annotation of behavioural models, in general, permits goal realization analysis. It is possible to obtain formal semantic annotations by instrumenting state monitors for objects impacted by a service, or involved in the delivery of a service.

It is possible to mine organizational intent from behaviour histories. An important element of service innovation is understanding enterprise intent (the goals/objectives that services must be designed to satisfy). Enterprise intent is often not articulated or carefully documented. It is possible to mine a history of prior behaviour to obtain a reasonably accurate representation of the intent that is manifested in such behaviour.

Innovation requires departure from past precedent. This can involve both harvested innovation and discovered innovation. Harvested innovation can be obtained by looking for clues for service design and the supporting enterprise architecture design outside enterprise boundaries. This permits the enterprise to innovate by departing from past enterprise precedent. More generally, the need to depart from past precedent also forms the basis for an important observation about the limits of big data analytics as a basis for innovation. Big data analytics offers a means for learning from, and thence perpetuating, past precedent. Innovation, on the other hand, requires us to explore designs that have never before been considered.

Discovered innovation must be underpinned by search. The limitations of big data analytics in service innovation can be surmounted via the application of systematic search through the space of potential designs of services and service delivery environments. We outline how this might be achieved in the context of innovation in business process design later in this paper.

3 Harvesting know-how

A key component of service design is *know-how* - knowledge about the means to achieve desired ends. The representation of know-how can take many forms. Goal models offer rich encodings of know-how. A goal model describes how high-level goals might be refined/decomposed to sub-goals, which are in turn further decomposed into sub-goals, with the lowest level of refinement relating sub-goals to artefacts describing behaviour/functionality (such as requirements, designs, procedures and so on). Goal models describe know-how in a *sequence-* or *coordination-agnostic* fashion, i.e., they describe how goals might be realized via sub-goals without specifying the sequencing (or coordination via more complex control structures) of these sub-goals. Other schemes for specifying know-how, such as procedures or process models, relate goals with sub-goals whose achievement is procedurally specified.

Legacy artefacts represent a rich, but often-ignored, source of know-how. We classify such artefacts (henceforth called *source artefacts*) into two categories: *text* and *model*. Text artefacts are documents such as memos, manuals, requirements documents, design documents, mission/vision statements, meeting minutes etc. Model artefacts could be models in a variety of notations, including UML design models, or enterprise models or rule models. Source artefacts for know-how might be found in enterprise repositories or outside enterprise boundaries (the latter are particularly important as sources of innovative insights).

In our earlier work on the *R-BPD* framework [4], we built a tool-kit that sought to improve the productivity of business analysts engaged in the exercise of business process modeling by shifting the focus in modeling from model-building to model-editing. Our aim was to address the *model acquisition bottleneck*, a version of the well-known *knowledge acquisition bottleneck*. *R-BPD*, from a different perspective, also represents a know-how harvesting tool. *R-BPD* defined two categories of model extraction techniques: text-to-model extraction (for extracting process models from text artefacts) and model-to-model extraction (for extracting process models from models in other notations). Our guiding premise was that most organizations maintain enterprise repositories of (sometimes legacy) documents and models which can provide rich sources of information that could be mined to extract "first-cut" process models. Our premise was also that by extracting such "first-cut" models (henceforth referred to as *proto-models*) and presenting them to an analyst for editing, such a toolkit could significantly enhance analyst productivity. Given that organizations are often loathe to invest the resources required for significant modeling exercises, the availability of such a toolkit could make modeling-in-the-large a viable option in many instances.

In addition to defining novel text-to-model and model-to-model extraction techniques, the *R-BPD* framework had to address a range of other challenges. The tool was able to extract a large number of (sometimes small) process proto-models from an enterprise repository. Some of these proto-models were in fact alternative descriptions of the same process. We developed heuristic techniques for establishing *model identity* to deal with such situations. When multiple models that seem to describe the same process are identified, we need to cross-validate these against each other. We used *model consistency* as a basis for cross-validation, i.e., alternative consistent descriptions of the same process are viewed as supporting each other.

The toolkit turned out to be effective in achieving its original goals. Our empirical evaluation suggested that using the tool could reduce modeling effort by as much as two-thirds.

R-BPD can thus form the basis for discovering know-how in the form of proto-process models from repositories of text and model artefacts. It is possible to conceive of "public good" repositories of such artefacts being created to support innovation harvesting of this form. Our recent work has explored the efficacy of text-to-model extraction techniques in the context of what might be viewed as the largest "public good" repository of all: the world-wide web. We instrumented web crawlers together with a natural language processing (NLP) toolkit NLTk [8] to trawl the web for commonly used natural language patterns for describing know-how. The *Know-How Miner* tool design consists of two key components:

- *The know-how extractor:* This is a high-throughput engine for extracting *know-how descriptors* from the textual content of the web. In a recent experiment, a single crawler, was able to produce 22 useful know-how descriptors in the space of about 30 minutes from 446 webpages. The extractor was instrumented to extract instances of a single textual pattern: `to <GOAL>, <STEP1>, <STEP2>...` (the steps to be separated by an AND or an OR). The following are two examples of the descriptors extracted:

  - `<to create a robust business plan>`
    `take a comprehensive view of the enterprise`
    `AND`
    `incorporate management-practice knowledge from every first-semester course`
  - `to <gain a broader community consensus>`
    `<raise issues at the Village Pump>`

```
OR
initiate a Request for Comment
```

The first descriptor is an example of a fairly general statement of know-how, while the second is an example of a descriptor that is quite context-specific. While the former is readily usable, even the latter can be leveraged for value if there is some modicum of human input (e.g., ontological markup that identifies the "Village Pump" as a community newsletter).

- *The know-how filter:* It is clear that machinery such as this can generate very large volumes of know-how descriptors. Leveraging value from such a repository requires the use of filters that retrieves from the repository those descriptors that are relevant for a specific goal or context. A detailed description of the design of such filters is outside the scope of this paper.

The key conclusion to be drawn from the preceding discussion is that machinery to mine relevant know-how from textual source on the web, as well as from repositories of legacy textual and model artefacts is feasible.

4 Mining semantics

In this section we will outline one approach to extracting semantic annotations for business process designs by using a network of object state sensors/monitors coupled with execution histories (e.g. process logs).

We will illustrate the machinery for mining process semantics by using the ProcessSEER framework [7], which permits us to determine, at design time, the answer to the following question that can be posed for any point in the process design: what would the effects of the process be if it were to execute up to this point? The answer is offered in the form of a set of assertions in the underlying formal language (typically first-order logic, but more expressive languages, such as CTL could be used in its place). The answer is necessarily non-deterministic, since a process might have taken one of many possible alternative paths through a process design to get to that point. The non-determinism also arises from the fact that the effects of certain process steps might "undo" the effects of prior steps - the inconsistencies that result in the "snapshot" of the domain that we seek to maintain might be resolved in multiple alternative ways (a large body of work in the reasoning about action community addresses this problem). The answer to the question is therefore provided via a set of effect scenarios, any one of which might eventuate in a process instance. The ProcessSEER approach simplifies the task of semantic effect annotation by only requiring that tasks (populating a capability library) be annotated with context-independent immediate effects. The ProcessSEER tool then contextualizes these effects by propagating them through a process model (specified in BPMN in the current instance) to determine the cumulative effect scenarios at the end of each task. ProcessSEER uses formal machinery (theorem-provers) to compute cumulative effects, but provides an analyst-friendly Controlled Natural Language (CNL) interface, coupled with a domain ontology, that permits the immediate effects of tasks to be specified in natural language (but with a restricted set of sentence formats). The use of CNL permits us to translate these natural language specifications into underlying formal representation, which in turn makes the use of theorem-provers possible. ProcessSEER also makes provision for local (task-specific) non-functional annotations to be propagated through a process design, so that we are able to determine the cumulative non-functional scenarios for each task in a process design as well.

This approach to semantic effect annotation of process models has turned out to be effective in supporting business process compliance analysis [3] (the effect scenarios specify all possible outcomes of the execution of a business process), goal markup of business processes [9], consistency analysis of inter-operating business processes [10] and the design of enterprise process architectures [11].

The acquisition of immediate effect annotation of tasks in the capability library remains an open problem. While specifying effects in a context-independent fashion for tasks in the capability library is significantly simpler than specifying the effects of every task in every process design of interest, analysts are typically reluctant to engage in any kind of semantic effect annotation exercise (indeed, the organizational and human impediments to the more basic exercise of process modeling are well-recognized). Our recent work addresses this problem by developing an automated machinery for generating a set of "first-cut" effect annotations, which can be subsequently refined/edited by analysts.

Our approach relies on identifying the set of objects (both physical and business objects) impacted by a business process and instrumenting a set of state sensors/monitors for each of these, such that it is possible to identify the state of each object (in terms of a pre-determined vocabulary of state variables) at any point in time. One might conceive of an *effect log* (in a manner akin to a process log) consisting of a series of time-stamped entries, with each entry describing the state of every object of interest and with an entry for every state change in any object of interest. Placing a process log and an effect log in temporal justification provides a rough indication of which state changes might have triggered by the execution of a given task. A simple solution is to identify state changes (in terms of changes of values assigned to state variables) as the effects of the execution of temporally co-occurring tasks. This solution works under two critical assumptions. First, we must assume that

only one process is executed at any point in time, so that any state changes manifested can be causally traced back to the execution of that process and no other. Second, we must assume that every state change is a direct effect of a process task (and not a *ramification* or indirect effect). Both assumptions are somewhat restrictive. In general, multiple processes might be co-incident on a given set of objects, and many of the state changes sensed might be the indirect effects of process tasks. A more sophisticated solution involves setting up an abductive reasoning problem, with the immediate effects of process tasks serving as the set of abducibles, and the set of relevant domain constraints and causal rules forming the knowledge base. A range of heuristics can be brought to bear on the problem of deciding which of the multiple possible abductive explanations for a set of co-occurring state changes represent the best description of the effects of a given task. A detailed description of this machinery is outside the scope of this paper, but an important conclusion from this account is that a large-scale instrumentation of object state sensors can be leveraged for obtaining formal semantic annotations of behavioural models.

5 Mining intent

An important question in the context of service innovation (and the related problem of documenting enterprise architectures) is identifying *enterprise intent*, i.e., the business or strategic objectives of the enterprise. Organizations often neglect to carefully document their goals and strategies, and what little is documented is often in the form of a very high-level strategy document defined at the board-level. It is also not uncommon to find situations where the externally manifested behaviour of the enterprise deviates from its declared high-level strategies.

Our recent work on strategy mining leverages a Business Objective Modeling Language (BOML) [5] which offers the following modeling constructs:

- A *goal*: Goals are descriptions of conditions that an organization seeks to achieve. Goals admit boolean evaluation, i.e., an organization is able to clearly determine whether it has achieved a goal or not (consider the following example: *"Our corporate strategy is to be the market leader in mobile handsets"*). This notion of a goal closely corresponds to that used in goal-oriented requirements engineering.

- An *objective function*: An objective function is a construct used in operations research techniques to define what the *preferred* or *optimal* solution to an optimization problem might be. These are typically articulated as *maximize f* or *minimize f*, where f is a function defined on the *decision variables* (using which the constraints that *feasible solutions* are required to satisfy are also written). Our analysis of a large number of actual corporate strategy documents, as well as the management literature, suggests that strategies involving *corporate performance measures* or *key performance indicators (KPIs)* are articulated in the form of maximization or minimization objectives. Consider the following statements of strategy: *"Our strategy is to minimize order lead times", or, "Our strategy is to maximize customer satisfaction"*. In the first of these, the intent is to minimize a function encoding order lead time while in the second, a function encoding some definition of customer satisfaction (for instance, using average customer wait times at the customer contact centre, the number of escalations, the number of product returns etc.) is maximized.

- A *plan*: A plan is a set of goals together with a set of sequencing and related coordination constraints. In the most general sense, a plan can be as complex as a process model. In this paper, we will view plans only as linear sequences of goals. This is because SML is designed to be used by senior management, i.e., individuals within an organization who might be involved in strategy formulation. Also, our analysis of a large number of actual corporate strategy documents suggests that strategies are typically articulated at a very high level of abstraction, where control structures more complex than linear sequencing is never required. A typical example is the following anonymized but actual strategy statement: *"Our strategy is to first gain market acceptance in NZ, then position ourselves in the UK market, then use the UK market credibility to enter the Australian market"*. There are three steps (goals) in this strategy, connected via a linear sequencing relation.

The ServAlign framework and tool [5] supports the automated analysis of alignment of service (or capability) designs with strategy models represented in BOML. Service designs are specified in terms of service post-conditions (in the same underlying formal language that one might use in the context of semantic effect annotation of process models) and QoS guarantees (typically in the form of linear inequalities involving QoS measures). The alignment machinery involves a novel combination of goal satisfaction analysis, plan conformance analysis and optimization (with respect to the objective functions defined in the BOML strategy model).

Our approach to strategy mining effectively inverts this machinery. We assume the existence of a *behaviour log* representing the strategic behaviour of the enterprise. A behaviour log is a process log extended with *decision entries*, one for each critical enterprise decision. Each decision entry documents the available choices and the selected option

for each decision. Our approach to *goal mining* involves an extension of the effect mining machinery for business processes described in the previous section. Our approach to *plan mining*, as described in [15], extends the process mining machinery implemented in ProM [14]. Our approach to the mining of objective functions involves plotting each decision entry in the behaviour log in an n-dimensional space (where n is the number of distinct QoS variables) and using support vector machines to infer the best expression (in terms of the available QoS variables) that discriminates between the available options and selected one. We do not detail this machinery here due to space constraints. The key observation here is that it is possible to devise automated machinery for inferring enterprise intent from logs of enterprise behaviour.

6 Systematized innovation

As discussed earlier, innovation requires a departure from past precedent. Yet many current approaches to supporting innovation, such as open innovation portals [**?**], are somewhat ad-hoc and have met with limited success. We argue that innovation must ultimately be underpinned by systematic search. The best metaphor is that of Thomas Edison's search for the material that would serve as the filament of the electric bulb. His search was systematic, through a space of several thousand materials, until he met with success with tungsten.

Service innovation can be similarly conceived of as a search problem. The search problem can be characterized by: (1) a search space, (2) innovation driver and (3) the search constraints. Consider the search for innovative process designs. We might be given an existing process design, with the need to improve processing time being the innovation driver. The search space in this instance would be the space of all possible process designs that might be generated from the current set of enterprise capabilities (even this set might be treated as extensible in more complex formulations of the problem). The set of search constraints would include the requirement to preserve the functionality of the current process. We would search through the space of process designs that realize the current process functionality (this might be specified in terms of the final effect scenarios of the current, semantically annotated, process design) to identify the design that minimizes processing time.

Our current research addresses this problem. The search space is typically quite large, and good heuristics are required to improve search efficiency. Yet, it is useful to note that there are few real-time constraints on this search, which might be executed in the background over a prolonged period of time.

Discovered innovations, not only from process designs, but for other types of artefacts involved in the specification of an enterprise architecture can be systematically derived via a process of search as outlined above. This is a critical ingredient of service innovation, where we recognize the limits of big data analytics, and seeks to surpass these using systematic search.

7 Conclusions

This paper argues that a novel approach to big data analytics offers interesting solutions in this space. The paper argues that the use of big data analytics for generating enterprise service insights is often ignored (while the extraction of insights about customers, the market and the enterprise context has received considerable attention). The paper offers a set of techniques (collectively referred to as *innovation harvesting*) which leverage big data in various forms, including object state sensor data, behaviour logs as well large-scale sources of open data such as the web to mine service innovation insights. The paper also outlines how systematic search might help overcome the limitations of big data analytics in this space.

References

[1] M. J. Bitner, A. L. Ostrom, and F. N. Morgan. Service blueprinting: A practical technique for service innovation. *California Management Review*, 50(3):66 – 94, 2008.

[2] R. K. et al. Breakthrough ideas for 2005. *Harvard Business Review*, 83(2):17 – 54, 2005.

[3] A. K. Ghose and G. Koliadis. Auditing business process compliance. In *Proceedings of the International Conference on Service-Oriented Computing (ICSOC-07)*, 2007.

[4] A. K. Ghose, G. Koliadis, and A. Cheung. Rapid business process discovery (r-bpd). In *Proc. of the 2007 ER Conference (ER-2007)*. SPringer LNCS, 2007.

[5] A. K. Ghose, E. Morrison, and L.-S. Le. Correlating services with business objectives in the servalign framework. In *Proc. of the CAISE-2013 Workshop on Business IT Alignment (BUSITAL-13)*. Springer LNBIP, 2013.

[6] P. D. Hertog. Knowledge-intensive business services as co-producers of innovation. *International Journal of Innovation Management*, 4(4):491, 2000.

[7] K. Hinge, A. Ghose, and G. Koliadis. Process seer: A tool for semantic effect annotation of business process models. In *Enterprise Distributed Object Computing Conference, 2009. EDOC '09. IEEE International*, pages 54 –63, 2009.

[8] E. Klein. Computational semantics in the natural language toolkit. In *Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006)*, 2006.

[9] G. Koliadis and A. K. Ghose. Relating business process models to goal-oriented requirements models in kaos. In *Proc. of the 2006 Pacific-Rim Knowledge Acquisition Workshop (PKAW-2006)*, 2006.

[10] G. Koliadis and A. K. Ghose. Semantic verification of business processes in inter-operation. In *Proc. of the 2007 IEEE*

Services Computing Conference. IEEE Computer Society Press, 2007.

[11] G. Koliadis and A. K. Ghose. Towards an enterprise business process architecture standard. In *Proc. of the 2008 IEEE Services Congress*. IEEE Computer Society Press, 2008.

[12] I. Miles. Patterns of innovation in service industries. *IBM Systems Journal*, 47(1):115–128, 2008.

[13] J. Spohrer and D. Riecken. Introduction. *Commun. ACM*, 49(7):30–32, July 2006.

[14] W. M. P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng*, 16(9):1128–1142, 2004.

[15] H. Xu, B. T. R. Savarimuthu, A. K. Ghose, and E. Morrison. Automatic bdi plan recognition from process and logs and event logs. In *Proc. of the AAMAS-2013 Workshop on Engineering Multi-Agent Systems*.