Correlating Services with Business Objectives in the ServAlign Framework

Aditya Ghose, Lam-Son Lê, and Evan Morrison

Decision Systems Lab, School of Computer Science and Software Engineering
University of Wollongong, New South Wales 2522, Australia
{aditya,edm92,lle}@uow.edu.au

Abstract. We present a novel approach to modeling business objectives (strategies) and a novel notion of alignment between strategy and service models leading to the successfully deployed ServAlign tool that supports automated alignment analysis.

Keywords: business objective modeling, intentional service design, strategic service alignment.

1 Introduction

The central role of the notion of *business objectives* in the theory and practice of management has long been recognized. A framework, methodology and supporting toolkit for strategic service alignment underpins services management in a variety of ways. The ability to assess alignment between the set of service offerings of an organization and its strategic objectives is critical as a "correctness check" for its operations. Alignment checking can help reveal *unrealized strategies* (i.e., strategies for which no operational support exists within the organization). It can also reveal *redundant services* (i.e., services that do not support any strategic intent). Alignment can provide the basis for service design by using the strategic landscape of an organization to guide the design of its services. The re-alignment of services to an altered strategic landscape provides the machinery for organizational response to dynamic business contexts. The alignment of strategies and business functionality has received considerable recent attention [1,2,3,4,5] but much remains to be done on 3 counts: (1) Offering a richer vocabulary for describing business objectives (2) Offering a formal definition of correlation (or alignment) between service designs and business objectives and (3) Developing automated support for the analysis of correlation or alignment. The*ServAlign* framework addresses these challenges. The tool has been implemented and evaluated successfully in industry contexts (tool and evaluation details, plus comparison with related work, ommitted here due to space constraints, can be found at www.uow.edu.au/∼aditya/servalign).

2 A Business Objective Modeling Language

We present a high-level Business Objective Modeling Language (BOML) (in the following we will use "business objective" and "strategy" interchangeably) in the

X. Franch and P. Soffer (Eds.): CAiSE 2013 Workshops, LNBIP 148, pp. 162–167, 2013.

following. Its expressive adequacy was validated via analysis of a large number of examples of corporate strategy documentation in the published literature, as well as about 11 actual strategy documents.
We view a *business objective model* as a set of *business objective statements* containing the following:
(1) A goal: Goals are descriptions of conditions that an organization seeks to achieve (e.g., *"Our corporate strategy is to be the market leader in mobile handsets"*).
(2) A measure (objective function or constraint): An objective function is a construct used in operations research techniques to define what the *preferred* or *optimal* solution(s) 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* (such as those found in the Balanced Scorecard framework) or *key performance indicators (KPIs)* are articulated in the form of maximization or minimization objectives (e.g., *"Our strategy is to maximize customer satisfaction"*). Business objectives involving performance measures are sometimes written as *constraints*. These might occur directly or may obtain from the transformation of an objective function into a constraint via the introduction of a threshold on the value of the function (e.g., one way to articulate a strategy to minimize average customer wait times at a customer care center is to agree that this strategic objective would be satisficed if the average customer wait time was 3 minutes or less). Business objectives thus articulated lead to 2 distinct ontological constructs: *(1) objective functions and (2) constraints.*
(3) A plan: A plan is a set of goals together with a set of sequencing and related coordination constraints. In the general case, a plan can be as complex as a process model (with conditional split-join constructs), but our analysis of a number of corporate strategy documents suggests that strategies are typically articulated at a very high level of abstraction, where control structures more complex than linear sequencing are almost never required.. In this paper, we will view plans only as linear sequences of goals (e.g. *"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.

In order to relate business objectives with service designs, we will need to refine business objectives to a point where they are expressed in a vocabulary that (approximately) matches that in which services are represented. In the following, we will refer to strategies formulated as functional goals and plans as *functional strategies.*

Objective function refinement: In an optimization problem, we seek values for a set of *decision variables*, such that these satisfy a set of *constraints* on the decision variables, and the value of an objective function is either maximized or

minimized. One can view the process of computing an optimal solution as search through a space of *feasible solutions* (i.e., solutions which satisfy the set of constraints) in a manner guided by the objective function (which tells us which amongst a set of feasible solutions is most preferred). An objective function thus (implicitly) provides a *preference relation* on a set of *solutions*. In our setting, these solutions are not feasible (constraint-satisfying) assignments of values to decision variables - instead they are alternative realizations of functional strategies (i.e., mappings from functional strategies to sets of services). For now, an objective function will be *semantically* viewed (note that the *syntactic* representation remains unchanged) as a set of pairs $\{\langle S_1, S_2\rangle, \langle S_2, S_3\rangle, \ldots\}$ where each pair of the form $\langle S_1, S_2\rangle$ is a statement of preference over sets of services (in this instance, that the set S_1 is preferred to the set S_2. In the following, we will require a notion of *consistency of a set of objective functions*. Objective functions O1 and O2 are *inconsistent* if and only if there exists a pair of solutions S1 and S2 such that S1 is preferred over S2 by O1 and the reverse holds under O2 (note that this notion of consistency in contingent on the available set of solutions - O1 and O2 may be consistent given one set of solutions, but inconsistent given another). Formally, a refinement of an objective function F into a set of objective functions $\{f_1, f_2, \ldots, f_n\}$ is *valid* if and only if: (1) For any $\langle s_1, s_2\rangle \in F$, there does not exist any $\langle s_2, s_1\rangle \in f_i$ for any i. (2) $F \subseteq f_1 \cup f_2 \cup \ldots f_n$. (3) There does not exist any $F' \subset \{f_1, f_2, \ldots, f_n\}$ such that $F \subseteq \bigcup F'$. (4) $\{f_1, f_2, \ldots, f_n\}$ is consistent. The first condition in this definition ensures that the refinement of an objective function does not make statements of preference that contradict the parent objective function. The second condition states that all of the preferences statements implicit in the parent objective function must also be made by the combination of the refined objective functions (i.e., no statement of preference goes missing as a consequence of refinement). The third condition requires that the refinement is non-redundant, i.e., no objective function is included in the refinement that does not contribute to ensuring that all of the preference statements in the parent objective function are included in the refinement. Finally, we require the refined set of objective functions to be consistent.

In many situations, an objective function of the form *maximize* x can be refined to obtain {*maximize* y,*maximize* z}, based on the observation that $x = y+z$. Other commonly occurring patterns exist, e.g., *maximize* x refined to obtained {*maximize* y, *minimize* z }, based on $x = y/z$. If the variables in question are constrained to take on positive real values (as is typically the case with most QoS metrics), then refinements such as these are guaranteed to satisfy the semantics above. There are situations, however, where the explicit application of these semantics can be used to detect invalid refinements. Consider two services S_1 and S_2, in a setting where services are characterized by QoS measures m_1, m_2 and m_3, all of which we seek to maximize. Assume that m_1 is a somewhat more abstract QoS metric than the other two, such that in most (but not all) settings, the following holds: $m_1(S) = m_2(S) + m_3(S)$, for some service S. If we erroneously refine the objective function *maximize* $m_1(S)$ to obtain {*maximize* $m_1(S)$,*maximize* $m_2(S)$}, we may end up with an anomalous situation where

$m_1(S_2) < m_1(S_1)$ and $m_2(S_2) < m_2(S_1)$, but $m_3(S_1) < m_3(S_2)$. The application of the semantics above helps detects situations such as these. The ServAlign toolkit is designed to support such checking.

Following the KAOS framework, we define a refinement of a goal G into a set of sub-goals $\{g_1, g_2, \ldots, g_n\}$ to be *valid* if and only if (we assume here that a set of goals refers to the conjunction of its elements): (1) $g_1 \wedge g_2 \wedge \ldots \wedge g_n \not\models \perp$, (2) $g_1 \wedge g_2 \wedge \ldots \wedge g_n \models G$ and (3) $G' \not\models G$ for any $G' \subset \{g_1, g_2, \ldots, g_n\}$. Note that the refinement of strategies articulated as constraints is identical to the goal refinement machinery above.

We require a notion of goal (or effect) accumulation in order to define plan refinement. Plan refinement involves the refinement of the constituent goals, but with the proviso that these sub-goals are also appropriately sequenced and that the cumulative effect of achieving these goals in the specified order leads to the achievement of the parent goal. We assume without loss of generality that goals are represented in conjunctive normal form (CNF) as *prime implicates*. Given two goals g_1 and g_2 (now viewed as sets of clauses) such that g_1 is sequenced before g_2 in a plan, the cumulative goal achieved by the achievement of these two goals in sequence is given by the pairwise goal accumulation function $acc(\langle g_1, g_2\rangle) = g_2 \cup \{g_1' \mid g_1' \subseteq g_1, g_1' \cup g_2$ is satisfiable and for any g_1'' where $g_1' \subset g_1'' \subseteq g_1, g_1'' \cup g_2$ is unsatisfiable$\}$. We use this to define a goal accumulation function over a sequence of goals, $Accm(\langle g_1, g_2, \ldots, g_n\rangle)$. If $Plan = \langle g_1, g_2\rangle$, then $Accm(Plan) = acc(Plan)$. If $Plan = \langle g_1, g_2, \ldots, g_n\rangle$ is a sequence with more than two goals, then $Accm(Plan) = acc(Accm(\langle g_1, g_2, \ldots, g_{n-1}\rangle), g_n)$. Note, *acc* and hence *Accm* are sets of sets of clauses in general (accumulation of goals may lead to multiple mutually-exclusive non-deterministic outcomes). The refinement of a plan $P = \langle g_1, g_2, \ldots, g_n\rangle$ to a sequence of sub-plans $\langle\langle g_{11}, g_{12}, \ldots, g_{1p}\rangle, \langle g_{21}, g_{22}, \ldots, g_{2q}\rangle, \ldots, \langle g_{n1}, g_{n2}, \ldots, g_{nr}\rangle\rangle$ is *valid* if and only if: (1)For each $g_i \in P$, $\{g_{i1}, g_{i2}, \ldots, g_{is}\}$ is a valid (goal) refinement of g_i. (2) For every $e \in Accm(P)$, there exists an $e' \in Accm(\langle g_{11}, g_{12}, \ldots, g_{1p}, g_{21}, g_{22}, \ldots, g_{2q}, \ldots, g_{n1}, g_{n2}, \ldots, g_{nr}\rangle$ s.t. $e' \models e$ These semantic constraints on *valid strategy refinements* can be leveraged in three ways in the ServAlign toolkit. First, it provides a means of generating strategy refinements in an automated fashion that is correct with respect to these semantics. Second, it provides a mechanism for checking user-generated refinements. Finally, it provides the basis for generating a library of *strategy refinement patterns* that a user may potentially use.

3 Strategic Service Alignment: Foundations

A *strategic service architecture* consists of:
(1) A strategy model: A 4-tuple $\langle Goals, Plans, Objectives, Constraints\rangle$. We will assume that strategies at varying levels of abstraction are related via a set of strategy refinement links.

(2) A services model: This consists of a set of business services (we require that these be represented, at a minimum, in terms of formal *post-conditions* and *QoS constraints*). We do not consider the sub-structure of services in this paper, or control flow relations between them (these are addressed elsewhere). *(2) A set of strategy-service realization links:* These relate strategies to the services that realize them and can be of two kinds: *basic realization links* (which directly associate strategies, at the most refined level, with services) and *derived realization links* (such a link would associate a strategy to a service via a refinement of the former, which be directly related to the latter via a basic realization link).

We will refer to any strategy associated with a service via a realization link as a *strategic antecedent* of the service. Let Con_{Serv} be the set of QoS constraints described in any service contained in a set of services $Serv$ (we assume a uniform *global* vocabulary for QoS variables). A set of services $Serv$ is *more preferred* than another set $Serv'$ under a set of consistent objective functions O if the value of the optimal solution of the optimization problem $\langle f, Con_{Serv}\rangle$ is greater than the value of the optimal solution of the optimization problem $\langle f, Con_{Serv'}\rangle$, for every distinct objective function f in O (assuming, without loss of generality, that every objective function is a maximization objective). Given a consistent set of objective functions O, and given a functional strategy Str, an *optimal realization of Str relative to O* is any subset $Serv$ of the set of available services such that $Serv$ realizes Str and for any alternative subset $Serv'$ of the set of available services that might also realize Str, $Serv'$ is less preferred to $Serv$ under O. Basic realization links are established using the following machinery: (1) A set of services $\{s_1, s_2, \ldots, s_n\}$ *realizes* a goal g if and only if it is a minimal (with respect to set inclusion) set of services such that $Postcond(s_1) \wedge Postcond(s_2) \wedge \ldots \wedge Postcond(s_n) \models g$. (2) A set of services realizes a plan if and only if it is the minimal set of services (i.e. there is no subset that achieves the same condition) that realizes each of its constituent goals. (3) A set of services $Serv$ realizes an objective function f if and only if, for each strategic antecedent Str of a service in $Serv$ that is not an objective function or constraint, the optimal realization of Str relative to f is included in $Serv$. (4) We will use a constraint negation operator (using the standard negation symbol $\neg$) which asserts the negation of the associated constraint predicate. Thus, $\neg(x < 20)$ is $x \geq 20$, and so on. A set of services $Serv$ realizes a constraint c iff $Con_{Serv} \cup \neg c$ is unsatisfiable. Testing the satisfiability of a set of constraints thus obtained can be done using standard constraint solving techniques.

We will require a notion of *internal alignment* of a strategy model. A strategy model $\langle Goals, Plans, Objectives, Constraints\rangle$ (a 4-tuple consisting of the sets of goals, plans, objective functions and constraints respectively) is aligned iff: (1) $Goals \cup \{Accumulate(p) \mid p \in Plans\}$ is satisfiable, and (2)$Constraints$ is satisfiable and (3)$Objectives$ is consistent. We note that satisfiability checking for $Constraints$ will require the use of constraint solvers (simple solvers for linear inequalities on reals will suffice for most cases, but more sophisticated solvers may be required depending on the expressivity/complexity of the language for defining constraints on QoS factors). We note that a similar model of internal

alignment for the services model is not of interest. This is because there is no requirement for concurrent execution of all of the services in a services model (unlike the strategy model - where there is an implicit assumption that an organization seeks to *concurrently* achieve *all* of its strategies). It is reasonable, therefore, for an organization to support a service which increases exposure to a certain market and concurrently support another that reduces exposure to that market - the crucial point is that these services are not executed concurrently.

Non-obstruction is another critical attribute of alignment. A service S *obstructs* a functional strategy Str under the following conditions: (1)If Str is a goal, then $postcond(S) \cup Str \models \perp$ or (2)if Str is a plan, then $postcond(S) \cup Accumulate(Str) \models \perp$.

A strategic service architecture is *aligned* iff: (1) The strategy model is *internally aligned*. (2) Every user-specified strategy (as distinct from a strategy obtained via refinement) is *realized* by some subset of the services model. (3) Every service in the services model has a user-specified *functional strategy* as a *strategic antecedent* (note that it would not make sense to consider services that exist only to realize objective functions, and no functional strategy). (4) No service in the services model *obstructs* any functional strategy in the strategy model.

Assessing strategic alignment involves progressive refinement of the user-specified strategies until these use the same vocabulary as the services model. At this point, the test for alignment of a strategic service architecture is applied. The critical question is: how do we decide when to stop refining strategies any further? Ideally, when the concept vocabulary used for the most refined set of strategies is included in the concept vocabulary used to define the services model. The problem with this approach is that if the models were inherently mis-aligned, we would continue strategy refinement without termination and never find the required *ontological match*. Our approach, instead, is to leverage the degree of ontological match. We iterate the strategy refinement step as long as there is monotonic improvement in the degree of match between the resultant vocabularies, and stop prior to the degree of match decreasing.

References

1. Halleux, P., Mathieu, L., Andersson, B.: A method to support the alignment of business models and goal models. In: Workshop on Business/IT-Alignment and Interoperability (BUSITAL 2008), Montpellier, France (June 2008)
2. Barone, D., Jiang, L., Amyot, D., Mylopoulos, J.: Composite indicators for business intelligence. In: Jeusfeld, M., Delcambre, L., Ling, T.-W. (eds.) ER 2011. LNCS, vol. 6998, pp. 448–458. Springer, Heidelberg (2011)
3. Etien, A., Rolland, C.: A Process for Generating Fitness Measures. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 277–292. Springer, Heidelberg (2005)
4. Thevenet, L.-H., Salinesi, C.: Aligning IS to organization's strategy: The INSTAL method. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 203–217. Springer, Heidelberg (2007)
5. Pijpers, V., Gordijn, J., Akkermans, H.: e^3 alignment: Exploring inter-organizational alignment in networked value constellations. Journal of Computer Science & Applications 6(5), 59–88 (2009)